GDAŃSK UNIVERSITY OF TECHNOLOGY

Subject card

Subject name and code	Programming in High Level Languages II, PG_00067435		
Field of study	Automatic Control, Cybernetics and Robotics		
Date of commencement of studies	October 2025	Academic year of realisation of subject	2026/2027
Education level	first-cycle studies	Subject group	Obligatory subject group in the field of study Subject group related to scientific research in the field of study
Mode of study	Full-time studies	Mode of delivery	at the university
Year of study	2	Language of instruction	Polish
Semester of study	3	ECTS credits	3.0
Learning profile	general academic profile	Assessment form	assessment
Conducting unit	Department of Decision Systems and Robotics -> Faculty of Electronics Telecommunications and Informatics -> Wydziały Politechniki Gdańskiej		
Name and surname of lecturer (lecturers)	Subject supervisor	dr hab. inż. Michał Czubenko	
	Teachers	dr hab. inż. Michał Czubenko	

Lesson types and methods of instruction

Lesson type	Lecture	Tutorial	Laboratory	Project	Seminar	SUM
Number of study hours	0.0	0.0	15.0	0.0	0.0	15

E-learning hours included: 0.0

Learning activity and number of study hours

Learning activity	Participation in didactic classes included in study plan	Participation in consultation hours	Self-study	SUM
Number of study hours	15	4.0	56.0	75

Subject objectives

The aim of the course is to introduce students to advanced programming techniques in a high-level language that combines syntactic clarity with great flexibility in creating modern applications. During the course, students will learn how to design and implement applications using different paradigms, with particular emphasis on object-oriented programming and graphical components using popular windowing frameworks (Qt or GTK). Issues related to data processing, such as serialization and deserialization, will also be discussed. Much emphasis will be placed on mastering advanced elements of the language, such as descriptors, decorators, protocols, functions as first-class objects, metalevel classes, and metaprogramming techniques that allow writing more flexible and reusable code. Aspects of design patterns (factories, singletons, adapters) will be discussed.

Learning outcomes

Course outcome	Subject outcome	Method of verification
[K6_U04] can apply knowledge of programming methods and techniques as well as select and apply appropriate programming methods and tools in computer software development or programming devices or controllers using microprocessors or programmable elements or systems specific to the field of study	is able to use a high-level programming language to a degree that allows the use of advanced programming methods	[SU1] Assessment of task fulfilment
[K6_U12] can analyze the operation of components, circuits and systems related to the field of study, as well as measure their parameters and examine technical specifications, and plan and conduct experiments related to the field of study, including computer simulations and measurements, and interpret obtained results and draw conclusions	can handle debuggers, analyze interpreter/compiler errors and program using tests	[SU2] Assessment of ability to analyse information
[K6_W04] knows and understands, to an advanced extent, the principles, methods and techniques of programming and the principles of computer software development or programming devices or controllers using microprocessors or programmable elements or systems specific to the field of study, and organisation of systems using computers or such devices	can handle methods related to metaprogramming and knows basic design patterns	[SW1] Assessment of factual knowledge

Subject contents

The course will include 7 computer exercises related to the following topics (the list is open):

1. Object-oriented programming and application structure (consolidation of object-oriented aspects, classes, inheritance, encapsulation, special methods, code organization)
2. Data processing - serialization and deserialization (support for formats such as JSON, CSV, Pickle, loading and saving objects, error handling and validation)
3. Graphical programming (simple windows, supporting logic, slot signals, event handling, communication between components, aspects of interface blocking)
4. Advanced language elements (creating meta-level classes, automatic modification of classes, descriptors)
5. Web applications (structure of web projects, routing and HTTP support, communication between frontend and backend)
6. REST API and testing (designing a simple REST server, unit tests)

Design patterns (singleton, factory, strategy, decorator, adapter and others)

Prerequisites and co-requisites

- has basic knowledge of mathematics, including mathematical analysis, algebra and the principles of spatial geometry
- can program in cpp and a selected high-level language
- knows the basic concepts of programming language syntax
- knows the basics of object-oriented programming and aspects of functional and structural programming

Assessment methods and criteria

Subject passing criteria	Passing threshold	Percentage of the final grade
Lab exercise	60.0%	100.0%

Recommended reading

Basic literature	Allen B. Downey. *Myśl w języku Python*. Helion, 2025. Print. Houlahan, Padraig. *Prototyping Python Dashboards for Scientists and Engineers: Build and Deploy a Complete Dashboard with Python*. 1st ed. Berkeley, CA: Apress L. P, 2024. Web.
Supplementary literature	Lanaro, G., Nguyen, Q., & Kasampalis, S. (2019). *Advanced Python Programming: Build high performance, concurrent, and multi-threaded apps with Python using proven design patterns*. Packt Publishing Ltd.

	eResources addresses	Basic https://realpython.com/ - Real Python - best webpage about pure Python https://docs.python.org/3/ - Python docs https://doc.qt.io/qtforpython-6/ - QT docs
Example issues/ example questions/ tasks being completed	1. Modeling a system (e.g. a library or a reservation system). 2. Implementation of a class that saves user data to a file and reads it from a file. 3. An application with a graphical input form and data validation. 4. An example of a class that dynamically creates methods. 5. A web application with a simple administration panel. 6. An interface for managing objects via API. 7. Implementation of a class that registers and manages objects (Factory).	
Work placement	Not applicable	

Document generated electronically. Does not require a seal or signature.