GDAŃSK UNIVERSITY OF TECHNOLOGY

Subject card

Subject name and code	Programming in High Level Languages, PG_00067327		
Field of study	Automatic Control, Cybernetics and Robotics		
Date of commencement of studies	October 2026	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	1	Language of instruction	Polish
Semester of study	2	ECTS credits	2.0
Learning profile	general academic profile	Assessment form	assessment
Conducting unit	Department of Decision Systems and Robotics -> Faculty of Electronics Telecommunications and Informatics -> Faculties of Gdańsk University of Technology		
Name and surname of lecturer (lecturers)	Subject supervisor	dr Piotr Dalka	
	Teachers	dr hab. inż. Michał Czubenko mgr inż. Tymoteusz Byrwa mgr inż. Jakub Kłopotek Główczewski dr Piotr Dalka mgr inż. Michał Kopczyński	

Lesson types	Lesson type	Lecture	Tutorial	Laboratory	Project	Seminar	SUM
	Number of study hours	15.0	0.0	15.0	0.0	0.0	30
	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	30	3.0	17.0	50

Subject objectives	
	The objective of the course is to comprehensively prepare the student to solve complex engineering problems by learning programming in a modern, versatile high-level language. During the course, the student will acquire the ability to create readable and efficient code, progressing from basic syntax, data types, and control structures to advanced paradigmsfunctional and object-oriented. The program covers key aspects of software engineering, such as code modularity, dependency and virtual environment management, as well as exception handling and input/output operations, including data serialization. The student will learn advanced techniques such as decorators, generators, metaprogramming, and the basics of asynchronous programming, enabling the creation of efficient and scalable solutions. Supplementing this knowledge will be an overview of standard and external libraries, the basics of creating web applications using REST architecture, and the ability to write unit tests.

Learning outcomes	Course outcome	Subject outcome	Method of verification
	[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	The student is able to analyze the operation of programming components of elements, systems and systems characteristic of automation and robotics, plan and conduct computer simulations and functional tests, as well as interpret the obtained results and draw technically justified conclusions. Is able to operate programming environments and debuggers, identify and analyze compiler or interpreter errors, as well as design and implement unit and functional tests to ensure the correctness and reliability of the created software.	[SU4] Assessment of ability to use methods and tools
	[K6_U09] can carry out a critical analysis of the functioning of existing technical solutions and assess these solutions, as well as apply experience related to the maintenance of technical systems, devices and facilities typical for the field of studies, gained in the professional engineering environment	The student is able to critically analyze the operation and structure of existing software and technical solutions used in automation and robotics systems. Is able to assess their effectiveness, readability, reliability and scalability, as well as indicate limitations and possibilities for improvement. Uses the experience gained in maintaining and testing programmable systems - both at the application and hardware level - to formulate recommendations and design new solutions consistent with the requirements of engineering practice.	[SU3] Assessment of ability to use knowledge gained from the subject
	[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	The student knows and understands at an advanced level the principles of software design and implementation in high-level languages, with particular emphasis on methods and techniques used in automation and robotics. Has knowledge of programming paradigms (procedural, object-oriented, functional), code modularity, dependency management, error handling and input-output operations, as well as creating code for systems based on microprocessors, microcontrollers and programmable devices. Understands the organization of computer systems and the architecture of execution environments used in embedded and robotic systems.	[SW1] Assessment of factual knowledge
	[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	The student is able to use the acquired knowledge of programming techniques to create clear and effective code, selecting appropriate tools and programming paradigms (procedural, object-oriented, functional) depending on the characteristics of the problem. Is able to use modern programming environments and libraries supporting the development of software for computer systems used in automation and robotics, as well as analyze and implement algorithms in accordance with the functional requirements of a given system.	[SU1] Assessment of task fulfilment

Subject contents

Course content – lecture
Lecture:

The course will include 15 lectures related to the following topics (the list of topics is not exhaustive):

- distinctive features of the language
- language syntax, basic data types, and collections
- flow control
- functional programming
- code organization, including modules and packages
- virtual environment management and package managers
- specific language elements: decorators, generators, context managers, and exception handling
- input/output operations and serialization and deserialization of data to/from popular formats such as CSV, JSON, XML, and YAML
- object-oriented programming
- implementation of selected design patterns
- concurrent and parallel processing and optimal execution of processor-bound and input/output-bound operations
- asynchronous programming
- advanced language features: descriptors, protocols, magic methods, metaclasses
- overview of the standard library and popular external libraries
- web frameworks and the basics of RESTful communication
- writing unit tests

Lab:

The list of exercises is not closed:
- Python basics data types (numbers, lists, dictionaries, tuples), logical and arithmetic operations, conditional statements and loops, introduction to working with sensor data.
- Functions and modularity defining functions, parameters, default values, returning data, processing and filtering sensor signals.
- Object-oriented programming creating classes and objects, inheritance, special methods, modeling robotic components (e.g. sensor, mobile robot, obstacle).
- File and stream operations, logger writing and reading data (.csv, .txt), storing logs and system configurations.
- Generators and iterators real-time data processing using yield, analysis of measurement streams using maps, filter, zip.
- Asynchronous programming and environments asyncio, simulation of parallel sensor readings, environment management (venv, pip, poetry, uv, requirements.txt).
- Robot motion simulation and visualization 2D and 3D trajectory calculations (numpy), motion visualization using matplotlib, analysis of robot position and orientation in space.

Prerequisites and co-requisites

- has basic knowledge of mathematics, including algebra and logic
- knows basic concepts related to functional and object-oriented programming, such as variable, function, class, inheritance, loop, condition
- knows good practices related to software development and basic design patterns
- knows what basic text file formats such as CSV, XML, or JSON look like and what characterizes them knows
- the basics of the HTTP protocol

Assessment methods and criteria

Subject passing criteria	Passing threshold	Percentage of the final grade
Final colloquium	60.0%	50.0%
Laboratory exercises	60.0%	50.0%

Recommended reading

Basic literature	• Mark Lutz. *Python. Wprowadzenie.* Helion, 2022. Print. • Allen B. Downey. *Myśl w języku Python*. Helion, 2025. Print.
Supplementary literature	• Luciano Ramalho. *Fluent Python. 2nd Edition*, 2022, Helion.
eResources addresses	

Example issues/ example questions/ tasks being completed

- Which data types are mutable?
- What types of arguments can a function accept?
- What is the difference between a module and a package?
- Which language elements should be used for effective resource cleanup at the end of a given task?
- For which applications is it advisable to use a generator, and when should its use be avoided
- By which language mechanisms can processor-intensive operations be optimally implemented?

Practical activites within the subject

Not applicable

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