COURSE DESCRIPTION CARD - SYLLABUS

Course name
Object-Oriented Programming in Python [S1DSwB1>POwP]

Course

Field of study
Data Science in Business

Year/Semester
1/2

Area of study (specialization)
–

Profile of study
general academic

Level of study
first-cycle

Course offered in
Polish

Form of study
full-time

Requirements
compulsory

Number of hours

Lecture
30

Laboratory classes
0

Other
0

Tutorials
30

Projects/seminars
0

Number of credit points

5,00

Coordinators

Grzegorz Nowak
grzegorz.nowak@put.poznan.pl

dr inż. Marcin Nowak
marcin.nowak@put.poznan.pl

Lecturers

Prerequisites

Students should have a basic understanding of programming in Python, including language syntax, data types, data structures, control statements, functions, and modular programming. The ability to work with files, basic libraries, and fundamental debugging and code testing skills is required.

Course objective

The aim of this course is to introduce students to the object-oriented programming (OOP) paradigm in Python. Students will learn to create and utilize classes and objects, apply the principles of encapsulation, inheritance, and polymorphism, as well as design and implement structures following the object-oriented approach. The course develops skills in organizing code into larger modules, working with advanced Python mechanisms, and using design patterns, preparing students for the efficient development and maintenance of applications.

Course-related learning outcomes

Knowledge:
Explains the principles of object-oriented programming (OOP) in Python, including the concepts of classes, objects, methods, and the differences compared to procedural programming [DSB1_W02].
Characterizes the mechanisms of encapsulation, inheritance, polymorphism, and abstraction, and their application in application design [DSB1_W02].
Describes the methods, techniques, and tools used in object-oriented programming, including design patterns and metaprogramming [DSB1_W07].

Skills:
Creates classes and objects in Python, using instance, class, and static methods to organize code [DSB1_U02].
Implements the principles of encapsulation, inheritance, and polymorphism to build flexible and scalable object-oriented applications [DSB1_U08].
Uses metaprogramming and decorators to automate code and dynamically modify program behavior [DSB1_U09].
Designs, implements, and tests object-oriented applications in Python, using design patterns and unit testing tools [DSB1_U08].
Analyzes code for correctness and efficiency, using unit tests and debugging [DSB1_U11].
Develops object-oriented programming skills by utilizing documentation and scientific literature [DSB1_U15].

Social competences:
Collaborates in programming teams on the design of object-oriented applications, applying best programming practices in Python [DSB1_K02].
Adheres to code quality and testing principles, considering readability, reusability, and compliance with OOP principles [DSB1_K05].

Methods for verifying learning outcomes and assessment criteria

Learning outcomes presented above are verified as follows:

Lecture:
There are two tests, each graded in the form of points-50 points per test. The final grade is determined by the sum of points from both tests. The first test takes place midway through the course, while the second is held at the end. The passing threshold is a total of 50 points from both tests.
Laboratories:
There are two tests, each graded in the form of points-50 points per test. The final grade is determined by the sum of points from both tests. The first test takes place midway through the course, while the second is held at the end. The passing threshold is a total of 50 points from both tests.

Programme content

The course covers the fundamentals of object-oriented programming (OOP) in Python, including a comparison with procedural programming. Students will learn the concepts of classes and objects, instance, class, and static methods, as well as the mechanisms of encapsulation, abstraction, inheritance, and polymorphism. Metaprogramming techniques, decorators, and design patterns such as Singleton, Factory, Observer, and Adapter will also be discussed. Additionally, the course includes unit testing and debugging of object-oriented code, preparing students for the effective design and implementation of applications following OOP principles.

Course topics

Introduction to OOP and comparison with procedural programming
Namespaces and scopes
Creating classes and objects in Python
Instance, class, and static methods
Encapsulation and data access control
Abstraction and abstract classes
Inheritance - how to avoid code repetition?
Polymorphism and operator overloading
Metaprogramming - dynamic code generation

Class decorators and metaprogramming
Introduction to design patterns in Python
Singleton and Factory Pattern
Observer and Adapter design patterns
Unit testing of classes and objects
Debugging object-oriented code in Python

Teaching methods

Lectures: Problem-based lecture, case study presentation
Laboratories: Problem-solving tasks, case study analysis, group workh

Bibliography

Basic:
Dusty, P., Lott, S. (2023). Programowanie zorientowane obiektowo w Pythonie. Tworzenie solidnych i łatwych w utrzymaniu aplikacji i bibliotek, Helion
Vasiliev, Y. (2024). Python w data science. Praktyczne wprowadzenie, Helion, Gliwice

Additional:
Kalb, I. (2022). Python zorientowany obiektowo. Programowanie gier i graficznych interfejsów użytkownika, Helion

Breakdown of average student's workload

	Hours	ECTS
Total workload	125	5,00
Classes requiring direct contact with the teacher	60	2,50
Student's own work (literature studies, preparation for laboratory classes/ tutorials, preparation for tests/exam, project preparation)	65	2,50