UAB
Universitat Autònoma de Barcelona

Object-oriented Programming

Code: 104389
ECTS Credits: 6

2024/2025

Degree	Type	Year
2503740 Computational Mathematics and Data Analytics	FB	1

Contact

Name: Silvana Silva Pereira

Email: silvana.silva@uab.cat

Teaching groups languages

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

Prerequisites

There are no official prerequisites, but you must know how to program in Python.

Objectives and Contextualisation

One of the most widespread design and programming methodologies is object-oriented, according to which software is organized into classes that contain methods (procedures) and attributes (data). Apart from the concept of class, other fundamental elements are inheritance, composition and the separation of the public and private parts of classes. These elements are, therefore, necessary for OO programming, but not sufficient: you have to know how to use them to arrive at solutions that are easily extensible in the future, since a law of software is that changes are inevitable. There are a series of design principles or heuristics that tell us how to use these elements in order to achieve this. Design patterns are built on these principles, which are OO design-level solutions to recurring problems that often appear when solving (apparently) independent problems. In this subject, OO concepts as well as design principles and patterns are introduced and practiced.

Learning revolves around a project --- practice --- in which it is necessary to apply all the topics explained to design and implement a software application. The programming language used in the subject will be Python.

Learning Outcomes

1. CM09 (Competence) Develop effective algorithmic solutions to computational problems in accordance with the established requirements for object-oriented programming.
2. CM09 (Competence) Develop effective algorithmic solutions to computational problems in accordance with the established requirements for object-oriented programming.
3. CM10 (Competence) Analyse the computational complexity of the algorithmic solutions to develop and implement the one that guarantees the best performance for object-oriented programming.
4. CM10 (Competence) Analyse the computational complexity of the algorithmic solutions to develop and implement the one that guarantees the best performance for object-oriented programming.

5. CM11 (Competence) Ensure the correct functioning of an algorithmic solution in accordance with the requirements of the problem to be solved for object-oriented programming.
6. KM09 (Knowledge) Recognise the methods, systems and technologies specific to object-oriented programming.
7. SM09 (Skill) Use operating systems and software commonly used in object-oriented programming.
8. SM09 (Skill) Use operating systems and software commonly used in object-oriented programming.
9. SM10 (Skill) Use algorithmic and data representation structures suitable for problem-solving in object-oriented programming.

Content

1. Object orientation concepts: encapsulation, inheritance, composition, interface. UML class diagrams
2. Object-oriented programming in Python
4. Principles of object-oriented design
5. Design patterns: creational, structural, behavioral
6. Coding style, logging, comments and documentation

Activities and Methodology

Title	Hours	ECTS	Learning Outcomes
Type: Directed			
Practicum sessions	13	0.52	CM09, CM10, CM11, KM09, SM09, SM10, CM09
Programming	50.5	2.02	CM09, CM10, CM11, SM09, SM10, CM09
Study	50.5	2.02	KM09, KM09
Theory lectures	36	1.44	CM09, CM10, CM11, KM09, SM09, SM10, CM09

- Theory classes. The professor introduces theoretical contents and provides useful references (books, articles, web pages) so that the student can continue with the learning process, as well as with the development of the proposed project. Therefore, the slides should not be taken as the unique source of information for studying. It is necessary to study the recommended books, articles, chapters of open access books and other available material published on the web page of the subject. In these sessions, we will also solve small problems, design and/or OO programming in Python.

- Practical sessions. The students should have been organized in small groups at the beginning of the course. In these sessions, students attend to 1) review the work done each week on the project and to solve doubts, and 2) to be evaluated continuously, in accordance with the scale in the statement of practice Therefore, the practical is not solved during these sessions but through a previous job.

All the materials of the subject along with their detailed planning will be found on virtual campus. All these materials are in English.

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
Exams	50%	0	0	CM09, CM10, CM11, KM09, SM09, SM10
Practicum	50%	0	0	CM09, CM10, CM11, KM09, SM09, SM10

There is a group assessment part and one individual assessment part. The note of the first (PR) corresponds to the practical. The second (EX), to the average of two exams during the term (EX1, EX2), or a retake exam (EXR).

The algorithm for calculating the final mark (FM) is the following (all marks over 10):

EX = max(0.5 * (EX1 + EX2), EXR)
if (EX >= 4.0) and (PR >= 4.0):
FM = 0.5 * (EX + PR)
else :
FM = min(EX, PR)

The practical

The realization of the practicum is essential to assimilate the concepts of theory. The first thing we want to make sure is that the practicum has really been done by the group that delivers it, and that each member of the group has contributed to the same extent as the others. The second is that the practicum is done progressively during the course, in parallel with the theory. That's why we will do the following:

- The mark for the practice is obtained by combining the grade of the parts into which it is divided. Each part will be evaluated in situ in one of the practical sessions that we will announce at the beginning of the course (the milestones). The points that can be obtained from each part will be specified in the handout. These assessments have no retake, afterwards.

- On the evaluation days of the practicum (milestones), if, as a result of the answers the questions asked by the teacher during the assessment, we have doubts about the authorship or contribution of some member of the group, then the mark will change from group to individual and depending of how much and what part of the practice each member has done.

The exercices

We will propose some exercises that are optional. These exercises can be done individually or in small groups, the same or different than the practice. The exercises may add points to thetheorynote EX, with a top of 10. We say 'may' because obtaining these extra points is conditioned to a validation test in order to verify that the student has effectively completed the exercise he has delivered. We will do these validation tests only once, in the EX1 and EX2 exams, and they do not have retake afterwards.

Other

We do not preserve the mark of the practicum, exercises or exams from previous courses. Students taking again the course do not receive any differentiated treatment.

If an exam or part of the practicum is not delivered, the mark for that part is zero. The final grade will be 'no-avaluable' if nothing to be evaluated has been delivered, either exam, practicum or exercice.

Milestones are assessments equivalent to written exams. If a student does not show up, the mark is zero.

MH marks will be awarded at the discretion of students with a final final grade of 9.0 and taking into account all kinds of work done (exams, practicum and exercises).

The dates of continuous evaluation and delivery of works will be published on the virtual campus of the subject and may be subject to change in response to possible incidents. It will always be reported through the virtual campus, which will be the communication mechanism between teachers and students.

For written exam-type assessment activities, a place, date and time of revision will be indicated in which the student will be able to review the activity with the teacher. If the studentdoes not show up for this review, this activity will not be reviewed later.

Thes course does not follow the "avaluació única" system.

Notwithstanding other disciplinary measures that are deemed appropriate, and in accordance with the currentacademic regulations, the irregularities committed by the student who can lead to a variation of the qualification of an act of self- evaluation. Therefore, plagiarizing,copying orletting copying the practice or any other evaluation activity will imply fail with a zero and can not be recovered in the same academic year. If this activity has a minimum associated mark, then the student won't pass the course.

Bibliography

Basic

Theory classes are based on a set of slides, which are a summary, a synthesis of concepts that are best explained in these books, which are well worth reading or consulting to better understand them and to expand knowledge.

Design patterns. E. Gamma, R. Helm, R. Johnson, J. Vlissides. Pearson Education 2003.
It is a classic of object-oriented design patterns, by the authors who proposed this concept. The original English version is from 1995 and so its examples and diagrams are old. But it explains the concepts very well.

Design patterns explained simply. Alexander Shvets. Book online,
https://sourcemaking.com/design-patterns-ebook. from the same author another online book at
https://refactoring.guru/design-patterns
Well explained and illustrated with examples, which are also programmed in Java, Python, C ++ and more.

Complementary

Head first Design Patterns. E. Freeman & E. Robson, 2021.
A very didactic book about design patterns that uses simple examples explained with humor.

Python: master the art of design patterns. D. Phillips, C. Giridhar, S. Kasampalis. Packt Publishing, 2016.
A very extensive book (700pp) on design patterns in Python. A very positive aspect is that it shows how to program them in Python in detail, one by one.

Object-oriented mastering Python. Steven F.Lott. Packt publishing, 2014.
Advanced Python, or rather, going into the details of the language. "Grasp the intricacies of object-oriented programming in Python in order to efficiently build powerful real-world applications." Of interest is the chapter on logging and some others.

Software

- Python Anaconda, with libraries numpy, matplotlib.
- Pycharm community edition, plus plugin PlantUML.

Language list

Name	Group	Language	Semester	Turn
(PLAB) Practical laboratories	1	Catalan/Spanish	second semester	morning-mixed
(PLAB) Practical laboratories	2	English	second semester	morning-mixed
(TE) Theory	1	Catalan/Spanish	second semester	morning-mixed