**Programming Laboratory**

Prof. Daniele Toti

COURSE AIMS AND INTENDED LEARNING OUTCOMES

The course aims to provide students with the basic knowledge of programming from both a theoretical and practical perspective. The chosen "teaching" language is Python (version 3+), but comparisons with other programming languages, particularly Java, will be made during the course.

Programming skills are essential in most modern technical and scientific activities. With this in mind, the course has three main objectives:

* Provide methodological and technological tools to enable students who have never taken a programming course to develop programs with a certain level of autonomy by the end of the course.
* Help students who have basic programming knowledge consolidate and apply it in the Python language, as well as provide insights into modeling and design.
* Prepare students for more advanced courses, such as Object-Oriented Programming, by laying the foundations to tackle different programming languages and paradigms.

By the end of the course, students will have gained familiarity with the fundamental techniques of programming and will be able to develop moderately complex programs in the Python language. They will be acquainted with the procedural programming paradigm, have a basic understanding of object-oriented programming and the object-oriented programming paradigm, and be able to use additional libraries to perform more complex operations. In general, they will be able to start thinking like a computer scientist..

COURSE CONTENT

* Introduction to programming
* Introduction to the Python language
* Basic syntax, variables, assignments and primitive types
* Brief overview of Input/Output: reading and writing from the keyboard
* Programming environment and IDE: PyCharm
* Fundamental mechanisms: comments, Boolean expressions, conditional statements, loops, operators
* Function definition: signature, return type, variable scope, variable-length arguments and optional parameters
* Built-in functions in Python
* Using libraries
* Methodological approaches to program writing and understanding
* Fundamental data structures: strings and collections
* String operations
* Collection operations: lists, sets, tuples and maps/dictionaries
* File and file system management
* Exception handling
* Object-oriented paradigm and object-oriented programming in Python: classes, constructors, methods, encapsulation, inheritance, polymorphism, special methods
* Elements of software design: multi-tier architecture and introduction to design patterns
* Recursion and brief introduction to the performance analysis of algorithms
* Introduction to date and time management in Python
* Introduction to additional libraries and data structures for scientific computing, data visualization and processing: Numpy, Matplotlib and Pandas.

READING LIST

Lecture slides, exercise notes, solutions to completed exercises and online resources communicated to the students.

Recommended textbook:

* *“Thinking Python,” Second Edition, Italian translation.*

The textbook will be made available to students on the Blackboard platform along with the rest of the materials. The course will not strictly follow the textbook but will cover, integrate and delve into various topics. Numerous in-class exercises will be proposed, and attendance is strongly recommended.

TEACHING METHOD

Theory lectures, exercises with discussions of solutions and assignment of homework.

ASSESSMENT METHOD AND CRITERIA

The exam consists of a written test containing open-ended programming exercises and closed-ended questions. The exam may include problems to be solved by developing programs, portions of code to read and interpret, as well as questions on theoretical aspects covered in the course. Depending on technical and logistical possibilities, programming exercises may be performed directly on a computer.

Programming exercises will be evaluated based on their syntactic and semantic correctness, compliance with their expected behavior, clarity, comprehensibility, code elegance and overall functioning.

The work is individual and group work is not allowed. It is not permitted to consult books or notes during the exam.

In exceptional cases and at the discretion of the instructor, an additional oral verification may be requested after the correction of the written test for certain candidates. In general, this may apply to written tests that fall just below the passing grade or situations where potential irregularities during the written test are suspected. In any case, students cannot request an oral examination.

The final grade of the exam is determined by the score obtained in the written test, supplemented by any bonus points earned through the completion of homework assignments.

Bonus points are awarded to students who successfully complete the homework assignments during the course and are valid for the first exam attempt. They are reset immediately after, regardless of the exam result.

NOTES AND PREREQUISITES

The course does not have any formal prerequisites. However, familiarity with computer usage and basic knowledge of the topics covered in the Fundamentals of Computer Science course are recommended.

*Office hours and location*

Information regarding student office hours can be found on the instructor’s personal page.

*Further information can be found on the lecturer's webpage at http://docenti.unicatt.it/web/searchByName.do?language=ENG or on the Faculty notice board.*