COURSE DESCRIPTION CARD - SYLLABUS

Course name
Informatics II [N1EiT1>INF]

Course

Field of study
Electronics and Telecommunications

Year/Semester
3/5

Area of study (specialization)
–

Profile of study
general academic

Level of study
first-cycle

Course offered in
Polish

Form of study
part-time

Requirements
compulsory

Number of hours

Lecture
30

Laboratory classes
0

Other
0

Tutorials
35

Projects/seminars
0

Number of credit points

8,00

Coordinators

prof. dr hab. inż. Grzegorz Danilewicz
grzegorz.danilewicz@put.poznan.pl

Lecturers

Prerequisites

Students taking the course should have a basic knowledge of computer design and number systems. They can use high-level programming languages C and C++. Additionally, students can acquire information from literature, databases, and other sources in Polish or English. They should be able to integrate, interpret, and analyze this information to draw conclusions and support their opinions.

Course objective

Introduce students to the core concepts of object-oriented programming, including classes, objects, inheritance, polymorphism, and encapsulation. Develop proficiency in using essential Python libraries for different tasks. Compare and contrast object-oriented and functional programming paradigms through practical Python examples, emphasizing their strengths and weaknesses in different problem-solving contexts.

Course-related learning outcomes

Knowledge:
1. Knows the principles of computer program construction, has knowledge of computer science, and knows the syntax of the C# programming language.

2. Knows the utilization of computer memory, including reservations and references.
Skills:
1. Can proficiently use the high-level programming language C#. Can develop simple console and graphical programs.
2. Capable of writing basic programs for network communication and accessing databases.
3. Demonstrates the ability to engage in independent learning.
4. Can effectively communicate in both Polish and English within professional and other contexts.
Social competencies:
1. Possesses a sense of responsibility for the design of electronic and telecommunications systems and recognizes the potential hazards associated with their improper use. Understands the principles of information storage and access control in databases to ensure data security.
2. Recognizes the influence of telecommunications and ICT systems and networks on the development of the information society.

Methods for verifying learning outcomes and assessment criteria

Learning outcomes presented above are verified as follows:

The knowledge gained in the lecture is verified by an exam. The exam can be taken in written or oral form.
In the written form, students must answer 7-10 questions (test and open) variously scored.
Number of questions: 7-10
Question types: A mix of short answer (test) and open-ended questions
Scoring: Questions have varying point values
There are three or four scoring groups.
Purpose of scoring groups: Questions are divided into groups based on difficulty or topic.
For each exam, there are prepared 50-60 questions.
Preparation: A large pool of questions is prepared to ensure fairness and variety.
The prerequisite for successful completion of the exam is the receipt of a minimum of 50% of the points possible.

In the case of an oral exam, the student draws one question from each scoring group.
Question selection: Students randomly select one question from each of the scoring groups.

In the oral form, for each question drawn, the student can receive an additional question (related to the drawn question).
Follow-up questions: For each drawn question, an additional question may be asked to further assess the student's understanding.

The evaluation of the question (includes the answer to both the drawn question and the additional question) includes the range of answers and the depth of understanding of the issue.

Evaluation criteria: The answer is evaluated based on its comprehensiveness, accuracy, and demonstration of understanding.

Passing threshold: To pass, students must achieve a minimum of 50% of the total possible points.

Skills acquired in the exercises are verified on the basis of the task carried out in the last class.

Assessment method: Practical skills are evaluated through a final class project.
The task is divided into 5-6 subtasks with different scoring.

Task structure: The project consists of multiple subtasks with varying point values.
Sub-tasks constitute a whole, but it is possible to perform them independently.

Task independence: While the subtasks contribute to the overall project, they can be evaluated separately.
Failure to perform a subtask does not affect the grade of the remaining subtasks.

Grading criteria: Each subtask is graded independently, and the overall grade is calculated based on the combined scores.
The passing threshold: 50% of the points.

Passing criteria: To pass, students must achieve a minimum of 50% of the total points for the project.

Criteria for evaluation of the exam and passing:
<= 50% = 2,0
51% - 60% = 3,0
61% - 70% = 3,5
71% - 80% = 4,0
81% - 90% = 4,5
91% - 100% = 5,0

Grading scale: This table outlines the correlation between the percentage of points obtained and the final grade.

Programme content

The program content covers:
1. compilation versus code interpretation,
2. structured versus object-oriented programming,
3. Python as a language for research and prototyping, and
4. Python programming basics.

Course topics

1. Introduction to Python programming
2. The Python programming environment
3. Commenting on code
4. Simple data types: integers, floats, booleans
5. Strings: formatting, methods, operations
6. Using libraries and modules
7. Functions: definition, calling, parameters, return values
8. Anonymous functions (lambda)
9. Input and output operations
10. Numerical computations and operators
11. Conditional statements and logical operators
12. Looping constructs
13. Data structures: lists, tuples, dictionaries, sets
14. File handling: text and binary files
15. Exception handling
16. Object-oriented programming: classes, objects, inheritance, polymorphism
17. Creating and using custom modules
18. Data exchange formats (JSON, XML)
19. Data visualization with Python libraries (Matplotlib)
20. Advanced Python topics
Laboratory: Practical Python programming projects

Teaching methods

1. Lectures
a. Traditional lecture
b. Lecture with Jupyter notebook
c. Discussion-based lecture
d. Experiment
e. Case study
f. Software development
2. Exercises
Students will complete practical exercises using Python programming language runtime environments. These exercises will be assigned by the instructor and accompanied by multimedia presentations.

Bibliography

Basic

1. Jesse Liberty "Programowanie C#", Helion 2005
2. Charles R. Severance, Python for Everybody. Exploring Data in Python 3, Charles Severance, University of Michigan
Additional
3. www.python.org

Breakdown of average student's workload

	Hours	ECTS
Total workload	175	8,00
Classes requiring direct contact with the teacher	75	3,00
Student's own work (literature studies, preparation for laboratory classes/ tutorials, preparation for tests/exam, project preparation)	100	5,00