Syllabus

Descrizione corso

Titolo insegnamento	Fondamenti di Programmazione
Codice insegnamento	42426
Titolo aggiuntivo	
Settore Scientifico-Disciplinare	INFO-01/A
Lingua	Inglese
Corso di Studio	Corso di laurea in Ingegneria elettronica e dell'Informazione
Altri Corsi di Studio (mutuati)	
Docenti	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
Assistente	dott. Muhammad Bilal Khan
Semestre	Tutti i semestri
Anno/i di corso	1
CFU	11
Ore didattica frontale	70
Ore di laboratorio	60
Ore di studio individuale	145
Ore di ricevimento previste	33
Sintesi contenuti	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.
Argomenti dell'insegnamento	Module 1: Module 1 introduces the fundamental concepts of programming and computational problem solving through Python and simple physical-computing platforms (e.g., Raspberry Pi Pico). The focus is on acquiring transferable programming skills while interacting with sensors and simple embedded systems. 1. Introduction to computing systems and programming 1.1. Hardware and software 1.2. Basic computer organisation 1.3. Data hierarchy and data representation 1.4. Machine language, assembly language, and high-level programming languages 1.5. Introduction to Python: interactive mode, script mode, and Jupyter notebooks 2. Fundamentals of algorithm and program design 2.1. Algorithms: input, output, definiteness, finiteness, and effectiveness 2.2. Algorithm specification and design using pseudocode and flowcharts 2.3. Fundamental notions of correctness, termination, and computational cost 2.4. Programs and programming paradigms 2.5. Structured programming paradigm: sequencing, selection, and iteration 3. Fundamental programming constructs in Python 3.1. Basic data types 3.2. Variables, constants, operators, and expressions 3.3. Standard input/output handling 3.4. Control-flow structures 3.5. Error and exception handling

	4. Basic data structures in Python 4.1. Lists 4.2. Tuples 4.3. Dictionaries 4.4. Sets 5. Functions, modules, and code reuse 5.1. Functions and subroutines 5.2. Parameters and return values 5.3. Variable scope 5.4. Modules and packages 6. File handling and Physical Computing with MicroPython 6.1. File input/output 6.2. Programming microcontroller-based systems using MicroPython (e.g., Raspberry Pi Pico/Pico W, ESP32) 6.3. Acquisition of data from sensors and input devices 6.4. Processing and storage of acquired data 6.5. Displaying data and display devices _____ 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
Parole chiave	Programming Fundamentals; Python; Physical Computing; C; Software management and testing
Prerequisiti	
Insegnamenti propedeutici	
Modalità di insegnamento	Lectures, exercises, laboratory activites
Obbligo di frequenza	Strongly recommended
Obiettivi formativi specifici e risultati di apprendimento attesi	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.
Obiettivi formativi specifici e risultati di apprendimento attesi (ulteriori 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.
Modalità di esame	MODULE 1: ASSESSMENT Module 1 is assessed through a written examination and, for attending students, an optional laboratory project. NOTA BENE: "attending students" are students who attend at least 70% of the ENTIRE course. The written examination consists of two parts: * Part I: theoretical questions and Python programming exercises; * Part II: MicroPython programming exercises related to physical-computing applications. Attending students are eligible to submit an optional mini-project developed during the laboratory activities. The project may contribute up to 3 additional marks to the final Module 1 grade (see details below). The mark of the project remains valid for one academic year and cannot be carried forward beyond that period. The mark of the written examination is valid only for the examination session in which it is achieved. ---------- MODULE 2: ASSESSMENT Assessment is divided in two parts:

	1. Written final closed-book exam, with review questions about the lecture material and programming exercises. 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.
Criteri di valutazione	PASS CRITERIA A student passes the course only if they: obtain a passing grade (minimum 18/30) in both Module 1 and Module 2; and complete all compulsory assessment components by the specified deadlines. --- MARKING SCHEME The final course grade is the weighted average of the grades obtained in Module 1 (60%) and Module 2 (40%). Module 1 (0–30): Written examination: graded on a scale from 0 to 30. Optional laboratory project: attending students (minimum 70% attendance) may submit a project developed during the laboratory activities. The project may contribute up to 3 additional marks to the Module 1 grade. Module 2 (0–30): The mark for Module 2 ranges from 0 to 30: the assignments count for 40%, and the written exam counts for 60% of the mark. ---

	EVALUATION CRITERIA Written examinations are assessed on the basis of correctness, completeness, and clarity. Assignments and projects are assessed on the basis of: quality of the solution, according to the criteria presented in class and documented in the course materials; problem-solving skills; communication skills; critical-thinking skills. --- NOTA BENE 1. All assignments/projects must be submitted by the specified deadline. Failure of doing so will result in an incomplete submission and non-admission to the final evaluation. 2. In the case of Module 2, only the assignments, clearly indicated beforehand, will contribute to the mark. 3. The award of cum laude is jointly decided by the course lecturers and can be granted only to students who obtain a grade of 30/30 in both Module 1 and Module 2.
Bibliografia obbligatoria	Material is provided during the course.
Bibliografia facoltativa	Material is provided during the course.
Altre informazioni	Subject Librarian: David Gebhardi, David.Gebhardi@unibz.it and Ilaria Miceli, Ilaria.Miceli@unibz.it
Obiettivi di Sviluppo Sostenibile (SDGs)	Innovazione e infrastrutture, Istruzione di qualità

Modulo del corso

Titolo della parte costituente del corso	Fondamenti di Programmazione I
Codice insegnamento	42426A
Settore Scientifico-Disciplinare	INFO-01/A
Lingua	Inglese
Docenti	prof. Rosella Gennari, Rosella.Gennari@unibz.it https://www.unibz.it/en/faculties/engineering/academic-staff/person/8607
Assistente	dott. Muhammad Bilal Khan
Semestre	Primo semestre
CFU	6
Docente responsabile	
Ore didattica frontale	40
Ore di laboratorio	20
Ore di studio individuale	90
Ore di ricevimento previste	18
Sintesi contenuti	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.
Argomenti dell'insegnamento	1. Introduction to computing systems and programming: hardware and software; basic computer organisation; data hierarchy and data representation; machine language, assembly language, and

	high-level programming languages. Introduction to Python: interactive mode, script mode, and Jupyter notebooks. 2. Fundamentals of algorithms and program design: algorithms (input, output, definiteness, finiteness, and effectiveness); algorithm specification and design using pseudocode and flowcharts; fundamental notions of correctness, termination, and computational cost; programs and programming paradigms, with a focus on the structured programming paradigm. 3. Fundamental programming constructs in Python: basic data types, variables, constants, operators and expressions; standard input/output handling; control-flow structures (sequencing, selection, and iteration); error and exception handling. 4. Basic data structures in Python: lists, tuples, dictionaries, and sets. 5. Functions, modules, and code reuse in Python: functions and subroutines (with and without parameters; with and without return values); variable scope; modules and packages. 6. File handling and physical computing with MicroPython: file input/output; programming microcontroller-based systems (e.g., Raspberry Pi Pico/Pico W, ESP32); acquisition of data from sensors and input devices; processing and storage of acquired data; displaying data through display devices.
Modalità di insegnamento	Interactive lectures, guided exercises and quizzes, hands-on Python and MicroPython laboratory activities, and optional project work.
Bibliografia obbligatoria	Made available in the course repository
Bibliografia facoltativa	Made available in the course repository

Modulo del corso

Titolo della parte costituente del corso	Fondamenti di Programmazione II

Codice insegnamento	42426B
Settore Scientifico-Disciplinare	INFO-01/A
Lingua	Inglese
Docenti	dr. Sergio Tessaris, Sergio.Tessaris@unibz.it https://www.unibz.it/en/faculties/engineering/academic-staff/person/2315
Assistente	dott. Muhammad Bilal Khan
Semestre	Secondo semestre
CFU	5
Docente responsabile	
Ore didattica frontale	30
Ore di laboratorio	20
Ore di studio individuale	75
Ore di ricevimento previste	15
Sintesi contenuti	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.
Argomenti dell'insegnamento	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
Modalità di insegnamento	Frontal lectures, practical assignments
Bibliografia obbligatoria	Made available in the course repository
Bibliografia facoltativa	Made available in the course repository