1. OVERVIEW

Subject area	Object-Oriented Programming
Degree	Bachelor's Degree in Data Science
School/Faculty	Faculty of Science, Engineering and Design
Year	First
ECTS	6 ECTS
Type	Core
Language(s)	Spanish
Delivery Mode	On campus
Semester	Second semester

2. INTRODUCTION

Data science has greater influence on our society by the day. Therefore, Object-Oriented Programming is aimed at providing a global impression of the most advanced programming techniques used in data analysis. This subject teaches students about design, development, monitoring and testing, as well as how to solve data extraction, transformation and load problems on a business level. This, all the while, following code quality and good practices.

Students learn how to design software which can be applied to different top-level programming languages, while always focusing on data management and processing. This subject will give students an overall view of data programming which will serve as a base for other subjects in the Bachelor's Degree in Data Science.

3. SKILLS AND LEARNING OUTCOMES

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

- CB1. Students have shown their knowledge and understanding of a study area originating from general secondary school education, and are usually at the level where, with the support of more advanced textbooks, they may also demonstrate awareness of the latest developments in their field of study.
- CB2. Students can apply their knowledge to their work or vocation in a professional manner and possess the skills which are usually evident through the forming and defending of opinions and resolving problems within their study area.
  CB3. Students must 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.

Cross-curricular skills (CT, by the acronym in Spanish):

- CT02. Independent learning: skills for choosing strategies to search, analyse, evaluate and manage information from different sources, as well as to independently learn and put into practice what has been learnt.

- CT04. Written/spoken communication: ability to communicate and gather information, ideas, opinions and viewpoints to understand and be able to act, spoken through words or gestures or written through words and/or graphic elements.
- CT05. Analysis and problem-solving: be able to critically assess information, break down complex situations, identify patterns and consider different alternatives, approaches and perspectives in order to find the best solutions and effective negotiations.

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

- CE3. Knowledge of the core principles and applications of software development and databases.
- CE4. Ability to successfully apply data type models and algorithms to create solutions to problems in the data science field.
- CE5. Ability to design, implement, gather, store and exploit databases and database management systems to create solutions to problems in the data science field.

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

- Develop algorithmic thinking. Transfer a problem into a sequence of actions to solve.
- Design and implement solutions to problems of medium-level complexity using databases (structured, semi-structured, non-structured), data structures and object-oriented programming.
- Use programming environments to compile, link and execute programs, as well as identify and correct errors in each stage.
- Suitably document the designs, as well as the introduction of comments in the code to ease understanding and further use of the software created.

Skills	Learning outcomes
CB1	Develop algorithmic thinking. Transfer a problem into a sequence of actions to solve. Use programming environments to compile, link and execute programs, as well as identify and correct errors in each stage.
CB2	Develop algorithmic thinking. Transfer a problem into a sequence of actions to solve. Design and implement solutions to problems of medium-level complexity using databases (structured, semi-structured, non-structured), data structures and object-oriented programming.
CT02	Develop algorithmic thinking. Transfer a problem into a sequence of actions to solve. Use programming environments to compile, link and execute programs, as well as identify and correct errors in each stage.
CT04	Suitably document the designs, as well as the introduction of comments in the code to ease understanding and further use of the software created.

CT05	Design and implement solutions to problems of medium-level complexity using databases (structured, semi-structured, non-structured), data structures and object-oriented programming.
CE3	Develop algorithmic thinking. Transfer a problem into a sequence of actions to solve.
CE4	Design and implement solutions to problems of medium-level complexity using databases (structured, semi-structured, non-structured), data structures and object-oriented programming.
CE5	Design and implement solutions to problems of medium-level complexity using databases (structured, semi-structured, non-structured), data structures and object-oriented programming. Use programming environments to compile, link and execute programs, as well as identify and correct errors in each stage.

4. CONTENTS

1. Implementation of classes Attributes, constructors, methods
2. Inheritance, collections and advanced class design
3. Overload and rewriting
4. Abstract classes, polymorphism and interfaces
5. Quality assurance and design patterns Graphical interfaces.

5. TEACHING/LEARNING METHODS

The types of teaching/learning methods are as follows:

- Collaborative learning: students learn to work with other people (colleagues and professors) to find creative, comprehensive and constructive solutions to questions and problems that arise from the given case studies, using relevant knowledge and available resources in relation to each subject.
- Problem-based learning: students face problems they must solve either working as a team or independently.
- Master Lecture: presentations by the professor using the appropriate technological tools to facilitate understanding of the subject matter.
- Directed academic activities: more independent tasks (individual or in groups), involving search for information, written summaries, debates and public defence of work.

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
Master classes	44
Problem solving and case studies	10
Laboratory work (exercises led by the teacher)	15
Knowledge tests	4
Independent working	50
Tutorials	6
Practical work at home	21
TOTAL	**150**

7. ASSESSMENT

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

On campus:

Assessment system	Weighting
Global knowledge test.	40%
Knowledge test.	25%
Practical 1	15%
Practical 2 (presentation and final submission)	20%
Total	100%
Global knowledge test.	40%

On the Virtual Campus, when you open the subject area, you can see all the details of your assessment activities and the deadlines and assessment procedures for each activity.

8. BIBLIOGRAPHY

The reference publication to accompany this subject area is:

- Boucheny Vincent (2021) Aprende la Programación Orientada a Objetos con el lenguaje Python

The recommended bibliography is indicated below:

- Pagés Mariano (2019). El Paradigma de Objetos a tu Alcance: Aprendiendo a resolver problemas, pensando en objetos
- Cerrada, J. A. y Collado, M. E. (2010). Fundamentos de Programación. Madrid: Editorial Universitaria Ramón Areces.
- Dixit, J. B. (2009). Computer Fundamentals and Programming in C. New Delhi: Laxmi Publications
- Jiménez, M. y Otero, B. (2013). Fundamentos de ordenadores Barcelona: Editorial Universitat Politècnica de Catalunya.
- Steven F. (2021) . Python Object-Oriented Programming: Build robust and maintainable object-oriented Python applications and libraries, 4th Edition
- Juganaru, M. (2014). Introducción a la programación. Ciudad de México: Grupo Editorial Patria.
- Peña, R. (2005). Diseño de Programas: Formalismo y Abstracción. Madrid: Pearson.
- Rodríguez, M. A. (1991). Metodología de programación a través de pseudocódigo. Madrid: McGraw-Hill.
- Romney, M. B. y Steinbart, P. J. (2011). Accounting Information Systems. New York: Prentice Hall.