UNIVERSITÀ DEGLI STUDI DI BARI ALDO MORO

CONSIGLIO INTERCLASSE IN MATEMATICA

COURSE OF STUDY THREE-YEAR BACHELOR PROGRAMME IN MATHEMATICS

ACADEMIC YEAR 2024-2025

ACADEMIC SUBJECT COMPUTER SCIENCE

General information	
Programme year	First
Term	First semester (September 30, 2024 – January 15, 2025)
European Credit Transfer and Accumulation System credits (ECTS)	6
SSD	INF/01
Language	Italian
Mode of attendance	Attendance is not mandatory, but it is strongly recommended.

Lecturer	
Name and surname	Ciro Castiello
E-mail	ciro.castiello@uniba.it
Telephone	+39 080 544 2135
Department and office	Department of Computer Science, room 669, 6th floor
Virtual meeting room	Piattaforma di e-learning UNIBA - https://elearning.uniba.it/
Web page	https://www.uniba.it/it/docenti/castiello-ciro
Office hours	The teacher will be available for consultations at the end of the lessons. Appointments can be also arranged by email (both in-person and online meetings can be organised).

Work schedule	Total	Lectures	Hands-on learning (recitations/laboratories)	Self-study
Hours	150	32	16	102
ECTS credits	6	4	2	

Learning objectives	
	Acquire basic concepts regarding computer architectures. Acquire basic concepts regarding programming methods and techniques. Apply these concepts to solve problems by using the Python language. Basic notions and concepts regarding algorithms and computational complexity.

Course prerequisites	
	No specific prior knowledge is required, apart from the basic secondary school subjects (in particular, the student must exhibit basic reasoning and logical-mathematical skills). Knowledge of English language (school level) is profitable.

Syllabus	
Course contents	Introduction to Computational Thinking - Noam Chomsky - The abstraction process - Trial-and-error learning and reuse of experience - Grammars, formalization and hierarchies, languages and computers

	Algorithms - Ada Lovelace - Definitions and preliminary concepts - Early machines and early programmers - Flowcharts and tools for designing algorithms Computability - Alan Turing - The Turing machine - Computational Complexity - The paradoxes and the halting problem Programming languages - Grace Hopper - Introduction to Python - Boolean values and logical operators - Test-driven code development Ordered structures - Donald Knuth - Data structures - Lists, stacks, queues Brute force - Betty Holberton - The sorting problem - The iteration concept - Linear search and insertion sort Unsorted structures - Jorge Luis Borges - The Library of Babel and the concept of infinity - Sets and dictionaries Recursion - Douglas Hofstadter - Little harmonic labyrinth - Recursion in the real world and in programs - Recursive algorithms Divide and conquer - John von Neumann - Introduction to computer architecture - Sorting large quantities of elements - The merge function and the merge sort algorithm - Complexity analysis The Python language - Python and Colab - From the first steps to the concept of variable - Basic syntax - Data types - Functions - Lists and tuples - Iterations - Sets and dictionaries - Recursion - File management - Guided exercises Other topics - Dynamic programming

	- Backtracking - Trees and graphs - Greedy algorithms
Reference books	S. Peroni – Computational Thinking and Programming (available online). Further material that may be suggested or provided by the teacher during the course.
Additional course materials	The teacher's personal learning material will be made available on the website dedicated to the course.
Repository	The website of the course is hosted on the e-learning platform of the University of Bari.

Expected learning outcomes	
Knowledge and understanding	- The student knows the high-level principles, as well as the historical and theoretical backgrounds, for solving problems efficiently by using computational tools and information-processing mechanisms. - Development of computational thinking. - Acquisition of basic programming principles.
Applying knowledge and understanding	- Understanding and using the main data structures for organising information. - Ability to develop algorithms for solving problems that are interesting from a computational point of view. - Ability to implement these algorithms in a specific programming language.
Soft skills	*Making judgements*: - Students must be able to navigate the general principles of computational thinking in order to apply techniques for the development and implementation of algorithms. - Students must be able to recognise and apply the appropriate data structures to be used in handling computational problems. - Students must be able to test the correctness of problem-solving models and code written in a programming language.
	Communication skills: - Student must be able to explain the topics included in the course programme using the specific vocabulary of the discipline.

Teaching methods	
	- Lectures with participatory teaching experiences. - Guided programming exercises.

Assessment	
Assessment methods	Carrying out a written test (to be performed, typically, on the computer in the laboratory). The test methods aim to ascertain the following skills: - Exposition (with reference to the theoretical topics forming part of the course programme) - Understanding (with reference to the techniques and models studied, with the proposal of case studies and exercises) - Development (with reference to the Python language, with the proposal of a description of a computational problem for which a solving algorithm must be defined, implemented and tested) The test has a minimum duration of 90 minutes. The use of teaching materials for Exposition and Understanding activities is not permitted.

	Consultation of all teaching materials for the Development activity is permitted. Booking on the esse3 platform (within the specified deadlines) is a prerequisite for participation in the test. The same platform is used for the publication of the test results. The results appear on esse3 within a few days of the conclusion of the test (generally no more than one week).
Evaluation criteria	With reference to the learning outcomes and the assessment methods presented above, the evaluation criteria aim specifically to verify: - Knowledge and understanding skills: Adequate knowledge of the formalisms and theoretical topics forming part of the course syllabus (Exposition activity), of the fundamental constructs and functions of the Python language (Development activity) and understanding of the models and techniques covered during the course (Understanding activity). - Applied knowledge and understanding: Effectiveness in developing solutions based on computational thinking by defining and/or analysing algorithms and applying appropriate data structures (Understanding and Development activities). - Making judgements: Ability to process and organise ideas critically and systematically (Exposition activity), ability to critically reason about the study carried out (Exposition and Understanding activities), ability to analyse own computational problem-solving proposals (Development activity). - Communication skills: Competence in the use of the specific vocabulary of the discipline with adequate quality of argumentation (Exposition activity), ability to synthesise and clarity of exposition in the presentation of one's proposals for solving the questions (Understanding and Development activities).
Grading policy	The different activities included in the test (Exposition, Understanding, Development) are marked. The final mark is derived from the sum of the marks obtained for each activity. The final mark is awarded in thirtieths. The examination is deemed passed when the mark is greater than or equal to 18.

Further information	
	Class attendance and interaction with the teacher is highly recommended.