CONSTRUCTOR
UNIVERSITY

Study
Program
Handbook

Software, Data and Technology

Bachelor of Science

Subject-specific Examination Regulations for Software, Data and Technology

(Fachspezifische Prüfungsordnung)

The subject-specific examination regulations for Software, Data and Technology are defined by this program handbook and are valid only in combination with the General Examination Regulations for Undergraduate degree programs (General Examination Regulations = Rahmenprüfungsordnung). This handbook also contains the program-specific Study and Examination Plan (Chapter 5).

Upon graduation, students in this program will receive a Bachelor of Science (BSc) degree with a scope of 180 ECTS (for specifics see Chapter 4 of this handbook).

Version	**Valid as of**	**Decision**	**Details**
Fall 2026 – V 1.0	Sep 01, 2026		
		June 28, 2023	Academic Senate approval of study program name change from “Data Science and Software Development” to “Software, Data and Technology”
	Sep 01, 2023	May 24, 2023	Originally approved by Academic Senate

Contents

1 Program Overview

1.1 Concept

1.1.1 The Constructor University Educational Concept

Constructor University aims to educate students for both an academic and a professional career by emphasizing three core objectives: academic excellence, personal development, and employability to succeed in the working world. Constructor University offers an excellent research driven education experience across disciplines to prepare students for graduate education as well as career success by combining disciplinary depth and interdisciplinary breadth with supplemental skills education and extra-curricular elements. Through a multi-disciplinary, wholistic approach and exposure to cutting-edge technologies and challenges, Constructor University develops and enables the academic excellence, intellectual competences, societal engagement, professional and scientific skills of tomorrows leaders for a sustainable and peaceful future.

In this context, it is Constructor University's aim to educate talented young people from all over the world, regardless of nationality, religion, and material circumstances, to become citizens of the world who are able to take responsible roles for the democratic, peaceful, and sustainable development of the societies in which they live. This is achieved through a high-quality teaching as well as manageable study loads and supportive study conditions. Study programs and related study abroad programs convey academic knowledge as well as the ability to interact positively with other individuals and groups in culturally diverse environments. The ability to succeed in the working world is a core objective for all study programs at Constructor University, both in terms of actual disciplinary subject matter and also to the social skills and intercultural competence. Study-program-specific modules and additional specializations provide the necessary depth, interdisciplinary offerings and the minor option provide breadth while the university-wide general foundation and methods modules, optional German language and Humanity modules, and an extended internship period strengthen the employability of students. The concept of living and learning together on an international campus with many cultural and social activities supplements students' education. In addition, Constructor University offers professional advising and counseling.

Constructor University's educational concept is highly regarded both nationally and internationally. While the university has consistently achieved top marks over the last decade in Germany's most comprehensive and detailed university ranking by the Center for Higher Education (CHE), it has also been listed by one of the most widely observed university rankings, the Times Higher Education (THE) ranking. More details on the current ranking positions can be found at https://constructor.university/more/about-us.

1.1.2 Program Concept

Software, Data and Technology are at the forefront of modern industries and play a major role in most areas of science and technology. The field is constantly evolving, but the fundamental principles underlying these technologies have now developed into a mature discipline. The BSc Software, Data and Technology program at Constructor University focuses on the understanding of these principles and their application in practice.

Students will obtain core software, data and technology competencies and skills (e.g., programming, data analysis, and machine learning) and they will learn about fundamental abstractions and abstract

notions of software, data and technology (e.g., data structures, algorithms, and software design principles). They will learn about the principles behind and the proper usage of core technologies (e.g., databases, parallel programming, compilers, and data analysis). Finally, students will develop an understanding of the limitations of technology and side effects of software, data and technology systems (e.g., security, privacy, and ethical aspects).

Because software, data and technology are rooted in mathematics and computer science, students will take mathematical and computer science methods modules covering calculus, linear algebra, probability theory, statistics, and numerical methods or discrete mathematics.

The job market for computer scientists has been very favorable in the last few years, and there is no indication that this will change in the near future. Because of the rapid changes in the field, it is important to focus the education on the fundamental principles, as well as, subfields of promising future relevance. Cross-disciplinary breadth and flexibility, as well as social and work organization skills are increasingly important. The program offers a major option in Software, Data and Technology designed for students who plan to work in the information technology industry or join graduate programs related to the discipline.

In summary, the BSc Software, Data and Technology program at Constructor University is designed to provide students with the foundational knowledge, skills and understanding of the principles and application of software, data and technology in modern industries. With an emphasis on cross-disciplinary breadth and flexibility, students will be well-prepared for a wide range of career opportunities in the field.

1.2 Specific Advantages of Software, Data and Technology at Constructor University

The Software, Data and Technology program at Constructor University aims to provide students with a comprehensive and rigorous education in the foundations of software, data and technology, while also keeping the curriculum contemporary and internationally oriented.

- The program will focus on relating the theoretical concepts to their practical application in industry and research, with instructors incorporating recent developments and trends in the field to demonstrate how basic methods and techniques are being used today.
- Early involvement in research projects will be an integral aspect of the program, providing students with the opportunity to gain hands-on experience and potentially develop interdisciplinary collaborations.
- The program will be constantly fine-tuned through direct and open dialogue with students and alumni, ensuring that the curriculum meets their specific needs and prepares them for internships and job opportunities.
- One of the specific advantages of the program is its carefully designed curriculum with a diverse range of course offerings, providing students with a well-rounded understanding of the field and preparing them for various career paths. The program covers foundational subjects such as Linear Algebra, Analysis, Core and Advanced Algorithms & Data Structures, as well as more specialized subjects. These courses are structured to ensure a progressive learning experience, with advanced modules building on the knowledge acquired in earlier modules.
- In addition to the core curriculum, the program also offers three specialized tracks: Machine Learning, Software Development, and Programming Languages. These specialized tracks

provide students with the opportunity to delve deeper into specific areas of interest and gain expertise in their chosen field. The Machine Learning specialization, for example, offers additional courses such as Optimization Methods, Stochastic Modeling and Financial Mathematics, Deep Learning, and NLP. The Software Development specialization includes courses such as Databases Internals, Integrated Development and IT Operations, Software Design, Parallel Programming, and Distributed Algorithms, while the Programming Languages specialization includes Formal Languages and Parsers, Compilers, and Semantics of Programming Languages.

The close ties and support and participation of JetBrains in the development of the program will provide students with unique opportunities for project work, internships, and access to special courses, as well as the chance to receive scholarships covering tuition, boarding, insurance, and a monthly allowance.

1.3 Program-Specific Educational Aims

1.3.1 Qualification Aims

The main subject-specific qualification aim of the BSc Software, Data and Technology program at Constructor University is to enable students to take up qualified employment in modern industries involving software, data and technology or to enter graduate programs related to these fields. Graduates of the program will have the following competencies:

- Software, Data and Technology competence

Graduates will be familiar with the theoretical foundations of software, data and technology and will be able to design and develop systems addressing a given application scenario. They will be able to analyze and structure complex problems and will be able to address them using program specific methods. Graduates will be able to construct and maintain complex systems using a structured, analytic, and creative approach.

- Communication competence

Graduates will be able to communicate subject-specific topics convincingly in both spoken and written form to fellow data scientists, software developers, or customers.

- Teamwork and project management competence

Graduates will be able to work effectively in a team and will be able to organize workflows in complex development efforts. They will be familiar with tools that support the development, testing, and maintenance of large systems and will be able to take design decisions in a constructive way.

- Learning competence

Graduates will have acquired a solid foundation enabling them to assess their own knowledge and skills, learn effectively, and remain up to date with the latest developments in the rapidly evolving fields of software, data and technology.

- Personal and professional competence

Graduates will be able to develop a professional profile, justify professional decisions based on theoretical and methodical knowledge, and critically reflect on their behavior with respect to their consequences for society. Additionally, the program being developed with the support and

participation of JetBrains will provide students with the opportunity for project work and internships in the company.

1.3.2 Intended Learning Outcomes

By the end of the BSc Software, Data and Technology program, students will be able to

1. work professionally in the field of software, data and technology and enter graduate programs related to these fields;
2. apply fundamental concepts of mathematics, statistics, and computer science while solving data-related problems;
3. analyze at multiple levels of abstraction and use appropriate mathematical and computational methods to model and analyze real-world problems;
4. develop, analyze and implement algorithms using modern software engineering methods and programming languages;
5. understand the characteristics of a range of computing platforms and their advantages and limitations;
6. choose from multiple programming paradigms, languages and algorithms to solve a given problem adequately;
7. apply the necessary mathematical methods, such as linear algebra, analysis, calculus, and discrete mathematics;
8. recognize the context in which data science and software systems operate, including interactions with people and the physical world;
9. describe the state of published knowledge in the field of software, data and technology and in a chosen specialization (Machine Learning, Software Development, Programming Languages);
10. analyze and model real-life scenarios in organizations and industries using contemporary techniques of data science and software development, also taking methods and insights of other disciplines into account;
11. appropriately communicate solutions of problems in software, data and technology in both spoken and written form to specialists and non-specialists;
12. draw scientifically founded conclusions that consider social, professional, scientific, and ethical aspects;
13. work effectively in a diverse team and take responsibility in a team;
14. take responsibility for their own learning, personal and professional development and role in society, reflecting on their practice and evaluating critical feedback;
15. adhere to and defend ethical, scientific, and professional standards.

1.4 Career Options and Support

The Software, Data and Technology program at Constructor University offers students a wide range of career opportunities in the rapidly growing fields of computer science. As two of the key disciplines of

the 21st century, software, data and technology affect almost all modern industries, making the job market highly favorable for graduates with a degree in this field. The program equips students with the skills and knowledge necessary to excel in various roles such as data scientist, data analyst, software engineer, full-stack developer, information systems manager, database administrator, application developer, machine learning engineer, IT consultant, and system analyst.

In addition to the broad range of career options available to graduates, the program also boasts strong industry partnerships with companies such as JetBrains, Acronis, Alemira, Virtuozzo, Rolos, and others. These partnerships provide students with valuable opportunities for internships, networking, and career development.

The Career Service Center (CSC) helps students in their career development. It provides students with high-quality training and coaching in CV creation, cover letter formulation, interview preparation, effective presenting, business etiquette, and employer research as well as in many other aspects, thus helping students identify and follow up on rewarding careers after graduating from Constructor University. Furthermore, the Alumni Office helps students establish a long-lasting and global network which is useful when exploring job options in academia, industry, and elsewhere.

1.5 Admission Requirements

Admission to Constructor University is selective and based on a candidate's school and/or university achievements, recommendations, self-presentation, and performance on standardized tests. Students admitted to Constructor University demonstrate exceptional academic achievements, intellectual creativity, and the desire and motivation to make a difference in the world.

The following documents need to be submitted with the application:

- Recommendation Letter (optional)
- Official or certified copies of high school/university transcripts
- Educational History Form
- Standardized test results (SAT/ACT) if applicable
- Motivation statement
- ZeeMee electronic resume (optional)
- Language proficiency test results (TOEFL Score: 90, IELTS: Level 6.5 or equivalent)

Formal admission requirements are subject to higher education law and are outlined in the Admission and Enrollment Policy of Constructor University.

For more detailed information about the admission visit: https://constructor.university/admission-aid/application-information-undergraduate

1.6 More information and contacts

For more information on the study program, please contact the Study Program Coordinator:

Prof. Dr. Alexander Omelchenko

Professor of Applied Mathematics, Data Science and Computing

Email: aomelchenko@constructor.university

or visit our program website: https://constructor.university/programs/undergraduate-education/data-science-software-development

For more information on Student Services please visit:

https://constructor.university/student-life/student-services

2 The Curricular Structure

2.1 General

The curricular structure provides multiple elements for enhancing employability, interdisciplinarity, and internationality. The unique CONSTRUCTOR Track, offered across all undergraduate study programs, provides comprehensive tailor-made modules designed to achieve and foster career competency. Additionally, a mandatory internship of at least two months after the second year of study and the possibility to study abroad for one semester give students the opportunity to gain insight into the professional world, apply their intercultural competences and reflect on their roles and ambitions for employment and in a globalized society.

All undergraduate programs at Constructor University are based on a coherently modularized structure, which provides students with an extensive and flexible choice of study plans to meet the educational aims of their major as well as minor study interests and complete their studies within the regular period.

The framework policies and procedures regulating undergraduate study programs at Constructor University can be found on the website (https://constructor.university/student-life/student-services/university-policies).

2.2 The Constructor University 4C Model

Constructor University offers study programs that comply with the regulations of the European Higher Education Area. All study programs are structured according to the European Credit Transfer System (ECTS), which facilitates credit transfer between academic institutions. The three-year undergraduate programs involve six semesters of study with a total of 180 ECTS credit points (CP). The undergraduate curricular structure follows an innovative and student-centered modularization scheme - the 4C Model. It groups the disciplinary content of the study program in three overarching themes, CHOICE-CORE-CAREER according to the year of study, while the university-wide CONSTRUCTOR Track is dedicated to multidisciplinary content dedicated to methods as well as intellectual skills and is integrated across all three years of study. The default module size is 5 CP, with smaller 2.5 CP modules being possible as justified exceptions, e.g. if the learning goals are more suitable for 2.5 CP and the overall student workload is balanced.

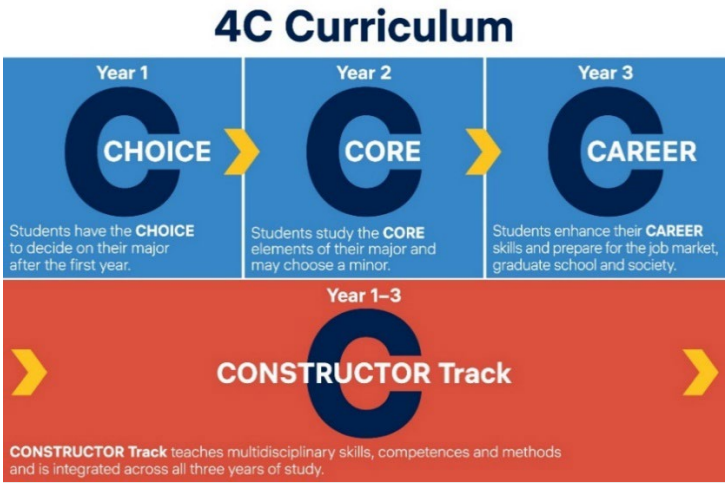

Figure 1: The Constructor University 4C-Model

2.2.1 Year 1 – CHOICE

The first study year is characterized by a university-specific offering of disciplinary education that builds on and expands upon the students' entrance qualifications. Students select introductory modules for a total of 45 CP from the CHOICE area of a variety of study programs, of which 15-45 CP will be from their intended major. A unique feature of our curriculum structure allows students to select their major freely upon entering Constructor University. The team of Academic Advising Services offers curriculum counseling to all Bachelor students independently of their major, while Academic Advisors, in their capacity as contact persons from the faculty, support students individually in deciding on their major study program.

To pursue a SDT major, the following CHOICE modules (30 CP) need to be taken as mandatory (m) modules during the first year of study:

- CHOICE Module: System Programming and Performance (m, 7.5 CP)
- CHOICE Module: Industrial Programming with Python (m, 5 CP)
- CHOICE Module: Core Algorithms and Data Structures (m, 7.5 CP)
- CHOICE Module: AI and Agentic Workflows (m, 5 CP)
- CHOICE Module: Compute System for AI and Performance (m, 2.5 CP)
- CHOICE Module: Practical Minimum (Git/Docker/CLI) (m, 2.5 CP)

The remaining two own CHOICE modules (15 CP) can be selected in the first year of study according to interest. For students not pursuing a minor the following modules are recommended in the first and second semesters:

- CHOICE module: Analysis (me, 7.5 CP)
- CHOICE module: Linear Algebra (me, 7.5 CP)

In total, the first-year modules lay the foundation for the second year of education within the SDT major.

Students can still change to another major at the beginning of their second year of studies, provided they have taken the corresponding mandatory CHOICE modules in their first year of studies. All students must participate in an entry advising session with their Academic Advisors to learn about their major change options and consult their Academic Advisor prior to changing their major.

Students that would like to retain a further option are strongly recommended to additionally register for the CHOICE modules of one of the following study programs in their first year:

- Computer Science (CS)
  CHOICE Module: Programming in C and C++ (7.5 CP)
  CHOICE Module: Algorithms and Data Structures (7.5 CP)
  CHOICE Module: Mathematical Foundations of Computer Science (7.5 CP)
  CHOICE Module: Digital Systems and Computer Architecture (7.5 CP)

- International Relations: Politics and History (IRPH)
  CHOICE Module: Introduction to International Relations Theory (m, 7.5 CP)

CHOICE Module: Introduction to Modern European History (m, 7.5 CP)

- Integrated Social and Cognitive Psychology (ISCP)
  CHOICE Module: Essentials of Cognitive Psychology (m, 7.5 CP)
  CHOICE Module: Essentials of Social Psychology (m, 7.5 CP)

To allow further major changes after the first semester the students are strongly recommended to register for the CHOICE modules of one of the following study programs:

- Physics and Data Science (PHDS)
  CHOICE Module: Classical Physics (m, 7.5 CP)
  CHOICE Module: Scientific Programming with Python (m, 7.5 CP)
  CHOICE Module: Modern Physics (m, 7.5 CP)
  CHOICE Module: Mathematical Modeling (m, 7.5 CP)

- Mathematics, Modeling and Data Analytics (MMDA)
  CHOICE Module: Analysis (m, 7.5 CP)
  CHOICE Module: Scientific Programming with Python (m, 7.5 CP)
  CHOICE Module: Linear Algebra (m, 7.5 CP)
  CHOICE Module: Mathematical Modelling (m, 7.5 CP)

- Robotics and Intelligent Systems (RIS)
  CHOICE Module: Programming in C and C++ (m, 7.5 CP)
  CHOICE Module: Digital Systems and Computer Architecture (m, 7.5 CP)
  CHOICE Module: General Electrical Engineering I (m, 7.5 CP)
  CHOICE Module: Algorithms and Data Structures (m, 7.5 CP)

The module descriptions can be found in the respective Study Program Handbook.

2.2.2 Year 2 – CORE

In their second year, students take a total of 45 CP from a selection of in-depth, discipline-specific CORE modules. Building on the introductory CHOICE modules and applying the methods and skills acquired thus far (see 2.3.1), these modules aim to expand students' critical understanding of the key theories, principles, and methods in their major for the current state of knowledge and best practice.

To pursue SDT as a major, the following mandatory (m) CORE modules (25 CP) must be taken:

- CORE Module: Operating Systems (m, 7.5 CP)
- CORE Module: Software Engineering and Design (m, 7.5 CP)
- CORE Module: Advanced Algorithms and Data Structures (m, 5 CP)
- CORE Module: Machine Learning (m, 5 CP)

Furthermore, students should take the following mandatory elective (me) modules:

- CORE Module: Functional Programming (me, 5 CP)
- CORE Module: Scientific Data Analysis (me, 5 CP)
- CORE Module: Database Fundamentals (me, 5 CP)
- CORE Module: Discrete Mathematics (me, 5 CP) OR Artificial Intelligence (me, 5 CP)

2.2.3 Year 3 – CAREER

During their third year, students prepare and make decisions about their career path after graduation. To explore available choices and to gain professional experience, students undertake a mandatory summer internship. The third year of studies allows SDT students to take Specialization modules within their discipline, but also focuses on the responsibility of students beyond their discipline (see CONSTRUCTOR Track).

The fifth semester also opens a mobility window for a diverse range of study abroad options. Finally, the sixth semester is dedicated to fostering the students' research experience by involving them in an extended Bachelor thesis project.

Internship / Start-up and Career Skills Module

As a core element of Constructor University's employability approach students are required to engage in a mandatory two-month internship of 15 CP that will usually be completed during the summer between the second and third years of study. This gives students the opportunity to gain first-hand practical experience in a professional environment, apply their knowledge and understanding in a professional context, reflect on the relevance of their major to employment and society, reflect on their own role in employment and society, and find a professional orientation. The internship can also establish valuable contacts for the students' Bachelor's thesis project, for the selection of a Master program graduate school or further employment after graduation. This module is complemented by career advising and several career skills workshops throughout all six semesters that prepare students for the transition from student life to professional life. As an alternative to the full-time internship, students interested in setting up their own company can apply for a start-up option to focus on developing of their business plans.

For further information, please contact the Career Service Center (CSC) (https://constructor.university/student-life/career-services).

Specialization Modules

In the third year of their studies, students take 15 CP from major-specific or major-related, advanced Specialization Modules to consolidate their knowledge and to be exposed to state-of-the-art research in the areas of their interest. This curricular component is offered as a portfolio of modules, from which students can make free selections within a track during their fifth and sixth semester. The three specialization tracks are: Data Science, Software Development and Programming languages. The default Specialization Module size is 5 CP, with smaller 2.5 CP modules being possible as justified exceptions.

To pursue SDT as a major, 15 CP from the following major-specific Specialization modules within one track need to be taken:

Data Science track:

- SDT Specialization: Optimization Methods (me, 5 CP)
- SDT Specialization: Stochastic Modeling and Financial Mathematics (me, 5 CP)
- MSc CSSE CORE: Deep Learning (me, 5 CP)
- SDT Specialization: Natural Language Processing (me, 5 CP)

Software Development track:

- SDT Specialization: Databases Internals (me, 5 CP)
- SDT Specialization: Integrated Development and IT Operations (me, 5 CP)
- SDT Specialization: Parallel Programming (me, 5 CP)
- CS Specialization: Distributed Algorithms (me, 5 CP)
- CS CORE: Computer Networks (me, 5 CP)

Programming Languages track:

- SDT Specialization: Formal Languages and Parsers (me, 5 CP)
- SDT Specialization: Compilers (me, 5 CP)
- SDT Specialization: Semantics of Programming Languages (me, 5 CP)
- SDT Specialization: Advanced Discrete Mathematics (me, 5 CP)

Specialization modules are designed to allow an SDT student to become more focused on a particular subject of their choice within the SDT program or an affiliated program. The intention is to simultaneously support their personal development and career choices.

Study Abroad

Students have the opportunity to study abroad for a semester to extend their knowledge and abilities, broaden their horizons and reflect on their values and behavior in a different context as well as on their role in a global society. For a semester abroad (usually the 5th semester), modules related to the major with a workload equivalent to 22.5 CP must be completed. Modules recognized as study abroad CP need to be pre-approved according to Constructor University study abroad procedures. Several exchange programs allow students to directly enroll at prestigious partner institutions worldwide. Constructor University's participation in Erasmus+, the European Union's exchange program, provides an exchange semester at a number of European universities that include Erasmus study abroad funding.

For further information, please contact the International Office (https://constructor.university/student-life/study-abroad/international-office).

SDT students that wish to pursue a study abroad in their fifth semester are required to select their modules at the study abroad partners such that they can be used to substitute between 10-15 CP of major-specific Specialization modules and between 5-15 CP of modules equivalent to the non-disciplinary New Skills modules (see CONSTRUCTOR Track). In their sixth semester, according to the study plan, returning study-abroad students complete the Bachelor Thesis/Seminar module (see next section), they take any missing Specialization modules to reach the required 15 CP in this area, and they take any missing New Skills modules to reach 15 CP in this area.

Bachelor Thesis/Seminar Module

This module is a mandatory graduation requirement for all undergraduate students. It consists of two module components in the major study program guided by a Constructor University faculty member:

the Bachelor Thesis (12 CP) and a Seminar (3 CP). The title of the thesis will appear on the students' transcripts.

Within this module, students apply the knowledge skills, and methods they have acquired in their major discipline to become acquainted with actual research topics, ranging from the identification of suitable (short-term) research projects, preparatory literature searches, the realization of discipline-specific research, and the documentation, discussion, and interpretation of the results.

With their Bachelor Thesis students demonstrate mastery of the contents and methods of their major-specific research field. Furthermore, students show the ability to analyze and solve a well-defined problem with scientific approaches, a critical reflection of the status quo in scientific literature, and the original development of their own ideas. With the permission of a Constructor University Faculty Supervisor, the Bachelor Thesis can also have an interdisciplinary nature. In the seminar, students present and discuss their theses in a course environment and reflect on their theoretical or experimental approach and conduct. They learn to present their chosen research topics concisely and comprehensively in front of an audience and to explain their methods, solutions, and results to both specialists and non-specialists.

2.3 The CONSTRUCTOR Track

The CONSTRUCTOR Track is another important feature of Constructor University's educational model. The CONSTRUCTOR Track runs parallel to the disciplinary CHOICE, CORE, and CAREER modules across all study years and is an integral part of almost all undergraduate study programs. It reflects a university-wide commitment to help transform late-stage adolescents into confident, competent and responsible young adults by providing an intellectual tool kit to become life-long learners and by giving them the capacity to employ a range of methodologies to approach potential solutions to problems across disciplines. The CONSTRUCTOR track comprises 45 CP in total and contains Methods, New Skills and German Language/Humanities modules.

2.3.1 Methods Modules

Methods such as mathematics, statistics, programming, data handling, presentation skills, academic writing, and scientific and experimental skills are offered to all students as part of the Methods area in their curriculum. The modules that are specifically assigned to each study program equip students with transferable academic skills. They convey and practice specific methods that are indispensable for each students' chosen study program. Students are required to take a minimum of 20 CP in the Methods area. The size of all Methods modules is 5 CP.

To pursue Software, Data and Technology as a major, the following Methods modules (20 CP) need to be taken as mandatory modules:

- Methods Module: Basics of Discrete Mathematics (m, 5 CP)
- Methods Module: JVM-Based Programming (m, 5 CP)
- Methods Module: Probability and Random Processes (m, 5 CP)
- Methods Module: Statistics and Data Analytics (m, 5 CP)

2.3.2 New Skills Modules

This part of the curriculum constitutes the intellectual and conceptual tool kit, and is designed to cultivate and nurture the capacity for a particular set of intellectual dispositions – curiosity, imagination, critical thought, transferability – as well as a range of individual and societal capacities – self-reflection, argumentation and communication – and to introduce students to the normative aspects of inquiry and research – including the norms governing sourcing, sharing, withholding materials and research results as well as others governing the responsibilities of expertise as well as the professional point of view.

All students are required to take 20 CP of New Skills modules in their second and third year of studies. Students choose New Skills Modules from the following portfolio:

- New Skills Module: Logic* (me, 2.5 CP)
- New Skills Module: Causation and Correlation* (me, 2.5 CP)
- New Skills Module: Argumentation, Data Visualization and Communication* (me, 5 CP)
- New Skills Module: Models, Matrices and Theories of Complex Systems (me, 5 CP)
- New Skills Module: Complex Problem Solving (me, 5 CP)
- New Skills Module: Agency, Leadership and Accountability (me, 5 CP)
- New Skills Module: Community Impact Project (me, 5 CP).

Some of these modules (marked by *) will be offered with two different perspectives from which the students can only choose one for the fulfillment of graduation requirements of their degree program. The module perspectives are independent modules which examine the topic from different points of view. Please see the module description in chapter 7 for more details such as scheduling, content and assessment.

2.3.3 German Language and Humanities Modules

Students are required to take 5 CP of German language or Humanities modules. German language abilities foster students' intercultural awareness and enhance their employability in their host country. They are also beneficial for securing mandatory internships (between the 2nd and 3rd year) in German companies and academic institutions. Constructor University supports its students in acquiring basic as well as advanced German skills in the first year of the Constructor Track. Non-native speakers of German are encouraged to take 2 German modules (2.5 CP each), but are not obliged to do so. Native speakers and other students who are not taking advantage of this offering take alternative modules in Humanities in each of the first two semesters:

- Humanities Module: Introduction to Philosophical Ethics (me, 2.5 CP)
- Humanities Module: Introduction to the Philosophy of Science (me, 2.5 CP)
- Humanities Module: Introduction to Visual Culture (me, 2.5 CP)

3 Software, Data and Technology Undergraduate Program Regulations

3.1 Scope of these Regulations

The regulations in this handbook are valid for all students who entered the Software, Data and Technology undergraduate program at Constructor University in Fall 2026. In case of conflict between the regulations in this handbook and the general policies for Bachelor Studies, the latter apply (see https://constructor.university/student-life/student-services/university-policies).

In exceptional cases, certain necessary deviations from the regulations of this study handbook might occur during the course of study (e.g., change of the semester sequence, assessment type, or the teaching mode of courses).

Updates to Study Program Handbooks are based on the policies approved by the Academic Senate on substantial and nonsubstantial changes to study programs. Students are integrated in the decision-making process through their respective committee representatives. All students affected by the changes will be properly informed.

In general, Constructor University therefore reserves the right to change or modify the regulations of the program handbook also after its publication at any time and in its sole discretion.

3.2 Examination Concept

According to the Policies for Bachelor and Master studies, modules generally carry at least five ECTS. Each program ensures appropriate examination frequency and organization, justified in an examination concept and regularly reviewed with student involvement.

Constructor University's examination concept follows the principle of Constructive Alignment (Biggs 1996), ensuring that learning outcomes, activities, and assessments are consistently aligned: students learn what is intended, and assessments both measure and shape learning. Where one assessment cannot cover all Intended Learning Outcomes (ILOs) complementary forms could be used (e.g., written exams plus lab reports). Module descriptions map ILOs to assessments.

In specific contexts, such as asynchronous online modules or courses emphasizing student engagement, Module Achievements or other types of formative assessments may support competence-oriented assessment.

Student feedback, embedded in the Quality Assurance System (QAS), systematically monitors workload, competence orientation, and alignment of ILOs and assessments. Student surveys and feedback are regulated in the Policy for student surveys and evaluations.

3.3 Degree

Upon successful completion of the study program, students are awarded a Bachelor of Science degree in Software, Data and Technology.

3.4 Graduation Requirements

In order to graduate, students need to obtain 180 CP. In addition, the following graduation requirements apply:

Students need to complete all mandatory components of the program as indicated in the Study and Examination Plan in Chapter 5 of this handbook.

4 Schematic Study Plan for Software, Data and Technology

Figure 2 schematically shows the sequence and types of modules required for the study program. A more detailed description, including the assessment types, is given in the Study and Examination Plan in the following section.

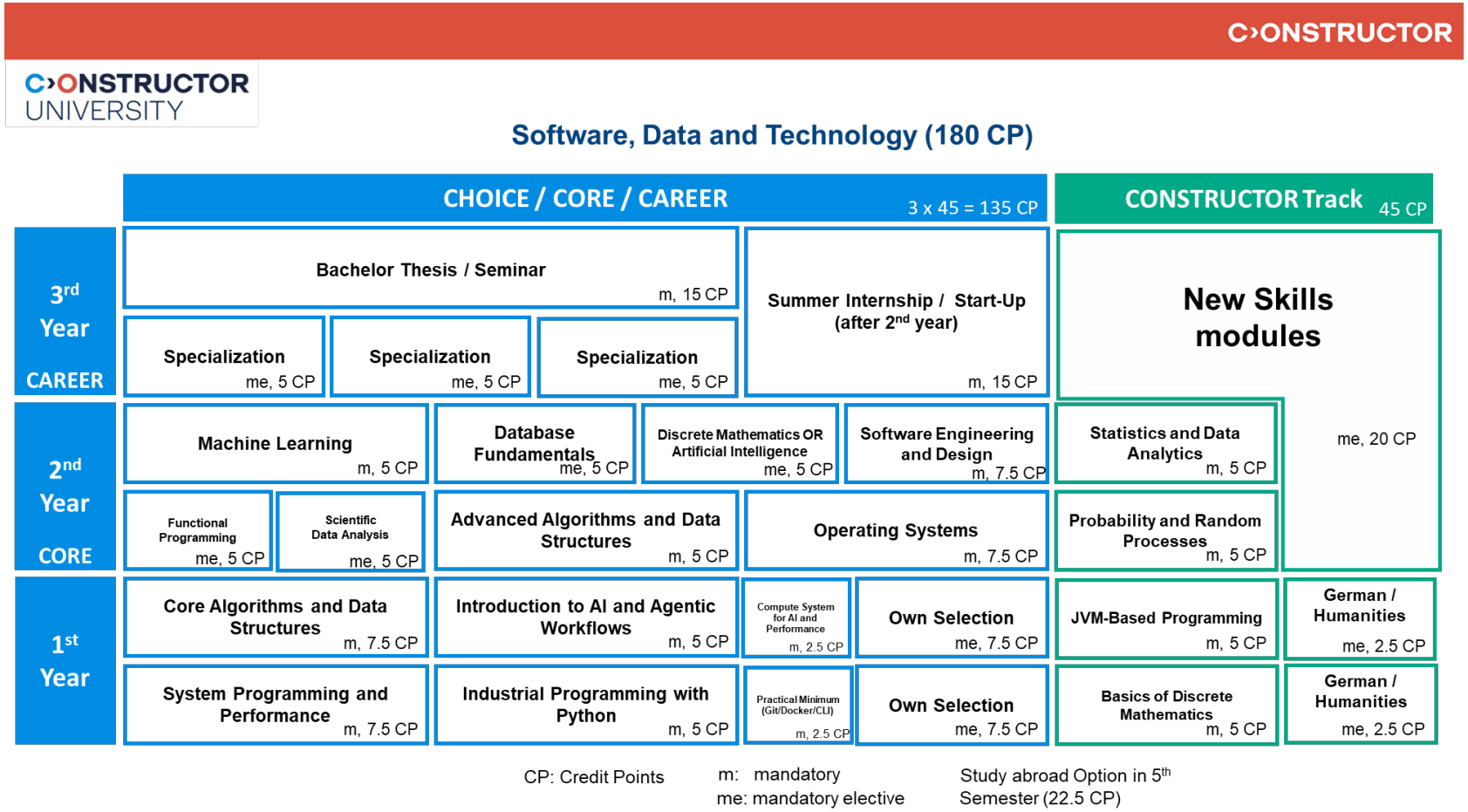

Figure 2: Schematic study plan for SDT

5 Study and Examination Plan

Software, Data, and Technology BSc

Matriculation Fall 2026

	Program-Specific Modules	Type	Assessment	Period	Status[1]	Sem.	ECTS
Year 1 - CHOICE							**45**
Take the mandatory CHOICE unit(s) listed below, this is a requirement for the SDT program.							
	Unit: Programming						**22.5**
SDT-106	**Module: System Programming and Performance**				**m**	**1**	**7,5**
SDT-106-A	System Programming and Performance	Lecture	Written Examination	Examination period			5
SDT-106-B	System Programming and Performance Lab	Lab	Program Code	During the semester			2,5
SDT-105	**Module: Industrial Programming with Python**				**m**	**1**	**5**
SDT-105-A	Industrial Programming with Python	Lecture	Written examination	Examination period			2.5
SDT-105-B	Industrial Programming with Python Lab	Lab	Program Code	During the semester			2.5
SDT-107	**Module: Practical Minimum (Git/Docker/CLI)**				**m**	**1**	**2,5**
SDT-107-A	Practical Minimum (Git/Docker/CLI)	Lab	Program Code	During the semester			
SDT-109	**Module: Introduction to AI and Agentic Workflows**				**m**	**2**	**5**
SDT-109-A	Intro to AI and Agentic Workflows	Lecture	Program Code	During the semester			2,5
SDT-109-B	Intro to AI and Agentic Workflows	Lab					2,5
SDT-110	**Module: Compute Systems for AI and Performance**				**m**	**2**	**2,5**
SDT-110-A	Compute Systems for AI and Performance	Lecture	Program Code	During the semester			
	Unit: Data Science						**7.5**
SDT-102	**Module: Core Algorithms & Data Structures**				**m**	**2**	**7.5**
SDT-102-A	Core Algorithms and Data Structures	Lecture	Written examination	Examination period			5
SDT-102-B	Core Algorithms and Data Structures Lab	Lab	Program Code	During the semester			2.5
	Unit: Further CHOICE modules				**me**		**15**
Students take two further CHOICE modules from those offered for all other study programs.							
It is recommended to take two of the following modules							
CH-150	**Module: Analysis**				**me**	**1**	**7.5**
CH-150-A	Analysis	Lecture	Written examination	Examination period			
CH-151	**Module: Linear Algebra**				**me**	**2**	**7.5**
CH-151-A	Linear Algebra	Lecture	Written examination	Examination period			
Year 2 - CORE							**45**
Take all units listed below							
	Unit: Software Development						**30**
CO-562	**Module: Operating Systems**				**m**	**3**	**7.5**
CO-562-A	Operating Systems	Lecture	Written examination	Examination period			
SDT-204	**Module: Software Engineering and Design**				**m**	**4**	**7.5**
SDT-204-A	Software Engineering and Design	Lecture	Written examination	Examination period			2.5
SDT-204-B	Software Engineering and Design Project	Project	Program Code	During the semester			5
SDT-205	**Module: Database Fundamentals**				**me**	**4**	**5**
SDT-205-A	Database Fundamentals	Lecture	Written examination	Examination period			2.5
SDT-205-B	Database Fundamentals Project	Project	Program Code	During the semester			2.5
SDT-202	**Module: Functional Programming**				**me**	**3**	**5**
SDT-202-A	Functional Programming	Lecture	Written examination	Examination period			2.5
SDT-202-B	Functional Programming Tutorial	Tutorial	Program Code	During the semester			2.5
CO-489	**Module: Scientific Data Analysis**				**me**	**3**	**5**
CO-489-A	Scientific Data Analysis	Lecture	Portfolio assessment	During the semester			
	Unit: Data Science				**m**		**15**
SDT-201	**Module: Advanced Algorithms and Data Structures**				**m**	**3**	**5**
SDT-201-A	Advanced Algorithms and Data Structures	Lecture	Written examination	Examination period			2.5
SDT-201-B	Advanced Algorithms and Data Structures Tutorial	Tutorial	Program Code	During the semester			2.5
CO-541	**Module: Machine Learning**				**m**	**4**	**5**
CO-541-A	Machine Learning	Lecture	Written examination	Examination period			
Take one of the two modules listed below							
CO-501	**Module: Discrete Mathematics**				**me**	**4**	**5**
CO-501-A	Discrete Mathematics	Lecture	Written examination	Examination period			
CO-547	**Module: Artificial Intelligence**				**me**	**4**	**5**
CO-547-A	Artificial Intelligence	Lecture	Written examination	Examination period			

	Construtor Track Modules (General Education)	Type	Assessment	Period	Status[1]	Sem.	ECTS
							15
	Unit: Methods						**10**
CTMS-MAT-26	**Module: Basics of Discrete Mathematics**				**m**	**1**	**5**
CTMS-26-A	Basics of Discrete Mathematics	Lecture	Written examination	Examination period			2.5
CTMS-26-B	Basics of Discrete Mathematics Tutorial	Lab					2.5
CTMS-SKI-27	**Module: JVM-Based Programming**				**m**	**2**	**5**
CTMS-27-A	JVM-Based Programming	Lecture	Written examination	Examination period			2.5
CTMS-27-B	JVM-Based Programming Lab	Lab	Program Code	During Semester			2.5
	Unit: German Language and Humanities (choose one module for each sememster)						**5**
CTLA-	**Module: Language 1**				**me**	**1**	**2.5**
CTLA-	Language 1	Seminar	Various	Various			
CTLA-	**Module: Language 2**				**me**	**2**	**2.5**
CTLA-	Language 2	Seminar	Various	Various			
CTHU-HUM-001	**Humanities Module: Introduction into Philosophical Ethics**				**me**	**2**	**2.5**
CTHU-001	Introduction into Philosophical Ethics	Lecture (online)	Written examination	Examination period			
CTHU-HUM-002	**Humanities Module: Introduction to the Philosophy of Science**				**me**	**1**	**2.5**
CTHU-002	Introduction to the Philosophy of Science	Lecture (online)	Written examination	Examination period			
CTHU-HUM-003	**Introduction to Visual Culture**				**me**	**2**	**2.5**
CTHU-003	Introduction to Visual Culture	Lecture (online)	Written examination	Examination period			
							15
	Unit: Methods					**3+4**	**10**
CTMS-MAT-12	**Module: Probability and Random Processes**				**m**	**3**	**5**
CTMS-12	Probability and Random Processes	Lecture	Written examination	Examination period			
CTMS-MET-21	**Module: Statistics and Data Analytics**				**m**	**4**	**5**
CTMS-21	Statistics and Data Analytics	Lecture	Written examination	Examination period			
	Unit: New Skills						**5**
Students must take a total 20 CP in New Skills modules in year 2 and 3							
CTNS-NSK-01	**Module: Logic (perspective I)***				**me**	**3/5**	**2.5**
CTNS-01	Logic (perspective I)	Lecture (online)	Written examination	Examination period			
CTNS-NSK-02	**Module: Logic (perspective II)***				**me**	**3/5**	**2.5**
CTNS-02	Logic (perspective II)	Lecture (online)	Written examination	Examination period			
CTNS-NSK-03	**Module: Causation and Correlation (perspective I)***				**me**	**4/6**	**2.5**
CTNS-03	Causation and Correlation (perspective I)	Lecture (online)	Written examination	Examination period			
CTNS-NSK-04	**Module: Causation and Correlation (perspective II)***				**me**	**4/6**	**2.5**
CTNS-04	Causation and Correlation (perspective II)	Lecture (online)	Written examination	Examination period			

*These modules are offered with two different perspectives, students may only select one

Year 3 - CAREER								45
CA-INT-900	**Module: Summer Internship / Startup and Career Skills**				**m**	**4/5**	**15**	
CA-INT-900-0	Internship / Startup and Career Skills	Internship	Report or Business plan	During the 5th semester				
SDT-400	**Module: Bachelor Thesis and Seminar SDT**				**m**	**6**	**15**	
SDT-400-T	Thesis SDT	Thesis	Thesis	15th of May			12	
SDT-400-S	Thesis Seminar SDT	Seminar	Presentation	During the semester			3	
	Unit: Specialization (take a total 15 ECTS of specialization modules)						**15**	
	Subunit: Machine Learning							
MCSSE-AI-01	**Module: Deep Learning**				**me**	**5**	**5**	
MCSSE-AI-01	Deep Learning	Lecture	Written examination	Examination period				
CA-S-MMDA-803	**Module: Stochastic Modeling and Financial Mathematics**				**me**	**6**	**5**	
CA-MMDA-803	Stochastic Modeling and Financial Mathematics	Lecture	Portfolio Assessment	During the semester				
SDT-301	**Module: Optimization Methods**				**me**	**5**	**5**	
SDT-301-A	Optimization Methods	Lecture	Written examination	Examination period			2.5	
SDT-301-B	Optimization Methods Tutorial	Tutorial	Program Code	During the semester			2.5	
SDT-305	**Module: Natural Language Processing**				**me**	**6**	**5**	
SDT-305-A	Natural Language Processing	Lecture	Written examination	Examination period				
	Subunit: Software Development							
SDT-302	**Module: Databases Internals**				**me**	**5**	**5**	
SDT-302-A	Databases Internals	Lecture	Written examination	Examination period				
SDT-306	**Module: Integrated Development and IT Operations**				**me**	**6**	**5**	
SDT-306-A	Integrated Development and IT Operations	Lecture	Written examination	Examination period				
SDT-303	**Module: Parallel Programming**				**me**	**5**	**5**	
SDT-303-A	Parallel Programming	Lecture	Written examination	Examination period				
CA-S-CS-803	**Module: Distributed Algorithms**				**me**	**6**	**5**	
CA-CS-803	Distributed Algorithms	Lecture	Written examination	Examination period				
CO-564	**Module: Computer Networks**				**me**	**5**	**5**	
CO-564-A	Computer Networks	Lecture	Written examination	Examination period				
	Subunit Programming Languages							
SDT-304	**Module: Formal Languages and Parsers**				**me**	**5**	**5**	
SDT-304-A	Formal Languages and Parsers	Lecture	Written examination	Examination period			2.5	
SDT-304-B	Formal Languages and Parsers Tutorial	Tutorial	Program Code	During the semester			2.5	
SDT-307	**Module: Compilers**				**me**	**6**	**5**	
SDT-307-A	Compilers	Lecture	Written examination	Examination period			2.5	
SDT-307-B	Compilers Project	Project	Program Code	During the semester			2.5	
SDT-308	**Module: Semantics of Programming Languages**				**me**	**6**	**5**	
SDT-308-A	Semantics of Programming Languages	Lecture	Written examination	Examination period			2.5	
SDT-308-B	Semantics of Programming Languages Tutorial	Tutorial	Program Code	During the semester			2.5	
SDT-309	**Module: Advanced Discrete Mathematics**				**me**	**5**	**5**	
SDT-309-A	Advanced Discrete Mathematics	Lecture	Written examination	Examination period				
Total ECTS								

[1] Status (m = mandatory, me = mandatory elective)

[2] For a full listing of all CHOICE / CORE / CAREER / Constructor Track modules please consult the **CampusNet** online catalogue and /or the study program handbooks.

[3] German native speakers will have alternatives to the language courses (in the field of Humanities). Humanities I and II are optional to all students, except for German native speakers.

								15
	Unit: New Skills							**15**
Students must take a total 20 CP in New Skills modules in year 2 and 3								
CTNS-NSK-05	**Module: Models, Matrices and Theories of Complex Systems**				**me**	**3/5**	**5**	
CTNS-05	Models, Matrices and Theories of Complex Systems	Online Lecture	Written examination	Examination period				
CTNS-NSK-06	**Module: Complex Problem Solving**				**me**	**3/5**	**5**	
CTNS-06	Complex Problem Solving	Online Lecture	Written examination	Examination period				
CTNS-NSK-07	**Module: Argumentation, Data Visualization and Communication (perspective I)***				**me**	**3/5**	**5**	
CTNS-07	Argumentation, Data Visualization and Communication (perspective I)	Online Lecture	Written examination	Examination period				
CTNS-NSK-08	**Module: Argumentation, Data Visualization and Communication (perspective II)***				**me**	**4/6**	**5**	
CTNS-08	Argumentation, Data Visualization and Communication (perspective II)	Online Lecture	Written examination	Examination period				
CTNS-NSK-09	**Module: Agency, Leadership, and Accountability**				**me**	**4/6**	**5**	
CTNS-09	Agency, Leadership, and Accountability	Online Lecture	Written examination	Examination period				
CTNS-CIP-10	**Module: Community Impact Project**				**me**	**3/5**	**5**	
CTNS-10	Community Impact Project	Project	Project	Examination period				

*These modules are offered with two different perspectives, students may only select one

	180

Figure 3: Study Examination plan for SDT

6 Software, Data and Technology Modules

6.1.1 System Programming and Performance

Module Name	System Programming and Performance
Module Code	SDT-106
Module ECTS	7.5
Program Owner	SDT-BSc (Software, Data and Technology)
Module Coordinator	Prof. Dr. Alexander Omelchenko

Study Semester		
Program	**Semester**	**Status**
SDT-BSc Software, Data and Technology	1	Mandatory

Student Workload	
Lecture	17.5
Laboratory	35
Independent Study	115
Exam Preparation	20
Total Hours	**187.5**

Module Components	**Number**	**Type**	**CP**
System Programming and Performance	SDT-106-A	Lecture	2.5
System Programming and Performance Lab	SDT-106-B	Laboratory	5

Module Description

This module introduces system programming on modern Unix-like platforms with a strong emphasis on performance engineering and evidence-based optimization. Students learn how programs are translated into executable binaries, how they interact with the operating system and hardware, and how runtime costs arise from memory access, concurrency, and I/O.

A systems programming language (primarily C/C++ on Linux) is used as a minimal vehicle to expose low-level mechanisms such as memory management, file I/O, and threading. The focus is not on mastering a specific language, but on understanding system-level concepts, professional tooling, and engineering practices that enable students to write correct, efficient, and debuggable software.

The module develops performance literacy: students learn to measure, profile, and reason about efficiency using data rather than intuition. These skills form a foundational competence for later courses in Operating Systems, Databases (systems perspective), AI infrastructure, and performance-critical software engineering.

Topics:

- Build toolchains and compilation pipeline: preprocessing, compilation, linking, debug symbols, optimization levels, and reproducible builds (CMake).

- Systems debugging practices: error handling, assertions, logging, core dumps, and debugging with gdb/lldb.

- Memory model in practice: stack vs. heap, pointers, ownership and lifetime, alignment, and common sources of undefined behavior.

- Dynamic memory management: allocation patterns, leaks, fragmentation, and detecting errors using sanitizers (ASan, UBSan).

- File I/O and OS interfaces: files and directories, streams vs. file descriptors, buffering strategies, and memory-mapped files.

- Concurrency fundamentals: threads, synchronization primitives, race conditions, deadlocks, and basic concurrent design patterns.

- Performance measurement: benchmarking methodology, profiling tools (sampling vs. instrumentation), flame graphs, and attribution of costs.

- Data-oriented performance: cache locality, data layout choices, allocation reduction, false sharing, and basic SIMD/vectorization awareness.

- Systems integration mindset: composing components, managing dependencies, and interfacing with higher-level software layers.

Rationale for assessment: The written examination assesses conceptual understanding of system-level principles, while the Program Code component assesses practical implementation, debugging, profiling, and performance engineering skills.

Recommended Knowledge

- Proficiency with basic programming constructs (variables, control flow, functions, and basic data structures) is required.

- Prior experience with any modern programming language is sufficient; the module does not teach programming syntax from scratch.

- Install a Linux environment (native Linux, dual boot, or WSL2).

- Install a native toolchain (gcc or clang), a debugger (gdb or lldb), and a build system (CMake).

- Install a modern IDE (JetBrains CLion recommended) and verify that basic debugging and profiling tools run locally.

Usability and Relationship to other Modules

This module establishes the system-level and performance baseline for subsequent modules in the SDT program. It directly prepares students for Operating Systems, Compute Systems for AI &

Performance, Database Fundamentals (systems perspective), and project-intensive software engineering and AI modules.

The skills acquired—debugging, profiling, memory reasoning, and performance analysis—are reused throughout the curriculum and are essential for building reliable and efficient systems, including AI-enabled developer tools and infrastructure.

Intended Learning Outcomes

No	Competence	ILO
1	Explain	Explain the native build pipeline (compilation and linking) and produce reproducible builds with appropriate debug and optimization settings.
2	Write	Write and debug system-level programs that manage memory manually and handle errors robustly.
3	Use	Use essential operating system interfaces for files, processes, and threads, and reason about their performance implications.
4	Detect	Detect and fix memory safety issues using professional tooling (debuggers and sanitizers) and explain their root causes.
5	Design	Design and implement simple concurrent programs, identify data races and deadlocks, and apply basic synchronization correctly.
6	Measure	Measure and profile program performance using standard profiling tools and identify performance bottlenecks.
7	Apply	Apply core performance optimization techniques (data layout, allocation reduction, cache-friendly access) while preserving correctness.
8	Communicate	Communicate performance results clearly in concise technical reports, including experimental setup, measurements, and trade-offs.

Indicative Literature

- Agner Fog — Optimizing Software in C++ (online resources)
- Brendan Gregg — Systems Performance (2nd ed.)
- Michael Kerrisk — The Linux Programming Interface
- Official documentation: Linux man pages, gcc/clang, gdb/lldb, and CMake documentation
- Randal E. Bryant, David R. O'Hallaron — Computer Systems: A Programmer's Perspective (3rd ed.)
- Ulrich Drepper — What Every Programmer Should Know About Memory

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
System Programming and Performance	Written Examination	60 minutes	33	45%	All theoretic ILOs
System Programming and Performance Lab	Program Code		67	45%	All Practical ILOs

Module Completion Requirements

- **Module Component Examination:** minimum pass 45% of each assessment.

Module Achievement

None.

6.1.2 Core Algorithms and Data Structures

Module Name	Core Algorithms and Data Structures
Module Code	SDT-102
Module ECTS	7.5
Program Owner	SDT-BSc (Software, Data and Technology)
Module Coordinator	Dr. Kinga Lipskoch

Study Semester		
Program	**Semester**	**Status**
SDT-BSc Software, Data and Technology	2	Mandatory

Student Workload	
Exam Preparation	20
Independent Study	115
Lecture	35
Tutorial	17.5
Total Hours	**187.5**

Module Components	**Number**	**Type**	**CP**
Core Algorithms and Data Structures	SDT-102-A	Lecture	5
Core Algorithms and Data Structures - Lab	SDT-102-B	Laboratory	2.5

Module Description

Algorithms and data structures are the foundation of computer science and are crucial for the design and implementation of efficient software programs. In this module, students will learn about fundamental algorithms for solving problems and about data structures for storing, accessing, and modifying data in an efficient manner. They will also learn techniques for analyzing the computational and memory complexities of algorithms and data structures. These concepts and techniques form the basis for almost all computer programs and are essential for success in the fields of software, data and technology.

Content:

- Introduction (asymptotic analysis of algorithms, analysis of recurrence relations, sums and integrals, time complexity, non-asymptotic optimizations, cache)

- Basic data structures (array, list, stack, queue, vector, hash tables, binary heap, heapsort, etc.)

- Sorting algorithms and heaps (quadratic sorting, stable sorting, mergesort, etc.)

- Graphs: depth-first search (DFS) and breadth-first search (BFS) algorithms.

- Graphs: matchings, colorings, flows, cuts.

- Graphs: shortest paths

- Introduction to Complexity Theory, Probabilistic Algorithms

- Numerical and Algebraic Algorithms

Recommended Knowledge

Students should have some knowledge of Industrial Programming with Python and Basics of Discrete Mathematics.

Usability and Relationship to other Modules

This module will provide students with a solid foundation for understanding how to design and analyze algorithms for solving problems, as well as data structures for efficiently storing and manipulating data.

Intended Learning Outcomes

No	Competence	ILO
1	Analyze	Analyze the time and space complexity of algorithms and optimize them using asymptotic analysis and non-asymptotic techniques such as cache optimization.
2	Implement	Implement and evaluate various data structures including arrays, lists, stacks, queues, vectors, hash tables, binary heaps, and heapsort.
3	Compare	Compare and contrast different sorting algorithms, including quadratic sorting, stable sorting, and mergesort, and understand the trade-offs involved in their use.
4	Implement	Implement depth-first search (DFS) and breadth-first search (BFS) algorithms and understand their applications in graph theory.
5	Analyze	Analyze matchings, colorings, flows, and cuts in graphs, and understand the algorithms and mathematical foundations used to solve these problems.
6	Implement	Implement shortest path algorithms in graphs and understand their applications in network design and routing.
7	Understand	Understand the fundamental concepts of complexity theory and probabilistic algorithms, and apply them in solving computational problems.
8	Analyze	Analyze and implement numerical and algebraic algorithms and understand their applications in a variety of fields.
9	Develop	Develop the ability to analyze, design, and implement algorithms for solving real-world problems and understand the trade-offs involved in their use.

Indicative Literature

- David E. Goldberg: Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, 1989.
- Donald E. Knuth: The Art of Computer Programming: Fundamental Algorithms, volume 1, 3rd edition, Addison Wesley Longman Publishing, 1997.
- Jon Kleinberg and Éva Tardos: Algorithm Design, 1st edition, Pearson, 2005.
- Michael T. Goodrich, Roberto Tamassia, and Michael H. Goldwasser: Data Structures and Algorithms in Python, John Wiley & Sons, 2013.
- Robert Sedgewick and Kevin Wayne: Algorithms, 4th edition, Addison-Wesley, 2011.
- Steven Skiena: The Algorithm Design Manual, 2nd edition, Springer, 2008.
- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein: Introduction to Algorithms, 3rd edition, MIT Press, 2009.

Entry Requirements

Prerequisites	SDT-106 System Programming and Performance
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Core Algorithms and Data Structures	Written Examination	120 Minutes	67	45%	All theoretical ILOs of the module
Core Algorithms and Data Structures - Lab	Program Code		33	45%	All practical ILOs of the module.

Module Completion Requirements

- **Module Component Examination:** minimum pass 45% of each assessment.

Module Achievement

None.

6.1.3 Industrial Programming with Python

Module Name	Industrial Programming with Python
Module Code	SDT-105
Module ECTS	5
Program Owner	SDT-BSc (Software, Data and Technology)
Module Coordinator	Prof. Dr. Alexander Omelchenko

Study Semester		
Program	**Semester**	**Status**
SDT-BSc Software, Data and Technology	1	Mandatory

Student Workload	
Lecture	17.5
Laboratory	17.5
Independent Study	75
Exam Preparation	15
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Industrial Programming with Python	SDT-105-A	Lecture	2.5
Industrial Programming with Python Lab	SDT-105-B	Laboratory	2.5

Module Description

This module builds professional, production-oriented Python skills for building small but robust components-data ingestion, automation, and API integration-that can later serve as building blocks for larger software and AI systems. The focus is on reproducible environments, testing, type-aware development, and clean module boundaries, so projects remain maintainable as they grow.

Topics:

- Python refresher for experienced programmers: idioms, modules/packages, and the standard library (as needed).

- Project structure and environments: venv, dependency pinning, lock files, and reproducible runs.

- Modern packaging: pyproject.toml, build wheels, versioning, and publishing/consuming internal packages.

- Code quality automation: formatting and linting, type checking (mypy/pyright), and pre-commit hooks.

- Testing with pytest: test design, fixtures, parametrization, mocking, and coverage.

- Continuous Integration: automated lint/test pipelines (e.g., GitHub Actions) and basic release workflows.

- Working with data: numpy arrays and pandas dataframes; reading/writing CSV/JSON/Parquet; basic validation.

- Interacting with web APIs: HTTP basics, authentication, rate limits, robust clients, and error handling.

- Glue code patterns: configuration management, logging, retries, and composing components into pipelines.

- Team workflows: code review basics, repository hygiene, and collaboration in Git-based projects.

Rationale for assessment: The written examination assesses conceptual understanding of professional Python development practices, while the Program Code component assesses practical implementation, testing, packaging, and integration skills.

Recommended Knowledge

- Proficiency with basic programming constructs is required. Prior exposure to Python (or another modern language) is strongly recommended; the module focuses on professional tooling and maintainable project structure rather than teaching syntax from scratch.

- Configure a GitHub account and enable a student pack.

- Install the latest version of Python, Git, IDE/editor.

Usability and Relationship to other Modules

This module is an alternative to SDT-104 “Scientific Programming with Python” with a greater focus on software craftsmanship and deployment rather than numerical computing. It provides essential groundwork for advanced topics such as Backend development, DevOps, and cloud computing.

Intended Learning Outcomes

No	Competence	ILO
1	Demonstrate	Demonstrate knowledge of the Python language and its most common standard libraries.
2	Describe	Describe the principles of clean code and common design patterns and articulate how they contribute to code readability and maintainability.
3	Understand	Understand the purpose and best practices of unit testing, and how these tests contribute to software reliability.
4	Set up	Set up and maintain a professional Python development environment with automated type-checking and style enforcement.
5	Implement	Implement clean, idiomatic, and maintainable Python code using modern external libraries.
6	Develop	Develop unit and integration test suites using pytest and integrate them into an automated continuous integration workflow.
7	Use	Use numpy and pandas to load, transform, validate, and inspect structured data for small analytics or preprocessing tasks.

8	Work	Work effectively in team-based development settings, using Git-based workflows, version control, and collaborative development practices.
9	Publish	Package, version, and publish Python libraries or applications, making them suitable for reuse in subsequent modules and team projects.

Indicative Literature

- Brett Slatkin - Effective Python
- Brian Okken — Python Testing with pytest
- Official documentation: pandas, numpy, and the Python standard library
- Python Packaging User Guide + relevant PEPs (PEP 518/517, PEP 440, PEP 621)

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Industrial Programming with Python	Written Examination	60 minutes	50	45%	1-4
Industrial Programming with Python Lab	Program Code		50	45%	5-9

Module Completion Requirements

- **Module Component Examination:** minimum pass 45% of each assessment.

Module Achievement

≥ 50 % of weekly assignments must be passed. Late/missed work may be compensated by extra tasks during the semester or an extra task in January.

6.1.4 Introduction to AI and Agentic Workflows

Module Name	Introduction to AI and Agentic Workflows
Module Code	SDT-109
Module ECTS	5
Program Owner	SDT-BSc (Software, Data and Technology)
Module Coordinator	Prof. Dr. Alexander Omelchenko

Study Semester		
Program	**Semester**	**Status**
SDT-BSc Software, Data and Technology	2	Mandatory

Student Workload	
Lecture	17.5
Laboratory	17.5
Independent Study	75
Exam Preparation	15
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Introduction to AI and Agentic Workflows	SDT-109-A	Lecture	2.5
Introduction to AI and Agentic Workflows	SDT-109-B	Laboratory	2.5

Module Description

This module introduces modern AI development with a focus on Large Language Models (LLMs) and agentic workflows. Students learn how to design and implement AI-enabled software systems that combine model reasoning with reliable tool use, retrieval from external knowledge sources, and integration into real-world developer workflows.

Topics:

- fundamentals of Large Language Models: tokens, embeddings, inference, context windows, and limitations

- agentic workflows: separation of planning and execution, tool calling, control flow, and orchestration patterns

- tool integration: structured tool interfaces, external APIs, and reliable interaction with software systems

- retrieval-augmented generation (RAG): indexing, retrieval strategies, ranking, and trade-offs

- state and memory in AI systems: short-term context, external memory, and persistence strategies

- evaluation of AI systems: automated tests, benchmarks, regression testing, and structured human evaluation

- observability and debugging: logging, tracing, prompt and program analysis, and failure mode inspection

- reproducibility and experimentation: configuration management, versioned prompts, and lightweight experiment tracking

- safety and security considerations: prompt injection, data privacy, output constraints, and misuse mitigation

- IDE-centric development workflows: iterative development, testing, and refactoring of AI-enabled systems

- optional implementation tracks: Python-based systems and selected Kotlin/JVM components for tooling-oriented projects

Recommended Knowledge

- Students are expected to be comfortable with basic programming concepts and Git-based workflows.

- Prior completion of Industrial Programming with Python and Practical Minimum (or equivalent experience) is recommended.

- Basic knowledge of linear algebra and discrete mathematics is helpful; linear algebra may be taken in parallel.

Usability and Relationship to other Modules

This module provides early, practice-oriented foundations for later modules in Machine Learning, Statistical Learning, Software Engineering and Design, Databases, and Operating Systems.

The focus on reproducibility, evaluation, and system integration supports project work across the SDT curriculum and aligns with modern AI-assisted software development workflows, particularly in IDE-centric and tooling-oriented environments.

Intended Learning Outcomes

No	Competence	ILO
1	Explain	Explain key concepts of LLM-based AI systems, including tokens, embeddings, inference, and fundamental limitations.
2	Design	Design agentic workflows that combine model reasoning with tool use, retrieval, and external services.
3	Implement	Implement AI-enabled applications that integrate structured tool calls and external APIs.
4	Implement	Implement a retrieval-augmented generation (RAG) pipeline and reason about its design trade-offs.
5	Evaluate	Evaluate AI system quality using automated tests, metrics, and structured human evaluation, and identify common failure modes.

6	Debug	Debug and improve agent behavior using logs, traces, prompt/program analysis, and controlled experiments.
7	Apply	Apply reproducibility practices such as configuration management, versioned prompts, and lightweight experiment tracking.
8	Apply	Apply safety and security measures including prompt injection mitigation, privacy-aware data handling, and output constraints.
9	Use	Use modern IDE-centric workflows and tooling to develop, test, and iterate on AI-enabled software systems.
10	Communicate	Communicate design decisions, limitations, and evaluation results clearly in technical documentation and code reviews.

Indicative Literature

- Chip Huyen — Designing Machine Learning Systems (selected chapters on evaluation and deployment)
- Dan Jurafsky, James H. Martin — Speech and Language Processing (selected chapters)
- Lewis Tunstall, Leandro von Werra, Thomas Wolf — Natural Language Processing with Transformers (selected chapters)
- Patrick Lewis et al. — Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks
- Selected recent research papers and technical documentation provided during the course
- Stuart Russell, Peter Norvig — Artificial Intelligence: A Modern Approach (selected chapters)

Entry Requirements

Prerequisites	SDT-105 Industrial Programming with Python SDT-107 Practical Minimum (Git/Docker/CLI)
Co-requisites	None
Additional Remarks	Industrial Programming with Python (or equivalent programming proficiency)

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Introduction to AI and Agentic Workflows	Program Code		100	45%	All ILOs
Introduction to AI and Agentic Workflows					

Module Completion Requirements

- **Module Examination:** minimum for pass- 45%.

Module Achievement

None.

6.1.5 Practical Minimum (Git/Docker/CLI)

Module Name	Practical Minimum (Git/Docker/CLI)
Module Code	SDT-107
Module ECTS	2.5
Program Owner	SDT-BSc (Software, Data and Technology)
Module Coordinator	Prof. Dr. Alexander Omelchenko

Study Semester		
Program	**Semester**	**Status**
SDT-BSc Software, Data and Technology	1	Mandatory

Student Workload	
Laboratory	17.5
Independent Study	40
Exam Preparation	5
Total Hours	**62.5**

Module Components	**Number**	**Type**	**CP**
Practical Minimum (Git/Docker/CLI)	SDT-107-A	Laboratory	2.5

Module Description

This module establishes a shared practical baseline for modern software engineering within the SDT program. It introduces the essential tooling and workflows required throughout the curriculum, including version control, command-line usage, containerized development environments, and basic automation.

The focus is not on individual tools in isolation, but on forming sustainable engineering habits that enable reproducible development, effective collaboration, and smooth onboarding into more advanced project-based modules such as Industrial Programming with Python, System Programming & Performance, and AI/agentic workflows.

The module is designed as a hands-on bootcamp and assumes no prior professional tooling experience.

Topics:

- Command-line fundamentals: filesystem navigation, processes, environment variables, pipes, and redirection

- Shell productivity: basic scripting and automation of recurring tasks

- Git fundamentals: repositories, commits, branches, merges, and conflict resolution

- Collaborative workflows: pull requests, basic code reviews, repository hygiene (.gitignore, README)

- Working with remotes: SSH authentication, tags, releases, and simple branching strategies

- Docker fundamentals: images vs. containers, running and inspecting containers, volumes and networking

- Building images: Dockerfile basics and reproducible builds

- Docker Compose: defining and running multi-service local development environments

- Reproducible setups: configuration files, environment variables, and deterministic project initialization

- Mini CI bootcamp: running automated checks in a simple CI workflow (e.g., GitHub Actions)

Hands-on sessions are conducted as guided workshops with short practical check-offs.

Weekly tasks culminate in a final practical setup demonstrating a reproducible, containerized workflow with version control and basic CI automation.

This module is designed as a compact, practice-oriented unit establishing foundational technical skills early in the program. Its reduced credit size allows tight integration with larger 5 CP modules while maintaining overall workload balance within the study program.

Recommended Knowledge

- No formal prerequisites are required.

- Students are expected to be able to install software and follow technical instructions.

- Create a GitHub account and configure SSH keys

- Install Git and Docker (Desktop or Engine)

- Bring a laptop with administrator rights

- Install a JetBrains IDE or comparable development environment

Usability and Relationship to other Modules

This module serves as a foundational prerequisite for all project-oriented and systems-oriented modules in the SDT program. It ensures that students share a common tooling baseline and can work in reproducible environments, collaborate via Git-based workflows, and submit assignments in a consistent technical format.

The skills acquired in this module are directly reused in Industrial Programming with Python, System Programming & Performance, AI Tooling & Workflow Optimization, and subsequent team projects.

Intended Learning Outcomes

No	Competence	ILO
1	Use	Use a command-line environment to navigate the file system, manage processes, and execute development tools effectively.

2	Use	Use shell features such as pipes, redirection, and environment variables to automate simple development tasks.
3	Use	Use Git for local version control, including creating commits, branching, merging, and resolving conflicts.
4	Work	Work effectively in collaborative software development settings using GitHub-based workflows, including pull requests and basic code review practices.
5	Use	Use Docker to run, inspect, and troubleshoot containerized development environments.
6	Create	Create Dockerfiles and simple Docker Compose configurations to define reproducible development environments.
7	Set up	Set up a minimal continuous integration workflow that runs automated checks on code changes.

Indicative Literature

- GitHub Docs: GitHub workflows and GitHub Actions
- Official Docker documentation (docs.docker.com)
- Scott Chacon, Ben Straub — Pro Git (free online edition)
- William Shotts — The Linux Command Line

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Practical Minimum (Git/Docker/CLI)	Program Code		100	45%	1-7

Module Completion Requirements

- **Module Examination:** minimum for pass- 45%.

Module Achievement

None.

6.1.6 Compute System for AI and Performance

Module Name	Compute System for AI and Performance
Module Code	SDT-110
Module ECTS	2.5
Program Owner	SDT-BSc (Software, Data and Technology)
Module Coordinator	Prof. Dr. Alexander Omelchenko

Study Semester		
Program	**Semester**	**Status**
SDT-BSc Software, Data and Technology	2	Mandatory

Student Workload	
Lecture	17.5
Independent Study	40
Exam Preparation	5
Total Hours	**62.5**

Module Components	**Number**	**Type**	**CP**
Compute Systems for AI and Performance	SDT-110-A	Lecture	2.5

Module Description

This module introduces the hardware-aware performance foundations required for modern AI-oriented software systems. Students learn how to reason about the cost of computation and data movement on contemporary CPU and GPU platforms, how to identify real performance bottlenecks using measurement and profiling, and how to make evidence-based optimization decisions.

The emphasis is on practical mental models rather than digital logic or circuit design. Core concepts include memory hierarchy and bandwidth limitations, parallel execution, and accelerator characteristics that dominate the runtime behavior of AI workloads.

Topics:

- modern compute platforms: CPU cores, caches, main memory, SIMD units, and GPU execution units

- memory hierarchy and data movement: latency, bandwidth, and their impact on performance

- compute-bound vs. memory-bound workloads and practical bottleneck identification

- basic performance models: Amdahl's law and roofline-style reasoning

- profiling and measurement: benchmarking methodology, sampling vs. instrumentation, flame graphs

- data-oriented performance techniques: cache-friendly data layouts, reduced allocations, and vectorization-aware loops

- parallel execution concepts: threads, basic parallel patterns, and associated overheads

- conceptual GPU execution model: SIMT execution, occupancy, and GPU memory hierarchy

- designing reproducible performance experiments and interpreting results critically

- performance reporting: documenting experimental setup, measurements, results, and trade-offs

This module is designed as a focused, practice-oriented unit introducing hardware-aware performance reasoning early in the curriculum. Its compact size allows integration with larger systems-oriented modules while maintaining overall workload balance within the study program.

Recommended Knowledge

- Students are expected to be comfortable with basic programming and command-line workflows.

- Completion of System Programming & Performance (or equivalent experience with compilation, memory, and profiling fundamentals) is recommended. No prior GPU programming experience is required.

- Basic knowledge of linear algebra and discrete mathematics is helpful.

Usability and Relationship to other Modules

This module provides the hardware and performance perspective that complements System Programming & Performance and prepares students for Operating Systems, where resource management, concurrency costs, and memory behavior are studied in depth.

It also supports AI-focused modules by establishing a shared performance vocabulary and benchmarking methodology for data-intensive and parallel workloads. The concepts introduced here are reused in later project-based modules and performance-critical software development.

Intended Learning Outcomes

No	Competence	ILO
1	Explain	Explain the main components of modern compute platforms (CPU cores, caches, memory, SIMD units, GPU execution units) and how they influence program performance.
2	Distinguish	Distinguish between compute-bound and memory-bound workloads and identify the dominant bottleneck using simple models and measurements.
3	Apply	Apply basic performance models (e.g., Amdahl's law and roofline-style reasoning) to assess expected speedups and scaling limits.
4	Use	Use profiling and measurement tools to identify hotspots and attribute performance costs to concrete code paths.
5	Implement	Implement and justify targeted optimizations such as cache-friendly data layouts, reduced allocations, vectorization-aware loops, and basic parallel patterns while preserving correctness.
6	Explain	Explain key GPU concepts at a conceptual level (SIMT execution, occupancy, memory hierarchy) and relate them to observed performance behavior.

7	Report	Produce a concise performance report that documents experimental setup, methodology, results, and trade-offs in a reproducible manner.

Indicative Literature

- Brendan Gregg — Systems Performance (selected chapters)
- John L. Hennessy, David A. Patterson — Computer Architecture: A Quantitative Approach (selected chapters)
- Official documentation and guides: Linux perf, flame graphs, and selected GPU architecture overviews
- Randal E. Bryant, David R. O'Hallaron — Computer Systems: A Programmer's Perspective (selected chapters)
- Ulrich Drepper — What Every Programmer Should Know About Memory (selected sections)

Entry Requirements

Prerequisites	SDT-106 System Programming and Performance SDT-107 Practical Minimum (Git/Docker/CLI)
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Compute Systems for AI and Performance	Program Code		100	45%	All ILOs

Module Completion Requirements

- **Module Examination:** minimum for pass- 45%.

Module Achievement

None.

6.1.7 Analysis

Module Name	Analysis
Module Code	CH-150
Module ECTS	7.5
Program Owner	MMDA-BSc (Mathematics, Modeling, and Data Analytics)
Module Coordinator	Prof. Dr. Sören Petrat

Study Semester		
Program	**Semester**	**Status**
minor-Mathematics Minor in Mathematics	1	Mandatory
MMDA-BSc Mathematics, Modeling, and Data Analytics	1	Mandatory
SDT-BSc Software, Data and Technology	1	Mandatory Elective

Student Workload	
Lecture	35
Tutorial	17.5
Independent Study	135
Total Hours	**187.5**

Module Components	**Number**	**Type**	**CP**
Analysis	CH-150-A	Lecture	7.5

Module Description

This module introduces fundamental concepts and techniques in a concise and rigorous way. The class conveys the pleasure of doing mathematics, and motivates mathematics concepts from problems and concrete examples, but also shows the power of abstraction and of formal reasoning.

The following topics will be covered:

- Proof by induction, and elementary combinatorics

- Groups, equivalence relations, and quotients

- Natural numbers, integers, rationals, and real numbers

- Sequences and series, and convergence

- Functions of a single real variable, continuity, and the intermediate value theorem

- Metric spaces, and the continuous functions as a metric space

- Differentiation, mean value theorem, and the inverse mapping theorem in one variable

- Riemann integral

- Fundamental theorem of Calculus, and the integration by parts with applications

- Integral mean value theorem

- Change of variables

- Taylor series with integral and Lagrange remainders

- Elementary point-set topology (neighborhoods, open and closed sets, compactness, and Heine-Borel)

Usability and Relationship to other Modules

- This module is part of the core education in Mathematics, Modeling and Data Analytics.

- It is also valuable for students in Physics, Computer Science, RIS, and ECE, either as part of a minor in Mathematics, or as an elective module.

- The curriculum is integrated with the curriculum of the module "Matrix Algebra and Advanced Calculus" in the following way: "Matrix Algebra and Advanced Calculus" emphasizes the operational aspects, computational skills, and intuitive understanding, while Analysis builds rigorous foundations of the field, emphasizing proof, abstraction, and mathematical rigor.

Recommended Knowledge

- Good command of high-school mathematics, in particular pre-calculus topics

- Good command of high-school calculus helps, but is not a prerequisite

- It is recommended to co-enroll in the Methods module “Matrix Algebra & Advanced Calculus I”

- Revise your high school mathematics

- Read general interest expositions about mathematics and mathematicians

- Work on mathematics problems over the summer

- For a detailed set of preparation instruction, references, and links, see http://math.Constructor-university.de/undergraduate/prepare/index

Intended Learning Outcomes

No	Competence	ILO
1	Formulate	Formulate mathematical concepts and results discussed in class cleanly.
2	Outline	Outline proofs which have been given in the lectures.
3	Prove	Prove results independently which are direct consequences of those proved in the lectures.
4	Understand	Understand and use fundamental mathematical terminology to communicate mathematics at a university level.

Indicative Literature

- T. Tao (2016) Analysis third edition. New Delhi: Hindustan Book Agency.
- W. Rudin (1976) Principles of Mathematical Analysis third edition New York: McGraw-Hill.

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Analysis	Written Examination	120 Minutes	100	45%	1-4

Module Completion Requirements

- **Module Examination:** minimum for pass- 45%.

Module Achievement

None.

6.1.8 Linear Algebra

Module Name	Linear Algebra
Module Code	CH-151
Module ECTS	7.5
Program Owner	MMDA-BSc (Mathematics, Modeling, and Data Analytics)
Module Coordinator	Dr. Ivan Ovsyannikov

Study Semester		
Program	**Semester**	**Status**
minor-Mathematics Minor in Mathematics	2	Mandatory
MMDA-BSc Mathematics, Modeling, and Data Analytics	2	Mandatory
SDT-BSc Software, Data and Technology	2	Mandatory Elective

Student Workload	
Lecture	35
Tutorial	17.5
Independent Study	135
Total Hours	**187.5**

Module Components	**Number**	**Type**	**CP**
Linear Algebra	CH-151	Lecture	7.5

Module Description

This module continues the introduction to Linear Algebra from the methods module "Matrix Algebra and Advanced Calculus I". The fundamental concepts and techniques of Linear Algebra are introduced in a rigorous and more abstract way. The first half of this module covers vector spaces and linear maps, while the second half covers inner products and geometry.

The following topics will be covered:

- Vector spaces

- Linear Operators

- Dual spaces

- Isomorphisms

- Connection to matrices

- Sums and direct sums

- Fundamental spaces of a linear operator

- Diagonalization of linear operators (on finite dimensional spaces)

- Cayley-Hamilton theorem

- Jordan decomposition

- Jordan normal form and its applications to linear differential equations

- Decomplexification and complexification

- Bilinear Forms and their classification

- Quadratic forms and orthogonalization

- Euclidean and unitary spaces

- Orthogonal and unitary operators

- Self-adjoint operators

Recommended Knowledge

- Basic matrix algebra at the level achieved in “Matrix Algebra and Advanced Calculus I”

- Revise your matrix algebra.

- Unless prepared otherwise, take the Methods module "Matrix Algebra and Advanced Calculus" in the first semester.

Usability and Relationship to other Modules

- This module is part of the core education in Mathematics

- This module is valuable for students in Computer Science, RIS, and ECE, either as part of a minor in Mathematics, or as an elective module.

- The curriculum is integrated with the curriculum of the module "Matrix Algebra and Advanced Calculus I and II” in the following way: "Matrix Algebra and Advanced Calculus I and II" emphasizes the operational aspects, computational skills, and intuitive understanding, while Linear Algebra builds rigorous foundations of the field, emphasizing proof, abstraction, and mathematical rigor.

Intended Learning Outcomes

No	Competence	ILO
1	Describe	Describe the concept of a vector space and linear operator in an abstract way.
2	Explain	Explain the connection of abstract linear algebra in the context of matrix algebra.
3	Discuss	Discuss the proofs of the major theorems from class.
4	Illustrate	Illustrate the use of bilinear forms and their role in geometry.

5	Distinguish	Distinguish bilinear forms in the context of Euclidean, unitary and symplectic spaces.

Indicative Literature

- G. Strang (2016) Introduction to Linear Algebra Wellesley: Wellesley-Cambridge Press fifth edition.
- P.K. Kostrikin Yu Manin (1997) Linear Algebra and Geometry London: Gordon and Breach.
- S. Axler (2005) Linear Algebra Done Right third edition Berlin: Springer.
- S. Lang (1986) Introduction to Linear Algebra second edition Berlin: Springer.

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Linear Algebra	Written Examination	120 Minutes	100	45%	1-5

Module Completion Requirements

- **Module Examination:** minimum for pass- 45%.

Module Achievement

None.

6.1.9 Mathematical Foundations of Computer Science

Module Name	Mathematical Foundations of Computer Science
Module Code	CH-233
Module ECTS	7.5
Program Owner	CS-BSc (Computer Science)
Module Coordinator	Prof. Dr. Jürgen Schönwälder

Study Semester		
Program	**Semester**	**Status**
CS-BSc Computer Science	1	Mandatory
SDT-BSc Software, Data and Technology	1	Mandatory elective

Student Workload	
Class Attendance	35
Tutorial	17.5
Independent Study	115
Exam Preparation	20
Total Hours	**187.5**

Module Components	**Number**	**Type**	**CP**
Mathematical Foundations of Computer Science	CH-233-A	Lecture	5
Mathematical Foundations of Computer Science Tutorial	CH-233-B	Tutorial	2.5

Module Description

The module introduces students to the mathematical foundations of computer science. Students learn to reason logically and clearly. They acquire the skill to formalize arguments and to prove propositions mathematically using elementary logic. Students are also introduced to fundamental concepts of graph theory and elementary graph algorithms.

After establishing the concept of algorithms, the first part covers basic elements of discrete mathematics, leading to

Boolean algebra, propositional logic, and predicate logic. Students learn how to use fundamental proof techniques to prove (or disprove) simple propositions. The second part of the module introduces students to basic concepts of algebraic structures like groups, rings, and fields and different structure preserving maps (homomorphisms). Students study how these abstract concepts relate to problems in computer science. The last part of the module covers the basic elements of graph theory and the

different representation of graphs. Elementary graph algorithms are introduced that have a wide range of applicability in computer science.

Recommended Knowledge

It is recommended that students revise mathematical concepts from their high school education.

Usability and Relationship to other Modules

This module introduces key mathematical concepts and teaches students to work with mathematical abstractions that are relevant for computer science. The acquired skills are relevant for subsequent courses covering theoretical or abstract

aspects of computer science.

Intended Learning Outcomes

No	Competence	ILO
1	Explain	Explain basic concepts and properties of algorithms.
2	Understand	Understand the concept of termination and complexity metrics.
3	Illustrate	Illustrate basic concepts of discrete math (sets, relations, functions).
4	Use	Use basic proof techniques and apply them to prove properties of algorithms.
5	Summarize	Summarize basic principles of Boolean algebra and propositional logic.
6	Use	Use predicate logic and outline concepts such as validity and satisfiability.
7	Distinguish	Distinguish abstract algebraic structures such as groups, rings and fields.
8	Classify	Classify different structure preserving maps (homomorphisms).
9	Understand	Understand calculations in finite fields and their applicability to computer science.
10	Explain	Explain elementary concepts of graph theory and use different graph representations.
11	Outline	Outline basic graph algorithms (e.g., traversal, search, spanning trees).

Indicative Literature

- Eric Lehmann, F. Thomson Leighton, Albert R. Meyer: Mathematics for Computer Science, online 2018.
- Winfried K. Grassmann, Jean-Paul Tremblay: Logic and Discrete Mathematics: A Computer Science Perspective, Pearson, 1996.

Entry Requirements

Prerequisites	None

Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Mathematical Foundations of Computer Science	Written Examination	120 Minutes	100	45%	All
Mathematical Foundations of Computer Science Tutorial					

Module Completion Requirements

- **Module Examination:** minimum for pass- 45%.

Module Achievement

None.

6.1.10 Operating Systems

Module Name	Operating Systems
Module Code	CO-562
Module ECTS	7.5
Program Owner	CS-BSc (Computer Science)
Module Coordinator	Prof. Dr. Jürgen Schönwälder

Study Semester		
Program	**Semester**	**Status**
CS-BSc Computer Science	3	Mandatory
SDT-BSc Software, Data and Technology	3	Mandatory

Student Workload	
Class Attendance	52.5
Exam Preparation	20
Independent Study	115
Total Hours	**187.5**

Module Components	**Number**	**Type**	**CP**
Operating Systems	CO-562-A	Lecture	7.5

Module Description

This module introduces concepts and principles used by operating systems to provide programming abstractions that enable an efficient and robust execution of application programs. Students will gain an understanding of how an operating system kernel manages hardware components and how it provides abstractions such as processes, threads, virtual memory, file systems, and inter-process communication facilities. Students learn the principles of event-driven and concurrent programming and the mechanisms that are necessary to solve synchronization and coordination problems, thereby avoiding race conditions, deadlocks, and resource starvation. The Linux kernel and runtime system will be used throughout the course to illustrate how key ideas and concepts have been implemented and how application programs can use them.

Recommended Knowledge

Students are expected to understand data representation and program execution at the machine instruction level as covered in the module Introduction to Computer Science.

Students are expected to have a working Linux installation, which allows them to compile and run sample programs provided by the instructor and to implement their own solutions for homework assignments.

Usability and Relationship to other Modules

This module enables students to write programs that make efficient use of the services provided by the operating system kernel. This is particularly important for advanced modules on computer networks, robotics, and embedded systems.

Intended Learning Outcomes

No	Competence	ILO
1	Explain	Explain the differences between processes, threads, application programs, libraries, and operating system kernels.
2	Describe	Describe well-known mutual exclusion and coordination problems.
3	Use	Use semaphores to achieve mutual exclusion and solve coordination problems.
4	Use	Use mutual exclusion locks and condition variables to solve synchronization and coordination problems.
5	Illustrate	Illustrate how deadlocks can be avoided, detected, and resolved.
6	Summarize	Summarize the different mechanisms to realize virtual memory and their trade-offs.
7	Solve	Solve basic inter-process communication problems using signals and pipes.
8	Use	Use socket inter-process communication primitives.
9	Use	Multiplex I/O activities using suitable system calls and libraries.
10	Describe	Describe file system programming interfaces and the design of file systems at the operating system kernel level.
11	Explain	Explain how memory mapping can improve I/O performance.
12	Restate	Restate the functionality of a linker and the difference between static linking and dynamic linking.
13	Outline	Outline how different device types are supported by Unix-like kernels.
14	Discuss	Discuss virtualization mechanisms such as containers or virtual machines.

Indicative Literature

- Abraham Silberschatz, Peter B. Galvin, Greg Gagne: Applied Operating System Concepts, John Wiley, 2000.
- Andrew S. Tanenbaum, Herbert Bos: Modern Operating Systems, Prentice Hall, 4th edition, Pearson, 2015.
- Robert Love: Linux Kernel Development, 3rd edition, Addison Wesley, 2010.
- Robert Love: Linux System Programming: Talking Directly to the Kernel and C Library, 2nd edition, O'Reilly, 2013.
- William Stallings: Operating Systems: Internals and Design Principles, 8th edition, Pearson, 2014.

Entry Requirements

Prerequisites	SDT-102 Core Algorithms and Data Structures **OR** CH-231 Algorithms and Data structures CH-234 Digital Systems and Computer Architecture
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Operating Systems	Written Examination	120 Minutes	100	45%	1-14

Module Completion Requirements

- **Module Examination:** minimum for pass- 45%.

Module Achievement

The module achievement ensures that students have obtained a sufficient level of practical system programming skills. The module achievement consists of hands-on homework assignments. Students have to obtain 50% out of 10 assignments. Two additional makeup assignments are offered during the semester and one more during the intersession.

6.1.11 Functional Programming

Module Name	Functional Programming
Module Code	SDT-202
Module ECTS	5
Program Owner	SDT-BSc (Software, Data and Technology)
Module Coordinator	Prof. Dr. Alexander Omelchenko

Study Semester		
Program	**Semester**	**Status**
CS-BSc Computer Science	3	Mandatory Elective
SDT-BSc Software, Data and Technology	3	Mandatory Elective

Student Workload	
Lecture	17.5
Tutorial	17.5
Independent Study	70
Exam Preparation	20
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Functional Programming	SDT-202-A	Lecture	2.5
Functional Programming Tutorial	SDT-202-B	Tutorial	2.5

Module Description

The goal of this discipline is to provide students with a solid foundation in functional programming principles and techniques, focusing on the theoretical knowledge and practical skills required to effectively work with functional languages. The module explores the core concepts, language structures, syntax, and semantic constructs of functional programming languages, emphasizing their applicability in modern software development.

Content:

- Fundamentals of functional programming: lambda calculus and combinatory logic.

- Haskell programming language: syntax, semantics, standard library.

- Manage effects using applicative functors and monads.

- Type systems of functional languages.

A single assessment type cannot sufficiently test all intended learning outcomes. The program code evaluates practical skills, whereas the written examination assesses understanding of theoretical knowledge, core principles, and analytical reasoning.

Recommended Knowledge

It is recommended that students install a Linux system such as Ubuntu on their notebooks and that they become familiar with basic tools such as editors (vim or emacs) and the basics of a shell. The Glasgow Haskell Compiler (GHC) will be used for implementing Haskell programs.

Usability and Relationship to other Modules

Familiarity with functional programming concepts and principles is increasingly important in fields such as data science, artificial intelligence, and software development. This module provides a solid foundation in functional programming techniques and languages, which are essential for advanced modules in computer science and data science. Additionally, this module introduces advanced concepts of functional programming that are needed in advanced programming-oriented modules in the 2nd and 3rd years of the SDT program.

Intended Learning Outcomes

No	Competence	ILO
1	Collaborate	Collaborate effectively within a team in the IT field, utilizing project management tools, communication skills, and software for team project activities to jointly develop projects.
2	Compare	Compare and contrast the advantages and disadvantages of the functional programming paradigm, and apply functional programming techniques to solve applied problems using languages such as Haskell
3	Understand	Understand and utilize the basic type systems of functional languages and their extensions with polymorphic and recursive types to create efficient, well-structured code in a functional programming context
4	Choose	Choose between lazy and eager evaluation strategies based on the specific requirements of a problem or application, and implement software solutions using a functional programming paradigm.
5	Explain	Explain the computational model underlying functional programming and implement algorithms in functional languages using key concepts such as immutable data structures, recursion, and pattern matching
6	Employ	Employ generic annotations and type classes to describe interfaces and ensure static control, promoting code reusability and maintainability in functional programming projects

Indicative Literature

- Hughes, John. "Why functional programming matters." The computer journal 32.2 (1989): 98-107.
- Miran Lipovača. Learn You a Haskell for Great Good.
- O'Sullivan, Bryan, John Goerzen, and Don Stewart. Real World Haskell. O'Reilly Media, Inc., 2008.

Entry Requirements

Prerequisites	SDT-102 Core Algorithms and Data Structures **OR** CH-231 Algorithms and Data structures
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Functional Programming	Written Examination	60 Minutes	50	45%	All theoretical ILOs of the module
Functional Programming Tutorial	Program Code		50	45%	All practical ILOs of the module

Module Completion Requirements

- **Module Component Examination:** minimum pass 45% of each assessment.

Module Achievement

None.

6.1.12 Scientific Data Analysis

Module Name	Scientific Data Analysis
Module Code	CO-489
Module ECTS	5
Program Owner	PHDS-BSc (Physics and Data Science)
Module Coordinator	Prof. Dr. Veit Wagner

Study Semester		
Program	**Semester**	**Status**
MMDA-BSc Mathematics, Modeling, and Data Analytics	3	Mandatory
PHDS-BSc Physics and Data Science	3	Mandatory Elective
SDT-BSc Software, Data and Technology	3	Mandatory Elective

Student Workload	
Homework	55
Independent Study	35
Lecture	35
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Scientific Data Analysis	CO-489-A	Lecture	5

Module Description

Interpretation of scientific data is at the core of knowledge creation in any science. Proper tools and analysis techniques are the foundation for new theory validation against experimental findings, parameter extraction from computational or experimental data, and to discover data relationships in given data sets. This holds for all fields of physics, for the natural sciences in general and for fields beyond. This module provides a calculus-based introduction to analytical techniques applied to scientific data sets. Topics include probability distributions, linear and non-linear least square estimation, Bayesian statistics, Fourier analysis, (time) sequence analysis including power spectra and convolution, principal component analysis, data visualization techniques, as well as error and outlier analysis. Exemplary datasets from experimental and computational sources are used throughout the course. The course introduces their proper handling and data organization in databases. The course is part of the core physics and data science as well as the core mathematics, modeling and data analytics education. It builds on the foundation of the programming lab, the data handling in first year lab courses and first year mathematics foundations. Essential practical experience in applying the various analysis techniques and their visualization will be supported by homework exercises in close coordination with the lectures. The aim of the module is to enable students to properly handle, store, analyze and visualize larger multidimensional scientific datasets by various methods and from various

fields, and to prepare students for the data handling in their BSc thesis research. At the same time, students' programming and mathematical repertoires as well as their problem-solving skills are developed. The module also serves as a foundation for specialization subject modules.

Recommended Knowledge

- Mathematics at the level of the Mathematical Modelling Module.

- Basic programming skills in Python.

- Review mathematics/linear algebra/statistics and programming at the level of the first-year courses.

Intended Learning Outcomes

No	Competence	ILO
1	Perform	Perform curve and model fitting.
2	Conduct	Conduct advanced data analysis including Fourier analysis and Bayesian statistics.
3	Understand	Understand error handling in multidimensional complex data analysis.
4	Store	Store, import, handle and visualize large data sets.

Indicative Literature

- Edward L. Robinson: Data Analysis for Scientists and Engineers, Princeton University Press, 2016.
- Graham Currell: Scientific Data Analysis, Oxford University Press, 2015.

Entry Requirements

Prerequisites	SDT-105 Industrial Programming with Python
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Scientific Data Analysis	Portfolio Assessment	(Assignments, Quizzes)	100	45%	1-4

Module Completion Requirements

- **Module Examination:** minimum pass 45%- portfolio.

Module Achievement

None.

6.1.13 Advanced Algorithms and Data Structures

Module Name	Advanced Algorithms and Data Structures
Module Code	SDT-201
Module ECTS	5
Program Owner	SDT-BSc (Software, Data and Technology)
Module Coordinator	Prof. Dr. Alexander Omelchenko

Study Semester		
Program	**Semester**	**Status**
SDT-BSc Software, Data and Technology	3	Mandatory

Student Workload	
Lecture	17.5
Tutorial	17.5
Independent Study	70
Exam Preparation	20
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Advanced Algorithms and Data Structures	SDT-201-A	Lecture	2.5
Advanced Algorithms and Data Structures Tutorial	SDT-201-B	Tutorial	2.5

Module Description

This module builds on the concepts and techniques covered in "Core Algorithms and Data Structures" and provides students with a deeper understanding of these important topics. The module will focus on more advanced algorithms and data structures that are commonly used in practice, and will provide students with a solid foundation for more advanced coursework in the program and for professional development in the field of software, data and technology.

Content:

- Advanced data structures such as B-trees, AVL trees, and hash tables

- Advanced algorithms for sorting, searching, and graph manipulation

- Techniques for parallel and distributed algorithms

- Algorithms for network flow and linear programming

- Algorithms for approximation and randomization

- Advanced algorithms for specific areas such as computational geometry, cryptography, and machine learning

- Techniques for analyzing the performance of algorithms and data structures and making trade-offs between time and space complexity

A single assessment type cannot sufficiently test all intended learning outcomes. The practical assessment evaluates practical skills, whereas the written examination assesses understanding of theoretical knowledge, core principles, and analytical reasoning.

Recommended Knowledge

Students should refresh their knowledge of the C++ and Python programming language and be able to solve simple programming problems in C++ and Python. Students are expected to have a working programming environment. Also, they should refresh their knowledge of the basics of algorithms and data structures.

Usability and Relationship to other Modules

Familiarity with advanced algorithms and data structures is essential for almost all advanced modules in SDT. This module builds upon the concepts covered in "Core Algorithms and Data Structures" and introduces more advanced algorithms and data structures that are commonly used in practice. Additionally, the module covers techniques for designing, implementing and analyzing efficient algorithms and data structures, and provides students with hands-on experience implementing these concepts in a programming language. This module is essential for students planning to continue their studies in the 2nd and 3rd years of the SDT program, as well as for those planning to pursue a career in the field of computer science.

Intended Learning Outcomes

No	Competence	ILO
1	Design	Design, implement and analyze advanced algorithms and data structures for various problems.
2	Understand	Understand the trade-offs between time and space complexity and make appropriate decisions when choosing algorithms and data structures.
3	Apply	Apply advanced algorithms and data structures to solve problems in different areas of computer science such as distributed systems, databases, and machine learning.
4	Understand	Understand and use parallel and distributed algorithms
5	Understand	Understand the concepts of computational complexity theory and use them to analyze the performance of algorithms and data structures
6	Understand	Understand the properties and use of specific algorithms and data structures used in different areas of computer science
7	Apply	Apply mathematical concepts and formalize algorithms to solve practical problems

Indicative Literature

- David E. Goldberg: Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, 1989.
- Jon Kleinberg and Éva Tardos: Algorithm Design, 1st edition, Pearson, 2005.
- Michael T. Goodrich, Roberto Tamassia, and Michael H. Goldwasser: Data Structures and Algorithms in Python, John Wiley & Sons, 2013.
- Robert Sedgewick and Kevin Wayne: Algorithms, 4th edition, Addison-Wesley, 2011.
- Steven Skiena: The Algorithm Design Manual, 2nd edition, Springer, 2008.
- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein: Introduction to Algorithms, 3rd edition, MIT Press, 2009.

Entry Requirements

Prerequisites	SDT-102 Core Algorithms and Data Structures **OR** CH-231 Algorithms and Data structures
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Advanced Algorithms and Data Structures	Written Examination	60 Minutes	50	45%	All theoretical ILOs of the module
Advanced Algorithms and Data Structures Tutorial	Practical Assessment		50	45%	All practical ILOs of the module

Module Completion Requirements

- **Module Component Examination:** minimum pass 45% of each assessment.

Module Achievement

None.

6.1.14 Machine Learning

Module Name	Machine Learning
Module Code	CO-541
Module ECTS	5
Program Owner	RIS-BSc (Robotics and Intelligent Systems)
Module Coordinator	Prof. Dr. Francesco Maurelli

Study Semester		
Program	**Semester**	**Status**
CS-BSc Computer Science	4	Mandatory Elective
MMDA-BSc Mathematics, Modeling, and Data Analytics	4	Mandatory
PHDS-BSc Physics and Data Science	4	Mandatory Elective
RIS-BSc Robotics and Intelligent Systems	4	Mandatory
SDT-BSc Software, Data and Technology	4	Mandatory
IEM-BSc Industrial Engineering & Management	6	Mandatory Elective

Student Workload	
Class Attendance	35
Exam Preparation	20
Independent Study	70
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Machine Learning	CO-541-A	Lecture	5

Module Description

Machine learning (ML) concerns algorithms that are fed with (large quantities of) real-world data, and which return a compressed "model" of the data. An example is the "world model" of a robot; the input data are sensor data streams, from which the robot learns a model of its environment, which is needed, for instance, for navigation. Another example is a spoken language model; the input data are speech recordings, from which ML methods build a model of spoken English; this is useful, for instance, in automated speech recognition systems. There exist many formalisms in which such models can be cast, and an equally large diversity of learning algorithms. However, there is a relatively small number of fundamental challenges that are common to all of these formalisms and algorithms. The lectures introduce such fundamental concepts and illustrate them with a choice of elementary model

formalisms (linear classifiers and regressors, radial basis function networks, clustering, online adaptive filters, neural networks, or hidden Markov models). Furthermore, the lectures also (re-)introduce required mathematical material from probability theory and linear algebra.

Usability and Relationship to other Modules

- This module gives a thorough introduction to the basics of machine learning. It complements the Artificial Intelligence module.

Intended Learning Outcomes

No	Competence	ILO
1	Understand	Understand the notion of probability spaces and random variables
2	Understand	Understand basic linear modeling and estimation techniques
3	Understand	Understand the fundamental nature of the "curse of dimensionality"
4	Understand	Understand the fundamental nature of the bias-variance problem and standard coping strategies
5	Use	Use elementary classification learning methods (linear discrimination, radial basis function networks, multilayer perceptrons)
6	Implement	Implement an end-to-end learning suite, including feature extraction and objective function optimization with regularization based on cross-validation

Indicative Literature

- C. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.
- S. Shalev-Shwartz, Shai Ben-David: Understanding Machine Learning, Cambridge University Press, 2014.
- T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd edition, Springer, 2008.
- T.M. Mitchell, Machine Learning, Mc Graw Hill, India, 2017.

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Machine Learning	Written Examination	120 Minutes	100	45%	1-6

Module Completion Requirements

- **Module Examination:** minimum for pass- 45%.

Module Achievement

None.

6.1.15 Discrete Mathematics

Module Name	Discrete Mathematics
Module Code	CO-501
Module ECTS	5
Program Owner	MMDA-BSc (Mathematics, Modeling, and Data Analytics)
Module Coordinator	Prof. Dr. Keivan Mallahi Karai

Study Semester		
Program	**Semester**	**Status**
MMDA-BSc Mathematics, Modeling, and Data Analytics	4	Mandatory
RIS-BSc Robotics and Intelligent Systems	4	Mandatory Elective
SDT-BSc Software, Data and Technology	4	Mandatory Elective

Student Workload	
Independent Study	90
Lecture	35
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Discrete Mathematics	CO-501-A	Lecture	5

Module Description

This module is an introductory lecture in discrete mathematics. The lecture consists of two main components, enumerative combinatorics and graph theory. The lecture emphasizes connections of discrete mathematics with other areas of mathematics such as linear algebra and basic probability, and outlines applications to areas of computer science, cryptography, etc. where employment of ideas from discrete mathematics has proven to be fruitful. The first part of the lecture—enumerative combinatorics—deals with several classical enumeration problems (Binomial coefficients, Stirling numbers), counting under group actions and generating function. The second half of the lecture—graph theory—includes a discussion of basic notions such as chromatic number, planarity, matchings in graphs, Ramsey theory, and expanders, and their applications.

Recommended Knowledge

- Basic university mathematics: can be acquired via the Methods Modules “Calculus and Elements of Linear Algebra I + II” or Matrix Algebra and Advanced Calculus.

- Some basic familiarity with linear algebra is useful, but not technically required.

- It is recommended to have taken the Methods module: Calculus and Elements of Linear Algebra I + II

Usability and Relationship to other Modules

- This module is recommended for students pursuing a minor in Mathematics.

- This module is a good option as an elective module for students in RIS.

Intended Learning Outcomes

No	Competence	ILO
1	Demonstrate	Demonstrate their mastery of basic tools in discrete mathematics
2	Develop	Develop the ability to use discrete mathematics concepts (such as graphs) to model problems in computer science
3	Analyze	Analyze the definition of basic combinatorial objects such as graphs, permutations, partitions, etc.
4	Formulate	Formulate and design methods and algorithms for solving applied problems based on concepts from discrete mathematics

Indicative Literature

- B. Bollobas (1998). Modern Graph Theory, Berlin: Springer.
- J.H. van Lint and R.M. Wilson (2001). A Course in Combinatorics, second edition. Cambridge: Cambridge University Press.

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Discrete Mathematics	Written Examination	120 Minutes	100	45%	1-4

Module Completion Requirements

- **Module Examination:** minimum for pass- 45%.

Module Achievement

None.

6.1.16 Artificial Intelligence

Module Name	Artificial Intelligence
Module Code	CO-547
Module ECTS	5
Program Owner	RIS-BSc (Robotics and Intelligent Systems)
Module Coordinator	Prof. Dr. Andreas Birk

Study Semester		
Program	**Semester**	**Status**
CS-BSc Computer Science	4	Mandatory Elective
Minor-RIS-BSc Minor Option in RIS	4	Mandatory
RIS-BSc Robotics and Intelligent Systems	4	Mandatory
SDT-BSc Software, Data and Technology	4	Mandatory Elective
CS-BSc Computer Science	6	Mandatory Elective

Student Workload	
Class Attendance	35
Exam Preparation	20
Independent Study	70
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Artificial Intelligence	CO-547-A	Lecture	5

Module Description

Artificial Intelligence (AI) is an important subdiscipline of Computer Science that deals with technologies to automate the performance of tasks that are usually associated with intelligence. AI methods have a significant application potential, as there is an increasing interest and need to generate artificial systems that can carry out complex missions in unstructured environments without permanent human supervision. The module teaches a selection of the most important methods in AI. In addition to general-purpose techniques and algorithms, it also includes aspects of methods that are especially targeted for physical systems such as intelligent mobile robots or autonomous cars.

Recommended Knowledge

Revise content of the pre-requisite modules.

Usability and Relationship to other Modules

This module gives an introduction to Artificial Intelligence (AI) excluding the aspects of machine learning (ML), which are covered in a dedicated module that complements this one.

Intended Learning Outcomes

No	Competence	ILO
1	Outline	Outline and explain the history, general developments, and application areas of AI.
2	Apply	Apply the basic concepts and methods of behavior-oriented AI.
3	Use	Use concepts and methods of search algorithms for problem-solving.
4	Explain	Explain the basic concepts of path-planning as an application example for domain-specific search.
5	Apply	Apply basic path-planning algorithms and compare their relations to general search algorithms.
6	Write	Write and explain concepts of propositional and first-order logic.
7	Use	Use logic representations and inference for basic examples of artificial planning systems.

Indicative Literature

- J.-C. Latombe, Robot Motion Planning, Springer, 1991.
- S. Russell and P. Norvig Artificial Intelligence: A Modern Approach, Prentice Hall, 2009.
- S.M. LaValle, Planning Algorithms. Cambridge University Press, 2006.

Entry Requirements

Prerequisites	SDT-102 Core Algorithms and Data Structures **OR** CH-231 Algorithms and Data structures
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Artificial Intelligence	Written Examination	120 Minutes	100	45%	1-7

Module Completion Requirements

- **Module Examination:** minimum for pass- 45%.

Module Achievement

None.

6.1.17 Software Engineering and Design

Module Name	Software Engineering and Design
Module Code	SDT-204
Module ECTS	7.5
Program Owner	SDT-BSc (Software, Data and Technology)
Module Coordinator	Prof. Dr. Timofey Bryksin

Study Semester		
Program	**Semester**	**Status**
SDT-BSc Software, Data and Technology	4	Mandatory

Student Workload	
Lecture	17.5
Tutorial	35
Independent Study	115
Exam Preparation	20
Total Hours	**187.5**

Module Components	**Number**	**Type**	**CP**
Software Engineering and Design	SDT-204-A	Lecture	2.5
Software Engineering and Design Project	SDT-204-B	Project	5

Module Description

Educational Aims:

1. To provide students with a comprehensive understanding of software engineering principles, practices, and techniques, as well as the software development life cycle.

2. To enable students to analyze and design complex software systems, and to apply appropriate software development methodologies and tools.

3. To teach students the best practices for software quality assurance, including testing, debugging, and software maintenance.

4. To foster the development of critical thinking skills, problem-solving skills, and communication skills required for software engineering and design.

5. To introduce students to the latest trends, technologies, and tools in software engineering and design, and to prepare them to work effectively in a rapidly evolving field.

6. To encourage students to apply software engineering and design principles to real-world problems and to develop solutions that meet business and user needs.

7. To prepare students for professional practice in software engineering and design, including the ability to work collaboratively, to manage software development projects, and to apply ethical principles in the workplace.

8. To promote the development of a lifelong learning mindset, and to encourage students to stay current with advances in software engineering and design throughout their careers.

Content:

1. Introduction to Software Engineering and Design

- Overview of software engineering and design principles and practices
- Software development life cycle and its phases
- Roles and responsibilities of software engineers and designers

2. Software Requirements Analysis and Specification

- Understanding and capturing software requirements
- Techniques for analyzing requirements
- Documentation and communication of requirements
- Requirements validation and verification

3. Software Design Principles and Patterns

- Object-oriented design principles
- Design patterns and their applications
- Modeling techniques and tools
- Design trade-offs and considerations

4. Software Architecture and Design

- Architectural styles and patterns
- Architecture modeling and documentation
- System and component design
- Integration and testing of software components

5. Software Testing and Quality Assurance

- Testing techniques and strategies
- Test planning and execution
- Quality metrics and measures

- Continuous integration and delivery

6. Software Project Management

- Project planning and estimation

- Risk management and mitigation

- Team organization and communication

7. Agile methodologies and practices

- Emerging Technologies and Trends in Software Engineering and Design

- Cloud computing and software-as-a-service (SaaS)

- Mobile and web application development

- Artificial intelligence and machine learning

- Blockchain technology and distributed systems

8. Ethical and Legal Issues in Software Engineering and Design

- Intellectual property and copyright laws

- Privacy and data protection

- Software piracy and licensing

- Ethical considerations in software development and use.

A single assessment type cannot sufficiently test all intended learning outcomes. The program code evaluates practical skills, whereas the written examination assesses understanding of theoretical knowledge, core principles, and analytical reasoning.

Recommended Knowledge

Students should be familiar with basic concepts of software development, such as data types, control structures, and object-oriented programming.

Usability and Relationship to other Modules

The knowledge and skills acquired in this module will be useful for students planning to pursue a career in software development or data science, or continue their studies in advanced software development or data science modules. The module will also be beneficial for students planning to pursue a career in other related fields such as IT management, software testing, or software project management.

Intended Learning Outcomes

No	Competence	ILO

1	Understand	Understand the software development life cycle and its phases, and be able to apply appropriate software development methodologies and tools to various stages of the process.
2	Apply	Apply software design principles and patterns to develop complex software systems that meet business and user needs.
3	Apply	Apply appropriate software quality assurance practices, including testing, debugging, and software maintenance, to ensure the quality of software products.
4	Develop	Develop critical thinking and problem-solving skills to identify, analyze, and solve software engineering and design problems.
5	Develop	Develop effective communication and collaboration skills required for professional practice in software engineering and design.
6	Analyze	Analyze and evaluate software requirements and specifications, and develop software that meets those requirements.
7	Apply	Apply appropriate software architecture and design principles and patterns to design and develop software systems
8	Apply	Apply risk management strategies to identify, analyze, and mitigate risks associated with software development projects.
9	Understand	Understand the ethical and legal considerations associated with software development, and apply ethical principles in the workplace.
10	Demonstrate	Stay current with advances in software engineering and design, and demonstrate a commitment to lifelong learning in the field.

Indicative Literature

- Craig Larman: Agile and Iterative Development: A Manager's Guide, Addison-Wesley, 2004.
- Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides: Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, 1994.
- Grady Booch, James Rumbaugh, Ivar Jacobson: The Unified Modeling Language User Guide, 2nd edition, Addison-Wesley, 2005.
- Ian Sommerville: Software Engineering, 10th edition, Pearson, 2015.
- Kent Beck: Test-Driven Development: By Example, Addison-Wesley, 2002.
- Martin Fowler: Patterns of Enterprise Application Architecture, Addison-Wesley, 2002.
- Michael Feathers: Working Effectively with Legacy Code, Prentice Hall, 2004.
- Robert Martin: Agile Software Development, Principles, Patterns, and Practices, Pearson, 2002.
- Roger S. Pressman: Software Engineering: A Practitioner's Approach, 8th edition, McGraw-Hill, 2014.
- Steve McConnell: Code Complete: A Practical Handbook of Software Construction, 2nd edition, Microsoft Press, 2004.

Entry Requirements

Prerequisites	CO-562 Operating Systems

Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Software Engineering and Design	Written Examination	60 Minutes	33	45%	All theoretical ILOs of the module
Software Engineering and Design Project	Program Code		67	45%	All practical ILOs of the module

Module Completion Requirements

- **Module Component Examination:** minimum pass 45% of each assessment.

Module Achievement

None.

6.1.18 Database Fundamentals

Module Name	Database Fundamentals
Module Code	SDT-205
Module ECTS	5
Program Owner	SDT-BSc (Software, Data and Technology)
Module Coordinator	Prof. Dr. Alexander Omelchenko

Study Semester		
Program	**Semester**	**Status**
SDT-BSc Software, Data and Technology	4	Mandatory Elective

Student Workload	
Lecture	17.5
Tutorial	17.5
Independent Study	70
Exam Preparation	20
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Database Fundamentals	SDT-205-A	Lecture	2.5
Database Fundamentals Project	SDT-205-B	Project	2.5

Module Description

Content:

- Introduction to databases and data management

- Database design and modeling

- Relational database management systems (RDBMS)

- SQL for data manipulation and querying

- Database normalization and optimization

- NoSQL databases and data warehousing

- Database security and administration

Educational Aims:

- To provide students with a strong foundation in database design, modeling and management

- To teach students how to use SQL to manipulate and query data in a relational database

- To enable students to design and implement efficient and normalized databases

- To familiarize students with the unique features of NoSQL databases and data warehousing

- To prepare students for using databases in various fields such as software development, data science, and business.

A single assessment type cannot sufficiently test all intended learning outcomes. The program code evaluates practical skills, whereas the written examination assesses understanding of theoretical knowledge, core principles, and analytical reasoning.

Recommended Knowledge

Working knowledge of basic algorithms and data structures, such as trees, is required as well as familiarity with an object-oriented programming language such as Kotlin. For the project work, students benefit from having basic hands-on skills using Linux and, ideally, basic knowledge of a scripting language such as Python.

Usability and Relationship to other Modules

Familiarity with databases is essential for students who wish to specialize in software development and data science. This module will provide a solid foundation in database design, modeling, and management, including the use of SQL to manipulate and query data. Additionally, this module will introduce advanced concepts of database management and NoSQL databases, which are needed in advanced programming-oriented modules in the 3rd year of the SDT program.

Intended Learning Outcomes

No	Competence	ILO
1	Design	Design and model databases.
2	Use	Use SQL to manipulate and query data in a relational database.
3	Implement	Implement and optimize databases using normalization techniques.
4	Use	Use NoSQL databases and data warehousing.
5	Use	Use databases in various fields such as software development, data science, and business.
6	Manage	Manage and secure databases

Indicative Literature

- C. J. Date: An Introduction to Database Systems, 8th edition, Addison-Wesley, 2004.
- Elmasri Navathe: Fundamentals of Database Systems, 7th edition, Addison-Wesley, 2014.
- Martin Kleppmann: Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems, O'Reilly Media, 2017.
- Ramez Elmasri, Shamkant B. Navathe: Database Systems: Concepts, Design and Applications, Prentice-Hall, 1999.

Entry Requirements

Prerequisites	SDT-102 Core Algorithms and Data Structures

Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Database Fundamentals	Written Examination	60 Minutes	50	45%	All theoretical ILOs of the module
Database Fundamentals Project	Program Code		50	45%	All practical ILOs of the module

Module Completion Requirements

- **Module Component Examination:** minimum pass 45% of each assessment.

Module Achievement

None.

6.1.19 Deep Learning

Module Name	Deep Learning
Module Code	MCSSE-AI-01
Module ECTS	5
Program Owner	CSSE-MSc (Computer Science & Software Engineering)
Module Coordinator	Dr. Dmitry Kropotov

Study Semester		
Program	**Semester**	**Status**
AST-MSc Advanced Software Technology	1	Mandatory Elective
CSSE-MSc Computer Science & Software Engineering	1	Mandatory Elective
DE-MSc Data Engineering	1	Mandatory Elective
CSSE-MSc Computer Science & Software Engineering	3	Mandatory Elective
DE-MSc Data Engineering	3	Mandatory Elective
PHDS-BSc Physics and Data Science	5	Mandatory Elective
SDT-BSc Software, Data and Technology	5	Mandatory Elective

Student Workload	
Exam Preparation	20
Independent Study	70
Lecture	35
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Deep Learning	MCSSE-AI-01	Lecture	5

Module Description

In machine learning we aim at extracting meaningful representations, patterns and regularities from high-dimensional data. In recent years, researchers from various disciplines have developed “deep” hierarchical models, i.e. models that consist of multiple layers of nonlinear processing. An important property of these models is that they can “learn” by reusing and combining intermediate concepts, so that these models can be used successfully in a variety of domains, including information retrieval, natural language processing, and visual object detection. After a brief introduction into core knowledge related to training, model evaluation and multilayer perceptrons, this module focuses on

exposing students to deep learning techniques including convolutional and recurrent neural networks, autoencoders, generative adversarial networks and reinforcement learning. The central aim is hence to enable students to critically assess and apply modern methods in machine learning.

Recommended Knowledge

- This module is recommended for students that have been exposed to core knowledge in machine learning / statistical learning on undergraduate level. Students without this background knowledge can still join since required core knowledge is re-introduced. Preparation via auxiliary literature or online courses will facilitate the start into the course.

- Strong knowledge and abilities in mathematics (linear algebra, calculus).

Usability and Relationship to other Modules

While the graduate level modules "Data Analytics" and "Machine Learning" provide an applied introduction to the field and are therefore recommended for students with a focus on Software Engineering or Cybersecurity, this module complements the undergraduate module "Machine Learning" or can be used independently as a strong introduction to the field of Deep Learning.

Intended Learning Outcomes

No	Competence	ILO
1	Understand	Understand core techniques to train neural networks.
2	Select	Select from modern neural network architectures the most appropriate method (e.g. convolutional and recurrent neural networks) based on given input data.
3	Contrast	Contrast different recent unsupervised learning methods including autoencoders and generative adversarial networks.
4	Describe	Describe techniques in reinforcement learning.

Indicative Literature

- Aurélien Géron: Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow, 2nd Edition, O'Reilly, 2019.
- Charu C. Aggarwal: Neural Networks and Deep Learning – A Textbook, Springer, 2018.
- Christopher M. Bishop: Pattern Recognition and Machine Learning, Springer, 2006.
- Ian Goodfellow, Yoshua Bengio, Aaron Courville: Deep Learning, MIT Press, 2016.

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Deep Learning	Written Examination	120 Minutes	100	45%	1-4

Module Completion Requirements

- **Module Examination:** minimum for pass- 45%.

Module Achievement

None.

6.1.20 Stochastic Modeling and Financial Mathematics

Module Name	Stochastic Modeling and Financial Mathematics
Module Code	CA-S-MMDA-803
Module ECTS	5
Program Owner	MMDA-BSc (Mathematics, Modeling, and Data Analytics)
Module Coordinator	Prof. Dr. Sören Petrat

Study Semester		
Program	**Semester**	**Status**
MMDA-BSc Mathematics, Modeling, and Data Analytics	5	Mandatory Elective
PHDS-BSc Physics and Data Science	5	Mandatory Elective
RIS-BSc Robotics and Intelligent Systems	5	Mandatory Elective
MMDA-BSc Mathematics, Modeling, and Data Analytics	6	Mandatory Elective
RIS-BSc Robotics and Intelligent Systems	6	Mandatory Elective
SDT-BSc Software, Data and Technology	6	Mandatory Elective

Student Workload	
Lecture	35
Independent Study	90
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Stochastic Modeling and Financial Mathematics	CA-MMDA-803	Lecture	5

Module Description

This module is a first hands-on introduction to stochastic modeling. Examples will mostly come from the area of Financial Mathematics, so that this module plays a central role in the education of students interested in Quantitative Finance and Mathematical Economics. The module is taught as an integrated lecture-lab, where short theoretical units are interspersed with interactive computation and computer experiments.

Topics include a short introduction to the basic notions of financial mathematics, binomial tree models, discrete Brownian paths, stochastic integrals and ODEs, Ito's Lemma, Monte-Carlo methods, finite

differences solutions, the Black-Scholes equation, and an introduction to time series analysis, parameter estimation, and calibration. Towards the end, the Fokker-Planck equation, Ornstein-Uhlenbeck processes, and nonlinear Stochastic Partial Differential Equations are discussed, and connections to applications in physics and other areas of mathematics are made. Students will program and explore all basic techniques in a numerical programming environment and apply these algorithms to real data whenever possible.

Recommended Knowledge

- Good command of Calculus, Linear Algebra, and basic probability basic Python programming

- Review the content of Matrix Algebra & Advanced Calculus II

- Review Python programming

- Pre-install Anaconda Python on your own laptop and know how to edit and start simple Python programs in a Python IDE like Spyder (which comes bundled as part of Anaconda Python).

Usability and Relationship to other Modules

- This module is part of the core education in Mathematics, Modeling and Data Analytics.

- It is also valuable for students in Physics and Data Science, Computer Science, Data Engineering, RIS, and ECE, either as part of a minor in Mathematics, or as an elective module.

Intended Learning Outcomes

No	Competence	ILO
1	Apply	Apply fundamental concepts of deterministic and stochastic modeling.
2	Design	Design, conduct, and interpret controlled in-silico scientific experiments.
3	Analyze	Analyze the basic concepts of financial mathematics and their role in finance.
4	Write	Write computer code for basic financial calculations, binomial trees, stochastic differential equations, stochastic integrals and time series analysis.
5	Compare	Compare their programs and predictions in the context of real data.
6	Demonstrate	Demonstrate the usage of a version control system for collaboration and the submission of code and reports.

Indicative Literature

- A. Etheridge (2002). A Course in Financial Calculus. Cambridge: Cambridge University Press.
- D.J. Higham (2001). An Algorithmic Introduction to Numerical Simulation of Stochastic Differential Equations, SIAM Rev. 43(3):525-546.
- D.J. Higham (2004). Black-Scholes Option Valuation for Scientific Computing Students, Computing in Science & Engineering 6(6):72-79.
- J.C. Hull (2015). Options, Futures and other Derivatives, 9th edition. New York: Pearson.

- Y.-D. Lyuu (2002). Financial Engineering and Computation - Principles, Mathematics, Algorithms. Cambridge: Cambridge University Press.

Entry Requirements

Prerequisites	CTMS-MAT-22 Matrix Algebra and Advanced Calculus I CTMS-MAT-23 Matrix Algebra and Advanced Calculus II
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Stochastic Modeling and Financial Mathematics	Portfolio Assessment	(Programming assessments, project)	100	45%	1-6

Module Completion Requirements

Module Examination: minimum pass 45%- portfolio.

Module Achievement

None.

6.1.21 Optimization Methods

Module Name	Optimization Methods
Module Code	SDT-301
Module ECTS	5
Program Owner	SDT-BSc (Software, Data and Technology)
Module Coordinator	Prof. Dr. Alexander Omelchenko

Study Semester		
Program	**Semester**	**Status**
SDT-BSc Software, Data and Technology	5	Mandatory Elective

Student Workload	
Lecture	17.5
Tutorial	35
Independent Study	52.5
Exam Preparation	20
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Optimization Methods	SDT-301-A	Lecture	2.5
Optimization Methods Tutorial	SDT-301-B	Tutorial	2.5

Module Description

Optimization methods are a set of mathematical techniques used to find the best solution to a problem, given a set of constraints. In this module, students will learn about different optimization techniques such as linear and non-linear programming, gradient-based and heuristic methods, and their applications in various fields such as engineering, finance, and operations research. They will also learn about the implementation of optimization algorithms using programming languages such as Python. This module serves as an introduction to optimization methods and provides students with a solid foundation for more advanced coursework in the program and the industry.

Content:

- Linear programming: Formulation of LP problems, simplex method, duality theory, sensitivity analysis, and applications.

- Non-linear programming: Unconstrained optimization, optimization under constraints, optimization with inequality constraints, optimization with equality constraints, optimization with nonlinear constraints, and

applications.

- Gradient-based optimization methods: Newton and quasi-Newton methods, conjugate gradient and conjugate direction methods, and optimization of multivariable functions.

- Heuristic optimization methods: Genetic algorithms, simulated annealing, tabu search, and other metaheuristics.

- Applications of optimization methods: Linear and non-linear programming in engineering, finance, and operations research.

- Implementation of optimization algorithms using programming languages such as Python.

- Analysis and interpretation of the results of optimization problems.

- Trade-offs and limitations of different optimization methods.

- Communication and presentation of optimization results using mathematical notation and technical language.

- Ethical and societal implications of optimization methods.

A single assessment type cannot sufficiently test all intended learning outcomes. The program code evaluates practical skills, whereas the written examination assesses understanding of theoretical knowledge, core principles, and analytical reasoning.

Recommended Knowledge

Familiarity with basic mathematical concepts such as linear algebra, and calculus. Familiarity with basic programming concepts and experience with a programming language such as Python.

Usability and Relationship to other Modules

Optimization methods are widely used in fields such as engineering, finance, operations research and computer science. This module provides a solid foundation in optimization techniques such as linear and non-linear programming, gradient-based and heuristic methods, and their applications in various fields. It also covers the implementation of optimization algorithms using programming languages such as Python, which is essential for students who wish to pursue advanced coursework in related fields or pursue careers in fields such as engineering, finance, operations research, and computer science. Understanding optimization methods is also important for making informed decisions in various fields as well as solving real-world problems.

Intended Learning Outcomes

No	Competence	ILO
1	Understand	Understand the concepts and principles of optimization methods, including linear and non-linear programming, gradient-based and heuristic methods.
2	Apply	Apply optimization techniques to solve real-world problems in various fields such as engineering, finance, and operations research.
3	Understand	Understand the trade-offs and limitations of different optimization methods and select the appropriate method for a given problem.
4	Implement	Implement optimization algorithms using programming languages such as Python.
5	Analyze	Analyze and interpret the results of optimization problems and make recommendations based on the results.

6	Communicate	Communicate effectively about optimization methods using mathematical notation and technical language.
7	Develop	Develop an understanding of the ethical and societal implications of optimization methods.

Indicative Literature

- Andreu Mas-Colell, Michael D. Whinston, Jerry R. Green: Microeconomic Theory, Oxford University Press, 1995.
- David G. Luenberger, Yinyu Ye: Linear and Nonlinear Programming, 3rd edition, Springer, 2008.
- Dimitri P. Bertsekas: Nonlinear Programming, 2nd edition, Athena Scientific, 1999.
- John C. Hull: Options, Futures, and Other Derivatives, 9th edition, Prentice Hall, 2014.
- Jorge Nocedal, Stephen J. Wright: Numerical Optimization, 2nd edition, Springer, 2006.

Entry Requirements

Prerequisites	CH-151 Linear Algebra **OR** CTMS-MAT-23 Matrix Algebra and Advanced Calculus II SDT-105 Industrial Programming with Python
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Optimization Methods	Written Examination	60 Minutes	50	45%	All theoretical ILOs of the module
Optimization Methods Tutorial	Program Code		50	45%	All practical ILOs of the module

Module Completion Requirements

- **Module Component Examination:** minimum pass 45% of each assessment.

Module Achievement

None.

6.1.22 Natural Language Processing

Module Name	Natural Language Processing
Module Code	SDT-305
Module ECTS	5
Program Owner	SDT-BSc (Software, Data and Technology)
Module Coordinator	Prof. Dr. Alexander Omelchenko

Study Semester		
Program	**Semester**	**Status**
SDT-BSc Software, Data and Technology	6	Mandatory Elective

Student Workload	
Lecture	35
Independent Study	70
Exam Preparation	20
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Natural Language Processing	SDT-305-A	Lecture	5

Module Description

Students learn the fundamental concepts and techniques of natural language processing (NLP) and how to apply them to analyze, understand and generate human language. They gain hands-on experience with popular NLP libraries and frameworks in Python. Students also learn about the key NLP tasks such as tokenization, part-of-speech tagging, syntactic parsing, named entity recognition, about the key NLP applications such as text classification, sentiment analysis, machine translation, and text generation, and about the challenges and limitations of NLP and the state-of-the-art research in the field.

Content:

- Introduction to NLP, including the history and current state of the field

- Key NLP tasks such as tokenization, part-of-speech tagging, syntactic parsing

- Key NLP applications such as text classification, sentiment analysis, machine translation, and text generation

- Techniques for working with large text corpora, such as text pre-processing, data cleaning and data preparation

- Techniques for building NLP systems, such as statistical models, and neural networks

- Techniques for evaluating NLP systems

- State-of-the-art research in NLP

- Hands-on experience with popular NLP libraries and frameworks in Python

Recommended Knowledge

- Familiarity with basic mathematical concepts such as linear algebra, and calculus. Familiarity with basic programming concepts and experience with a programming language such as Python.

- Familiarity with basics of Deep Learning, Python and Pytorch is recommended.

Usability and Relationship to other Modules

This module will provide students with a solid understanding of the fundamental concepts and techniques of natural language processing, as well as hands-on experience with popular NLP libraries and frameworks in Python. The module will prepare students for more advanced coursework in the program and for professional development in the field of software, data and technology, particularly in areas such as text mining, information retrieval, and language-based AI.

Intended Learning Outcomes

No	Competence	ILO
1	Understand	Understand and apply fundamental concepts and techniques of natural language processing (NLP) to analyze, understand, and generate human language.
2	Identify	Identify and perform the key NLP tasks such as tokenization, part-of-speech tagging, syntactic parsing, semantic role labeling, named entity recognition, and coreference resolution.
3	Identify	Identify and apply the key NLP applications such as text classification, sentiment analysis, machine translation, and text generation.
4	Evaluate	Evaluate NLP systems and understand the challenges and limitations of NLP.
5	Understand	Understand the state-of-the-art research in NLP, learn how to read and reproduce modern NLP research papers.
6	Use	Use popular NLP libraries and frameworks in Python to implement NLP tasks and applications.

Indicative Literature

- Dan Jurafsky and James H. Martin: Speech and Language Processing, 3rd edition, Prentice Hall, 2020.
- Jacob Eisenstein: Natural Language Processing, 1st edition, Cambridge University Press, 2020.
- James H. Martin: Natural Language Processing with GATE, 1st edition, Cambridge University Press, 2016.
- Yoav Goldberg: Neural Network Methods for Natural Language Processing, 1st edition, Morgan & Claypool Publishers, 2017.

Entry Requirements

Prerequisites	CH-151 Linear Algebra **OR** CTMS-MAT-23 Matrix Algebra and Advanced Calculus II SDT-105 Industrial Programming with Python CO-541 Machine Learning
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Natural Language Processing	Written Examination	120 Minutes	100	45%	1-6

Module Completion Requirements

- **Module Examination:** minimum for pass- 45%.

Module Achievement

None.

6.1.23 Distributed Algorithms

Module Name	Distributed Algorithms
Module Code	CA-S-CS-803
Module ECTS	5
Program Owner	CS-BSc (Computer Science)
Module Coordinator	Dr. Kinga Lipskoch

Study Semester		
Program	**Semester**	**Status**
CS-BSc Computer Science	6	Mandatory Elective
RIS-BSc Robotics and Intelligent Systems	6	Mandatory Elective
SDT-BSc Software, Data and Technology	6	Mandatory Elective

Student Workload	
Class Attendance	35
Exam Preparation	20
Independent Study	70
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Distributed Algorithms	CA-CS-803	Lecture	5

Module Description

Distributed algorithms are the foundation of modern distributed computing systems. They are characterized by a lack of knowledge of a global state, a lack of knowledge of a global time, and inherent non-determinism in their execution. The course introduces basic distributed algorithms using an abstract formal model, which is centered on the notion of a transition system. The topics covered are logical clocks, distributed snapshots, mutual exclusion algorithms, wave algorithms, election algorithms, reliable broadcast algorithms, and distributed consensus algorithms. Process algebras are introduced as another formalism to describe distributed and concurrent systems.

The distributed algorithms introduced in this module form the foundation of computing systems that have to be scalable and fault-tolerant, e.g., large-scale distributed non-standard databases or distributed file systems. The course is recommended for students interested in the design of scalable distributed computing systems.

Recommended Knowledge

Students should refresh their knowledge of the C, C++ and Python programming language and be able to solve simple programming problems in C, C++ and Python. Students are expected to have a working programming environment.

Intended Learning Outcomes

No	Competence	ILO
1	Describe	Describe and analyze distributed algorithms using formal methods such as transition systems.
2	Explain	Explain different algorithms to solve election problems.
3	Illustrate	Illustrate the limitations of time to order events and how logical clocks and vector clocks overcome these limitations.
4	Apply	Apply distributed algorithms to produce consistent snapshots of distributed computations.
5	Describe	Describe the differences among wave algorithms for different topologies.
6	Analyze	Analyze and implement distributed consensus algorithms such as Paxos and Raft.
7	Use	Use a process algebra such as communicating sequential processes or π-calculus to model distributed algorithms.

Indicative Literature

- Maarten van Steen, Andrew S. Tanenbaum: Distributed Systems, 3rd edition, Pearson Education, 2017.
- Nancy A. Lynch: Distributed Algorithms, Morgan Kaufmann, 1996.

Entry Requirements

Prerequisites	SDT-102 Core Algorithms and Data Structures **OR** CH-231 Algorithms and Data structures
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs

Distributed Algorithms	Written Examination	120 Minutes	100	45%	1-7

Module Completion Requirements

- **Module Examination:** minimum for pass- 45%.

Module Achievement

None.

6.1.24 Computer Networks

Module Name	Computer Networks
Module Code	CO-564
Module ECTS	5
Program Owner	CS-BSc (Computer Science)
Module Coordinator	Dr. Kinga Lipskoch

Study Semester		
Program	**Semester**	**Status**
CS-BSc Computer Science	5	Mandatory Elective
SDT-BSc Software, Data and Technology	5	Mandatory Elective

Student Workload	
Class Attendance	35
Independent Study	70
Exam Preparation	20
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Computer Networks	CO-564-A	Lecture	5

Module Description

Computer networks such as the Internet play a critical role in today's connected world. This module discusses the technology of Internet services in depth to enable students to understand the core issues involved in the design of modern computer networks. Fundamental algorithms and principles are explained in the context of existing protocols as they are used in today's Internet. Students taking this module should finally understand the technical complexity behind everyday online services such as Google or YouTube.

Students taking this module will understand how computer networks work and they will be able to assess communication networks, including aspects such as performance but also robustness and security. Students will learn that the design of communication networks is not only influenced by technical constraints but also by the necessity to define common standards, which often requires to take engineering decisions that reflect non-technical requirements.

Recommended Knowledge

Students are expected to be familiar with the C programming language and to learn basics of higher-level scripting languages such as Python (the official Python documentation is available on https://docs.python.org/)

Usability and Relationship to other Modules

It is recommended to take the CORE module Operating Systems in semester 3 in which network programming APIs are covered, because a significant portion of the communication technology is implemented at the operating system level. An understanding of operating system concepts and abstractions will help students to understand how computer network technology is commonly implemented and made available to applications. The specialization module Distributed Algorithms discusses algorithms for solving problems commonly found in distributed systems that use computer networks to exchange information. The module Secure and Dependable Systems introduces cryptographic mechanisms that can be used to secure communication over computer networks.

Intended Learning Outcomes

No	Competence	ILO
1	Recall	Recall layering principles and the OSI reference model.
2	Articulate	Articulate the organization of the Internet and the organization involved in providing Internet services.
3	Describe	Describe media access control, flow control, and congestion control mechanisms
4	Explain	Explain how local area networks differ from global networks.
5	Illustrate	Illustrate how frames are forwarded in local area networks.
6	Contrast	Contrast addressing mechanisms and translations between addresses used at different layers.
7	Demonstrate	Demonstrate how the Internet network layer forwards packets.
8	Present	Present how routing algorithms and protocols are used to determine and select routes.
9	Describe	Describe how the Internet transport layer provides different end-to-end services.
10	Demonstrate	Demonstrate how names are resolved to addresses and vice versa.
11	Summarize	Summarize how application layer protocols send and access electronic mail or access resources on the world-wide web.
12	Design	Design and implement simple application layer protocols.
13	Recognize	Recognize to which extent computer networks are fragile and evaluate strategies to cope with the fragility.
14	Analyze	Analyze traffic traces produced by a given computer network.

Indicative Literature

- Andrew S. Tanenbaum: Computer Networks, 4th Edition, Prentice Hall, 2002.
- James F. Kurose, Keith W. Ross: Computer Networking: A Top-Down Approach Featuring the Internet, 3rd Edition, Addison-Wesley, 2004.

Entry Requirements

Prerequisites	SDT-102 Core Algorithms and Data Structures

	OR CH-231 Algorithms and Data structures
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Computer Networks	Written Examination	120 Minutes	100	45%	1-14

Module Completion Requirements

- **Module Examination:** minimum for pass- 45%.

Module Achievement

None.

6.1.25 Databases Internals

Module Name	Databases Internals
Module Code	SDT-302
Module ECTS	5
Program Owner	SDT-BSc (Software, Data and Technology)
Module Coordinator	Prof. Dr. Alexander Omelchenko

Study Semester		
Program	**Semester**	**Status**
SDT-BSc Software, Data and Technology	5	Mandatory Elective

Student Workload	
Lecture	17.5
Self-Organized Teamwork	35
Independent Study	52.5
Exam Preparation	20
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Databases Internals	SDT-302-A	Lecture	5

Module Description

This discipline is aimed at gaining skills in building data storage systems and operating with them in an effective way. During the course, students will get familiar with how a relational database engine works internally. They will look at the data structures, algorithms and mathematics, which allow for efficient execution of complex search queries in pretty big databases, and will try to implement a few components of a toy database.

Content:

- Principles of building relational DBMS

- Some issues of building distributed DBMS

- Columnar DBMS

- Non-classical DBMS types: XML, graph, object

- Elements of multidimensional indexing

- The task of configuring DBMS

Recommended Knowledge

- Familiarity with basic concepts of database management systems (DBMS) and SQL.

- Familiarity with basic concepts of data structures and algorithms.

- Familiarity with basic programming concepts and experience with a programming language such as Kotlin

Usability and Relationship to other Modules

An understanding of the internal workings of databases is essential for students pursuing careers in computer science, software engineering, and related fields. This module provides a solid foundation in the internal workings of database management systems, including storage structures, query processing, and transaction management. It also covers the implementation of databases using programming languages such as SQL. This knowledge is crucial for students who wish to pursue advanced coursework in computer science and related fields, as well as for those who wish to pursue careers in fields such as software development, data management and data administration.

Intended Learning Outcomes

No	Competence	ILO
1	Describe	Describe the principles of storing and accessing data records on disk, performing relational operations such as selections and joins, building and using indexes appropriately
2	Apply	Apply cost-based optimization techniques to the problems of choosing the optimal way of query execution
3	Implement	Implement lock-based and timestamp-ordering based schedulers in transactional systems

Indicative Literature

- C. J. Date: An Introduction to Database Systems, 8th edition, Addison-Wesley, 2004.
- Hector Garcia-Molina, Jeffrey D. Ullman, Jennifer Widom: Database Systems: The Complete Book, 2nd edition, Prentice Hall 2002.
- Michael Stonebraker, Joseph M. Hellerstein, James Hamilton: Readings in Database Systems, 5th edition, Morgan Kaufmann Publishers, 2018.
- Ramez Elmasri, Shamkant B Navathe: Fundamentals of Database Systems, 7th edition, Pearson, 2018.
- Thomas Connolly, Carolyn Begg: Database Systems: A Practical Approach to Design, Implementation and Management, 6th edition, Addison-Wesley, 2016.

Entry Requirements

Prerequisites	SDT-205 Database Fundamentals CTMS-SKI-27

	JVM-Based Programming
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Databases Internals	Written Examination	120 Minutes	100	45%	All theoretical ILOs of the module

Module Completion Requirements

- **Module Examination:** minimum for pass- 45%.

Module Achievement

None.

6.1.26 Integrated Development and IT Operations

Module Name	Integrated Development and IT Operations
Module Code	SDT-306
Module ECTS	5
Program Owner	SDT-BSc (Software, Data and Technology)
Module Coordinator	Prof. Dr. Alexander Omelchenko

Study Semester		
Program	**Semester**	**Status**
SDT-BSc Software, Data and Technology	6	Mandatory Elective

Student Workload	
Lecture	35
Independent Study	70
Exam Preparation	20
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Integrated Development and IT Operations	SDT-306-A	Lecture	5

Module Description

This module provides an introduction to the principles, practices, and tools of DevOps, a set of methodologies that aim to improve the collaboration, communication and integration between software development and IT operations teams. In this module, students will learn about the key concepts and practices of DevOps, including continuous integration, continuous delivery, and infrastructure as code. They will also learn about the tools and technologies used in DevOps, such as version control systems, containerization, and cloud computing. This module will give students a solid understanding of the principles and practices of DevOps and how they can be applied to improve the software development process and increase efficiency.

Content:

- Key concepts of DevOps, such as continuous integration, continuous delivery, and infrastructure as code

- Collaboration, communication and integration between software development and IT operations teams

- Version control systems and their role in DevOps

- Containerization and its usage in DevOps

- Cloud computing and its role in DevOps

- Automation and monitoring in DevOps

- Security considerations in DevOps

- Best practices and real-world examples of DevOps in action.

Recommended Knowledge

Students should be familiar with basic concepts of software development, including version control, testing, and deployment, with basic Linux command line usage and basic administration tasks.

Usability and Relationship to other Modules

DevOps is a rapidly growing field that combines software development and IT operations to improve the software development process and increase efficiency. This module provides a solid foundation in the principles, practices, and tools of DevOps, including continuous integration, continuous delivery, and infrastructure as code. It also covers the tools and technologies used in DevOps, such as version control systems, containerization, and cloud computing. This knowledge is crucial for students who wish to pursue careers in software development, IT operations, and related fields. It will also help students to understand how to improve the software development process and increase efficiency in their organizations.

Intended Learning Outcomes

No	Competence	ILO
1	Understand	Understand the key principles and practices of DevOps and how they can be applied to software development projects.
2	Use	Use version control systems such as Git to manage and track changes to code.
3	Set up	Set up and configure continuous integration and delivery pipelines using tools like Jenkins.
4	Use	Use infrastructure as code tools like Docker and Terraform to automate the deployment and management of applications and infrastructure.
5	Monitor	Monitor and log the performance and reliability of applications and systems using tools such as Splunk and Grafana.
6	Collaborate	Collaborate effectively with development and operations teams to improve the efficiency and reliability of software delivery.

Indicative Literature

- Gene Kim, Kevin Behr, George Spafford: The Phoenix Project: A Novel About IT DevOps and Helping Your Business Win IT Revolution, 2013.
- Jez Humble, David Farley: Continuous Delivery: Reliable Software Releases through Build Test and Deployment Automation, Addison-Wesley, 2010.

- Kim, Gene and John Willis. The DevOps Adoption Playbook: A Guide to Adopting DevOps in a Multi-Speed IT Enterprise. IT Revolution Press, 2018.
- Michael Nygard: Release It!: Design and Deploy Production-Ready Software, Pragmatic Bookshelf, 2007.
- Patrick Debois: The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations IT Revolution Press, 2016.

Entry Requirements

Prerequisites	CO-562 Operating Systems SDT-204 Software Engineering and Design
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Integrated Development and IT Operations	Written Examination	120 Minutes	100	45%	All theoretical ILOs of the module

Module Completion Requirements

- **Module Examination:** minimum for pass- 45%.

Module Achievement

None.

6.1.27 Parallel Programming

Module Name	Parallel Programming
Module Code	SDT-303
Module ECTS	5
Program Owner	SDT-BSc (Software, Data and Technology)
Module Coordinator	Prof. Dr. Kirill Krinkin

Study Semester		
Program	**Semester**	**Status**
SDT-BSc Software, Data and Technology	5	Mandatory Elective

Student Workload	
Lecture	35
Independent Study	70
Exam Preparation	20
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Parallel Programming	SDT-303-A	Lecture	5

Module Description

This module introduces the principles and practices of concurrent programming, including the state-of-the-art and modern approaches to building concurrent algorithms.

Content:

- Key concepts in concurrent programming, such as the shared-memory model, linearizability correctness property, and the standard progress guarantees

- Lock-based synchronization

- Basic non-blocking concurrent algorithms, such as the Michael-Scott queue

- Modern concurrent algorithms, such as the Fetch-And-Add-based queues

- Various techniques used in concurrent algorithms, such as helping, descriptors, and flat combining

Recommended Knowledge

- Fundamentals of computer science, including computer architecture, algorithms and data structures, and programming skills in Java or Kotlin.

- Strong knowledge of basic data structures and algorithms, basic programming skills in Kotlin.

Usability and Relationship to other Modules

Nowadays, concurrent programming is crucial for students pursuing careers in computer science, software engineering, and related fields. This module provides a solid foundation in the principles and practices of developing concurrent algorithms.

Intended Learning Outcomes

No	Competence	ILO
1	Understand	Understand the basic concepts of concurrent programming.
2	Able	Able to develop new concurrent algorithms either using the learned techniques and approaches or even by introducing new ones.
3	Able	Able to implement existing concurrent algorithms.
4	Able	Able to reason about the correctness of concurrent algorithms.

Indicative Literature

- Andrew S. Tanenbaum, Martin Casado: Computer Networks, 5th edition, Prentice Hall, 2010.
- Grama Ananth et al., Introduction to parallel computing. Pearson Education India, 2003.
- Herlihy, Maurice, Nir Shavit, Victor Luchangco, and Michael Spear. The art of multiprocessor programming. Newnes, 2020.
- Michael J. Quinn: Parallel Programming in C with MPI and OpenMP, McGraw-Hill, 2003.
- Peter Pacheco: An Introduction to Parallel Programming, Morgan Kaufmann Publishers, 2011.
- William Gropp, Ewing Lusk, Anthony Skjellum: Using MPI: Portable Parallel Programming with the Message-Passing Interface 2nd edition, MIT Press, 1999.

Entry Requirements

Prerequisites	CO-562 Operating Systems CTMS-SKI-27 JVM-Based Programming
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Parallel Programming	Written Examination	120 Minutes	100	45%	All theoretical ILOs of the module

Module Completion Requirements

- **Module Examination:** minimum for pass- 45%.

Module Achievement

None.

6.1.28 Formal Languages and Parsers

Module Name	Formal Languages and Parsers
Module Code	SDT-304
Module ECTS	5
Program Owner	SDT-BSc (Software, Data and Technology)
Module Coordinator	Prof. Dr. Anton Podkopaev

Study Semester		
Program	**Semester**	**Status**
SDT-BSc Software, Data and Technology	5	Mandatory Elective

Student Workload	
Lecture	17.5
Tutorial	35
Independent Study	52.5
Exam Preparation	20
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Formal Languages and Parsers	SDT-304-A	Lecture	2.5
Formal Languages and Parsers Tutorial	SDT-304-B	Tutorial	2.5

Module Description

The aim of this discipline is to give students theoretical knowledge and practical skills in the fundamentals of the theory of formal languages. Considerable attention is paid to issues related to the theoretical aspects of the syntax and semantics of programming languages, as well as to the creation of effective algorithms for lexical and syntactic analysis of program code. During the course, students will:

− learn basic methods of parsing; main approaches used for generating object code.

− be able to describe programming language syntax, using various approaches; construct language semantics using

different approaches; apply regular expressions for lexical analysis; create algorithms for efficient syntactic analysis of program code; develop JIT compilers.

− Have gained skills in application of methods for describing the syntax and semantics of programming languages using various approaches; methods for creating effective algorithms for lexical and syntactic analysis of program code.

Content:

- Deterministic and nondeterministic finite automata

- Deterministic and nondeterministic pushdown automata

- Turing machines and linear-bounded turing machines

- Recursive descent parsing

- Lexer and parser generators

- LL and LR grammars

- Parser expression grammars

- Combinatory parsing

- Regular languages and expressions

- Context-free languages and grammars

- Context-sensitive, recursively enumerable languages and their useful subsets

A single assessment type cannot sufficiently test all intended learning outcomes. The program code evaluates practical skills, whereas the written examination assesses understanding of theoretical knowledge, core principles, and analytical reasoning.

Recommended Knowledge

- Familiarity with basic mathematical concepts such as set theory, logic, and discrete mathematics. Familiarity with basic programming concepts and experience with a programming language such as Python or Kotlin.

Usability and Relationship to other Modules

Formal languages and automata form the theoretical foundations of computer science and are essential for

understanding the properties and limits of computation. This module provides a solid foundation in formal

languages and automata theory, including regular languages, context-free languages, Turing machines, and

decidability. It also covers the applications of formal languages and automata theory in various fields such as compilers, parsing, and verification. This knowledge is crucial for students who wish to pursue advanced coursework in computer science, theoretical computer science, and related fields, as well as for those who wish

to pursue careers in fields such as software engineering, verification, and natural language processing.

Intended Learning Outcomes

No	Competence	ILO

1	Know	Know the basic modern principles and approaches to the construction of formal languages. Have gained skill in finding relevant information on formal languages and grammars.
2	Can	Can design a language with syntax which can be effectively parsed using modern methods. Know the classes of grammars most useful in practice. Know the theoretical complexity bounds for relevant classes of grammars.
3	Know	Know the algorithms for syntactic analysis. Generate a lexer and a parser from the language specification. Effectively implement recursive descent parsers. Effectively implement parser combinators.
4	Know	Know the basic methods of working with finite state machines. Be able to present and justify the choice of methods of working with a given formal language. Have the skills to describe a given programming language syntax with regular expressions and context-free grammars

Indicative Literature

- Dexter Kozen: Automata and Computability, Springer, 1997.
- Harry Lewis, Christos Papadimitriou: Elements of the Theory of Computation, 2nd edition, Prentice Hall, 1998.
- John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman: Introduction to Automata Theory, Languages, and Computation, 3rd edition, Addison-Wesley, 2007.
- Martin Davis, Ron Sigal, Elaine J. Weyuker: Computability, Complexity, and Languages: Fundamentals of Theoretical Computer Science, 2nd edition, Elsevier, 1994.
- Michael Sipser: Introduction to the Theory of Computation, 3rd edition, Cengage Learning, 2013.

Entry Requirements

Prerequisites	SDT-106 System Programming and Performance CO-501 Discrete Mathematics SDT-105 Industrial Programming with Python
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Formal Languages and Parsers	Written Examination	60 Minutes	50	45%	All theoretical ILOs of the module
Formal Languages and Parsers Tutorial	Program Code		50	45%	All practical ILOs of the module

Module Completion Requirements

- **Module Component Examination:** minimum pass 45% of each assessment.

Module Achievement

None.

6.1.29 Compilers

Module Name	Compilers
Module Code	SDT-307
Module ECTS	5
Program Owner	SDT-BSc (Software, Data and Technology)
Module Coordinator	Prof. Dr. Anton Podkopaev

Study Semester		
Program	**Semester**	**Status**
SDT-BSc Software, Data and Technology	6	Mandatory Elective

Student Workload	
Lecture	17.5
Tutorial	17.5
Exam Preparation	20
Independent Study	70
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Compilers	SDT-307-A	Lecture	2.5
Compilers Project	SDT-307-B	Project	2.5

Module Description

This module provides students with a deep understanding of the principles and techniques used in compilers, including lexical analysis, syntax analysis, semantic analysis, and code generation, give them hands-on experience with the implementation of a compiler using a modern compiler construction toolkit, teach students about the important trade-offs involved in the design and implementation of a compiler, including performance, code size, and maintainability.

Content:

- Introduction to compilers and the compilation process. Programming languages and machine architectures. Compiler, interpreter.

- Syntax analysis. Stack machine.

- x86 command system. Code generator, control structures, procedures and functions.

- Dynamic data structures and symbolic expressions.

- Program Runtime

A single assessment type cannot sufficiently test all intended learning outcomes. The program code evaluates practical skills, whereas the written examination assesses understanding of theoretical knowledge, core principles, and analytical reasoning.

Recommended Knowledge

Students should have a solid understanding of at least one programming language and its syntax before taking this module. They should review basic data structures and algorithms, as well as math skills, formal languages and automata, functional programming.

Usability and Relationship to other Modules

This module is suitable for computer science students with a solid understanding of programming and algorithms, as well as an interest in the inner workings of programming languages and the software development process. Knowledge of the module may be used to develop and improve the compiler, to write code generators for specific processors, to design and implement new languages, to improve code optimization, and to design and implement run-time systems.

Intended Learning Outcomes

No	Competence	ILO
1	Know	Know the basic algorithms for parsing code. Implements these algorithms. Know how to implement a compiler stack machine.
2	Know	Know the basic principles of dynamic data structures. Know the disciplines of dynamic memory management.
3	Know	Know the basic principles of compiler structure. Be able to organize and conduct a scientific discussion on issues from the professional sphere. Know how to discuss a choice of the optimal compiler for solving practical tasks.
4	Know	Know basic stages of compiler development. Implement all compiler components.

Indicative Literature

- "Compilers: Principles Techniques and Tools" by Alfred V Aho Monica S Lam Ravi Sethi and Jeffrey D Ullman commonly known as the "Dragon book" published by Addison-Wesley Professional (1986).
- "Compiling with Continuations" by Andrew W Appel published by Cambridge University Press (1992).
- "Engineering a Compiler" by Keith D Cooper and Linda Torczon published by Morgan Kaufmann (2012).
- "Modern Compiler Implementation in Java" by Andrew W Appel published by Cambridge University Press (1998).
- "Principles of Compiler Design" by Alfred V Aho and Jeffrey D Ullman published by Addison-Wesley Professional (1977).
- gccgnuorg - compiler GCC
- gccgnuorg/wiki/ListOfCompilerBooks - a list of books on compiler construction

- llvmorg - infrastructure LLVM

Entry Requirements

Prerequisites	SDT-202 Functional Programming SDT-304 Formal Languages and Parsers
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Compilers	Written Examination	60 Minutes	50	45%	All theoretical ILOs of the module
Compilers Project	Program Code		50	45%	All practical ILOs of the module

Module Completion Requirements

- **Module Component Examination:** minimum pass 45% of each assessment.

Module Achievement

None.

6.1.30 Semantics of Programming Languages

Module Name	Semantics of Programming Languages
Module Code	SDT-308
Module ECTS	5
Program Owner	SDT-BSc (Software, Data and Technology)
Module Coordinator	Prof. Dr. Anton Podkopaev

Study Semester		
Program	**Semester**	**Status**
SDT-BSc Software, Data and Technology	6	Mandatory Elective

Student Workload	
Lecture	17.5
Tutorial	17.5
Independent Study	50
Exam Preparation	20
Total Hours	**105**

Module Components	**Number**	**Type**	**CP**
Semantics of Programming Languages	SDT-308-A	Lecture	2.5
Semantics of Programming Languages Tutorial	SDT-308-B	Tutorial	2.5

Module Description

The range of topics covered in this module includes approaches for formally describing semantics of a programming language as well as methods for proving the correctness of program transformations.

Content:

- Big-step and small-step operational semantics of imperative programming languages,

- Denotational semantics,

- Hoare's Axiomatic Semantics,

- Semantics of languages with multithreading.

A single assessment type cannot sufficiently test all intended learning outcomes. The program code evaluates practical skills, whereas the written examination assesses understanding of theoretical knowledge, core principles, and analytical reasoning.

Recommended Knowledge

- Understanding of propositional logic.

- To master the module students need the knowledge obtained as a result of studying "Formal languages and Parsers" and "Functional programming".

Usability and Relationship to other Modules

In terms of career prospects, knowledge of semantics of programming languages is relevant for anyone working in the field of programming languages, compilers, programming tools, programming languages design, formal verification, and many other areas. This course is used for developing a deeper understanding of how programming languages are designed, implemented, and used, for gaining a solid foundation in formal semantics and type systems, which can be applied to programming languages and other formal systems.

Intended Learning Outcomes

No	Competence	ILO
1	Know	Know the basic styles of programming language semantics: denotational, operational, axiomatic. Is able to reasonably describe the chosen style and the advantages of its use for a particular task
2	Know	Know the notion of semantic equivalence of programs and expressions. Knows variants of definitions and their relationships, justification of properties of program transformations.
3	Know	Know the concept of operational semantics. Knows how to use big-step and small-step semantics.
4	Know	Know the formalization of various semantics and proofs of their properties in Coq as well as verify programs using Hoare logic.

Indicative Literature

- "Introduction to the Theory of Programming Languages" by Michel Parigot published by Cambridge University Press (1992).
- "Programming Language Foundations" by Brian A Malloy published by Cambridge University Press (2018).
- "Semantics Engineering with PLT Redex" by Matthew Flatt Robert Bruce Findler and Shriram Krishnamurthi published by MIT Press (2013).
- "Semantics of Programming Languages" by Carl A Gunter published by MIT Press (1992).
- "Semantics with Applications: An Appetizer" by Hanne Riis Nielson and Flemming Nielson published by Springer (2007)
- Benjamin Pierce et al Software Foundations (Vol 1 2).
- FNielson H-RNielson Semantics with Applications A formal introduction.
- Glynn Winskel The Formal Semantics of Programming Languages.
- https://softwarefoundations.cis.upenn.edu/

Entry Requirements

Prerequisites	SDT-202 Functional Programming SDT-304 Formal Languages and Parsers
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Semantics of Programming Languages	Written Examination	60 Minutes	50	45%	All theoretical ILOs of the module
Semantics of Programming Languages Tutorial	Program Code		50	45%	All practical ILOs of the module

Module Completion Requirements

- **Module Component Examination:** minimum pass 45% of each assessment.

Module Achievement

None.

6.1.31 Advanced Discrete Mathematics

Module Name	Advanced Discrete Mathematics
Module Code	SDT-309
Module ECTS	5
Program Owner	SDT-BSc (Software, Data and Technology)
Module Coordinator	Prof. Dr. Alexander Omelchenko

Study Semester		
Program	**Semester**	**Status**
SDT-BSc Software, Data and Technology	5	Mandatory Elective

Student Workload	
Class Attendance	35
Exam Preparation	20
Independent Study	70
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Advanced Discrete Mathematics	SDT-309-A	Lecture	5

Module Description

This module offers an in-depth exploration of advanced topics in discrete mathematics, building upon foundational knowledge from the introductory module. It primarily focuses on sophisticated methods and concepts in enumerative combinatorics and graph theory. The module highlights advanced theoretical frameworks, emphasizes interdisciplinary connections with areas such as algebra, probability theory, optimization, and theoretical computer science, and introduces applications relevant to cryptography, algorithm design, network analysis, and data science. The course content is structured flexibly to accommodate current trends, research interests, and practical applications within discrete mathematics.

Recommended Knowledge

- Ability to follow and construct mathematical arguments and proofs.

- Basic proficiency in mathematical modeling and algorithmic reasoning.

- It is recommended to have taken the Discrete Mathematics module.

Usability and Relationship to other Modules

- This module is highly recommended for students pursuing advanced studies in mathematics or computer science.

- It serves as an ideal elective for students specializing in theoretical computer science, data science, or related interdisciplinary areas.

Intended Learning Outcomes

No	Competence	ILO
1	Demonstrate	Demonstrate a deep understanding of advanced combinatorial methods and graph-theoretic concepts.
2	Apply	Apply advanced discrete mathematical tools to model and analyze complex problems in computer science and related disciplines.
3	Evaluate	Evaluate critically and construct proofs involving advanced discrete structures.
4	Develop	Develop and refine algorithms that leverage sophisticated discrete mathematics techniques to solve practical computational problems.

Indicative Literature

- B. Bollobás (1998). Modern Graph Theory, Berlin: Springer.
- D.B. West (2000). Introduction to Graph Theory, second edition. Prentice Hall.
- J.H. van Lint and R.M. Wilson (2001). A Course in Combinatorics, second edition. Cambridge: Cambridge University Press..
- R.P. Stanley (2011). Enumerative Combinatorics, Volume 1 & 2, Cambridge: Cambridge University Press.

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Advanced Discrete Mathematics	Written Examination	120 Minutes	100	45%	1-4

Module Completion Requirements

- **Module Examination:** minimum for pass- 45%.

Module Achievement

None.

6.1.32 Internship / Startup and Career Skills

Module Name	Internship / Startup and Career Skills
Module Code	CA-INT-900
Module ECTS	15
Program Owner	Career (Career Module for UG programs)
Module Coordinator	Dr. Tanja Woebs Clémentine Senicourt

Study Semester		
Program	**Semester**	**Status**
BCCB-BSc Biochemistry and Cell Biology	5	Mandatory
CBT-BSc Chemistry and Biotechnology	5	Mandatory
CS-BSc Computer Science	5	Mandatory
ECE-BSc Electrical and Computer Engineering	5	Mandatory
ESSMER-BSc Earth Sciences and Sustainable Management of Environmental Resources	5	Mandatory
F-ACS-BSc Applied Computer Science	5	Mandatory
IBA-BA International Business Administration	5	Mandatory
IRPH-BA International Relations: Politics and History	5	Mandatory
ISCP-BA Integrated Social and Cognitive Psychology	5	Mandatory
MCCB-BSc Medicinal Chemistry and Chemical Biology	5	Mandatory
MDDA-BSc Management, Decisions and Data Analytics	5	Mandatory
MMDA-BSc Mathematics, Modeling, and Data Analytics	5	Mandatory
PHDS-BSc Physics and Data Science	5	Mandatory
BCCB-BSc Biochemistry and Cell Biology	6	Mandatory
CBT-BSc Chemistry and Biotechnology	6	Mandatory
CS-BSc Computer Science	6	Mandatory
ECE-BSc	6	Mandatory

Electrical and Computer Engineering		
ESSMER-BSc Earth Sciences and Sustainable Management of Environmental Resources	6	Mandatory
GEM-BA Global Economics and Management	6	Mandatory
IBA-BA International Business Administration	6	Mandatory
IRPH-BA International Relations: Politics and History	6	Mandatory
ISCP-BA Integrated Social and Cognitive Psychology	6	Mandatory
MCCB-BSc Medicinal Chemistry and Chemical Biology	6	Mandatory
MMDA-BSc Mathematics, Modeling, and Data Analytics	6	Mandatory
PHDS-BSc Physics and Data Science	6	Mandatory
RIS-BSc Robotics and Intelligent Systems	6	Mandatory
S-ACS-BSc Applied Computer Science	6	Mandatory
SDT-BSc Software, Data and Technology	6	Mandatory

Student Workload	
Internship	308
Internship Event	2
Independent Study	32
Interactive Learning	33
Total Hours	**375**

Module Components	**Number**	**Type**	**CP**
Internship	CA-INT-900-0	Internship	15

Module Description

The aims of the internship module are reflection, application, orientation, and development: for students to reflect on their interests, knowledge, skills, their role in society, the relevance of their major subject to society, to apply these skills and this knowledge in real life whilst getting practical experience, to find a professional orientation, and to develop their personality and in their career. This module supports the programs' aims of preparing students for gainful, qualified employment and the development of their personality.

The full-time internship must be related to the students' major area of study and extends lasts a minimum of two consecutive months, normally scheduled just before the 5th semester, with the internship event and submission of the internship report in the 5th semester. Upon approval by the SPC and SCS, the internship may take place at other times, such as before teaching starts in the 3rd semester or after teaching finishes in the 6th semester. The Study Program Coordinator or their faculty delegate approves the intended internship a priori by reviewing the tasks in either the Internship Contract or Internship Confirmation from the respective internship institution or company. Further regulations as set out in the Policies for Bachelor Studies apply.

Students will be gradually prepared for the internship in semesters 1 to 4 through a series of mandatory information sessions, seminars, and career events.

The purpose of the Career Services Information Sessions is to provide all students with basic facts about the job market in general, and especially in Germany and the EU, and services provided by the Student Career Support.

In the Career Skills Seminars, students will learn how to engage in the internship/job search, how to create a competitive application (CV, Cover Letter, etc.), and how to successfully conduct themselves at job interviews and/or assessment centers. In addition to these mandatory sections, students can customize their skill set regarding application challenges and their intended career path in elective seminars.

Finally, during the Career Events organized by the Career Service Center (e.g. the annual Constructor Career Fair and single employer events on and off campus), students will have the opportunity to apply their acquired job market skills in an actual internship/job search situation and to gain their desired internship in a high-quality environment and with excellent employers.

As an alternative to the full-time internship, students can apply for the StartUp Option. Following the same schedule as the full-time internship, the StartUp Option allows students who are particularly interested in founding their own company to focus on the development of their business plan over a period of two consecutive months. Participation in the StartUp Option depends on a successful presentation of the student's initial StartUp idea. This presentation will be held at the beginning of the 4th semester. A jury of faculty members will judge the student's potential to realize their idea and approve the participation of the students. The StartUp Option is supervised by the Faculty StartUp Coordinator. At the end of StartUp Option, students submit their business plan. Further regulations as outlined in the Policies for Bachelor Studies apply.

The concluding Internship Event will be conducted within each study program (or a cluster of related study programs) and will formally conclude the module by providing students the opportunity to present on their internships and reflect on the lessons learned within their major area of study. The purpose of this event is not only to self-reflect on the whole internship process, but also to create a professional network within the academic community, especially by entering the Alumni Network after graduation. It is recommended that all three classes (years) of the same major are present at this event to enable networking between older and younger students and to create an educational environment for younger students to observe the "lessons learned" from the diverse internships of their elder fellow students.

Recommended Knowledge

- Information provided on CSC

- Major specific knowledge and skills

- Please see the section “Knowledge Center” at JobTeaser Career Center for information on Career Skills seminar and workshop offers and for online tutorials on the job market preparation and the application process. For more information, please see https://constructor.university/student-life/career-services

- Participating in the internship events of earlier classes

Usability and Relationship to other Modules

This module applies skills and knowledge acquired in previous modules to a professional environment and provides an opportunity to reflect on their relevance in employment and society. It may lead to thesis topics.

Intended Learning Outcomes

No	Competence	ILO
1	Describe	Describe the scope and the functions of the employment market and personal career development.
2	Apply	Apply professional, personal, and career-related skills for the modern labor market, including self-organization, initiative and responsibility, communication, intercultural sensitivity, team and leadership skills, etc.
3	Manage	Manage their own career orientation processes independently by identifying personal interests, selecting appropriate internship locations or start-up opportunities, conducting interviews, succeeding at pitches or assessment centers, negotiating related employment, managing their funding or support conditions (such as salary, contract, funding, supplies, work space, etc.).
4	Apply	Apply specialist skills and knowledge acquired during their studies to solve problems in a professional environment and reflect on their relevance in employment and society.
5	Justify	Justify professional decisions based on theoretical knowledge and academic methods.
6	Reflect	Reflect on their professional conduct in the context of the expectations of and consequences for employers and their society.
7	Reflect	Reflect on and set their own targets for the further development of their knowledge, skills, interests, and values.
8	Establish	Establish and expand their contacts with potential employers or business partners, and possibly other students and alumni, to build their own professional network to create employment opportunities in the future.
9	Discuss	Discuss observations and reflections in a professional network.

Indicative Literature

Entry Requirements

Prerequisites	At least 15 CP from CORE modules in the major
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Internship	Project Report	3500 words	100	Pass/Fail	1-9

Module Completion Requirements

- **Module Examination:** pass/fail grading.

Module Achievement

None.

6.1.33 Bachelor Thesis and Seminar SDT

Module Name	Bachelor Thesis and Seminar SDT
Module Code	SDT-400
Module ECTS	15
Program Owner	SDT-BSc (Software, Data and Technology)
Module Coordinator	Study Program Chair

Study Semester		
Program	**Semester**	**Status**
SDT-BSc Software, Data and Technology	6	Mandatory

Student Workload	
Independent Study/Laboratory Work	350
Seminar	25
Total Hours	**375**

Module Components	**Number**	**Type**	**CP**
Thesis SDT	SDT-400-S	Thesis	12
Thesis Seminar SDT	SDT-400-T	Seminar	3

Module Description

This module is a mandatory graduation requirement for all undergraduate students to demonstrate their ability to address a problem from their respective major subject independently using academic/scientific methods within a set time frame. Although supervised, this module requires students to be able to work independently and systematically and set their own goals in exchange for the opportunity to explore a topic that excites and interests them personally and that a faculty member is interested in supervising. Within this module, students apply their acquired knowledge about their major discipline and their learned skills and methods for conducting research, ranging from the identification of suitable (short-term) research projects, preparatory literature searches, the realization of discipline-specific research, and the documentation, discussion, interpretation, and communication of research results.

This module consists of two components, an independent thesis and an accompanying seminar. The thesis component must be supervised by a Constructor University faculty member and requires short-term research work, the results of which must be documented in a comprehensive written thesis including an introduction, a justification of the methods, results, a discussion of the results, and a conclusion. The seminar provides students with the opportunity to practice their ability to present, discuss, and justify their and other students' approaches, methods, and results at various stages of their research in order to improve their academic writing, receive and reflect on formative feedback, and therefore grow personally and professionally.

Recommended Knowledge

- Identify an area or a topic of interest and discuss this with your prospective supervisor in a timely manner.

- Create a research proposal including a research plan to ensure timely submission.

- Ensure you possess all required technical research skills or are able to acquire them on time.

- Review the University's Code of Academic Integrity and Guidelines to Ensure Good Academic Practice.

Usability and Relationship to other Modules

This module builds on all previous modules in the undergraduate program. Students apply the knowledge, skills, and competencies they have acquired and practiced during their studies, including research methods and their ability to acquire additional skills independently as and if required.

Intended Learning Outcomes

No	Competence	ILO
1	Plan	Plan and organize advanced learning processes independently.
2	Design	Design and implement appropriate research methods, taking full account of the range of alternative techniques and approaches.
3	Collect	Collect, assess, and interpret relevant information.
4	Draw	Draw scientifically-founded conclusions that consider social, scientific, and ethical factors.
5	Apply	Apply their knowledge and understanding to a context of their choice.
6	Develop	Develop, formulate, and advance solutions to problems and debates within their subject area, and defend these through argument.
7	Discuss	Discuss information, ideas, problems, and solutions with specialists and non-specialists.

Indicative Literature

Entry Requirements

Prerequisites	Students must have taken and successfully passed a total of at least 30 CP from advanced modules, and of those, at least 20 CP from advanced modules in the major.
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Thesis SDT	Thesis	Approx. 6.000 – 8.000 words (15 – 25 pages), excluding front and back matter.	80	45%	All ILOs, mainly 1-6.
Thesis Seminar SDT	Presentation	15-30 minutes	20	45%	The presentation focuses mainly on ILOs 6 and 7, but by nature of these ILOs it also touches on the others.

Module Completion Requirements

- **Module Component Examination:** minimum pass 45% of each assessment.

Module Achievement

None.

7 Constructor Track Modules

7.1 Methods Modules

7.1.1 JVM-Based Programming

Module Name	JVM-Based Programming
Module Code	CTMS-SKI-27
Module ECTS	5
Program Owner	SDT-BSc (Software, Data and Technology)
Module Coordinator	Prof. Dr. Alexander Omelchenko

Study Semester		
Program	**Semester**	**Status**
SDT-BSc Software, Data and Technology	2	Mandatory

Student Workload	
Lecture	17.5
Laboratory	17.5
Independent Study	75
Exam Preparation	15
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
JVM-Based Programming	CTMS-27-A	Lecture	2.5
JVM-Based Programming Lab	CTMS-27-B	Laboratory	2.5

Module Description

This module provides a practical introduction to JVM-based software development with a focus on the Kotlin programming language and the JetBrains tooling ecosystem. The module is positioned as an early “pre-course” to ensure that students can contribute to Kotlin/JVM codebases and become eligible for JetBrains internships after the first year.

The course emphasizes professional programming practices rather than learning syntax for its own sake. Students learn Kotlin fundamentals through hands-on development tasks that reflect common industry workflows: working with standard libraries and collections, using Kotlin’s type system and null-safety, designing small modular programs, and writing tests. Students also learn basic JVM project tooling, including Gradle-based builds, dependency management, and working effectively in IntelliJ IDEA.

The lab component is project-driven. Students implement a small Kotlin/JVM application or library with clear module structure, automated tests, and basic packaging. Where appropriate, examples

connect to later AI-oriented modules (e.g., building reliable API clients, structured data handling, and reusable tooling components), without turning this module into an AI course.

Topics:

- Kotlin fundamentals for production code: types, control flow, functions, and idiomatic style

- null-safety and type system: safe handling of optional data and error-prone cases

- collections and generics: idiomatic data processing and reusable abstractions

- object-oriented and modular design in Kotlin: classes, interfaces, data classes, encapsulation

- functional constructs in Kotlin: lambdas, higher-order functions, and expressive APIs

- exceptions and error handling; basic debugging techniques and tooling

- basic I/O and data formats: files, JSON (conceptual and practical use cases)

- introduction to Kotlin coroutines (conceptual level): async tasks and structured concurrency basics (lightweight)

- JVM project tooling: Gradle, dependencies, project structure, build and run configurations

- testing Kotlin code: unit tests, test structure, and practical debugging of failures

- collaborative workflow: Git-based development, code review hygiene, and simple CI-style quality gates (conceptual)

Rationale for assessment: The written examination assesses Kotlin/JVM language and tooling concepts, while the Program Code component assesses practical implementation, testing, build configuration, and project structure.

Recommended Knowledge

- Completion of Industrial Programming with Python and Practical Minimum (Git/Docker/CLI) is recommended.

- Students are expected to be comfortable with basic programming concepts (functions, loops, data structures) and to have a working development environment.

- Familiarity with basic data structures and algorithmic thinking (from Core Algorithms and Discrete Mathematics) is helpful.

Usability and Relationship to other Modules

This module provides Kotlin/JVM competence early in the program to support JetBrains internship readiness and to align the curriculum with the JetBrains ecosystem. The module also prepares students for later electives and implementation choices in subsequent modules, where Kotlin/JVM may be used for building tooling components, services, or integrations.

In particular, the module supports advanced work in Advanced Development in JVM Languages (elective in later semesters) and can be used as an implementation foundation in modules related to developer tooling and AI-enabled systems (e.g., projects involving code intelligence pipelines, workflow automation, and agent/tool integrations).

Intended Learning Outcomes

No	Competence	ILO
1	Write	Write, understand, and debug Kotlin code effectively using modern JVM tooling and an IDE-based workflow.
2	Use	Use Kotlin's type system and null-safety features to write robust and maintainable programs.
3	Implement	Implement small to medium programming tasks using Kotlin collections, generics, and idiomatic language constructs.
4	Design	Design modular Kotlin/JVM programs with clear responsibilities, interfaces, and clean project structure.
5	Build	Set up and manage a Kotlin/JVM project using Gradle, including dependencies, build configuration, and packaging basics.
6	Test	Develop unit tests for Kotlin code and apply basic quality practices to ensure correctness and maintainability.
7	Apply	Apply basic concurrency concepts using Kotlin coroutines at an introductory level (async tasks, cancellation awareness, error handling basics).
8	Work	Work effectively in a small team using Git-based workflows and code review practices on a Kotlin codebase.

Indicative Literature

- Dmitry Jemerov, Svetlana Isakova — Kotlin in Action (selected chapters)
- Gradle documentation (basic build and dependency management workflows)
- Marcin Moskala — Effective Kotlin (selected chapters)
- Official Kotlin documentation and guides (language basics, collections, coroutines)
- Venkat Subramaniam — Programming Kotlin (selected chapters)

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
JVM-Based Programming	Written Examination	90 minutes	50	45%	All theoretical ILOs

JVM-Based Programming Lab	Program Code		50	45%	All practical ILOs

Module Completion Requirements

- **Module Component Examination:** minimum pass 45% of each assessment.

Module Achievement

None.

7.1.2 Basics of Discrete Mathematics

Module Name	Basics of Discrete Mathematics
Module Code	CTMS-MAT-26
Module ECTS	5
Program Owner	SDT-BSc (Software, Data and Technology)
Module Coordinator	Prof. Dr. Alexander Omelchenko

Study Semester		
Program	**Semester**	**Status**
SDT-BSc Software, Data and Technology	1	Mandatory

Student Workload	
Lecture	17.5
Tutorial	17.5
Independent Study	75
Exam Preparation	15
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Basics of Discrete Mathematics	CTMS-26-A	Lecture	2.5
Basics of Discrete Mathematics Tutorial	CTMS-26-B	Tutorial	2.5

Module Description

This module provides a foundational introduction to discrete mathematics tailored to computer science and modern AI-oriented software development. It equips students with core discrete structures and reasoning techniques required for algorithms, reliable programming, and principled reasoning about uncertainty.

Topics:

- modelling problems using discrete structures such as graphs and trees,

- designing and analyzing simple graph algorithms (BFS/DFS) as a bridge to algorithmic thinking,

- using logic, invariants, and proof techniques to argue correctness of programs, and

- understanding combinatorial growth and asymptotic behavior (why exponential complexity is infeasible),

- basic probability concepts (probability, independence, expectation) for randomized methods and uncertainty in evaluation.

Recommended Knowledge

No formal prerequisites are required. Students should be comfortable with basic high-school algebra and functions. Basic familiarity with programming (variables, loops, functions) is helpful for optional exploratory exercises, but is not required.

Usability and Relationship to other Modules

This module provides essential groundwork for Core Algorithms and Data Structures, Intro to AI & Agentic Workflows, and later systems and software engineering modules where reasoning with invariants and correctness arguments is required.

The concepts introduced here form a shared mathematical and conceptual language that is reused throughout the SDT curriculum.

Intended Learning Outcomes

No	Competence	ILO
1	Explain	Explain and use propositional logic and Boolean algebra to formalize conditions, constraints, and simple specifications.
2	Apply	Apply standard proof techniques (direct proof, contradiction, induction) and use loop invariants to argue correctness of simple algorithms.
3	Analyze	Analyze the growth of functions and use asymptotic notation (O, Ω, Θ) to compare runtimes and recognize combinatorial explosion.
4	Solve	Solve basic counting problems using sum and product rules, permutations and combinations, binomial coefficients, and the pigeonhole principle.
5	Model	Model problems using graphs and trees and choose appropriate representations (adjacency lists or matrices) for a given task.
6	Design	Design, trace, and analyze BFS and DFS, including their time and space complexity, and explain typical use cases.
7	Compute	Compute basic probabilities and expectations for discrete random variables and interpret results in the context of randomized methods and uncertainty in evaluation.
8	Communicate	Communicate solutions clearly and rigorously, including definitions, intermediate steps, and justification of claims.

Indicative Literature

- Kenneth H. Rosen — Discrete Mathematics and Its Applications (latest edition)
- Miklós Bóna — A Walk Through Combinatorics (selected chapters)
- Selected lecture notes and exercise sheets provided by the instructor
- Sheldon Ross — A First Course in Probability (selected sections)
- Thomas H. Cormen et al. — Introduction to Algorithms (CLRS), selected sections

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Basics of Discrete Mathematics	Written Examination	60 minutes	50	45%	All theoretical ILOs
Basics of Discrete Mathematics Tutorial	Assignments		50	45%	All practical ILOs

Module Completion Requirements

- **Module Component Examination:** minimum pass 45% of each assessment.

Module Achievement

Students must pass at least 50% of the written tutorial assignments / problem sets. This module achievement is required to ensure continuous engagement with mathematical problem-solving and preparation for the written examination. Late or missed assignments may be compensated by extra tasks during the semester.

7.1.3 Probability and Random Processes

Module Name	Probability and Random Processes
Module Code	CTMS-MAT-12
Module ECTS	5
Program Owner	CT (CONSTRUCTOR Track Area)
Module Coordinator	Prof. Dr. Keivan Mallahi Karai

Study Semester		
Program	**Semester**	**Status**
CS-BSc Computer Science	3	Mandatory
ECE-BSc Electrical and Computer Engineering	3	Mandatory
F-ACS-BSc Applied Computer Science	3	Mandatory
MMDA-BSc Mathematics, Modeling, and Data Analytics	3	Mandatory
PHDS-BSc Physics and Data Science	3	Mandatory
RIS-BSc Robotics and Intelligent Systems	3	Mandatory
SDT-BSc Software, Data and Technology	3	Mandatory
S-ACS-BSc Applied Computer Science	4	Mandatory

Student Workload	
Independent Study	90
Lecture	35
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Probability and random processes	CTMS-12	Lecture	5

Module Description

This module aims to provide a basic knowledge of probability theory and random processes suitable for students in engineering, Computer Science, and Mathematics. The module provides students with basic skills needed for formulating real-world problems dealing with randomness and probability in mathematical language, and methods for applying a toolkit to solve these problems. Mathematical rigor is used where appropriate. A more advanced treatment of the subject is deferred to the third-year module Stochastic Processes.

The lecture comprises the following topics:

- Brief review of number systems, elementary functions, and their graphs

- Outcomes, events and sample space

- Combinatorial probability

- Conditional probability and Bayes' formula

- Binomials and Poisson-Approximation

- Random Variables, distribution and density functions

- Independence of random variables

- Conditional Distributions and Densities

- Transformation of random variables

- Joint distribution of random variables and their transformations

- Expected Values and Moments, Covariance

- High dimensional probability: Chebyshev and Chernoff bounds

- Moment-Generating Functions and Characteristic Functions

- The Central limit theorem

- Random Vectors and Moments, Covariance matrix, Decorrelation

- Multivariate normal distribution. Markov chains, stationary distributions.

Recommended Knowledge

- Review all of the first-year calculus and linear algebra modules as indicated in "Entry Requirements - Knowledge, Ability, or Skills" above.

- Knowledge of calculus at the level of a first-year calculus module (differentiation, integration with one and several variables, trigonometric functions, logarithms and exponential functions).

- Knowledge of linear algebra at the level of a first-year university module (eigenvalues and eigenvectors, diagonalization of matrices).

- Some familiarity with elementary probability theory at the high school level.

Usability and Relationship to other Modules

Students taking this module are expected to be familiar with basic tools from calculus and linear algebra.

Intended Learning Outcomes

No	Competence	ILO
1	Command	Command the methods described in the content section of this module description to the extent that they can solve standard text-book problems reliably and with confidence.

2	Recognize	Recognize the probabilistic structures in an unfamiliar context and translate them into a mathematical problem statement.
3	Recognize	Recognize common mathematical terminology used in textbooks and research papers in the quantitative sciences, engineering, and mathematics to the extent that they fall into the content categories covered in this module.

Indicative Literature

- J. Hwang and J.K. Blitzstein (2019). Introduction to Probability, second edition. London: Chapman & Hall.
- S. Ghahramani. Fundamentals of Probability with Stochastic Processes, fourth edition. Upper Saddle River: Prentice Hall.

Entry Requirements

Prerequisites	CTMS-MAT-22 Matrix Algebra and Advanced Calculus I CTMS-MAT-23 Matrix Algebra and Advanced Calculus II CTMS-MAT-24 Elements of Linear Algebra CTMS-MAT-25 Elements of Calculus
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Probability and random processes	Written Examination	120 Minutes	100	45%	1-3

Module Completion Requirements

- **Module Examination:** minimum for pass- 45%.

Module Achievement

None.

7.1.4 Statistics and Data Analytics

Module Name	Statistics and Data Analytics
Module Code	CTMS-MET-21
Module ECTS	5
Program Owner	CT (CONSTRUCTOR Track Area)
Module Coordinator	Dr. Ivan Ovsyannikov

Study Semester		
Program	**Semester**	**Status**
CS-BSc Computer Science	4	Mandatory Elective
MMDA-BSc Mathematics, Modeling, and Data Analytics	4	Mandatory
PHDS-BSc Physics and Data Science	4	Mandatory
SDT-BSc Software, Data and Technology	4	Mandatory

Student Workload	
Independent Study	105
Lecture	35
Total Hours	**140**

Module Components	**Number**	**Type**	**CP**
Statistics and Data Analytics	CTMS-21	Lecture	5

Module Description

The aim of this module is to introduce students to basic ideas and methods used for analysing large and complex datasets. While the first modern statistical toolkits date back to the beginning of the twentieth century, the advent of the computer age and the availability of fast computations has led to dramatic changes in the field.

Statistical models have found applications in many areas ranging from business and healthcare to astrophysics and speech recognition. Such models are used to make predictions, draw inferences and support policy decisions in all these areas.

This module draws on students' knowledge from the module Probability and Random Processes to help them build and analyze statistical models, ranging in their degree of sophistication from basis to more advanced ones, and apply them to real-world situations.

The module will cover the following topics:

- Classical statistics: descriptive and inferential modes, parameter estimation and hypothesis testing.

- Linear regressions, multiple linear regressions

- Classification: logistic regression, generative models for classification

- Resampling methods, bootstrap

- Non-linear models, splines

- Support vector machines

- Basic ideas of deep learning

Recommended Knowledge

- Good command of basic probability

- Recap Probability and Random Processes

Usability and Relationship to other Modules

- This module is part of the core education in Mathematics, Modeling and Data Analytics and Physics and Data Science.

- It is also valuable for students in Computer Science, RIS, and ECE, either as part of a minor in Mathematics, or as an elective module.

Intended Learning Outcomes

No	Competence	ILO
1	Formulate	Formulate statistical models for real world problems.
2	Describe	Describe statistical methods for analyzing real world problems.
3	Explain	Explain the importance of linear and non-linear models.
4	Recognize	Recognize different solution methods for modeling problems.
5	Illustrate	Illustrate the use of regressions, resampling, support vector machines and other statistical tools to describe phenomena in the real world.
6	Describe	Describe basic ideas of deep learning.

Indicative Literature

- James, Witten, Hastie, Tibshirani. An introduction to Statistical learning, second edition.

Entry Requirements

Prerequisites	CTMS-MAT-12 Probability and Random Processes
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Statistics and Data Analytics	Written Examination	120 Minutes	100	45%	1-6

Module Completion Requirements

- **Module Examination:** minimum for pass- 45%.

Module Achievement

None.

7.2 New Skills Modules

7.2.1 Logic (perspective I)

Module Name	Logic (perspective I)
Module Code	CTNS-NSK-01
Module ECTS	2.5
Program Owner	CT (CONSTRUCTOR Track Area)
Module Coordinator	Dr. Marina Christodoulou

Study Semester		
Program	**Semester**	**Status**
BCCB-BSc Biochemistry and Cell Biology	3	Mandatory Elective
CBT-BSc Chemistry and Biotechnology	3	Mandatory Elective
CS-BSc Computer Science	3	Mandatory Elective
ECE-BSc Electrical and Computer Engineering	3	Mandatory Elective
ESSMER-BSc Earth Sciences and Sustainable Management of Environmental Resources	3	Mandatory Elective
F-ACS-BSc Applied Computer Science	3	Mandatory Elective
GEM-BA Global Economics and Management	3	Mandatory Elective
IBA-BA International Business Administration	3	Mandatory Elective
IEM-BSc Industrial Engineering & Management	3	Mandatory Elective
IEM-Online-BSc Industrial Engineering & Management (Online)	3	Mandatory Elective
IRPH-BA International Relations: Politics and History	3	Mandatory Elective
ISCP-BA Integrated Social and Cognitive Psychology	3	Mandatory Elective
MCCB-BSc Medicinal Chemistry and Chemical Biology	3	Mandatory Elective
MDDA-BSc Management, Decisions and Data Analytics	3	Mandatory Elective
MMDA-BSc Mathematics, Modeling, and Data Analytics	3	Mandatory Elective
PHDS-BSc Physics and Data Science	3	Mandatory Elective

RIS-BSc Robotics and Intelligent Systems	3	Mandatory Elective
SDT-BSc Software, Data and Technology	3	Mandatory Elective
IRPH-BA International Relations: Politics and History	4	Mandatory Elective
ISCP-BA Integrated Social and Cognitive Psychology	4	Mandatory Elective
S-ACS-BSc Applied Computer Science	4	Mandatory Elective

Student Workload	
Tutorial	17.5
Independent Study	27.5
Online Lecture	17.5
Total Hours	**62.5**

Module Components	**Number**	**Type**	**CP**
Logic (perspective I)	CTNS-01	Lecture (Online)	2.5

Module Description

Suppose a friend asks you to help solve a complicated problem? Where do you begin? Arguably, the first and most difficult task you face is to figure out what the heart of the problem actually is. In doing that you will look for structural similarities between the problem posed and other problems that arise in different fields that others may have addressed successfully. Those similarities may point you to a pathway for resolving the problem you have been asked to solve. But it is not enough to look for structural similarities. Sometimes relying on similarities may even be misleading. Once you've settled tentatively on what you take to be the heart of the matter, you will naturally look for materials, whether evidence or arguments, that you believe is relevant to its potential solution. But the evidence you investigate of course depends on your formulation of the problem, and your formulation of the problem likely depends on the tools you have available - including potential sources of evidence and argumentation. You cannot ignore this interactivity, but you can't allow yourself to be hamstrung entirely by it. But there is more. The problem itself may be too big to be manageable all at once, so you will have to explore whether it can be broken into manageable parts and if the information you have bears on all or only some of those parts. And later you will face the problem of whether the solutions to the particular sub problems can be put together coherently to solve the entire problem taken as a whole.

What you are doing is what we call engaging in computational thinking. There are several elements of computational thinking illustrated above. These include: Decomposition (breaking the larger problem down into smaller ones); Pattern recognition (identifying structural similarities); Abstraction (ignoring irrelevant particulars of the problem): and Creating Algorithms), problem-solving formulas.

But even more basic to what you are doing is the process of drawing inferences from the material you have. After all, how else are you going to create a problem-solving formula, if you draw incorrect

inferences about what information has shown and what, if anything follows logically from it. What you must do is apply the rules of logic to the information to draw inferences that are warranted.

We distinguish between informal and formal systems of logic, both of which are designed to indicate fallacies as well as warranted inferences. If I argue for a conclusion by appealing to my physical ability to coerce you, I prove nothing about the truth of what I claim. If anything, by doing so I display my lack of confidence in my argument. Or if the best I can do is berate you for your skepticism, I have done little more than offer an ad hominem instead of an argument. Our focus will be on formal systems of logic, since they are at the heart of both scientific argumentation and computer developed algorithms. There are in fact many different kinds of logic and all figure to varying degrees in scientific inquiry. There are inductive types of logic, which purport to formalize the relationship between premises that if true offer evidence on behalf of a conclusion and the conclusion and are represented as claims about the extent to which the conclusion is confirmed by the premises. There are deductive types of logic, which introduce a different relationship between premise and conclusion. These variations of logic consist in rules that if followed entail that if the premises are true then the conclusion too must be true.

There are also modal types of logic which are applied specifically to the concepts of necessity and possibility, and thus to the relationship among sentences that include either or both those terms. And there is also what are called deontic logic, a modification of logic that purport to show that there are rules of inference that allow us to infer what we ought to do from facts about the circumstances in which we find ourselves. In the natural and social sciences most of the emphasis has been placed on inductive logic, whereas in math it is placed on deductive logic, and in modern physics there is an increasing interest in the concepts of possibility and necessity and thus in modal logic. The humanities, especially normative discussions in philosophy and literature are the province of deontic logic.

This module will also take students through the central aspects of computational thinking, as it is related to logic; it will introduce the central concepts in each, their relationship to one another and begin to provide the conceptual apparatus and practical skills for scientific inquiry and research.

This 2.5 CP module supports Academic Excellence by providing essential contextual knowledge that underpins the program's qualification aims, while appropriately limiting academic depth relative to the core technical competencies of the field

Intended Learning Outcomes

No	Competence	ILO
1	Apply	Apply the various principles of logic and expand them to computational thinking.
2	Understand	Understand the way in which logical processes in humans and in computers are similar and different at the same time.
3	Apply	Apply the basic rules of first-order deductive logic and employ them rules in the context of creating a scientific or social scientific study and argument.
4	Employ	Employ those rules in the context of creating a scientific or social scientific study and argument.

Indicative Literature

- Frege, Gottlob (1879), Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens [Translation: A Formal Language for Pure Thought Modeled on that of Arithmetic],Halle an der Salle: Verlag von Louis Nebert.
- Gödel, Kurt (1986), Russels mathematische Logik. In: Alfred North Whitehead, Bertrand Russell: Principia Mathematica. Vorwort, S. V–XXXIV. Suhrkamp.
- Kubica, Jeremy. Computational fairy tales. Jeremy Kubica, 2012.
- Leeds, Stephen. "George Boolos and Richard Jeffrey. Computability and logic. Cambridge University Press, New York and London1974, x+ 262 pp." The Journal of Symbolic Logic 42.4 (1977): 585-586.
- McCarthy, Timothy. "Richard Jeffrey. Formal logic: Its scope and limits. of XXXVIII 646. McGraw-Hill Book Company, New York etc. 1981, xvi+ 198 pp." The Journal of Symbolic Logic 49.4 (1984): 1408-1409.

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Logic (perspective I)	Written Examination	60 Minutes	100	45%	All

Module Completion Requirements

- **Module Examination:** minimum for pass- 45%.

Module Achievement

None.

7.2.2 Logic (perspective II)

Module Name	Logic (perspective II)
Module Code	CTNS-NSK-02
Module ECTS	2.5
Program Owner	CT (CONSTRUCTOR Track Area)
Module Coordinator	Prof. Dr. Jules Coleman

Study Semester		
Program	**Semester**	**Status**
BCCB-BSc Biochemistry and Cell Biology	3	Mandatory Elective
CBT-BSc Chemistry and Biotechnology	3	Mandatory Elective
CS-BSc Computer Science	3	Mandatory Elective
ECE-BSc Electrical and Computer Engineering	3	Mandatory Elective
ESSMER-BSc Earth Sciences and Sustainable Management of Environmental Resources	3	Mandatory Elective
F-ACS-BSc Applied Computer Science	3	Mandatory Elective
GEM-BA Global Economics and Management	3	Mandatory Elective
IBA-BA International Business Administration	3	Mandatory Elective
IBA-Online-BA International Business Administration (Online)	3	Mandatory Elective
IEM-BSc Industrial Engineering & Management	3	Mandatory Elective
IEM-Online-BSc Industrial Engineering & Management (Online)	3	Mandatory Elective
IRPH-BA International Relations: Politics and History	3	Mandatory Elective
ISCP-BA Integrated Social and Cognitive Psychology	3	Mandatory Elective
MCCB-BSc Medicinal Chemistry and Chemical Biology	3	Mandatory Elective
MDDA-BSc Management, Decisions and Data Analytics	3	Mandatory Elective
MMDA-BSc Mathematics, Modeling, and Data Analytics	3	Mandatory Elective
PHDS-BSc Physics and Data Science	3	Mandatory Elective

RIS-BSc Robotics and Intelligent Systems	3	Mandatory Elective
SDT-BSc Software, Data and Technology	3	Mandatory Elective
IRPH-BA International Relations: Politics and History	4	Mandatory Elective
ISCP-BA Integrated Social and Cognitive Psychology	4	Mandatory Elective
S-ACS-BSc Applied Computer Science	4	Mandatory Elective

Student Workload	
Independent Study	45
Online Lecture	17.5
Total Hours	**62.5**

Module Components	**Number**	**Type**	**CP**
Logic (perspective II)	CTNS-02	Lecture (Online)	2.5

Module Description

The focus of this module is on formal systems of logic, since they are at the heart of both scientific argumentation and computer developed algorithms. There are in fact many kinds of logic and all figure to varying degrees in scientific inquiry. There are inductive types of logic, which purport to formalize the relationship between premises that if true offer evidence on behalf of a conclusion and the conclusion and are represented as claims about the extent to which the conclusion is confirmed by the premises. There are deductive types of logic, which introduce a different relationship between premise and conclusion. These variations of logic consist in rules that if followed entail that if the premises are true then the conclusion too must be true.

This module introduces logics that go beyond traditional deductive propositional logic and predicate logic and as such it is aimed at students who are already familiar with basics of traditional formal logic. The aim of the module is to provide an overview of alternative logics and to develop a sensitivity that there are many different logics that can provide effective tools for solving problems in specific application domains.

The module first reviews the principles of a traditional logic and then introduces many-valued logics that distinguish more than two truth values, for example true, false, and unknown. Fuzzy logic extends traditional logic by replacing truth values with real numbers in the range 0 to 1 that are expressing how strong the believe into a proposition is. Modal logics introduce modal operators expressing whether a proposition is necessary or possible. Temporal logics deal with propositions that are qualified by time. One can view temporal logics as a form of modal logics where propositions are qualified by time constraints. Interval temporal logic provides a way to reason about time intervals in which propositions are true.

The module will also investigate the application of logic frameworks to specific classes of problems. For example, a special subset of predicate logic, based on so-called Horn clauses, forms the basis of

logic programming languages such as Prolog. Description logics, which are usually decidable logics, are used to model relationships and they have applications in the semantic web, which enables search engines to reason about resources present on the Internet.

This 2.5 CP module supports Academic Excellence by providing essential contextual knowledge that underpins the program's qualification aims, while appropriately limiting academic depth relative to the core technical competencies of the field

Intended Learning Outcomes

No	Competence	ILO
1	Apply	Apply the various principles of logic.
2	Explain	Explain practical relevance of non-standard logic.
3	Describe	Describe how many-valued logic extends basic predicate logic.
4	Apply	Apply basic rules of fuzzy logic to calculate partial truth values.
5	Sketch	Sketch basic rules of temporal logic.
6	Implement	Implement predicates in a logic programming language.
7	Prove	Prove some simple non-standard logic theorems.

Indicative Literature

- Baader, Franz. "The Description Logic Handbook: Theory Implementation and Applications", Cambridge University Press, 2nd edition, May 2010.
- Bergmann, Merry. "An Introduction to Many-Valued and Fuzzy Logic: Semantics, Algebras, and Derivation Systems", Cambridge University Press, April 2008.
- Fisher, Michael. "An Introduction to Practical Formal Methods Using Temporal Logic", Wiley, Juli 2011.
- Sterling, Leon S., Ehud Y. Shapiro, Ehud Y. "The Art of Prolog", 2nd edition, MIT Press, March 1994.

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Logic (perspective II)	Written Examination	60 Minutes	100	45%	All

Module Completion Requirements

- **Module Examination:** minimum for pass- 45%.

Module Achievement

None.

7.2.3 Causation and Correlation (perspective I)

Module Name	Causation and Correlation (perspective I)
Module Code	CTNS-NSK-03
Module ECTS	2.5
Program Owner	CT (CONSTRUCTOR Track Area)
Module Coordinator	Dr. Marina Christodoulou

Study Semester		
Program	**Semester**	**Status**
BCCB-BSc Biochemistry and Cell Biology	4	Mandatory Elective
CBT-BSc Chemistry and Biotechnology	4	Mandatory Elective
CS-BSc Computer Science	4	Mandatory Elective
ECE-BSc Electrical and Computer Engineering	4	Mandatory Elective
ESSMER-BSc Earth Sciences and Sustainable Management of Environmental Resources	4	Mandatory Elective
F-ACS-BSc Applied Computer Science	4	Mandatory Elective
GEM-BA Global Economics and Management	4	Mandatory Elective
IBA-BA International Business Administration	4	Mandatory Elective
IBA-Online-BA International Business Administration (Online)	4	Mandatory Elective
IEM-BSc Industrial Engineering & Management	4	Mandatory Elective
IEM-Online-BSc Industrial Engineering & Management (Online)	4	Mandatory Elective
IRPH-BA International Relations: Politics and History	4	Mandatory Elective
ISCP-BA Integrated Social and Cognitive Psychology	4	Mandatory Elective
MCCB-BSc Medicinal Chemistry and Chemical Biology	4	Mandatory Elective
MDDA-BSc Management, Decisions and Data Analytics	4	Mandatory Elective
MMDA-BSc Mathematics, Modeling, and Data Analytics	4	Mandatory Elective
PHDS-BSc Physics and Data Science	4	Mandatory Elective

RIS-BSc Robotics and Intelligent Systems	4	Mandatory Elective
SDT-BSc Software, Data and Technology	4	Mandatory Elective

Student Workload	
Tutorial	17.5
Independent Study	27.5
Online Lecture	17.5
Total Hours	**62.5**

Module Components	**Number**	**Type**	**CP**
Causation and Correlation	CTNS-03	Lecture (Online)	2.5

Module Description

In many ways, life is a journey. And also, as in other journeys, our success or failure depends not only on our personal traits and character, our physical and mental health, but also on the accuracy of our map. We need to know what the world we are navigating is actually like, the how, why and the what of what makes it work the way it does. The natural sciences provide the most important tool we have developed to learn how the world works and why it works the way it does. The social sciences provide the most advanced tools we have to learn how we and other human beings, similar in most ways, different in many others, act and react and what makes them do what they do. In order for our maps to be useful, they must be accurate and correctly reflect the way the natural and social worlds work and why they work as they do.

The natural sciences and social sciences are blessed with enormous amounts of data. In this way, history and the present are gifts to us. To understand how and why the world works the way it does requires that we are able to offer an explanation of it. The data supports a number of possible explanations of it. How are we to choose among potential explanations? Explanations, if sound, will enable us to make reliable predictions about what the future will be like, and also to identify many possibilities that may unfold in the future. But there are differences not just in the degree of confidence we have in our predictions, but in whether some of them are necessary future states or whether all of them are merely possibilities? Thus, there are three related activities at the core of scientific inquiry: understanding where we are now and how we got here (historical); knowing what to expect going forward (prediction); and exploring how we can change the paths we are on (creativity).

At the heart of these activities are certain fundamental concepts, all of which are related to the scientific quest to uncover immutable and unchanging laws of nature. Laws of nature are thought to reflect a causal nexus between a previous event and a future one. There are also true statements that reflect universal or nearly universal connections between events past and present that are not laws of nature because the relationship they express is that of a correlation between events. A working thermostat accurately allows us to determine or even to predict the temperature in the room in which it is located, but it does not explain why the room has the temperature it has. What then is the core difference between causal relationships and correlations? At the same time, we all recognize that given where we are now there are many possible futures for each of us, and even had our lives gone

just the slightest bit differently than they have, our present state could well have been very different than it is. The relationship between possible pathways between events that have not materialized but could have is expressed through the idea of counterfactual.

Creating accurate roadmaps, forming expectations we can rely on, making the world a more verdant and attractive place requires us to understand the concepts of causation, correlation, counterfactual explanation, prediction, necessity, possibility, law of nature and universal generalization. This course is designed precisely to provide the conceptual tools and intellectual skills to implement those concepts in our future readings and research and ultimately in our experimental investigations, and to employ those tools in various disciplines.

This 2.5 CP module supports Academic Excellence by providing essential contextual knowledge that underpins the program's qualification aims, while appropriately limiting academic depth relative to the core technical competencies of the field

Intended Learning Outcomes

No	Competence	ILO
1	Formulate	Formulate testable hypotheses that are designed to reveal causal connections and those designed to reveal interesting, important and useful correlations.
2	Distinguish	Distinguish scientifically interesting correlations from unimportant ones.
3	Apply	Apply critical thinking skills to evaluate information.
4	Understand	Understand when and why inquiry into unrealized possibility is important and relevant.

Indicative Literature

- Goodman, Nelson. Fact, fiction, and forecast. Harvard University Press, 1983.
- Quine Willard, Van Orman, and Joseph Silbert Ullian. The web of belief. Vol 2. New York: Random house, 1978.
- Thomas S. Kuhn: The Structure of Scientific Revolutions. Nelson, fourth edition, 2012.

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs

Causation and Correlation	Written Examination	60 Minutes	100	45%	1-4

Module Completion Requirements

- **Module Examination:** minimum for pass- 45%.

Module Achievement

None.

7.2.4 Causation and Correlation (perspective II)

Module Name	Causation and Correlation (perspective II)
Module Code	CTNS-NSK-04
Module ECTS	2.5
Program Owner	CT (CONSTRUCTOR Track Area)
Module Coordinator	Dr. Eoin Ryan Dr. Irina Chiaburu Prof. Dr. Keivan Mallahi Karai

Study Semester		
Program	**Semester**	**Status**
ISCP-BA Integrated Social and Cognitive Psychology	3	Mandatory Elective
S-ACS-BSc Applied Computer Science	3	Mandatory Elective
BCCB-BSc Biochemistry and Cell Biology	4	Mandatory Elective
CBT-BSc Chemistry and Biotechnology	4	Mandatory Elective
CS-BSc Computer Science	4	Mandatory Elective
ECE-BSc Electrical and Computer Engineering	4	Mandatory Elective
ESSMER-BSc Earth Sciences and Sustainable Management of Environmental Resources	4	Mandatory Elective
F-ACS-BSc Applied Computer Science	4	Mandatory Elective
GEM-BA Global Economics and Management	4	Mandatory Elective
IBA-BA International Business Administration	4	Mandatory Elective
IBA-Online-BA International Business Administration (Online)	4	Mandatory Elective
IEM-BSc Industrial Engineering & Management	4	Mandatory Elective
IEM-Online-BSc Industrial Engineering & Management (Online)	4	Mandatory Elective
IRPH-BA International Relations: Politics and History	4	Mandatory Elective
ISCP-BA Integrated Social and Cognitive Psychology	4	Mandatory Elective
MCCB-BSc Medicinal Chemistry and Chemical Biology	4	Mandatory Elective

MDDA-BSc Management, Decisions and Data Analytics	4	Mandatory Elective
MMDA-BSc Mathematics, Modeling, and Data Analytics	4	Mandatory Elective
PHDS-BSc Physics and Data Science	4	Mandatory Elective
RIS-BSc Robotics and Intelligent Systems	4	Mandatory Elective
SDT-BSc Software, Data and Technology	4	Mandatory Elective
MMDA-BSc Mathematics, Modeling, and Data Analytics	5	Mandatory Elective

Student Workload	
Independent Study	45
Online Lecture	17.5
Total Hours	**62.5**

Module Components	**Number**	**Type**	**CP**
Causation and Correlation (perspective II)	CTNS-04	Lecture (Online)	2.5

Module Description

Causality or causation is a surprisingly difficult concept to understand. David Hume famously noted that causality is a concept that our science and philosophy cannot do without, but it is equally a concept that our science and philosophy cannot describe. Since Hume, the problem of cause has not gone away, and sometimes seems to get even worse (e.g., quantum mechanics confusing previous notions of causality). Yet, ways of doing science that lessen our need to explicitly use causality have become very effective (e.g., huge developments in statistics). Nevertheless, it still seems that the concept of causality is at the core of explaining how the world works, across fields as diverse as physics, medicine, logistics, the law, sociology, and history - and ordinary daily life - through all of which, explanations and predictions in terms of cause and effect remain intuitively central.

Causality remains a thorny problem but, in recent decades, significant progress has occurred, particularly in work by or inspired by Judea Pearl. This work incorporates many 20th century developments, including statistical methods - but with a reemphasis on finding the why, or the cause, behind statistical correlations -, progress in understanding the logic, semantics and metaphysics of conditionals and counterfactuals, developments based on insights from the likes of philosopher Hans Reichenbach or biological statistician Sewall Wright into causal precedence and path analysis, and much more. The result is a new toolkit to identify causes and build causal explanations. Yet even as we get better at identifying causes, this raises new (or old) questions about causality, including metaphysical questions about the nature of causes (and effects, events, objects, etc), but also questions about what we really use causality for (understanding the world as it is or just to glean predictive control of specific outcomes), about how causality is used differently in different fields and

activities (is cause in physics the same as that in history?), and about how other crucial concepts relate to our concept of cause (space and time seem to be related to causality, but so do concepts of legal and moral responsibility).

This course will introduce students to the mathematical formalism derived from Pearl's work, based on directed acyclic graphs and probability theory. Building upon previous work by Reichenbach and Wright, Pearl defines a "a calculus of interventions" of "do-calculus" for talking about interventions and their relation to causation and counterfactuals. This model has been applied in various areas ranging from econometrics to statistics, where acquiring knowledge about causality is of great importance.

At the same time, the course will not forget some of the metaphysical and epistemological issues around cause, so that students can better critically evaluate putative causal explanations in their full context. Abstractly, such issues involve some of the same philosophical questions Hume already asked, but more practically, it is important to see how metaphysical and epistemological debates surrounding the notion of cause affect scientific practice, and equally if not more importantly, how scientific practice pushes the limits of theory. This course will look at various ways in which empirical data can be transformed into explanations and theories, including the variance approach to causality (characteristic of the positivistic quantitative paradigm), and the process theory of causality (associated with qualitative methodology). Examples and case studies will be relevant for students of the social sciences but also students of the natural/physical world as well.

This 2.5 CP module supports Academic Excellence by providing essential contextual knowledge that underpins the program's qualification aims, while appropriately limiting academic depth relative to the core technical competencies of the field

Recommended Knowledge

Basic probability theory

Intended Learning Outcomes

No	Competence	ILO
1	Have	Have a clear understanding of the history of causal thinking.
2	Form	Form a critical understanding of the key debates and controversies surrounding the idea of causality.
3	Recognize	Recognize and apply probabilistic causal models.
4	Explain	Explain how understanding of causality differs among different disciplines.
5	Demonstrate	Demonstrate how theoretical thinking about causality has shaped scientific practices.

Indicative Literature

- Ilari, Phyllis McKay and Federica Russo. Causality: Philosophical Theory Meets Scientific Practice. Oxford University Press 2014.
- Paul, L. A. and Ned Hall. Causation: A User's Guide. Oxford University Press 2013.

- Pearl, Judea, Glymour Madelyn and Jewell, Nicolas. Causal Inference in Statistics: A Primer. Wiley 2016.
- Pearl, Judea. Causality: Models, Reasoning and Inference. Cambridge University Press 2009.

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Causation and Correlation (perspective II)	Written Examination	60 Minutes	100	45%	1-5

Module Completion Requirements

- **Module Examination:** minimum for pass- 45%.

Module Achievement

None.

7.2.5 Models, Matrices and Theories of Complex Systems

Module Name	Models, Matrices and Theories of Complex Systems
Module Code	CTNS-NSK-05
Module ECTS	5
Program Owner	CT (CONSTRUCTOR Track Area)
Module Coordinator	Prof. Dr. Marc-Thorsten Hütt

Study Semester		
Program	**Semester**	**Status**
BCCB-BSc Biochemistry and Cell Biology	5	Mandatory Elective
CBT-BSc Chemistry and Biotechnology	5	Mandatory Elective
CS-BSc Computer Science	5	Mandatory Elective
ECE-BSc Electrical and Computer Engineering	5	Mandatory Elective
ESSMER-BSc Earth Sciences and Sustainable Management of Environmental Resources	5	Mandatory Elective
GEM-BA Global Economics and Management	5	Mandatory Elective
IBA-BA International Business Administration	5	Mandatory Elective
IBA-Online-BA International Business Administration (Online)	5	Mandatory Elective
IRPH-BA International Relations: Politics and History	5	Mandatory Elective
ISCP-BA Integrated Social and Cognitive Psychology	5	Mandatory Elective
MCCB-BSc Medicinal Chemistry and Chemical Biology	5	Mandatory Elective
MDDA-BSc Management, Decisions and Data Analytics	5	Mandatory Elective
MMDA-BSc Mathematics, Modeling, and Data Analytics	5	Mandatory Elective
PHDS-BSc Physics and Data Science	5	Mandatory Elective
RIS-BSc Robotics and Intelligent Systems	5	Mandatory Elective
SDT-BSc Software, Data and Technology	5	Mandatory Elective

Student Workload	
Independent Study	90
Online Lecture	35
Total Hours	**125**

Module Components	Number	Type	CP
Models, Matrices and Theories of Complex Systems	CTNS-05	Seminar (Online)	5

Module Description

Linear and nonlinear models together with matrix approaches can be useful in diverse disciplines to implement quantitative, computational approaches to exploring complex systems. Some of the most popular data and systems analysis strategies are built upon matrices. Examples include principal component analysis (PCA), the optimization techniques used in Operations Research (OR), the assessment of stable and unstable states in nonlinear dynamical systems, as well as aspects of machine learning.

Here we introduce the toolbox of linear models and matrix-based methods embedded in a wide range of transdisciplinary applications (part 1). We describe its foundation in linear algebra and graph theory (part 2) and the embedding of these concepts in nonlinear dynamics (part 3). At the end of the course, we outline applications to graph theory and machine learning (part 4). Along the way we will encounter some of the classical approaches in complex systems: self-organization, deterministic chaos, cellular atomata and fractals. Throughout the course, examples from neuroscience, social sciences, medicine, biology, physics, chemistry, and other fields are used to illustrate these methods.

A strong emphasis of the course is on the sensible usage of linear approaches in a nonlinear world. We will critically reflect the advantages as well as the disadvantages and limitations of this method. Guiding questions are: How appropriate is a linear approximation of a nonlinear system? What do you really learn from PCA? How reliable are the optimal states obtained via linear programming (LP) techniques?

In the end, students will have a clearer understanding of matrix approaches as well as linear and nonlinear models in their own discipline, but they will also see the full transdisciplinarity of this topic. They will make better decisions in their choice of data analysis methods and become mindful of the challenges when going from a linear to a nonlinear thinking.

Recommended Knowledge

The modules Logic and Causation & Correlation are highly recommended.

Intended Learning Outcomes

No	Competence	ILO
1	Apply	Apply the concept of linear modeling in their own discipline.
2	Distinguish	Distinguish between linear and nonlinear interpretation strategies and understand the range of applicability of linear models.

3	Make	Make use of data analysis / data interpretation strategies from other disciplines, which are derived from linear algebra.
4	Know	Be aware of the ties that linear models have to machine learning and network theory,
5	Note	Note that these four ILOs can be loosely associated with the four parts of the course indicated above.

Indicative Literature

- Part 1: material from Linear Algebra for Everyone, Gilbert Strang, Wellesley-Cambridge Press, 2020.
- Part 2: material from Introduction to Linear Algebra (5th Edition), Gilbert Strang, Cambridge University Press, 2021.
- Part 3: Mainzer, Klaus. "Introduction: from linear to nonlinear thinking." Thinking in Complexity: The Computational Dynamics of Matter, Mind and Mankind (2007): 1-16.; material from Mathematics of Big Data: Spreadsheets, Databases, Matrices, and Graphs, Jeremy Kepner, Hayden Jananthan, The MIT Press, 2018.; material from Introduction to Linear Algebra (5th Edition), Gilbert Strang, Cambridge University Press, 2021.
- Part 4: material from Linear Algebra and Learning from Data, Gilbert Strang, Wellesley-Cambridge Press, 2019.

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Models, Matrices and Theories of Complex Systems	Written Examination	120 Minutes	100	45%	1-4

Module Completion Requirements

- **Module Examination:** minimum for pass- 45%.

Module Achievement

None.

7.2.6 Complex Problem Solving

Module Name	Complex Problem Solving
Module Code	CTNS-NSK-06
Module ECTS	5
Program Owner	CT (CONSTRUCTOR Track Area)
Module Coordinator	Prof. Dr. Marco Verweij

Study Semester		
Program	**Semester**	**Status**
BCCB-BSc Biochemistry and Cell Biology	5	Mandatory Elective
CBT-BSc Chemistry and Biotechnology	5	Mandatory Elective
CS-BSc Computer Science	5	Mandatory Elective
ECE-BSc Electrical and Computer Engineering	5	Mandatory Elective
ESSMER-BSc Earth Sciences and Sustainable Management of Environmental Resources	5	Mandatory Elective
GEM-BA Global Economics and Management	5	Mandatory Elective
IBA-BA International Business Administration	5	Mandatory Elective
IBA-Online-BA International Business Administration (Online)	5	Mandatory Elective
IRPH-BA International Relations: Politics and History	5	Mandatory Elective
ISCP-BA Integrated Social and Cognitive Psychology	5	Mandatory Elective
MCCB-BSc Medicinal Chemistry and Chemical Biology	5	Mandatory Elective
MDDA-BSc Management, Decisions and Data Analytics	5	Mandatory Elective
MMDA-BSc Mathematics, Modeling, and Data Analytics	5	Mandatory Elective
PHDS-BSc Physics and Data Science	5	Mandatory Elective
RIS-BSc Robotics and Intelligent Systems	5	Mandatory Elective
SDT-BSc Software, Data and Technology	5	Mandatory Elective

Student Workload	
Tutorial	17.5
Independent Study	72.5
Online Lecture	35
Total Hours	**125**

Module Components	Number	Type	CP
Complex Problem Solving	CTNS-06	Lecture (Online)	5

Module Description

Complex problems are, by definition, non-linear and/or emergent. Some fifty years ago, scholars such as Herbert Simon began to argue that societies around the world had developed an impressive array of tools with which to solve simple and even complicated problems, but still needed to develop methods with which to address the rapidly increasing number of complex issues. Since then, a variety of such methods has emerged. These include 'serious games' developed in computer science, 'multisector systems analysis' applied in civil and environmental engineering, 'robust decision-making' proposed by the RAND Corporation, 'design thinking' developed in engineering and business studies, 'structured problem-solving' used by McKinsey & Co., 'real-time technology assessment' advocated in science and technology studies, and 'deliberative decision-making' emanating from political science.

In this course, students first learn to distinguish between simple, complicated and complex problems. They also become familiar with the ways in which a particular issue can sometimes shift from one category into another. In addition, the participants learn to apply several tools for resolving complex problems. Finally, the students are introduced to the various ways in which natural and social scientists can help stakeholders resolve complex problems. Throughout the course examples and applications will be used. When possible, guest lectures will be offered by experts on a particular tool for tackling complex issues. For the written, take-home exam, students will have to select a specific complex problem, analyse it and come up with a recommendation - in addition to answering several questions about the material learned.

Recommended Knowledge

- Being able to read primary academic literature

- Willingness to engage in teamwork

- Camillus, J. (2008). Strategy as a wicked problem. Harvard Business Review 86: 99-106;

- Rogers, P. J. (2008). Using programme theory to evaluate complicated and complex aspects of interventions. Evaluation, 14, 29–48.

Intended Learning Outcomes

No	Competence	ILO
1	Identify	Identify a complex problem.
2	Develop	Develop an acceptable recommendation for resolving complex problems.

3	Understand	Understand the roles that natural and social scientists can play in helping stakeholders resolve complex problems.

Indicative Literature

- Camillus, J. (2008). Strategy as a wicked problem. Harvard Business Review 86: 99-106; Rogers, P. J. (2008). Using programme theory to evaluate complicated and complex aspects of interventions. Evaluation, 14, 29–48.
- Chia, A. (2019). Distilling the essence of the McKinsey way: The problem-solving cycle. Management Teaching Review 4(4): 350-377.
- Den Haan, J., van der Voort, M.C., Baart, F., Berends, K.D., van den Berg, M.C., Straatsma, M.W., Geenen, A.J.P., & Hulscher, S.J.M.H. (2020). The virtual river game: Gaming using models to collaboratively explore river management complexity, Environmental Modelling & Software 134, 104855.
- Folke, C., Carpenter, S., Elmqvist, T., Gunderson, L., Holling, C.S., & Walker, B. (2002). Resilience and sustainable development: Building adaptive capacity in a world of transformations. AMBIO: A Journal of the Human Environment 31(5): 437-440.
- Ostrom, E. (2010). Beyond markets and states: Polycentric governance of complex economic systems. American Economic Review 100(3): 641-72.
- Pielke, R. Jr. (2007). The honest broker: Making sense of science in policy and politics. Cambridge: Cambridge University Press.
- Project Management Institute (2021). A guide to the project management body of knowledge (PMBOK® guide).
- Schon, D. A., & Rein, M. (1994). Frame reflection: Toward the resolution of intractable policy controversies. New York: Basic Books.
- Simon, H. A. (1973). The structure of ill structured problems. Artificial Intelligence 4(3-4): 181-201.
- Verweij, M. & Thompson, M. (Eds.) (2006). Clumsy solutions for a complex world. London: Palgrave Macmillan.

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Complex Problem Solving	Written Examination	120 Minutes	100	45%	1-3

Module Completion Requirements

- **Module Examination:** minimum for pass- 45%.

Module Achievement

The module achievement is designed to encourage students to regularly engage with the online course materials throughout the semester, thereby providing them with a sufficient grounding in the concepts and applications of complex problem-solving to take the exam. The module achievement consists of online quizzes tied to the course content. In order to fulfill the module achievement — and to be eligible to sit for the final exam — students are required to participate in at least 80% of the available online quizzes. The module achievement is assessed on a pass/non-pass basis and does not affect the overall module grade. Given the flexible, online nature of the quizzes, students are able to complete them at their own convenience within the set timeframes. Students who do not meet the 80% participation threshold before their first exam attempt will have the opportunity to fulfill the module achievement prior to their second exam attempt.

7.2.7 Argumentation, Data Visualization and Communication (perspective I)

Module Name	Argumentation, Data Visualization and Communication (perspective I)
Module Code	CTNS-NSK-07
Module ECTS	5
Program Owner	CT (CONSTRUCTOR Track Area)
Module Coordinator	Prof. Dr. Arvid Kappas

Study Semester		
Program	**Semester**	**Status**
BCCB-BSc Biochemistry and Cell Biology	5	Mandatory Elective
CBT-BSc Chemistry and Biotechnology	5	Mandatory Elective
CS-BSc Computer Science	5	Mandatory Elective
ECE-BSc Electrical and Computer Engineering	5	Mandatory Elective
ESSMER-BSc Earth Sciences and Sustainable Management of Environmental Resources	5	Mandatory Elective
F-ACS-BSc Applied Computer Science	5	Mandatory Elective
GEM-BA Global Economics and Management	5	Mandatory Elective
IBA-BA International Business Administration	5	Mandatory Elective
IBA-Online-BA International Business Administration (Online)	5	Mandatory Elective
IRPH-BA International Relations: Politics and History	5	Mandatory Elective
ISCP-BA Integrated Social and Cognitive Psychology	5	Mandatory Elective
MCCB-BSc Medicinal Chemistry and Chemical Biology	5	Mandatory Elective
MDDA-BSc Management, Decisions and Data Analytics	5	Mandatory Elective
MMDA-BSc Mathematics, Modeling, and Data Analytics	5	Mandatory Elective
PHDS-BSc Physics and Data Science	5	Mandatory Elective
RIS-BSc Robotics and Intelligent Systems	5	Mandatory Elective

S-ACS-BSc Applied Computer Science	5	Mandatory Elective
SDT-BSc Software, Data and Technology	5	Mandatory Elective

Student Workload	
Tutorial	17.5
Independent Study	72.5
Online Lecture	35
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Argumentation, Data Visualization and Communication (perspective I)	CTNS-07	Lecture (Online)	5

Module Description

One must be careful not to confuse argumentation with being argumentative. The latter is an unattractive personal attribute, whereas the former is a requirement of publicly holding a belief, asserting the truth of a proposition, the plausibility of a hypothesis, or a judgment of the value of a person or an asset. It is an essential component of public discourse. Public discourse is governed by norms and one of those norms is that those who assert the truth of a proposition or the validity of an argument or the responsibility of another for wrongdoing open themselves up to good faith requests to defend their claims. In its most general meaning, argumentation is the requirement that one offer evidence in support of the claims they make, as well as in defense of the judgments and assessments they reach. There are different modalities of argumentation associated with different contexts and disciplines. Legal arguments have a structure of their own as do assessments of medical conditions and moral character. In each case, there are differences in the kind of evidence that is thought relevant and, more importantly, in the standards of assessment for whether a case has been successfully made. Different modalities of argumentation require can call for different modes of reasoning. We not only offer reasons in defense of or in support of beliefs we have, judgments we make and hypotheses we offer, but we reason from evidence we collect to conclusions that are warranted by them.

This course is divided into three sections. Section one develops argumentation skills, including how to construct and evaluate claims, how to reason from evidence to conclusions, and how to recognize the cognitive biases and fallacies that can cause arguments to fail. Section two focuses on data visualization as a tool for supporting arguments. A well-designed graph can clarify complex evidence and make an argument accessible, while a poorly designed one can distort the truth. Section three addresses communication. Even a well-designed and supported argument will not persuade its audience if it is not communicated well. You will explore models of communication, the importance of rhetorics, and the ethics of attempting to change the mind of others, including the new and exciting challenges posed by generative AI.

The thread that runs through these sections is a consistent concern with how to support what we claim effectively. Some reasoning is linear: from A we infer B, and from A and B, we infer C, building stepwise towards a conclusion. But not all reasoning takes this form. Sometimes we argue for a claim by showing

that it slots neatly, or fits coherently, with other claims for which we have independent support, or by establishing that those individual pieces of evidence mutually reinforce one another. A key goal of this course is to introduce you to these different structures of justification so that you are aware of both the norms of your discipline and to what makes knowledge trustworthy in the first place.

There are three fundamental ideas that we want to extract from this segment of the course. These are (1) that argumentation is itself a requirement of being a researcher who claims to have made findings of one sort or another; (2) that there are different forms of appropriate argumentation for different domains and circumstances; and (3) that there are different forms of reasoning on behalf of various claims or from various bits of evidence to conclusions: whether those conclusions are value judgments, political beliefs, or scientific conclusions. Our goal is to familiarize you with all three of these deep ideas and to help you gain facility with each.

Recommended Knowledge

The modules Logic and Causation & Correlation are highly recommended.

Intended Learning Outcomes

No	Competence	ILO
1	Distinguish	Distinguish among different modalities of argument, e.g. legal arguments, vs. scientific ones.
2	Construct	Construct arguments using tools of data visualization.
3	Communicate	Communicate conclusions and arguments concisely, clearly and convincingly.

Indicative Literature

- Cairo, A (2012). The Functional Art: An introduction to information graphics and visualization. New Ridders.
- Knaflic, C.N. (2015). Storytelling with data: A data visualization guide for business professionals. John Wiley & Sons.
- Tufte, E.R. (1985). The visual display of quantitative information. The Journal for Healthcare Quality (JHQ), 7(3), 15.

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs

Argumentation, Data Visualization and Communication (perspective I)	Written Examination	120 minutes	100	45%	1-3

Module Completion Requirements

- **Module Examination:** minimum for pass- 45%.

Module Achievement

None.

7.2.8 Argumentation, Data Visualization and Communication (perspective II)

Module Name	Argumentation, Data Visualization and Communication (perspective II)
Module Code	CTNS-NSK-08
Module ECTS	5
Program Owner	CT (CONSTRUCTOR Track Area)
Module Coordinator	Prof. Dr. Arvid Kappas

Study Semester		
Program	**Semester**	**Status**
BCCB-BSc Biochemistry and Cell Biology	6	Mandatory Elective
CBT-BSc Chemistry and Biotechnology	6	Mandatory Elective
CS-BSc Computer Science	6	Mandatory Elective
ECE-BSc Electrical and Computer Engineering	6	Mandatory Elective
ESSMER-BSc Earth Sciences and Sustainable Management of Environmental Resources	6	Mandatory Elective
F-ACS-BSc Applied Computer Science	6	Mandatory Elective
GEM-BA Global Economics and Management	6	Mandatory Elective
IBA-BA International Business Administration	6	Mandatory Elective
IBA-Online-BA International Business Administration (Online)	6	Mandatory Elective
IRPH-BA International Relations: Politics and History	6	Mandatory Elective
ISCP-BA Integrated Social and Cognitive Psychology	6	Mandatory Elective
MCCB-BSc Medicinal Chemistry and Chemical Biology	6	Mandatory Elective
MDDA-BSc Management, Decisions and Data Analytics	6	Mandatory Elective
MMDA-BSc Mathematics, Modeling, and Data Analytics	6	Mandatory Elective
PHDS-BSc Physics and Data Science	6	Mandatory Elective
RIS-BSc Robotics and Intelligent Systems	6	Mandatory Elective

S-ACS-BSc Applied Computer Science	6	Mandatory Elective
SDT-BSc Software, Data and Technology	6	Mandatory Elective

Student Workload	
Independent Study	80
Online Lecture	35
Tutorial	10
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Argumentation, Data Visualization and Communication (perspective II)	CTNS-08	Lecture (Online)	5

Module Description

Humans are a social species, and interaction is crucial throughout the entire life span. While much of human communication involves language, there is a complex multichannel system of nonverbal communication that enriches linguistic content, provides context, and is also involved in structuring dynamic interaction. Interactants achieve goals by encoding information that is interpreted in the light of current context in transactions with others. This complexity implies also that there are frequent misunderstandings as a sender's intention is not fulfilled. Students in this course will learn to understand the structure of communication processes in a variety of formal and informal contexts. They will learn what constitutes challenges to achieving successful communication and to how to communicate effectively, taking the context and specific requirements for a target audience into consideration. These aspects will be discussed also in the scientific context, as well as business, and special cases, such as legal context - particularly with view to argumentation theory.

Communication is a truly transdisciplinary concept that involves knowledge from diverse fields such as biology, psychology, neuroscience, linguistics, sociology, philosophy, communication and information science. Students will learn what these different disciplines contribute to an understanding of communication and how theories from these fields can be applied in the real world. In the context of scientific communication, there will also be a focus on visual communication of data in different disciplines. Good practice examples will be contrasted with typical errors to facilitate successful communication also with view to the Bachelor's thesis.

Recommended Knowledge

-The modules Logic and Causation & Correlation are highly recommended.

- Ability and openness to engage in interactions

- Media literacy, critical thinking and a proficient handling of data sources

- Own research in academic literature

Intended Learning Outcomes

No	Competence	ILO
1	Analyze	Analyze communication processes in formal and informal contexts.
2	Identify	Identify challenges and failures in communication.
3	Design	Design communications to achieve specified goals to specific target groups.
4	Understand	Understand the principles of argumentation theory.
5	Use	Use data visualization in scientific communications.

Indicative Literature

- Douglas Walton: Argumentation Theory – A Very Short Introduction. In: Simari, G., Rahwan, I. (eds) Argumentation in Artificial Intelligence. Springer, Boston, MA, 2009.
- Joseph A. DeVito: The Interpersonal Communication Book (Global edition, 16th edition), 2022.
- Steven L. Franconeri, Lace M. Padilla, Priti Shah, Jeffrey M. Zacks, and Jessica Hullman: The Science of Visual Data Communication: What Works Psychological Science in the Public Interest, 22(3), 110–161, 2022.

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Argumentation, Data Visualization and Communication (perspective II)	Presentation	7-11 mins	100	45%	1-5

Module Completion Requirements

Module Examination: minimum for pass- 45%.

-

Module Achievement

None.

7.2.9 Agency, Leadership, and Accountability

Module Name	Agency, Leadership, and Accountability
Module Code	CTNS-NSK-09
Module ECTS	5
Program Owner	CT (CONSTRUCTOR Track Area)
Module Coordinator	Dr. Marina Christodoulou

Study Semester		
Program	**Semester**	**Status**
S-ACS-BSc Applied Computer Science	5	Mandatory
BCCB-BSc Biochemistry and Cell Biology	6	Mandatory Elective
CBT-BSc Chemistry and Biotechnology	6	Mandatory Elective
CS-BSc Computer Science	6	Mandatory Elective
ECE-BSc Electrical and Computer Engineering	6	Mandatory Elective
ESSMER-BSc Earth Sciences and Sustainable Management of Environmental Resources	6	Mandatory Elective
F-ACS-BSc Applied Computer Science	6	Mandatory Elective
GEM-BA Global Economics and Management	6	Mandatory Elective
IBA-BA International Business Administration	6	Mandatory Elective
IBA-Online-BA International Business Administration (Online)	6	Mandatory
IRPH-BA International Relations: Politics and History	6	Mandatory Elective
ISCP-BA Integrated Social and Cognitive Psychology	6	Mandatory Elective
MCCB-BSc Medicinal Chemistry and Chemical Biology	6	Mandatory Elective
MDDA-BSc Management, Decisions and Data Analytics	6	Mandatory Elective
MMDA-BSc Mathematics, Modeling, and Data Analytics	6	Mandatory Elective
PHDS-BSc Physics and Data Science	6	Mandatory Elective
RIS-BSc Robotics and Intelligent Systems	6	Mandatory Elective

SDT-BSc Software, Data and Technology	6	Mandatory Elective

Student Workload	
Tutorial	17.5
Independent Study	72.5
Online Lecture	35
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Agency, Leadership, and Accountability	CTNS-09	Lecture (Online)	5

Module Description

Each of us is judged by the actions we undertake and held to account for the consequences of them. Sometimes we may be lucky and our bad acts don't have harmful effects on others. Other times we may be unlucky and reasonable decisions can lead to unexpected or unforeseen adverse consequences for others. We are therefore held accountable both for choices and for outcomes. In either case, accountability expresses the judgment that we bear responsibility for what we do and what happens as a result. But our responsibility and our accountability in these cases is closely connected to the idea that we have agency.

Agency presumes that we are the source of the choices we make and the actions that result from those choices. For some, this may entail the idea that we have free will. But there is scientific world view that holds that all actions are determined by the causes that explain them, which is the idea that if we knew the causes of your decisions in advance, we would know the decision you would make even before you made it. If that is so, how can your choice be free? And if it is not free, how can you be responsible for it? And if you cannot be responsible, how can we justifiably hold you to account for it?

These questions express the centuries old questions about the relationship between free will and a determinist world view: for some, the conflict between a scientific world view and a moral world view.

But we do not always act as individuals. In society we organize ourselves into groups: e.g. tightly organized social groups, loosely organized market economies, political societies, companies, and more. These groups have structure. Some individuals are given the responsibility of leading the group and of exercising authority. But one can exercise authority over others in a group merely by giving orders and threatening punishment for non-compliance.

Exercising authority is not the same thing as being a leader? For one can lead by example or by encouraging others to exercise personal judgment and authority. What then is the essence of leadership?

The module has several educational goals. The first is for students to understand the difference between actions that we undertake for which we can reasonably held accountable and things that we do but which we are not responsible for. For example, a twitch is an example of the latter, but so too may be a car accident we cause as a result of a heart attack we had no way of anticipating or

controlling. This suggests the importance of control to responsibility. At the heart of personal agency is the idea of control. The second goal is for students to understand what having control means. Some think that the scientific view is that the world is deterministic, and if it is then we cannot have any personal control over what happens, including what we do. Others think that the quantum scientific view entails a degree of indeterminacy and that free will and control are possible, but only in the sense of being unpredictable or random. But then random outcomes are not ones we control either. So, we will devote most attention to trying to understand the relationships between control, causation and predictability.

But we do not only exercise agency in isolation. Sometimes we act as part of groups and organizations. The law often recognizes ways in which groups and organizations can have rights, but is there a way in which we can understand how groups have responsibility for outcomes that they should be accountable for. We need to figure out then whether there is a notion of group agency that does not simply boil down to the sum of individual actions. We will explore the ways in which individual actions lead to collective agency.

Finally we will explore the ways in which occupying a leadership role can make one accountable for the actions of others over which one has authority.

Intended Learning Outcomes

No	Competence	ILO
1	Understand	Understand and reflect how the social and moral world views that rely on agency and responsibility are compatible, if they are, with current scientific world views.
2	Understand	Understand how science is an economic sector, populated by large powerful organizations that set norms, fund research agendas.
3	Identify	Identify the difference between being a leader of others or of a group - whether a research group or a lab or a company - and being in charge of the group.
4	Learn	Learn to be a leader of others and groups. Understand that when one graduates one will enter not just a field of work but a heavily structured set of institutions and that one's agency and responsibility for what happens, what work gets done, its quality and value, will be affected accordingly.

Indicative Literature

- Feinberg, Joel. "Doing & deserving; essays in the theory of responsibility." (1970).
- Hull, David L. "Science as a Process." Science as a Process. University of Chicago Press, 2010.

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Agency, Leadership, and Accountability	Written Examination	120 Minutes	100	45%	1-4

Module Completion Requirements

- **Module Examination:** minimum for pass- 45%.

Module Achievement

None.

7.2.10 Community Impact Project

Module Name	Community Impact Project
Module Code	CTNC-CIP-10
Module ECTS	5
Program Owner	CT (CONSTRUCTOR Track Area)
Module Coordinator	CIP Faculty Coordinator

Study Semester		
Program	**Semester**	**Status**
PHDS-BSc Physics and Data Science	5	Mandatory Elective
SDT-BSc Software, Data and Technology	5	Mandatory Elective
PHDS-BSc Physics and Data Science	6	Mandatory Elective
RIS-BSc Robotics and Intelligent Systems	6	Mandatory Elective
SDT-BSc Software, Data and Technology	6	Mandatory Elective

Student Workload	
Self-Organized Teamwork	115
Introductory, Accompanying, and Final Events	10
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Community Impact Project	CTNC-10	Project	5

Module Description

CIPs are self-organized, major-related, and problem-centered applications of students' acquired knowledge and skills. These activities will ideally be connected to their majors so that they will challenge the students' sense of practical relevance and social responsibility within the field of their studies. Projects will tackle real issues in their direct and/or broader social environment. These projects ideally connect the campus community to other communities, companies, or organizations in a mutually beneficial way.

Students are encouraged to create their own projects and find partners (e.g., companies, schools, NGOs), but will get help from the CIP faculty coordinator team and faculty mentors to do so. They can

join and collaborate in interdisciplinary groups that attack a given issue from different disciplinary perspectives. Student activities are self-organized but can draw on the support and guidance of both faculty and the CIP faculty coordinator team.

Recommended Knowledge

- Basic knowledge of the main concepts and methodological instruments of the respective disciplines

- Develop or join a community impact project before the 5th or 6th semester based on the introductory events during the 4th semester by using the database of projects, communicating with fellow students and faculty, and finding potential companies, organizations, or communities to target.

Usability and Relationship to other Modules

Students who have accomplished their CIP (6th semester) are encouraged to support their fellow students during the development phase of the next year's projects (4th semester)

Intended Learning Outcomes

No	Competence	ILO
1	Understand	Understand the real-life issues of communities, organizations, and industries and relate them to concepts in their own discipline
2	Enhance	Enhance problem-solving skills and develop critical faculty, create solutions to problems, and communicate these solutions appropriately to their audience
3	Apply	Apply media and communication skills in diverse and non-peer social contexts
4	Develop	Develop an awareness of the societal relevance of their own scientific actions and a sense of social responsibility for their social surroundings
5	Reflect	Reflect on their own behavior critically in relation to social expectations and consequences
6	Work	Work in a team and deal with diversity, develop cooperation and conflict skills, and strengthen their empathy and tolerance for ambiguity

Indicative Literature

Entry Requirements

Prerequisites	CTNC-CIP-10 Community Impact Project At least 15 CP from CORE modules in the major
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Community Impact Project	Project Assessment	not numerically graded (pass/fail)	100		All intended learning outcomes of the module.

Module Completion Requirements

- **Module Examination:** minimum for pass- 45%.

Module Achievement

None.

7.3 Languages

The descriptions of the language modules are provided in a separate document, the "Language Module Handbook" that can be accessed from the Constructor University's Language & Community Center internet sites (https://constructor.university/student-life/language-community-center/learning-languages).

7.4 Humanity Modules

7.4.1 Introduction to Visual Culture

Module Name	Introduction to Visual Culture
Module Code	CTHU-HUM-003
Module ECTS	2.5
Program Owner	CT (CONSTRUCTOR Track Area)
Module Coordinator	Dr. Irina Chiaburu

Study Semester		
Program	**Semester**	**Status**
BCCB-BSc Biochemistry and Cell Biology	1	Mandatory Elective
CBT-BSc Chemistry and Biotechnology	1	Mandatory Elective
CS-BSc Computer Science	1	Mandatory Elective
ECE-BSc Electrical and Computer Engineering	1	Mandatory Elective
ESSMER-BSc Earth Sciences and Sustainable Management of Environmental Resources	1	Mandatory Elective
IBA-BA International Business Administration	1	Mandatory Elective
IEM-Online-BSc Industrial Engineering & Management (Online)	1	Mandatory Elective
IRPH-BA International Relations: Politics and History	1	Mandatory Elective
ISCP-BA Integrated Social and Cognitive Psychology	1	Mandatory Elective
MDDA-BSc Management, Decisions and Data Analytics	1	Mandatory Elective
MMDA-BSc Mathematics, Modeling, and Data Analytics	1	Mandatory Elective
PHDS-BSc Physics and Data Science	1	Mandatory Elective

RIS-BSc Robotics and Intelligent Systems	1	Mandatory Elective
SDT-BSc Software, Data and Technology	1	Mandatory Elective
BCCB-BSc Biochemistry and Cell Biology	2	Mandatory Elective
CBT-BSc Chemistry and Biotechnology	2	Mandatory Elective
CS-BSc Computer Science	2	Mandatory Elective
ECE-BSc Electrical and Computer Engineering	2	Mandatory Elective
ESSMER-BSc Earth Sciences and Sustainable Management of Environmental Resources	2	Mandatory Elective
GEM-BA Global Economics and Management	2	Mandatory Elective
IBA-BA International Business Administration	2	Mandatory Elective
IBA-Online-BA International Business Administration (Online)	2	Mandatory Elective
IEM-BSc Industrial Engineering & Management	2	Mandatory Elective
IEM-Online-BSc Industrial Engineering & Management (Online)	2	Mandatory Elective
IRPH-BA International Relations: Politics and History	2	Mandatory Elective
ISCP-BA Integrated Social and Cognitive Psychology	2	Mandatory Elective
MCCB-BSc Medicinal Chemistry and Chemical Biology	2	Mandatory Elective
MDDA-BSc Management, Decisions and Data Analytics	2	Mandatory Elective
MMDA-BSc Mathematics, Modeling, and Data Analytics	2	Mandatory Elective
PHDS-BSc Physics and Data Science	2	Mandatory Elective
RIS-BSc Robotics and Intelligent Systems	2	Mandatory Elective
SDT-BSc Software, Data and Technology	2	Mandatory Elective

Student Workload	
Independent Study	45
Online Lecture	17.5

Total Hours	62.5

Module Components	Number	Type	CP
Introduction to Visual Culture	CTHU-003	Lecture (Online)	2.5

Module Description

Of the five senses, the sense of sight has for a long time occupied the central position in human cultures. As John Berger has suggested this could be because we can see and recognize the world around us before we learn how to speak. Images have been with us since the earliest days of the human history. In fact, the earliest records of human history are images found on cave walls across the world. We use images to capture abstract ideas, to catalogue and organize the world, to represent the world, to capture specific moments, to trace time and change, to tell stories, to express feelings, to better understand, to provide evidence and more. At the same time, images exert their power on us, seducing us into believing in their 'innocence', that is into forgetting that as representations they are also interpretations, i.e., a particular version of the world.

The purpose of this course is to explore multiple ways in which images and the visual in general mediate and structure human experiences and practices from more specialized discourses, e.g., scientific discourses, to more informal and personal day-to-day practices, such as self-fashioning in cyberspace. We will look at how social and historical contexts affect how we see, as well as what is visible and what is not. We will explore the centrality of the visual to the intellectual activity, from early genres of scientific drawing to visualizations of big data. We will examine whether one can speak of visual culture of protest, look at the relationship between looking and subjectivity and, most importantly, ponder the relationship between the visual and the real.

This 2.5 CP module is part of a portfolio of several 2.5 CP modules designed to allow students to have a wider choice of electives. To compensate for the smaller size of this module, the assessment is of smaller scope.

Intended Learning Outcomes

No	Competence	ILO
1	Understand	Understand a range of key concepts pertaining to visual culture, art theory and cultural analysis.
2	Understand	Understand the role visuality plays in development and maintenance of political, social, and intellectual discourses.
3	Think	Think critically about images and their contexts.
4	Reflect	Reflect critically on the connection between seeing and knowing.

Indicative Literature

- Berger, J., Blomberg, S., Fox, C., Dibb, M., & Hollis, R. (1973). Ways of seeing.
- Foucault, M. (2002). The order of things: an archaeology of the human sciences (Ser. Routledge classics). Routledge.

- Hunt, L. (2004). Politics, culture, and class in the French revolution: twentieth anniversary edition, with a new preface (Ser. Studies on the history of society and culture, 1). University of California Press.
- Miller, V. (2020). Understanding digital culture (Second). SAGE.
- Thomas, N. (1994). Colonialism's culture: anthropology, travel and government. Polity Press.

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Introduction to Visual Culture	Written Examination	60 Minutes	100	45%	1-4

Module Completion Requirements

- **Module Examination:** minimum for pass- 45%.

Module Achievement

None.

7.4.2 Introduction to the Philosophy of Science

Module Name	Introduction to the Philosophy of Science
Module Code	CTHU-HUM-002
Module ECTS	2.5
Program Owner	CT (CONSTRUCTOR Track Area)
Module Coordinator	Dr. Eoin Ryan

Study Semester		
Program	**Semester**	**Status**
BCCB-BSc Biochemistry and Cell Biology	1	Mandatory Elective
CBT-BSc Chemistry and Biotechnology	1	Mandatory Elective
CS-BSc Computer Science	1	Mandatory Elective
ECE-BSc Electrical and Computer Engineering	1	Mandatory Elective
ESSMER-BSc Earth Sciences and Sustainable Management of Environmental Resources	1	Mandatory Elective
GEM-BA Global Economics and Management	1	Mandatory Elective
IBA-BA International Business Administration	1	Mandatory Elective
IBA-Online-BA International Business Administration (Online)	1	Mandatory Elective
IEM-Online-BSc Industrial Engineering & Management (Online)	1	Mandatory Elective
IRPH-BA International Relations: Politics and History	1	Mandatory Elective
ISCP-BA Integrated Social and Cognitive Psychology	1	Mandatory Elective
MCCB-BSc Medicinal Chemistry and Chemical Biology	1	Mandatory Elective
MDDA-BSc Management, Decisions and Data Analytics	1	Mandatory Elective
MMDA-BSc Mathematics, Modeling, and Data Analytics	1	Mandatory Elective
PHDS-BSc Physics and Data Science	1	Mandatory Elective
RIS-BSc Robotics and Intelligent Systems	1	Mandatory Elective
SDT-BSc Software, Data and Technology	1	Mandatory Elective

BCCB-BSc Biochemistry and Cell Biology	2	Mandatory Elective
CBT-BSc Chemistry and Biotechnology	2	Mandatory Elective
CS-BSc Computer Science	2	Mandatory Elective
ECE-BSc Electrical and Computer Engineering	2	Mandatory Elective
ESSMER-BSc Earth Sciences and Sustainable Management of Environmental Resources	2	Mandatory Elective
IBA-BA International Business Administration	2	Mandatory Elective
IBA-Online-BA International Business Administration (Online)	2	Mandatory Elective
IEM-BSc Industrial Engineering & Management	2	Mandatory Elective
IEM-Online-BSc Industrial Engineering & Management (Online)	2	Mandatory Elective
IRPH-BA International Relations: Politics and History	2	Mandatory Elective
ISCP-BA Integrated Social and Cognitive Psychology	2	Mandatory Elective
MCCB-BSc Medicinal Chemistry and Chemical Biology	2	Mandatory Elective
MDDA-BSc Management, Decisions and Data Analytics	2	Mandatory Elective
MMDA-BSc Mathematics, Modeling, and Data Analytics	2	Mandatory Elective
PHDS-BSc Physics and Data Science	2	Mandatory Elective
RIS-BSc Robotics and Intelligent Systems	2	Mandatory Elective
SDT-BSc Software, Data and Technology	2	Mandatory Elective

Student Workload	
Independent Study	45
Online Lecture	17.5
Total Hours	**62.5**

Module Components	**Number**	**Type**	**CP**
Introduction to the Philosophy of Science	CTHU-002	Lecture (Online)	2.5

Module Description

This humanities module will introduce students to some of the central ideas in philosophy of science. Topics will include distinguishing science from pseudo-science, types of inference and the problem of induction, the pros and cons of realism and anti-realism, the role of explanation, the nature of scientific change, the difference between natural and social sciences, scientism and the values of science, as well as some examples from philosophy of the special sciences (e.g., physics, biology).

The course aims to give students an understanding of how science produces knowledge, and some of the various contexts and issues which mean this process is never entirely transparent, neutral, or unproblematic. Students will gain a critical understanding of science as a human practice and technology; this will enable them both to better understand the importance and success of science, but also how to properly critique science when appropriate.

This 2.5 CP module is part of a portfolio of several 2.5 CP modules designed to allow students to have a wider choice of electives. To compensate for the smaller size of this module, the assessment is of smaller scope.

Intended Learning Outcomes

No	Competence	ILO
1	Understand	Understand key ideas from the philosophy of science.
2	Discuss	Discuss different types of inference and rational processes.
3	Describe	Describe differences between how the natural sciences, social sciences and humanities discover knowledge.
4	Identify	Identify ways in which science can be more and less value-laden.
5	Illustrate	Illustrate some important conceptual leaps in the history of science.

Indicative Literature

- James Ladyman, Understanding Philosophy of Science (2002).
- Paul Song, Philosophy of Science: Perspectives from Scientists (2022).
- Peter Godfrey-Smith Theory and Reality (2021)

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Introduction to the Philosophy of Science	Written Examination	60 Minutes	100	45%	1-5

Module Completion Requirements

- **Module Examination:** minimum for pass- 45%.

Module Achievement

None.

7.4.3 Introduction to Philosophical Ethics

Module Name	Introduction to Philosophical Ethics
Module Code	CTHU-HUM-001
Module ECTS	2.5
Program Owner	CT (CONSTRUCTOR Track Area)
Module Coordinator	Dr. Eoin Ryan

Study Semester		
Program	**Semester**	**Status**
BCCB-BSc Biochemistry and Cell Biology	1	Mandatory Elective
CBT-BSc Chemistry and Biotechnology	1	Mandatory Elective
CS-BSc Computer Science	1	Mandatory Elective
ECE-BSc Electrical and Computer Engineering	1	Mandatory Elective
ESSMER-BSc Earth Sciences and Sustainable Management of Environmental Resources	1	Mandatory Elective
IBA-BA International Business Administration	1	Mandatory Elective
IBA-Online-BA International Business Administration (Online)	1	Mandatory Elective
IEM-BSc Industrial Engineering & Management	1	Mandatory Elective
IEM-Online-BSc Industrial Engineering & Management (Online)	1	Mandatory Elective
IRPH-BA International Relations: Politics and History	1	Mandatory Elective
ISCP-BA Integrated Social and Cognitive Psychology	1	Mandatory Elective
MCCB-BSc Medicinal Chemistry and Chemical Biology	1	Mandatory Elective
MDDA-BSc Management, Decisions and Data Analytics	1	Mandatory Elective
MMDA-BSc Mathematics, Modeling, and Data Analytics	1	Mandatory Elective
PHDS-BSc Physics and Data Science	1	Mandatory Elective
RIS-BSc Robotics and Intelligent Systems	1	Mandatory Elective
SDT-BSc Software, Data and Technology	1	Mandatory Elective

BCCB-BSc Biochemistry and Cell Biology	2	Mandatory Elective
CBT-BSc Chemistry and Biotechnology	2	Mandatory Elective
CS-BSc Computer Science	2	Mandatory Elective
ECE-BSc Electrical and Computer Engineering	2	Mandatory Elective
ESSMER-BSc Earth Sciences and Sustainable Management of Environmental Resources	2	Mandatory Elective
GEM-BA Global Economics and Management	2	Mandatory Elective
IBA-BA International Business Administration	2	Mandatory Elective
IBA-Online-BA International Business Administration (Online)	2	Mandatory Elective
IEM-Online-BSc Industrial Engineering & Management (Online)	2	Mandatory Elective
IRPH-BA International Relations: Politics and History	2	Mandatory Elective
ISCP-BA Integrated Social and Cognitive Psychology	2	Mandatory Elective
MCCB-BSc Medicinal Chemistry and Chemical Biology	2	Mandatory Elective
MDDA-BSc Management, Decisions and Data Analytics	2	Mandatory Elective
MMDA-BSc Mathematics, Modeling, and Data Analytics	2	Mandatory Elective
PHDS-BSc Physics and Data Science	2	Mandatory Elective
RIS-BSc Robotics and Intelligent Systems	2	Mandatory Elective
SDT-BSc Software, Data and Technology	2	Mandatory Elective

Student Workload	
Independent Study	45
Online Lecture	17.5
Total Hours	**62.5**

Module Components	**Number**	**Type**	**CP**
Introduction to Philosophical Ethics	CTHU-001	Lecture (Online)	2.5

Module Description

The nature of morality - how to lead a life that is good for yourself, and how to be good towards others - has been a central debate in philosophy since the time of Socrates, and it is a topic that continues to be vigorously discussed. This course will introduce students to some of the key aspects of philosophical ethics, including leading normative theories of ethics (e.g. consequentialism or utilitarianism, deontology, virtue ethics, natural law ethics, egoism) as well as some important questions from metaethics (are useful and generalizable ethical claims even possible; what do ethical speech and ethical judgements actually do or explain) and moral psychology (how do abstract ethical principles do when realized by human psychologies). The course will describe ideas that are key factors in ethics (free will, happiness, responsibility, good, evil, religion, rights) and indicate various routes to progress in understanding ethics, as well as some of their difficulties.

This 2.5 CP module is part of a portfolio of several 2.5 CP modules designed to allow students to have a wider choice of electives. To compensate for the smaller size of this module, the assessment is of smaller scope.

Intended Learning Outcomes

No	Competence	ILO
1	Describe	Describe normative ethical theories such as consequentialism, deontology and virtue ethics.
2	Discuss	Discuss some metaethical concerns.
3	Analyze	Analyze ethical language.
4	Highlight	Highlight complexities and contradictions in typical ethical commitments.
5	Indicate	Indicate common parameters for ethical discussions at individual and social levels.
6	Analyze	Analyze notions such as objectivity, subjectivity, universality, pluralism, value.

Indicative Literature

- Mark van Roojen Metaethicas: A Contemporary Introduction (2015).
- Russ Shafer-Landay A Concise Introduction to Ethics (2019).
- Simon Blackburn Being Good (2009).

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Introduction to Philosophical Ethics	Written Examination	60 Minutes	100	45%	1-6

Module Completion Requirements

- **Module Examination:** pass/fail grading.

Module Achievement

None.

8 Appendix

8.1 Intended Learning Outcomes Assessment-Matrix

Software, Data and Technology BSc					System Programming and Performance	Analysis	Industrial Programming with Python	Practical Minimum (Git/Docker/CLI)	Core Algorithms and Data Structures	Intro to AI & Agentic Workflows	Compute Systems for AI & Performance,	Linear Algebra	Operating Systems	Software Engineering and Design	Advanced Algorithms and Data Structures	Machine Learning	Computer Networks	Functional Programming	Scientific Data Analysis	Database Fundamentals	Discrete Mathematics	Artificial Intelligence	Specialization modules	Bachelor Thesis	CT Methods	Basics of Discrete Mathematics	JVM Based Programming	CT New Skills	CT Language and Humanities
Semester					1	1	1	1	2	2	2	2	3	4	3	4	3	4	3	4	4	4	5-6	6	1-4	1	2	3-6	1-2
Mandatory/ mandatory elective					m	me	m	m	m	m	m	me	m	m	m	m	me	me	me	me	me	me	me	m	me	m	m	me	me
Credits					7,5	7,5	5,0	2,5	7,5	5,0	2,5	7,5	7,5	7,5	5,0	5,0	5,0	5,0	5,0	5,0	5,0	5,0	15,0	15	20	5	5	20	5,0
	Competencies*																												
Program Learning Outcomes	**A**	**E**	**P**	**S**																									
work professionally in the field of software, Data and Technology and enter graduate programs related to these fields	x	x	x		x	x	x	x	x	x	x	x	x	x	x	x		x	x	x	x	x	x	x	x	x	x		
apply fundamental concepts of mathematics, statistics, and computer science while solving data-related problems;	x	x				x	x		x	x	x	x			x	x	x		x		x	x	x	x	x	x	x	x	
analyze at multiple levels of abstraction and use appropriate mathematical and computational methods to model and analyze real-world problems;	x	x	x	x	x	x	x		x	x	x	x	x	x	x	x		x	x	x	x	x	x	x	x	x	x	x	
develop, analyze and implement algorithms using modern software engineering methods and programming languages;	x	x			x		x		x	x					x			x	x	x		x	x	x				
understand the characteristics of a range of computing platforms and their advantages and limitations;	x	x			x		x	x		x	x		x	x						x			x	x					
choose from multiple programming paradigms, languages and algorithms to solve a given problem adequately;	x	x			x		x		x	x			x	x	x			x		x			x	x					
apply the necessary mathematical methods, such as linear algebra, analysis, calculus, and discrete mathematics;	x					x	x		x			x			x	x		x			x		x	x	x	x			
recognize the context in which data science and software systems operate, including interactions with people and the physical world;	x	x	x	x	x		x	x		x	x		x	x		x		x	x	x		x	x	x	x			x	
describe the state of published knowledge in the field of software, Data and Technology and in a chosen specialization (Machine Learning, Software Development, Programming Languages);	x	x	x		x		x			x	x		x	x		x		x	x	x	x	x	x	x	x	x			
analyze and model real-life scenarios in organizations and industries using contemporary techniques of data science and software development, also taking methods and insights of other disciplines into account;	x	x	x		x		x		x	x	x		x	x	x	x		x	x	x	x	x	x	x	x	x			
appropriately communicate solutions of problems in software, Data and Technology in both spoken and written form to specialists and non-specialists;	x	x	x	x				x		x			x										x	x				x	x
draw scientifically founded conclusions that consider social, professional, scientific, and ethical aspects;	x	x	x	x	x	x	x		x	x	x	x	x	x	x	x		x	x	x	x	x	x	x	x	x	x	x	x
work effectively in a diverse team and take responsibility in a team;	x	x	x	x				x		x													x	x	x			x	x
take responsibility for their own learning, personal and professional development and role in society, reflecting on their practice and evaluating critical feedback;								x																x	x			x	x
adhere to and defend ethical, scientific, and professional standards;	x	x	x	x						x														x	x			x	
Assessment Type																													
Written examination					x	x	x		x			x	x	x	x	x	x	x		x	x	x			x	x	x	x	
Term paper																													
Essay																							x		x				
Project report																													
Poster presentation																													
Laboratory report																													
Program Code					x		x	x	x	x	x			x	x			x		x			x						
Oral examination																													
Presentation																								x			x	x	
Practical Assessments																													
Project Assessments																												x	
Portfolio Assessments																			x				x						
Bachelor Thesis																								x					
Module achievements							x						x															x	x

*Competencies: A-scientific/academic proficiency; E-competence for qualified employment; P-development of personality; S-competence for engagement in society

Figure 4: Intended Learning Outcomes Assessment-Matrix