COURSE DESCRIPTION CARD - SYLLABUS

Course name
Object Oriented Programming [S2Elmob1>PO]

Course

Field of study
Electromobility

Year/Semester
1/1

Area of study (specialization)
–

Profile of study
general academic

Level of study
second-cycle

Course offered in
Polish

Form of study
full-time

Requirements
compulsory

Number of hours

Lecture
15

Laboratory classes
15

Other
0

Tutorials
0

Projects/seminars
0

Number of credit points

2,00

Coordinators

dr inż. Stanisław Mikulski
stanislaw.mikulski@put.poznan.pl

Lecturers

Prerequisites

Basics knowledge about programming in any programming language, including issues such as: loops, conditional statements, data types, functions. Basic knowledge of mathematics on the following issues: approximation, matrix calculus and statistical analysis.

Course objective

The aim of the course is to broaden the knowledge of object-oriented programming techniques, in such a way that the student can using already developed Python packages and create object-oriented software characterized by high agility and efficiency.

Course-related learning outcomes

Knowledge:
The student has extended knowledge in the field of programming techniques and the use of modern IT tools for the analysis and synthesis of electric systems, hybrid and electric vehicles, including traction

Skills:
The student is able to use modern information and communication tools, advanced programming

techniques and machine learning methods to collect, process and analyze data

Social competences:
The student understands that in the area of technology, knowledge and skills quickly devalue, which requires constant replenishment

Methods for verifying learning outcomes and assessment criteria

Learning outcomes presented above are verified as follows:

Lectures:
through short tests (quizzes) placed on the university e-learning platform and final credit taking place at the end of the semester;
Laboratory:
through tasks performed independently during classes.

Programme content

Programming paradigms, code execution mechanisms, code interpretation vs. compilation, discussion of the basic structure of Python, including classes, objects, abstraction, inheritance and encapsulation. Discussion of selected design patterns and their exemplary implementation. UML class diagrams.

Course topics

The following topics are discussed during the lectures:
1) basic definitions such as: algorithm, program code, etc.
2) differences between code compilation and interpretation
3) data storage cycle in python
4) instruction flow control (loops and conditional instructions)
5) class and instance definition
6) inheritance, encapsulation, polymorphism and abstraction in Python
7) types of UML diagrams
8) class diagrams and their equivalents in Python code

The following issues are discussed during the laboratories:
1) introduction to Python and discussion of the Visual Studio Code environment
2) data types and operators used in Python
3) creating classes
4) use of encapsulation (definition of class properties and access functions)
5) inheritance
6) abstract classes
7) creating code based on UML diagrams

Teaching methods

Presentations, quizzes and e-learning tasks, initiating discussions during the lecture.
Lecture conducted on-line with using synchronous access methods.

Bibliography

Basic:
1. Johansson R., Matematyczny Python. Obliczenia naukowe i analiza danych z użyciem NumPy, SciPy i Matplotlib, Helion 2021
2. Python documentation, https://docs.python.org/3/
3. Lott S. F., Phillips D., Programowanie zorientowane obiektowo w Pythonie. Tworzenie solidnych i łatwych w utrzymaniu aplikacji i bibliotek. Wydanie IV, Helion 2023

Additional:
1. Chollet F., Deep learning. Praca z językiem Python i biblioteką Keras, Helion 2019

Breakdown of average student's workload

	Hours	ECTS
Total workload	55	2,00
Classes requiring direct contact with the teacher	30	1,00
Student's own work (literature studies, preparation for laboratory classes/ tutorials, preparation for tests/exam, project preparation)	25	1,00