Syllabus

Course Description

Course Title	Fundamentals of Programming
Course Code	42426
Course Title Additional	
Scientific-Disciplinary Sector	INFO-01/A
Language	English
Degree Course	Bachelor in Electronic and Information Engineering
Other Degree Courses (Loaned)	
Lecturers	Dr. Sergio Tessaris, Sergio.Tessaris@unibz.it https://www.unibz.it/en/faculties/engineering/academic-staff/person/2315 Prof. Rosella Gennari, Rosella.Gennari@unibz.it https://www.unibz.it/en/faculties/engineering/academic-staff/person/8607
Teaching Assistant	Dott. Muhammad Bilal Khan
Semester	All semesters
Course Year/s	1
CP	11
Teaching Hours	70
Lab Hours	40
Individual Study Hours	165
Planned Office Hours	33
Contents Summary	The course refers to the basic educational activities and belongs to the scientific area of Computer Science. The course is designed for acquiring professional skills and knowledge. The objective of the course is to teach the fundamental principles of programming, with a focus on structured programming, and tools to support the development of software.

	Students will learn how to solve computational problems with well-designed programs The learning will be based on examples and practical assignments, from very simple ones to more complex. The final objective for the student is to acquire the ability to translate a set of functional and non-functional requirements into a software solution.
Course Topics	Module 1: 1. Introduction to: hardware and software, with computer organisation; data hierarchy; machine languages, assembly languages, high-level programming languages. Introduction to Python: interactive mode, script mode, Jupyter. 2. Introduction to different programming paradigms, focusing on the structured programming paradigm. 3. Structured programming: basic data types, variables, constants, operators and expressions; standard input/output handling; control flow structures; file and error handling. 4. Basic data structures/types of Python: (1) lists, (2) dictionaries, (3) tuples, (4) sets. 5. Subroutines and functions in Python (with/without parameters; with/without return). 6. Basics on modules and packages in Python. The above will be delivered meanwhile acquiring practical knowledge, through programming exercises, of how to program a simple physical-computing board with a Python-based language (e.g., Raspberry PicoH or PicoWH, ESP32, running MicroPython). Programming exercises cover the following: - how to perceive data via basic physical input devices (e.g., temperature sensor, humidity sensor), - how to process and store data, - how to plot data depending on their features. Module 2: The following topics will be covered by focusing on the C programming language and its specific features. Differences and similarities with Python will be outlined. 1. Introduction to C programming and toolchain 1.1. Understanding and using the compiler toolchains 1.2. Understanding cross-compilation

	1.3. Tools to support modern software development 2. C language: syntax and data types 2.1. C standards 2.2. Control flow 2.3. Basic and derived types 3. C memory management and activation record 3.1. C memory organisation 3.2. Dynamic memory management 4. C programming techniques 4.1. Organisation of software artifacts in C 4.2. Effective use of C constructs and data types 4.3. Defensive programming techniques 5. Debugging and software testing 5.1. Techniques and strategies for effective debugging of code 5.2. Debugging tools 5.3. Unit testing
Keywords	Programming Fundamentals; Python; Physical Computing; C; Software management and testing
Recommended Prerequisites	
Propaedeutic Courses	
Teaching Format	Lectures, exercises, laboratory activites
Mandatory Attendance	Strongly recommended
Specific Educational Objectives and Learning Outcomes	The course refers to the basic educational activities and belongs to the scientific area of Computer Science. The course is designed for acquiring professional skills and knowledge. The objective of the course is to teach the fundamental principles of programming, with a focus on structured programming, and tools to support the development of software. Students will learn how to solve computational problems with well-designed programs The learning will be based on examples and practical assignments, from very simple ones to more complex. The final objective for the student is to acquire the ability to translate a set of functional and non-functional requirements into a software solution.
Specific Educational Objectives and Learning Outcomes (additional info.)	Knowledge and understanding • Know the fundamental principles of programming. • Know different programming paradigms and models of

	computation. • Have a solid knowledge of the most important data structures and programming techniques. Applying knowledge and understanding • Be able to solve problems using programming. • Be able to develop small and medium size programs starting from given requirements. Making judgements • Be able to collect and interpret useful data and to judge information systems and their applicability. • Be able to identify an appropriate programming paradigm and data structures to solve a given problem. Communication skills • Be able to describe and motivate the software design choices. • Be able to properly document a software artifact to ensure its integration in more complex systems. Learning skills Be able to learn how to use different procedural programming languages in autonomy, by identifying and understanding the relevant literature.
Assessment	Module 1 Attending students are those that - attend at least 70% of the exercise classes of the module, i.e., at least 14 hours (hard constraint), - participate in class with a positive and reflective attitude, - show a committment in tackling the class exercises for learning, taking due care of deadlines and instructions. The assessment is as follows: - For the lecture part: a written, closed-book exam with closed-ended and open-ended questions for all students; - For the exercise part: a programming project for attending students; a written exam for non-attending students. The results of written exams are only valid for the session of the examination. The result of the project will be valid for 1 academic year and cannot be carried over beyond that time-frame. - Note: in case of a positive outcome, the intermediate exam, assignments and project work are valid for 1 academic year only and cannot be carried over beyond that time-frame.

	Module 2 Assessment will be the same for attending and non-attending students. It's divided in two parts: 1. Written final exam, with review questions about the lecture material (closed-ended questions and closed-book exam). The results of the written exam are only valid for the session of the examination. 2. Lab practical assignments to be submitted online; contributions will be valid for 1 academic year and cannot be carried over beyond that time-frame.
Evaluation Criteria	A student passes the exam only if the student has a positive result (i.e., not less than 18) and tackles all parts of the exam (see Assessment above) by the appointed deadlines. The result is the average of the marks for Modules 1 and 2. The marks for Modules 1 and 2 are given as follows: - the mark for Module 1 ranges from 0 to 30: the assignments/projects count for 20% (min is 0, max is 6), and the written exam for 80% of the mark (min is 0, max is 24); - the mark for Module 2 ranges from 0 to 30: the assignments count for 60%, and the written exam counts for 40% of the mark: o All assignments must be submitted before the date of the written exam, failure of doing so will result in an incomplete submission and non-admission to the final evaluation; o Only some of the assignments, clearly indicated beforehand, will contribute to the mark. Laude is jointly decided by the course lecturers in case the marks for both modules is 30. E.g., suppose marks per Module are as follows: - Module 1's mark is 28 (25 for the written exam, 3 for the project); - Module 2's mark is 30. The result for the student is then 29, the average of 28 and 30. Written exam questions are evaluated in terms of correctness and clarity.

	Assignments/projects are evaluated in terms of: - quality, according to the criteria illustrated and explained in class, and recorded in the companion materials (e.g., code quality criteria), - displayed problem-solving skills, - displayed communication skills, - displayed critical-thinking skills.
Required Readings	Material is provided during the course.
Supplementary Readings	Material is provided during the course.
Further Information	Subject Librarian: David Gebhardi, David.Gebhardi@unibz.it and Ilaria Miceli, Ilaria.Miceli@unibz.it
Sustainable Development Goals (SDGs)	Industry, innovation and infrastructure, Quality education

Course Module

Course Constituent Title	Fundamentals of Programming I
Course Code	42426A
Scientific-Disciplinary Sector	INFO-01/A
Language	English
Lecturers	Prof. Rosella Gennari, Rosella.Gennari@unibz.it https://www.unibz.it/en/faculties/engineering/academic-staff/person/8607
Teaching Assistant	Dott. Muhammad Bilal Khan
Semester	First semester
CP	6
Responsible Lecturer	
Teaching Hours	40
Lab Hours	20
Individual Study Hours	90
Planned Office Hours	

Contents Summary	The course refers to the basic educational activities and belongs to the scientific area of Computer Science. The course is designed for acquiring professional skills and knowledge. The objective of the course is to teach the fundamental principles of programming, with a focus on structured programming, and tools to support the development of software. Students will learn how to solve computational problems with well-designed programs The learning will be based on examples and practical assignments, from very simple ones to more complex. The final objective for the student is to acquire the ability to translate a set of functional and non-functional requirements into a software solution.
Course Topics	1. Introduction to: hardware and software, with computer organisation; data hierarchy; machine languages, assembly languages, high-level programming languages. Introduction to Python: interactive mode, script mode, Jupyter. 2. Introduction to different programming paradigms, focusing on the structured programming paradigm. 3. Structured programming: basic data types, variables, constants, operators and expressions; standard input/output handling; control flow structures; file and error handling. 4. Basic data structures/types of Python: (1) lists, (2) dictionaries, (3) tuples, (4) sets. 5. Subroutines and functions in Python (with/without parameters; with/without return). 6. Basics on modules and packages in Python. The above will be delivered meanwhile acquiring practical knowledge, through programming exercises, of how to program a simple physical-computing board with a Python-based language (e.g., Raspberry PicoH or PicoWH, ESP32, running MicroPython). Programming exercises cover the following: - how to perceive data via basic physical input devices (e.g., temperature sensor, humidity sensor), - how to process and store data, - how to plot data depending on their features.
Teaching Format	Frontal lectures, exercises, projects.

Required Readings	Made available in the course repository
Supplementary Readings	Made available in the course repository

Course Module

Course Constituent Title	Fundamentals of Programming II
Course Code	42426B
Scientific-Disciplinary Sector	INFO-01/A
Language	English
Lecturers	Dr. Sergio Tessaris, Sergio.Tessaris@unibz.it https://www.unibz.it/en/faculties/engineering/academic-staff/person/2315
Teaching Assistant	Dott. Muhammad Bilal Khan
Semester	Second semester
CP	5
Responsible Lecturer	
Teaching Hours	30
Lab Hours	20
Individual Study Hours	75
Planned Office Hours	15
Contents Summary	The course refers to the basic educational activities and belongs to the scientific area of Computer Science. The course is designed for acquiring professional skills and knowledge. The objective of the course is to teach the fundamental principles of programming, with a focus on structured programming, and tools to support the development of software. Students will learn how to solve computational problems with well-designed programs The learning will be based on examples and practical assignments, from very simple ones to more complex. The final objective for the student is to acquire the ability to translate a set of functional and non-functional requirements into a software solution.
Course Topics	The following topics will be covered by focusing on the C

	programming language and its specific features. Differences and similarities with Python will be outlined. 1. Introduction to C programming and toolchain 1.1. Understanding and using the compiler toolchains 1.2. Understanding cross-compilation 1.3. Tools to support modern software development 2. C language: syntax and data types 2.1. C standards 2.2. Control flow 2.3. Basic and derived types 3. C memory management and activation record 3.1. C memory organisation 3.2. Dynamic memory management 4. C programming techniques 4.1. Organisation of software artifacts in C 4.2. Effective use of C constructs and data types 4.3. Defensive programming techniques 5. Debugging and software testing 5.1. Techniques and strategies for effective debugging of code 5.2. Debugging tools 5.3. Unit testing
Teaching Format	Frontal lectures, practical assignments
Required Readings	Made available in the course repository
Supplementary Readings	Made available in the course repository