COURSE DESCRIPTION CARD - SYLLABUS

Course name
Introduction to Programming - Python

Course

Field of study
Computing

Year/Semester
1/1

Area of study (specialization)

Profile of study
general academic

Level of study
First-cycle studies

Course offered in
Polish

Form of study
full-time

Requirements
elective

Number of hours

Lecture	Laboratory classes	Other (e.g. online)
24	30	
Tutorials	Projects/seminars	

Number of credit points

4

Lecturers

Responsible for the course/lecturer:
mgr. inż. Jan Badura
email: jan.badura@cs.put.poznan.pl
Wydział Informatyki i Telekomunikacji
ul. Piotrowo 3, 60-965, Poznań

Responsible for the course/lecturer:

Prerequisites

According to the core curriculum for general education available at: http://cke.gov.pl, it is assumed that when starting a subject, a student has basic skills in:
- mathematics: 4th educational stage, basic scope extended by calculus (extended scope);
- computer science: 4th educational stage, basic scope.
In terms of social competences, the student must demonstrate such attitudes as honesty, responsibility, perseverance, cognitive curiosity, creativity, manners, and respect for other people.

Course objective

The aim of the course is to familiarize students with the basics of computer programming and to teach programming in Python3. In particular, this includes:
- providing students with basic information about the development of programming languages, structured programming, the principles of object-oriented programming and the construction of text and window programs,
- developing students' ability to algorithmize problems, including in the form of functions,

- teaching students how to use the integrated programming systems fluently,
- mastering by students the technique of object-oriented programming, including the creation of various program units and access to the data and codes contained in them,
- teaching students to create and use modules and packages and use them in programs,
- acquisition by students of the skills of programmatic protection of codes against program execution errors.

Course-related learning outcomes

Knowledge

1. Has extended and deepened knowledge of mathematics useful for formulating and solving complex IT tasks related to programming in Python3, formal specification and verification of software (K1st_W1)
2. Has a structured and theoretically based general knowledge of the key issues of computer science, which are programming techniques, and detailed knowledge of selected issues of this discipline (K1st_W4)
3. Knows the basic techniques, methods and tools used in the process of solving IT (programming) tasks, mainly of an engineering nature (K1st_W7)

Skills

1. Can formulate and solve programming tasks, apply appropriately selected methods, including analytical, simulation or experimental methods implemented in Python3 (K1st_U4)
2. Has the ability to formulate algorithms and implement them using at least one of the popular tools, which is the Python3 language (K1st_U11)
3. Is able to organize, cooperate and work in a group, assuming different roles in it, and is able to properly set priorities for the implementation of a task defined by himself or others in order to develop the most efficient artificial intelligence algorithm for playing computer games (K1st_U18)

Social competences

1. Based on the history of development of the Python3 language understands that in computer science knowledge and skills become obsolete very quickly (K1st_K1)
2. Is aware of the importance of knowledge in solving engineering problems, especially related to programming on the example of Python3 and knows examples and understands the causes of malfunctioning IT systems (K1st_K2)

Methods for verifying learning outcomes and assessment criteria

Learning outcomes presented above are verified as follows:

Formative assessment:

- on the basis of answers given regarding the implementation of tasks within the laboratories;

Summative assessment:

The assumed learning outcomes are checked by:

- assessment of skills related to the implementation of laboratory tasks (to obtain a 3.0 grade, during the semester each student has to solve problems available in the HackerRank system, defined by the teacher at the beginning of the semester. Additionally, the student can increase they grade by 0.5 for every 10 tasks solved from the set of additional tasks - also indicated by the teacher at the beginning of

the semester,
- assessment of knowledge and skills demonstrated in a written (on a computer) final test of a problematic and practical nature. The test consists of programming tasks checked automatically on a computer, which must be solved in order to pass the subject. The student must solve at least one problem.

Programme content

The program of the course covers the following issues:
- basic concepts related to programming (programming, algorithm, program, programming language, machine-oriented language, command, higher-order language, universal language, specialized language),
- review of programming languages (Ada, Algol, assemblers, Basic, C, C ++, Cobol, Fortran, HTML, Java, Lisp, Logo, Pascal, PHP, PL/1, Prolog),
- workflows (block diagrams) and symbols used in them,
- general principles of object-oriented programming (heredity, hermeticity and polymorphism),
- general characteristics of the PyCharm package,
- basic concepts related to constructing programs in the PyCharm integrated programming system (project, package, module),
- using the integrated PyCharm programming package,
- overview of the Python3 language structure (program, module, package, functions, classes and objects, data types, variables, instructions),
- program and package structure,
- basic elements of the language (basic symbols, language keywords and directives, identifiers, numbers, strings, including Unicode strings, logical literals, comments and separators),
- data types and their description (defining types, simple types, string types, describing objects, type compatibility, decorators),
- variables (variable declarations, indexed variables, object variables, dynamic variables, function variables, with initial value, variable overlapping, constant and variable literals),
- expressions (types of operators and their precedence, expression syntax, constant expression),
- instructions (simple, structured, lambda expressions),
- functions (definitions, types of parameters, overloading, calling),
- object processing (constructors and destructors, static, virtual, dynamic and abstract methods, message handling, properties),
- publishing packages in the Python Package Index,
- file processing, serialization,
- multithreading (thread synchronization, priorities, waiting for completion).

During laboratory classes, students, after familiarizing themselves with the PyCharm integrated programming environment, write programs that use the learned elements of the language.

Teaching methods

1. Lecture: multimedia presentation and presentation of writing and executing selected programs directly in the PyCharm or Interactive package.

2. Laboratory exercises: practical exercises on elements of the Python3 language, writing programs in this language.

Bibliography

Basic

1. Python 3: kompletne wprowadzenie do programowania, Mark Summerfield, Helion, 2010

Additional

1. Python. Wprowadzenie. Wydanie IV, Mark Lutz, Helion 2010

2. How to Think Like a Computer Scientist: Interactive Edition (https://runestone.academy/ns/books/published/thinkcspy/index.html)

Breakdown of average student's workload

	Hours	ECTS
Total workload	100	4,0
Classes requiring direct contact with the teacher	54	2,0
Student's own work (literature studies, preparation for laboratory classes/tutorials, preparation for tests/exam, project preparation) [1]	46	2,0

[1] delete or add other activities as appropriate