SYLLABUS

Fundamental algorithms

University year 2025-2026

1. Information regarding the programme

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 of study	Computer Science
1.5. Study cycle	Bachelor
1.6. Study programme/Qualification	Artificial Intelligence
1.7. Form of education	Full time

2. Information regarding the discipline

2.1. Name of the discipline	**Fundamental algorithms**					Discipline code	**MLE5200**
2.2. Course coordinator				Asist. dr. Anamaria Briciu			
2.3. Seminar coordinator				Asist. dr. Anamaria Briciu			
2.4. Year of study	1	2.5. Semester	1	2.6. Type of evaluation	E	2.7. Discipline regime	Compulsory

3. Total estimated time (hours/semester of didactic activities)

3.1. Hours per week	**6**	of which: 3.2 course	**2**	3.3 seminar/laboratory/project	**2 S** **2 LP**
3.4. Total hours in the curriculum	84	of which: 3.5 course	28	3.6 seminar/laboratory/project	**56**
Time allotment for individual study (ID) and self-study activities (SA)					**hours**
Learning using manual, course support, bibliography, course notes (SA)					14
Additional documentation (in libraries, on electronic platforms, field documentation)					12
Preparation for seminars/labs, homework, papers, portfolios and essays					14
Tutorship					8
Evaluations					18
Other activities:					
3.7. Total individual study hours	**66**				
3.8. Total hours per semester	**150**				
3.9. Number of ECTS credits	**6**				

4. Prerequisites (if necessary)

4.1. curriculum	
4.2. competencies	

5. Conditions (if necessary)

5.1. for the course	
5.2. for the seminar /lab activities	

6. Specific competencies acquired [1]

[1] One can choose either competences or learning outcomes, or both. If only one option is chosen, the row related to the other option will be deleted, and the kept one will be numbered 6.

Professional/essential competencies	C1.1 Definition and description of programming paradigms and of language specific mechanisms, as well as identification of syntactic and semantic differences. C1.2 Description of existing software applications, on different levels of abstraction (architecture, classes, methods) using adequate basic knowledge. C1.3 Elaboration of adequate source code and testing of components in a well-known programming language, based on given specifications. C1.4 Testing applications based on testing plans. C1.5 Development of units of programs and corresponding documentation
Transversal competencies	TC1 Application of efficient and rigorous working rules, manifest responsible attitudes towards the scientific and didactic fields, underlying the individual potential and respecting professional and ethical principles. TC2 Use of efficient methods and techniques for learning, information, research and development of abilities for knowledge exploitation, for adapting to the needs of a dynamic society and for communication in a widely used foreign language.

7. Objectives of the discipline (outcome of the acquired competencies)

7.1 General objective of the discipline	• To introduce the basic concepts of software engineering (design,implementation and maintenance) and to learn Python programming language
7.2 Specific objective of the discipline	• To introduce the key concepts of programming • To introduce the basic concepts of software engineering • To gain understanding of basic software tools used in development of programs • To learn Python programming language and tools to develop, run, test and debug programs • To acquire and improve a programming style according to the best practical recommendations

8. Content

8.1 Course	Teaching methods	Remarks
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 ● Example: calculator	● 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	
3. Modular programming ● What is a module: Python module definition, variable scope in a module,	● Interactive exposure ● Explanation ● Conversation	

packages, standard module libraries, deployment ● PyCharm	● 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, infrastructure layer ● How to organize source code: responsibilities, single responsibility principle, separation of concerns, dependency, coupling, cohesion	● Interactive exposure ● Explanation ● Conversation ● Examples ● Didactical demonstration	
6. Object-oriented programming ● UML diagrans ● 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	● Interactive exposure ● Explanation	

• Examples	• 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 Seminar / laboratory	Teaching methods	Remarks
1. Simple Python programs	• Interactive exposure • Explanation • Conversation • Didactical demonstration	
2. Procedural Programming		
3. Modular Programming		
4. 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/

9. Corroborating the content of the discipline with the expectations of the epistemic community, professional associations and representative employers within the field of the program

- The course respects the IEEE and ACM Curricula Recommendations for Computer Science studies.
- The course exists in the studying program of all major universities in Romania and abroad.
- The content of the course is considered by the software companies as important for average programming skills.

10. Evaluation

Activity type	10.1 Evaluation criteria	10.2 Evaluation methods	10.3 Percentage of final grade
10.4 Course	The correctness and completeness of the accumulated knowledge and the capacity to design and implement correct Python programs.	Written exam	40%
10.5 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%
10.6. Seminar	Seminar activity	Active participation at the discussions during the seminar (asking and answering questions, volunteering to solve problems, etc.)	Maximum 0.5 points bonus, added to the final grade
10.6 Minimum standard of performance			
● 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.			

11. Labels ODD (Sustainable Development Goals)[2]

Not applicable.

Date:
22.09.2025

Signature of course coordinator

Asist. dr. Anamaria Briciu

Signature of seminar coordinator

Asist. dr. Anamaria Briciu

Date of approval:
...

Signature of the head of department

Assoc.prof.phd. Adrian STERCA

[2] Keep only the labels that, according to the *Procedure for applying ODD labels in the academic process*, suit the discipline and delete the others, including the general one for *Sustainable Development* – if not applicable. If no label describes the discipline, delete them all and write „*Not applicable.*".