

COURSE DESCRIPTION

Computer programming and programming languages

Academic year 2026-2027

1. Programme-related data

1.1. Higher Education Institution	Babeş-Bolyai University
1.2. Faculty	Faculty of Mathematics and Computer Science
1.3. Department	Department of Computer Science
1.4. Field	Computers and Information Technology
1.5. Level of study	Bachelor
1.6. Degree programme / Qualification	Information Engineering
1.7. Form of education	Full time

2. Course-related data

2.1. Course title	Algorithms and programming					Course code	MLE5171
2.2. Course coordinator		Asist. dr. Briciu Anamaria					
2.3. Seminar coordinator		Asist. dr. Briciu Anamaria					
2.4. Year of study	1	2.5. Semester	1	2.6. Type of assessment		Progress check	
2.7. Course status		Compulsory		2.8. Course type		Core subject	

3. Total estimated time (hours per semester of teaching activities)

3.1. Number of hours per week	6	of which: 3.2. course	2	3.3. seminar/ laboratory/ project	1 S 2 LP
3.4. Total of hours in the curriculum	70	of which: 3.5. course	28	3.6. seminar/ laboratory	42
Time allocation for individual study (IS) and self-taught activities (ST)					hours
Learning from textbooks, course materials, bibliography, and notes (IS)					20
Additional research in the library, on subject-specific electronic platforms, and on-site					4
Preparing seminars/ laboratories/ projects, assignments, reports, portfolios, and essays					34
Tutoring (professional guidance)					9
Examinations					8
Other activities					5
3.7. Total hours of individual study (IS) and self-taught activities (ST)				80	
3.8. Total hours per semester				150	
3.9. Number of credits				6	

4. Prerequisites (where applicable)

4.1. curriculum-related	
4.2 skills-related	

5. Specific conditions (where applicable)

5.1. course-related	
5.2. seminar/laboratory-related	

6.1. Competencies resulting from the completion of the degree programme (as referred to in the curriculum)¹

Professional competencies	
Competency code	Competency
CP1	create software
CP3	analyse software specifications
CP5	design information system
Transversal competencies	
Competency code	Competency
CT2	Solve problems
CT3	Think analytically

6.2. Learning outcomes relevant to the degree programme (as referred to in the curriculum)²

Learning outcomes targeted by the subject		
Competency code	Knowledge and comprehension	Specific academic skills
CP1 CP3	1. The student/graduate describes, identifies, and summarizes basic concepts and methods related to programming languages, programming environments, programming techniques, databases, artificial intelligence, and software engineering, as well as how they are applied to real-world problems.	1. The student/graduate selects and explains key concepts specific to algorithm design, object-oriented programming, and logical and functional programming. The student/graduate specifies requirements, analyzes, designs, develops, and tests programs in general-purpose programming languages (C, etc.) and/or object-oriented languages (C++, Java, etc.), applying specific elements of software engineering.
CP5	2. The graduate has adequate knowledge of the use of integrated development environments in order to create large-scale complex applications. The graduate has knowledge of programming, mathematics, engineering and technology and has the skills to use them in creating complex computer systems.	2. The graduate has the ability to develop, design and create new applications, systems or products using best practices in the field of computer science.

7. Subject-specific learning outcomes

Knowledge and comprehension
1. The student understands the fundamental concepts of software engineering (such as design, implementation, and maintenance).
2. The student understands the structure and functionality of software applications at different levels of abstraction, including architecture, modules, classes, and functions.
3. The student understands the fundamentals of the Python programming language and is familiar with the tools used for developing, executing, testing, and debugging programs.
Specific academic skills

¹ The professional and/or transversal skills targeted by the subject for which the course description is prepared will be copied from the curriculum of the degree programme. For each competency, the complete entry, including the competency code, will be copied with the exact wording that appears in the curriculum, without any changes. If no competency is copied from either of the two categories, the row corresponding to that category is deleted from the table.

² The learning outcomes relevant for the degree programme and targeted by the subject for which the course description is prepared will be listed. The entries, copied without any changes from the Curriculum by subject type (Core Subject/Specialisation Subject/Complementary Subject), are listed under the corresponding competency.

1. The student can design and implement software components in a programming language, following given specifications and applying appropriate programming principles.
2. The student can analyze, interpret, and explain existing source code, identifying its structure, functionality, and potential issues.
3. The student can organize source code into reusable and maintainable components and produce appropriate technical documentation.

8. Contents

8.1. Course	Teaching and learning methods	Remarks ³
1. Introduction to software development processes - What is programming: algorithm, program, basic elements of the Python language, Python interpreter, basic roles in software engineering - How to write programs: problem statement, requirements, feature driven development process	- Interactive exposure - Explanation - Conversation - Examples - Didactical demonstration	
2. Procedural programming - Compound types: list, tuple, dictionary - Functions: test cases, definition, variable scope, calling, parameter passing - Test-driven development (TDD), refactoring	- Interactive exposure - Explanation - Conversation - Examples - Didactical demonstration	
1. Modular programming - What is a module: Python module definition, variable scope in a module, packages, standard module libraries, deployment - PyCharm	- Interactive exposure - Explanation - Conversation - Examples - Didactical demonstration	
4. User defined types - How to define new data types: encapsulation, data hiding in Python, guidelines - Introduction to object-oriented programming - Exceptions	- Interactive exposure - Explanation - Conversation - Examples - Didactical demonstration	
5. Software design guidelines Layered architecture: UI layer, application layer, domain layer,	- Interactive exposure - Explanation - Conversation - Examples - Didactical demonstration	

³ For example, organisational aspects, recommendations for students, specific aspects relating to the course/seminar, such as inviting experts in the field, etc.

infrastructure layer ● How to organize source code: responsibilities, single responsibility principle, separation of concerns, dependency, coupling, cohesion		
6. Object-oriented programming ● UML diagrams ● Implementation of classes in Python ● Objects and classes: classes, objects, fields, methods, Python scope and namespace	● Interactive exposure ● Explanation ● Conversation ● Examples ● Didactical demonstration	
7. Design aspects ● Top down and bottom up strategies ● UI elements	● Interactive exposure ● Explanation ● Conversation ● Examples ● Didactical demonstration	
8. Program testing and inspection ● Testing methods: exhaustive testing, black box testing, white box testing ● Automated testing, TDD ● File operations in Python	● Interactive exposure ● Explanation ● Conversation ● Examples ● Didactical demonstration	
9. Recursion ● Notion of recursion ● Direct and indirect recursion ● Examples Computational complexity	● Interactive exposure ● Explanation ● Conversation ● Examples ● Didactical demonstration	
10. Search algorithms ● Problem definition ● Search methods: sequential, binary ● Complexity of algorithms Sorting algorithms ● Problem definition ● Sort methods: Bubble Sort, Selection Sort, Insertion Sort, Quick Sort ● Complexity of algorithms	● Interactive exposure ● Explanation ● Conversation ● Examples ● Didactical demonstration	

11. Backtracking ● General presentation of the method ● Algorithms and complexity ● Examples	● Interactive exposure ● Explanation ● Conversation ● Examples ● Didactical demonstration	
12. Divide et impera, Greedy ● General presentation of the methods ● Algorithms and complexity ● Examples	● Interactive exposure ● Explanation ● Conversation ● Examples ● Didactical demonstration	
13. Dynamic programming ● Method description ● Examples	● Interactive exposure ● Explanation ● Conversation ● Examples ● Didactical demonstration	
14. Revision		

Bibliography

1. M. Frentiu, H.F. Pop, Fundamentals of Programming, Cluj University Press, 2006.
2. K. Beck, Test Driven Development: By Example. Addison-Wesley Longman, 2002.
http://en.wikipedia.org/wiki/Test-driven_development
3. M. Fowler, Refactoring. Improving the Design of Existing Code, Addison-Wesley, 1999.
<http://refactoring.com/catalog/index.html>
4. The Python Programming Language - <https://www.python.org/>
5. The Python Standard Library - <https://docs.python.org/3/library/index.html>
6. The Python Tutorial - <https://docs.python.org/3/tutorial/>

8.2. Laboratory	Teaching and learning methods	Remarks
1. Simple Python programs	● Interactive exposure ● Explanation ● Conversation ● Didactical demonstration	
1. Procedural Programming		
2. Modular Programming		
3. Feature-driven software development		
5. Abstract data types		
6. Design principles		
7. Object-oriented programming		
8. Program design. Layered architecture		
9. Inspection and testing		
10. Recursion. Complexity of algorithms		
11. Search and sorting algorithms		

12. Problem solving methods: Backtracking		
13. Problem solving methods: Greedy		
14. Practical test		
Bibliography		
1. M. Frentiu, H.F. Pop, Fundamentals of Programming, Cluj University Press, 2006.		
2. K. Beck, Test Driven Development: By Example. Addison-Wesley Longman, 2002.		
http://en.wikipedia.org/wiki/Test-driven_development		
3. M. Fowler, Refactoring. Improving the Design of Existing Code, Addison-Wesley, 1999.		
http://refactoring.com/catalog/index.html		
4. The Python Programming Language - https://www.python.org/		
5. The Python Standard Library - https://docs.python.org/3/library/index.html		
6. The Python Tutorial - https://docs.python.org/3/tutorial/		
8.3. Seminar	Teaching and learning methods	Remarks
1. Simple Python programs. Procedural Programming	● Interactive exposure ● Explanation ● Conversation ● Didactical demonstration	
2. Modular Programming. Feature-driven software development		
3. Abstract data types. Design principles		
4. Object-oriented programming. Program design. Layered architecture		
5. Inspection and testing. Recursion. Complexity of algorithms		
6. Search and sorting algorithms. Problem solving methods: Backtracking		
7. Problem solving methods: Greedy		
Bibliography		
1. M. Frentiu, H.F. Pop, Fundamentals of Programming, Cluj University Press, 2006.		
2. K. Beck, Test Driven Development: By Example. Addison-Wesley Longman, 2002.		
http://en.wikipedia.org/wiki/Test-driven_development		
3. M. Fowler, Refactoring. Improving the Design of Existing Code, Addison-Wesley, 1999.		
http://refactoring.com/catalog/index.html		
4. The Python Programming Language - https://www.python.org/		
5. The Python Standard Library - https://docs.python.org/3/library/index.html		
6. The Python Tutorial - https://docs.python.org/3/tutorial/		

9. Evaluation

Type of activity	9.1 Evaluation criteria ⁴	9.2 Evaluation methods ⁵	9.3 Percentage in the final grade
9.4. Course	The correctness and completeness of the accumulated knowledge and the capacity to design and implement correct Python programs.	Written exam	40%
9.5. Laboratory	Be able to design, implement and test a Python program	Practical exam	30%
	Correctness of laboratory assignments and documentation delivered during the semester	Program and documentation	30%
9.6. Seminar	Seminar activity	Active participation at the discussions during the seminar (asking and answering questions, volunteering to solve problems, etc.)	Maximum 0.5 points bon
9.6 Minimum standard for passing			
- Each student must demonstrate an acceptable level of knowledge and understanding of the domain, the ability to present knowledge in a coherent manner and the ability to establish connections and use this knowledge to solve different problems in Python. - It is mandatory for each student to attend minimum 10 seminars and 12 labs. - A minimum grade of 5 should be obtained at the lab activity, practical test and written examination			

10. SDG labels (Sustainable Development Goals)⁶

		Sustainable Development Generic Label						
								

⁴ The evaluation criteria must directly reflect the learning outcomes targeted at the level of the degree programme respectively at the level of the subject. More specifically, the learning outcomes set out in the expected learning outcomes are assessed.

⁵ Both final evaluation methods and ongoing evaluation strategies should be established.

⁶ Select a single label which, according to the *Implementation of SDG labels in the academic process*, best matches the subject. If the subject addresses sustainable development in a generic manner (i.e. by presenting/introducing the general framework of sustainable development, etc.), then the Sustainable Development generic label may be applied. If none of the labels describe the subject, select the last option: "No label applies."

☐	☐	☐	☐	☐	☐	☐	☐	x
10 INEGALITĂȚI REDUSE 	11 ORĂȘE ȘI COMUNITĂȚI DURABILE 	12 CONSUM ȘI PRODUCȚIE RESPONSABILĂ 	13 ACȚIUNE CLIMATICĂ 	14 VIAȚĂ ACVATICĂ 	15 VIAȚĂ TERESTRĂ 	16 PACE, JUSTIȚIE ȘI INSTITUȚII EFICIENTE 	17 PARTENERIATE PENTRU REALIZAREA OBIECTIVELOR 	No label applies
☐	☐	☐	☐	☐	☐	☐	☐	☐

Date of entry:
19.10.2026

Signature of course coordinator

Asist. Dr. Anamaria BRICIU

Signature of seminar coordinator

Asist. Dr. Anamaria BRICIU

Date of approval in the department:

...

Signature of the head of department

Conf. Dr. Adrian STERCA