UAB
Universitat Autònoma de Barcelona

Advanced Programming

Code: 104338
ECTS Credits: 6

2025/2026

Degree	Type	Year
Data Engineering	FB	1

Contact

Name: Jordi Casas Roma

Email: jordi.casas.roma@uab.cat

Teachers

(External) Cristobal Pio Garcia

Teaching groups languages

You can view this information at the end of this document.

Prerequisites

The subject does not have any official prerequisite. However, students should be familiar with the most basic programming structures. Thus, it is recommended for students to have taken the course "Fundamentals of programming".

Objectives and Contextualisation

This subject must be regarded as the logical continuation of the subject of "Fundamentals of Programming". The basic objective is to expand the basic notions of programming introduced in "Fundamentals of Programming" and to complete them with the principles of object-oriented programming (OOP).

Object-oriented programming is one of the most widespread design and programming methodologies. In object-oriented programming, software is organized into classes that contain methods (procedures) and attributes (data). The instances or objects of these classes send messages to each other (they make "calls" to methods of other objects) thus achieving the desired functionality. Apart from the concept of class, three other fundamental elements of the OOP are inheritance, composition and separation of the public and private parts of the classes.

During the course, we will explain all these concepts of the OOP and we will see how we can use them to build solutions that are easily extensible, since a basic law of software is that changes are inevitable. Therefore, we will also explain a set of design principles or heuristics that tell us how the elements of the OOP have to be used. These principles are used to design common solutions to recurrent problems, which often appear in seemingly independent applications.

In this way, the training objectives of the subject are the following:

- Understand and correctly apply the basic principles of object-oriented programming: concept of class, object and data encapsulation.
- Understand and apply the principles of inheritance and class composition to real problems.
- Use the most appropriate programming and representation structures for the implementation of an algorithm.
- Acquire the ability to design and program algorithms to solve complex problems, applying object oriented design patterns and principles.
- Understand the software life cycle for the resolution of programming problems: problem analysis, design, implementation and testing.
- Develop the programs following a coding style aimed at achieving high-quality programs.

Competences

- Demonstrate sensitivity towards ethical, social and environmental topics.
- Design efficient algorithmic solutions to computational problems, implement them in the form of robust software developments which are structured and easy to maintain, and verify their validity.
- Make a critical evaluation of work carried out.
- Plan and manage the available time and resources.
- Search, select and manage information and knowledge responsibly.
- Students must have and understand knowledge of an area of study built on the basis of general secondary education, and while it relies on some advanced textbooks it also includes some aspects coming from the forefront of its field of study.

Learning Outcomes

1. Apply strategies for debugging, testing and correcting programmes.
2. Apply the basic principles of computer structure and programming.
3. Demonstrate sensitivity towards ethical, social and environmental topics.
4. Develop programmes that are well documented, using a good programming style.
5. Implement medium-difficulty algorithmic problems in a programming language.
6. Make a critical evaluation of work carried out.
7. Plan and manage the available time and resources.
8. Search, select and manage information and knowledge responsibly.
9. Students must have and understand knowledge of an area of study built on the basis of general secondary education, and while it relies on some advanced textbooks it also includes some aspects coming from the forefront of its field of study.

Content

- Review of basic programming structures
  - Variables and types
  - Iterators, functions and files
  - Coding style
  - Debugging
- Object-Oriented Programming (OOP)
  - Abstraction and encapsulation
  - Inheritance
  - Polymorphism
  - Abstract classes
  - Class and sequence diagrams
- Scientific libraries
  - Numpy

    - Pandas
- Logging, documentation and testing
    - Logging
    - Documentation
    - Error checking and detection
- OO design
    - GRAPS patterns
    - Design principles

Activities and Methodology

Title	Hours	ECTS	Learning Outcomes
Type: Directed			
Lectures	50	2	2, 6, 8, 3, 5, 9
Type: Supervised			
Implementation of the programming project	50	2	2, 1, 4, 5, 7
Type: Autonomous			
Individual study	14	0.56	8, 7
Problem resolution	32	1.28	2, 1, 6, 4, 5, 7

The teaching methodology of the subject starts from the principle that "programming is the only way to learn to program" and, therefore, will be mainly focused on the practical work of the student. Classroom sessions will be organized to discuss the theoretical contents of the subject, from a very practical perspective with examples, and exercises and programming problems that must be solved in class directly with the computer. On the other hand, a programming project will be carried out that will have to be developed mainly autonomously throughout the course (with follow-up and control by the professor in specific sessions) and which will involve practically integrating almost all concepts and programming tools introduced in lectures to solving a complex real problem. In addition, a set of exercises will have to be solved individually throughout the course (some of them will be evaluated and discussed in lectures). They should serve to understand, integrate and apply the concepts developed during the course.

Lectures will be organized in four hours a week with no distinction among theory, problems and practices. It is encouraged that the student bring his own laptop to class if he has one. In the lectures, the concepts of the syllabus of the subject will be discussed. In some cases, explanatory videos or other supplementary material may be made available to the student to watch before the class session. Lectures will have a very practical approach with examples and exercises that will be presented to students to facilitate understanding and learning of the concepts.

The student must complete the lectures with autonomous personal work to do the exercises that are proposed and that should serve to understand the contents of the course. It must be borne in mind that the syllabus of the subject hasa logical continuity throughout the course, so that to follow correctly a class it is necessary to understand what was explained in the previous sessions.

In addition, students must develop in groups (of two people) a programming project that will be developed autonomously throughout the course apart from the lectures. The programming project will allow addressing a programming problem of a certain complexity that integrates most of the concepts explained during the course. During the course, some lectures sessions will be devoted to the control, monitoring and evaluation of the work carried out by the student in the programming project.

Annotation: Within the schedule set by the centre or degree programme, 15 minutes of one class will be reserved for students to evaluate their lecturers and their courses or modules through questionnaires.

Assessment

Continous Assessment Activities

Title	Weighting	Hours	ECTS	Learning Outcomes
Individual assessment	60%	4	0.16	2, 5, 9
Programming project	40%	0	0	2, 1, 6, 8, 3, 4, 5, 7

The single assessment model is not applied in this subject.

Subject assessment will take into account two types of assessment activities: individual assessment and programming project.

The final grade of the subject is obtained by combining the grade of these activities as follows:

- Final Grade = (0.4 * Project) + (0.6 * Individual assessment)

Project

Includes the assessment of all the work carried out in the programming project:

- A minimum grade of 5 is required in the final delivery of the project in order to pass the subject.
- The project is developed in groups of 2 people.
- Each group will have to make a public presentation of the project in front of the rest of the class.
- The project must be delivered within the established deadline, with no possibility of recovery beyond the deadline.

Individual assessment

This section includes the results of the individual tests that will be done throughout the course. There will be two partial tests that will be done during the period of lectures in thecourse during class time and a final test during the official exam period. This final test will permit to compensate any of the partial tests and will only have to be done by students who have not passed any of the two partial test. If one of the two partial test has been passed, but the other does not, in this test only the part of the subject corresponding to the partial test that has not been passed must be re-assessed.

- A minimum grade of 5 is required in each of the two partial tests in order to pass the subject.
- Thefinal grade will be the average of the two partial tests:

Individual assessment grade = (0.5 * Partial Test 1) + (0.5 * Partial Test 2)

- A minimum grade of 5 is required in the final grade of the individual assessment in order to pass the subject.

Final grades

- Not assessed: A student will be considered not assessed (NA) if he does not submit at least 50% of the deliveries of exercises and does not do any of the evaluation tests: partial test 1, partial test 2, final test, final delivery of the project.

- Suspended: If the final grade is equal to or greater than 5, but the student does not reach the minimum required in any of the assessment activities, the final grade will be suspended and the grade in the Transcript of Records (ToR) will be 4.5.
- Pass the course with honours: In order to pass the course with honours, the final grade must be a 9.0 or higher. Because the number of students with this distinction cannot exceed 5% of the number of students enrolled in the course, this distinction will be awarded to whoever has the highest final grade.

Compensations

For the students who are taking the courses for thesecond time or more, the project of the previous year can be compensated if these conditions are met:

- The final grade of the project of the previous year is greater than or equal to 7
- The grade of the individual assessment of the previous year is greater than or equal to 3

In this case, the grade of the project will be 5, independently of the grade of the previous year.

Review of assessment

For each assessment activity, a place, date and time of review will be indicated allowing students to review the activity with the lecturer. In this context, students may discuss the activity grade awarded by the lecturers responsible for the subject. If students do not take part in this review, no further opportunity will be made available.

Important note: copies and plagiarism

Notwithstanding other disciplinary measures deemed appropriate, and in accordance with the academic regulations in force, assessment activities will receive a zero whenever a student commits academic irregularities that may alter such assessment. Assessment activities graded in this way and by this procedure will not be re-assessable. If passing the assessment activity or activities in question is required to pass the subject, the awarding of a zero for disciplinary measures will also entail a direct fail for the subject, with no opportunity to re-assess this in the same academic year. Irregularities contemplated in this procedure include, among others:

- the total or partial copying of a practical exercise, report, or any other evaluation activity;
- allowing others to copy;
- presenting group work that has not been done entirely by the members of the group;
- presenting any materials prepared by a third party as one's own work, even if these materials are translations or adaptations,including work that is not original or exclusively that of the student;
- having communication devices (such as mobile phones, smart watches, etc.) accessible during theoretical-practical assessment tests (individual exams).
- talk with peers during the theoretical evaluation tests-individual practices (exams);
- copy or attempt to copy other students during the theoretical-practical assessment tests (exams);
- use or attempt to use writings related to the subject during the performance of theoretical-practical assessment tests (exams), when these have not beenexplicitly allowed.

In these cases, the grade in the Transcript of Records (ToR) will be the lowest value between 3.0 and the weighted average grade (and therefore re-assessment will not be possible).

In the assessment of problems and the project, tools to detect code plagiarism will be used.

Important note: Use of AI

In this subject, the use of Artificial Intelligence (AI) technologies is allowed as an integral part of the development of the work, provided that the final result reflects a significant contribution of the student in the analysis and personal reflection. The student must clearly identify which parts have been generated with this technology, specify the tools used and include a critical reflection on how these have influenced the process and the final result of the activity. The lack of transparency in the use of AI will be considered a lack of academic honesty and may lead to a penalty in the grade of the activity, or greater sanctions in serious cases

Bibliography

- Introduction to Computation and Programming Using Python (Third Edition). John V. Guttag. The MIT Press, 2021.
- Python: master the art of design patterns. D. Phillips, C. Giridhar, S. Kasampalis. Packt Publishing, 2016.
- Mastering object-oriented Python. Steven F. Lott. Packt publishing, 2014.
- Head first object oriented analysis and design. B.D. McLaughlin, G. Pollice, D. West. O'Reilly, 2007.
- Design patterns explained simply. Alexander Shvets. https://sourcemaking.com/design-patterns-ebook.
- Head first design patterns. E. Freeman, E. Freeman, K. Sierra, B. Bates. O'Reilly, 2004.
- Clean code: a handbook of agile software craftmanship. R.C. Martin. Prentice Hall, 2008. O la versió en castellà, Código limpio, d'Anaya
  Multimedia 2012.

Software

Any programming environment in Python could be used, such as PyCharm, Spyder or Visual Studio Code.

Groups and Languages

Please note that this information is provisional until 30 November 2025. You can check it through this link. To consult the language you will need to enter the CODE of the subject.

Name	Group	Language	Semester	Turn
(PAUL) Classroom practices	81	Catalan	second semester	morning-mixed
(PAUL) Classroom practices	82	Catalan	second semester	morning-mixed