

# Syllabus

## *Descrizione corso*

<b>Titolo insegnamento</b>	Fondamenti di Programmazione
<b>Codice insegnamento</b>	42426
<b>Titolo aggiuntivo</b>	
<b>Settore Scientifico-Disciplinare</b>	INF/01
<b>Lingua</b>	Inglese
<b>Corso di Studio</b>	Corso di laurea in Ingegneria elettronica e dell'Informazione
<b>Altri Corsi di Studio (mutuati)</b>	
<b>Docenti</b>	dr. Sergio Tessaris, Sergio.Tessararis@unibz.it <a href="https://www.unibz.it/en/faculties/engineering/academic-staff/person/2315">https://www.unibz.it/en/faculties/engineering/academic-staff/person/2315</a> prof. Rosella Gennari, Rosella.Gennari@unibz.it <a href="https://www.unibz.it/en/faculties/engineering/academic-staff/person/8607">https://www.unibz.it/en/faculties/engineering/academic-staff/person/8607</a>
<b>Assistente</b>	dott. Muhammad Bilal Khan
<b>Semestre</b>	Tutti i semestri
<b>Anno/i di corso</b>	1
<b>CFU</b>	11
<b>Ore didattica frontale</b>	70
<b>Ore di laboratorio</b>	40
<b>Ore di studio individuale</b>	165
<b>Ore di ricevimento previste</b>	33
<b>Sintesi contenuti</b>	<p>The course refers to the basic educational activities and belongs to the scientific area of Computer Science.</p> <p>The course is designed for acquiring professional skills and knowledge.</p> <p>The objective of the course is to teach the fundamental principles of programming, with a focus on structured programming, and</p>

	<p>tools to support the development of software.</p> <p>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.</p> <p>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.</p>
<b>Argomenti dell'insegnamento</b>	<p>Module 1:</p> <ol style="list-style-type: none"> <li>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.</li> <li>2. Introduction to different programming paradigms, focusing on the structured programming paradigm.</li> <li>3. Structured programming: basic data types, variables, constants, operators and expressions; standard input/output handling; control flow structures; file and error handling.</li> <li>4. Basic data structures/types of Python: (1) lists, (2) dictionaries, (3) tuples, (4) sets.</li> <li>5. Subroutines and functions in Python (with/without parameters; with/without return).</li> <li>6. Basics on modules and packages in Python.</li> </ol> <p>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:</p> <ul style="list-style-type: none"> <li>- how to perceive data via basic physical input devices (e.g., temperature sensor, humidity sensor),</li> <li>- how to process and store data,</li> <li>- how to plot data depending on their features.</li> </ul> <p>Module 2:</p> <p>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.</p> <ol style="list-style-type: none"> <li>1. Introduction to C programming and toolchain <ol style="list-style-type: none"> <li>1.1. Understanding and using the compiler toolchains</li> </ol> </li> </ol>

	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
<b>Parole chiave</b>	Programming Fundamentals; Python; Physical Computing; C; Software management and testing
<b>Prerequisiti</b>	
<b>Insegnamenti propedeutici</b>	
<b>Modalità di insegnamento</b>	Lectures, exercises, laboratory activities
<b>Obbligo di frequenza</b>	Strongly recommended
<b>Obiettivi formativi specifici e risultati di apprendimento attesi</b>	<p>The course refers to the basic educational activities and belongs to the scientific area of Computer Science.</p> <p>The course is designed for acquiring professional skills and knowledge.</p> <p>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.</p> <p>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.</p> <p>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.</p>
<b>Obiettivi formativi specifici e risultati di apprendimento</b>	Knowledge and understanding <ul style="list-style-type: none"> <li>Know the fundamental principles of programming.</li> </ul>

<b>attesi (ulteriori info.)</b>	<ul style="list-style-type: none"> <li>• Know different programming paradigms and models of computation.</li> <li>• Have a solid knowledge of the most important data structures and programming techniques.</li> </ul> <p>Applying knowledge and understanding</p> <ul style="list-style-type: none"> <li>• Be able to solve problems using programming.</li> <li>• Be able to develop small and medium size programs starting from given requirements.</li> </ul> <p>Making judgements</p> <ul style="list-style-type: none"> <li>• Be able to collect and interpret useful data and to judge information systems and their applicability.</li> <li>• Be able to identify an appropriate programming paradigm and data structures to solve a given problem.</li> </ul> <p>Communication skills</p> <ul style="list-style-type: none"> <li>• Be able to describe and motivate the software design choices.</li> <li>• Be able to properly document a software artifact to ensure its integration in more complex systems.</li> </ul> <p>Learning skills</p> <p>Be able to learn how to use different procedural programming languages in autonomy, by identifying and understanding the relevant literature.</p>
<b>Modalità di esame</b>	<p>Module 1</p> <p>Attending students are those that</p> <ul style="list-style-type: none"> <li>- attend at least 70% of the exercise classes of the module, i.e., at least 14 hours (hard constraint),</li> <li>- participate in class with a positive and reflective attitude,</li> <li>- show a commitment in tackling the class exercises for learning, taking due care of deadlines and instructions.</li> </ul> <p>The assessment is as follows:</p> <ul style="list-style-type: none"> <li>- For the lecture part: a written, closed-book exam with closed-ended and open-ended questions for all students;</li> <li>- For the exercise part: a programming project for attending students; a written exam for non-attending students.</li> </ul> <p>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.</p> <p>-</p> <p>Note: in case of a positive outcome, the intermediate exam, assignments and project work are valid for 1 academic year only</p>

	<p>and cannot be carried over beyond that time-frame.</p> <p>Module 2</p> <p>Assessment will be the same for attending and non-attending students. It's divided in two parts:</p> <ol style="list-style-type: none"> <li>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.</li> <li>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.</li> </ol>
<b>Criteri di valutazione</b>	<p>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.</p> <p>The result is the average of the marks for Modules 1 and 2. The marks for Modules 1 and 2 are given as follows:</p> <ul style="list-style-type: none"> <li>- 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);</li> <li>- 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: <ul style="list-style-type: none"> <li>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;</li> <li>o Only some of the assignments, clearly indicated beforehand, will contribute to the mark.</li> </ul> </li> </ul> <p>Laude is jointly decided by the course lecturers in case the marks for both modules is 30.</p> <p>E.g., suppose marks per Module are as follows:</p> <ul style="list-style-type: none"> <li>- Module 1's mark is 28 (25 for the written exam, 3 for the project);</li> <li>- Module 2's mark is 30.</li> </ul> <p>The result for the student is then 29, the average of 28 and 30.</p> <p>Written exam questions are evaluated in terms of correctness and clarity.</p>

	<p>Assignments/projects are evaluated in terms of:</p> <ul style="list-style-type: none"> <li>- quality, according to the criteria illustrated and explained in class, and recorded in the companion materials (e.g., code quality criteria),</li> <li>- displayed problem-solving skills,</li> <li>- displayed communication skills,</li> <li>- displayed critical-thinking skills.</li> </ul>
<b>Bibliografia obbligatoria</b>	Material is provided during the course.
<b>Bibliografia facoltativa</b>	Material is provided during the course.
<b>Altre informazioni</b>	Subject Librarian: David Gebhardi, David.Gebhardi@unibz.it and Ilaria Miceli, Ilaria.Miceli@unibz.it
<b>Obiettivi di Sviluppo Sostenibile (SDGs)</b>	Innovazione e infrastrutture, Istruzione di qualità

## *Modulo del corso*

<b>Titolo della parte costituente del corso</b>	Fondamenti di Programmazione I
<b>Codice insegnamento</b>	42426A
<b>Settore Scientifico-Disciplinare</b>	INF/01
<b>Lingua</b>	Inglese
<b>Docenti</b>	prof. Rosella Gennari, Rosella.Gennari@unibz.it <a href="https://www.unibz.it/en/faculties/engineering/academic-staff/person/8607">https://www.unibz.it/en/faculties/engineering/academic-staff/person/8607</a>
<b>Assistente</b>	dott. Muhammad Bilal Khan
<b>Semestre</b>	
<b>CFU</b>	6
<b>Docente responsabile</b>	
<b>Ore didattiche frontali</b>	40

Ore di laboratorio	20
Ore di studio individuale	90
Ore di ricevimento previste	
Sintesi contenuti	<p>The course refers to the basic educational activities and belongs to the scientific area of Computer Science.</p> <p>The course is designed for acquiring professional skills and knowledge.</p> <p>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.</p> <p>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.</p> <p>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.</p>
Argomenti dell'insegnamento	<ol style="list-style-type: none"> <li>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.</li> <li>2. Introduction to different programming paradigms, focusing on the structured programming paradigm.</li> <li>3. Structured programming: basic data types, variables, constants, operators and expressions; standard input/output handling; control flow structures; file and error handling.</li> <li>4. Basic data structures/types of Python: (1) lists, (2) dictionaries, (3) tuples, (4) sets.</li> <li>5. Subroutines and functions in Python (with/without parameters; with/without return).</li> <li>6. Basics on modules and packages in Python.</li> </ol> <p>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).</p> <p>Programming exercises cover the following:</p> <ul style="list-style-type: none"> <li>- how to perceive data via basic physical input devices (e.g., temperature sensor, humidity sensor),</li> <li>- how to process and store data,</li> </ul>

	- how to plot data depending on their features.
<b>Modalità di insegnamento</b>	Frontal lectures, exercises, projects.
<b>Bibliografia obbligatoria</b>	Made available in the course repository
<b>Bibliografia facoltativa</b>	Made available in the course repository

## *Modulo del corso*

<b>Titolo della parte costituente del corso</b>	Fondamenti di Programmazione II
<b>Codice insegnamento</b>	42426B
<b>Settore Scientifico-Disciplinare</b>	INF/01
<b>Lingua</b>	Inglese
<b>Docenti</b>	dr. Sergio Tessaris, Sergio.Tessaris@unibz.it <a href="https://www.unibz.it/en/faculties/engineering/academic-staff/person/2315">https://www.unibz.it/en/faculties/engineering/academic-staff/person/2315</a>
<b>Assistente</b>	dott. Muhammad Bilal Khan
<b>Semestre</b>	
<b>CFU</b>	5
<b>Docente responsabile</b>	
<b>Ore didattica frontale</b>	30
<b>Ore di laboratorio</b>	20
<b>Ore di studio individuale</b>	75
<b>Ore di ricevimento previste</b>	15
<b>Sintesi contenuti</b>	<p>The course refers to the basic educational activities and belongs to the scientific area of Computer Science.</p> <p>The course is designed for acquiring professional skills and knowledge.</p> <p>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.</p> <p>Students will learn how to solve computational problems with well-designed programs The learning will be based on examples and</p>

	<p>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.</p>
<b>Argomenti dell'insegnamento</b>	<p>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.</p> <ol style="list-style-type: none"> <li>1. Introduction to C programming and toolchain               <ol style="list-style-type: none"> <li>1.1. Understanding and using the compiler toolchains</li> <li>1.2. Understanding cross-compilation</li> <li>1.3. Tools to support modern software development</li> </ol> </li> <li>2. C language: syntax and data types               <ol style="list-style-type: none"> <li>2.1. C standards</li> <li>2.2. Control flow</li> <li>2.3. Basic and derived types</li> </ol> </li> <li>3. C memory management and activation record               <ol style="list-style-type: none"> <li>3.1. C memory organisation</li> <li>3.2. Dynamic memory management</li> </ol> </li> <li>4. C programming techniques               <ol style="list-style-type: none"> <li>4.1. Organisation of software artifacts in C</li> <li>4.2. Effective use of C constructs and data types</li> <li>4.3. Defensive programming techniques</li> </ol> </li> <li>5. Debugging and software testing               <ol style="list-style-type: none"> <li>5.1. Techniques and strategies for effective debugging of code</li> <li>5.2. Debugging tools</li> <li>5.3. Unit testing</li> </ol> </li> </ol>
<b>Modalità di insegnamento</b>	Frontal lectures, practical assignments
<b>Bibliografia obbligatoria</b>	Made available in the course repository
<b>Bibliografia facoltativa</b>	Made available in the course repository