1. OVERVIEW

Subject area	Scientific Computing I
Degree	Bachelor's Degree in Physics
School/Faculty	School of Architecture, Engineering and Design
Year	First
ECTS	6 ECTS
Type	Core
Language(s)	Spanish
Delivery mode	On campus
Semester	First semester
Academic year	2022/2023
Coordinating professor	Manuel Martín Bravo
Teacher	Manuel Martín Bravo

2. INTRODUCTION

In this subject area, students learn the fundamentals of computer programming, focusing on its applications in the field of science. The aim is for students to become familiar with classical programming paradigms and to apply them in the design and development of simple applied programs. We have chosen to work with Python 3 as it is currently one of the most common programming languages and because it offers a good balance between being powerful and easy to use. We should stress that the subject area focuses more on algorithmic thinking and basic data structures than on the specifics of the chosen programming language. It also highlights the importance of neat programming style and good programming practices so students can go on to create clean, well-structured code.

As the subject area is taught in the first semester of the first year of the degree, students are not expected to have any previous knowledge of coding. The natural continuation of this subject area is Scientific Computing II, which is scheduled for the second semester of the first year. Together, these two subject areas provide a foundation in computational physics. Students can expand their knowledge of computational physics in elective subject areas in later years of the degree.

3. SKILLS AND LEARNING OUTCOMES

General skills (CG, by the acronym in Spanish):

- CG2. Ability to plan and perform independent work when managing projects associated with different areas of physics.

Key skills (CB, by the acronym in Spanish):

- CB3. Students have the ability to gather and interpret relevant data (usually within their study area) to form opinions which include reflecting on relevant social, scientific or ethical matters.

- CB5. Students have developed the learning skills necessary to undertake further study in a much more independent manner.

Transversal skills (CT, by the acronym in Spanish):

- CT5. Problem solving: Be able to critically evaluate information, separate complex situations into their constituent parts, recognise patterns, and consider alternatives, different approaches and perspectives in order to find optimal solutions and negotiate efficiently.
- CT6. Adaptability: Being able to accept, appreciate and integrate different positions, being able to adapt one's own approach as required by the situation, as well as working effectively in ambiguous situations.

Specific skills (CE, by the acronym in Spanish):

- CE05. To understand and know how to use the mathematical and numerical methods used in physics and in handling experimental data.
- CE07. To use the most suitable electronic instruments and IT tools to study physical problems and search for solutions.

Learning outcomes (RA, by the acronym in Spanish):

- RA1. To understand the physical and logical structure of a computer.
- RA2. To build a solid foundation in the use of programming languages that are commonly used in science, being able to solve applied physical and/or mathematical problems by coding simple programs.
- RA3. To understand the principles of both generic and object-orientated procedural programming.

The following table shows how the skills developed in the subject area match up with the intended learning outcomes:

Skills	Learning outcomes
CB5, CE07	**RA1.** To understand the physical and logical structure of a computer.
CG2, CB3, CT5, CE05	**RA2.** To build a solid foundation in the use of programming languages that are commonly used in science, being able to solve applied physical and/or mathematical problems by coding simple programs.
CB5, CT6, CE07	**RA3.** To understand the principles of both generic and object-orientated procedural programming.

4. CONTENTS

This subject is organised into learning units.

Unit 1. Operating systems and programming languages.

- 1.1. Overview of operating systems.
- 1.2. Overview of programming languages.

Unit 2. Basic elements of programming.

- 2.1. Conditional loops and control structures. ▯ 2.2. Functions.

Unit 3. Input and output.

- 3.1. Formats. Data input.
- 3.2. Screen output, writing files.

Unit 4. Introduction to object-orientated programming.

- 4.1. Classes, scope of definitions. 4.2. Inheritance.

Unit 5. Data structures and algorithms.

- 5.1. Natural language, pseudocode, flowcharts.
- 5.2. Data structures: tuples, lists, sequences, sets, dictionaries.

5. TEACHING/LEARNING METHODS

The types of teaching/learning methods are as follows:

- Collaborative learning: Students learn to collaborate with other people (classmates and professors) in order to find creative, comprehensive and constructive solutions to questions and problems that arise from the given case studies, using all relevant knowledge and material resources available.
- Problem-based learning: Students are given problems and asked to solve them, working individually or in groups.
- Lectures: Presentations by the professor with the necessary technological tools to maximise comprehension of the learning content.
- Workshop-based learning: Students acquire knowledge through learning to use the tools and equipment needed in their profession. In other words, "learning by doing".
- Guided academic activities: Individual and group work that is more independent, including information searches, written summaries, debates and the public defence of projects.

6. LEARNING ACTIVITIES

The types of learning activities, plus the amount of time spent on each activity, are as follows:

On campus:

Learning activity	Number of hours
Lectures	14
Asynchronous lectures	12
Oral presentations of projects and debates	2
Report writing	13.5
Assessment	6
Practical activities (problems, written work, projects, workshops and/or lab work)	38
Group tutorials	10
Independent working	54.5
TOTAL	**150**

7. ASSESSMENT

The assessment systems, plus their weighting in the final grade for the subject area, are as follows:

On campus:

Assessment system	Weighting
Individual on-campus knowledge tests (theory and/or practice)	50%
Oral defence	5%
Submission of group and/or individual reports, written work, projects or exercises	35%
Performance observation	10%

On the Virtual Campus, when you open the subject area, you'll find details of your assessable tasks, including the submission dates and assessment procedures for each task.

8. BIBLIOGRAPHY

There are many introductory guides to Python programming that are suitable for this subject area. This bibliography includes a basic guide that includes lots of detailed examples (Gaddis 2019) and a more advanced guide that focuses on data analysis (Guttag 2016).

- Tony Gaddis (2021). *Starting Out with Python,* 5th edition. Editorial Pearson Education. Nueva York.
- John V. Guttag (2016). *Introduction to Computation and Programming Using Python: With Application to Understanding Data*, 2nd edition. Editoral The MIT Press. Cambridge, MA.

In addition to specialised guides, there are lots of high-quality online resources that are also very useful for this subject area. We recommend the following:

- Official Python documentation, available on the Python Software Foundation website [US]: https://www.python.org/doc/
- Online Python Tutor (Philip Guo), a free online tool where users can execute Python 3 code line by line: http://www.pythontutor.com/
- Python Course (Bernd Klein), a free, high-quality and up-to-date online course, available ad-free in English and German: https://www.python-course.eu/