Syllabus
Course description

Course title	Fundamentals of Programming
Course code	42426
Scientific sector	INF/01
Degree	Bachelor in Electronics and Cyberphysical Systems
Semester	1 and 2
Academic year	1
Year	2024/2025
Credits	6+5
Modular	Yes

Total lecturing hours	40+30
Total lab hours	20+20
Total exercise hours	
Attendance	Highly recommended, in general. Mandatory for projects in the exercise classes of Module 1.
Prerequisites	Not foreseen.
Course page	Provided by teachers during the first classes.

Specific educational objectives	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 structural programming, and tools to support the development of software. Students will learn how to solve computational problems with well-designed programs for basic cyber-physical solutions for them. 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 for basic cyber-physical systems.

Module 1	
Lecturer	Rosella Gennari, gennari@inf.unibz.it
Scientific sector of the lecturer	INF/01
Teaching language	English

Office hours	By prior appointment via **mail** during the Module class timespan.
Lecturing assistant	_
Teaching assistant	Bilal Muhammad Khan <MuhammadBilal.Khan@student.unibz.it>
Office hours	By prior appointment via **mail** during the Module class timespan.
List of topics covered	1. Introduction to: hardware and software, with computer organisation; data hierarchy; machine languages, assembly languages, high-level programming languages. 2. Introduction to Python: interactive mode, script mode, Jupyter. 3. Introduction to different programming paradigms, focusing on the structured programming paradigm. 4. Structured programming: basic data types, variables, constants, operators and expressions; standard input/output handling; control flow structures; file and error handling. 5. Basic data structures/types of Python: (1) lists, (2) dictionaries, (3) tuples, (4) sets. 6. Subroutines and functions in Python (with/without parameters; with/without return); functions and basic recursion in Python, e.g., some combinatorics. 7. Basics of computational thinking to solve a computational problem and program a resolution in Python and Python-based languages, via physical-computing boards. The above will be delivered meanwhile acquiring practical knowledge, through programming exercises, of how to program an IoT board with a Python-based language (e.g., Raspberry PicoH or PicoWH, ESP32, running MicroPython). Specifically, programming exercises cover the following: - how to perceive data via physical input devices (e.g., temperature sensor, humidity sensor), - how to process, communicate and store data, - how to react via physical output devices (e.g., LEDs, buzzers), - how to plot data depending on their features.
Teaching format	Frontal lectures, exercises, projects.

Module 2	
Lecturer	Sergio Tessaris <sergio.tessaris@unibz.it> (lectures) Muhammad Azfar Yaqub <muhammadazfar.yaqub@unibz.it> (labs)
Scientific sector of the lecturer	INF/01

Teaching language	English
Office hours	
Lecturing assistant	N/A
Teaching assistant	TBA
Office hours	By prior appointment via **mail** during the Module class timespan.
List of topics covered	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. Introduction Memory management and activation record 1. Introduction to software development a. Organisation of software artifacts in C b. Effective use of C constructs and data types c. Defensive programming techniques 2. Software development toolchain a. Understanding and using the compiler toolchains b. Understanding cross-compilation 3. Tools to support modern software development (IDEs; software management tools: DVCS and cloud-based tools) a. The use of editing tools for software development b. Tools for collaborative software development c. Best practices for developing software artifacts 4. Debugging and software testing (debugging tools; writing safe and secure programs; type checking) a. Understanding risks associated to poor programming practices b. Use of tools for identifying common problems in programs c. Techniques and strategies for effective debugging of code
Teaching format	Frontal lectures, practical assignments.

Learning outcomes	**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. **Assessment for non-attending students**. The assessment is divided into 2 parts: 1. Module 1: a final written exam, with knowledge-related questions, their understanding and application; it is a pencil-and-paper, closed-book exam; 2. Module 1: assignments, based on exercises of the module, for applying knowledge, making judgments and communicating; assignments are with pencil and paper, closed book. **Assessment for attending students**. The assessment is divided into 4 parts. However, attending students can - take a mid-semester intermediate exam for Module 1, which is similar to the final-written exam for Module 1, and replaces this (only this, and not that of Module 2),

	- develop a mid-semester project, based on exercises of the module, that replaces its assignments (only these, not those of Module 2); they submit the project and deliver a mid-semester presentation to demonstrate a working prototype for their project and assess it critically. *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.
Assessment language	Course language.
Evaluation criteria and criteria for awarding marks	A student passes the exam only if the student has a positive result (i.e., not less than 18) and tackles <u>all</u> 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 50%, and the written exam counts for 50% 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 Submission of assignments within their published deadlines will grant extra points for the mark;

	○ Only some of the assignments, clearly indicated beforehand, will contribute to the mark; ○ Intermediate evaluations might be issued using different grading scales (e.g. percentage) and converted to the 0-30 scale using linear interpolation. 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. Subject Librarian: David Gebhardi, David.Gebhardi@unibz.it and Ilaria Miceli, Ilaria.Miceli@unibz.it
Supplementary readings	Material is provided during the course.