OBIETTIVI FORMATIVI

Synthetic general description

The course presents the foundation of Computer Science and Programming by adopting Python as reference language.

The basic concepts of imperative programming are presented without neglecting the use of specific libraries for scientific computing. In particular, the learner is led towards the acquisition of the classical conceptual tools of structured programming through the description and use of built-in structures for data processing (lists, tuples, dictionaries), the coding of remarkable algorithms and the use of the technique of recursion. Throughout the course we will use a professional development environment for the Python language.

General educational objectives of teaching in terms of expected learning

1. Knowledge and understanding: the primary objective of the course is the students' acquisition of the "philosophy" of structured programming, as well as the detailed knowledge of syntax and semantics of the Python programming language. The course pays particular attention to the development of well-written and well-structured code using the basic techniques for software development in the imperative paradigm.

2. Applying knowledge and understanding: it intends to provide the tools to achieve the following practical and professional skills:
   - To analyze computational problems and to code algorithmic ideas for their resolution;
   - To design, to describe, to implement and debug Python programs with professional tools;
   - To use built-in data structures for data management in scientific computing;
   - Understanding simple recursive algorithms;
   - To use specific libraries for scientific calculation;
   - To read, to understand and analyze third-party Python code also in terms of efficiency;
   - To be able to read documentation of libraries.
3. **Making judgements**: through the examination of code examples and numerous practice exercises, the learner will be able, both independently and in a cooperative manner, to analyze problems and design and implement related software solutions.

4. **Communication skills**: the student will acquire the necessary communication skills and expressive appropriateness in the use of technical verbal language in the context of computer programming.

5. **Learning skills**: the course aims to provide the learner with the necessary theoretical and practical methodologies to be used in professional contexts and, in particular, the ability to formulate and implement *ad-hoc* algorithms for solving new computational problems as well as the possibility of easily and quickly acquiring other programming languages.

**MODALITÀ DI SVOLGIMENTO DELL’INSEGNAMENTO**

All the material shown (slides and code fragments) is made available to students in the *Teams platform*. Video projector is used for lectures in the classroom. The slides are not intended to replace the reference texts but represent a precise guide to the course topics.

In order to pursue the educational objectives of the course set out in the descriptors 2) *Applying knowledge and understanding* and 3) *Making judgments*:

- many lessons will be carried out in an interactive teacher-learners mode;
- practical classroom exercises are planned to familiarize the learners with the drafting of Python code;
- some case studies in the field of small software design will be discussed.

**PREREQUISITI RICHIESTI**

No prerequisite required.

**FREQUENZA LEZIONI**

Optional but suggested.

**CONTENUTI DEL CORSO**

1. **Introduction to programming**
   - c. Representation of information: integers and floating-point numbers, characters, strings, images and sounds (outline).

2. **Programming languages**
   - a. Programming languages: machine, assembly and high level.
c. Installation of the development environment for the Python language. First program: Editing, Running, Debugging.

3. **Constructs of Python language**
   a. Basic syntax, data types, predefined operators, I/O management.
   b. Numbers and mathematical functions.
   c. Flow control: constructs of selection and iterative.
   d. Functions.

4. **Built-in data structures in Python**
   a. Strings.
   b. Lists, Tuples, Dictionaries.

5. **Advanced topics**
   b. Recursive functions.
   c. Modules. Basics on the mathematical libraries *NumPy* and *SciPy* and the graphic library *PlotPy*.

---

**TESTI DI RIFERIMENTO**


**ALTRO MATERIALE DIDATTICO**

All the material shown (slides and code fragments) is made available to students in the Teams repository as well as on the Teacher's webpage.

---

**PROGRAMMAZIONE DEL CORSO**

<table>
<thead>
<tr>
<th>Argomenti</th>
<th>Riferimenti testi</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Introduction to Computer Science</td>
<td></td>
</tr>
<tr>
<td>2 Introduction to Programming</td>
<td></td>
</tr>
<tr>
<td>3 Programming Languages</td>
<td></td>
</tr>
<tr>
<td>4 Construct of Python Languages</td>
<td></td>
</tr>
<tr>
<td>5 Built-in data structures in Python</td>
<td></td>
</tr>
<tr>
<td>6 Advanced Topics</td>
<td></td>
</tr>
</tbody>
</table>
VERIFICA DELL'APPRENDIMENTO

MODALITÀ DI VERIFICA DELL'APPRENDIMENTO
Programming exercises.

Should teaching be carried out in mixed mode or remotely, it may be necessary to introduce changes with respect to previous statements, in line with the programme planned and outlined in the syllabus.

Learning assessment may also be carried out on line, should the conditions require it.

ESEMPI DI DOMANDE E/O ESERCIZI FREQUENTI
To implement a python program to solve a specific problem about data.