SYLLABUS

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	**Computers and Information Technology**
1.5 Study cycle	**Bachelor**
1.6 Study programme / Qualification	**Information Engineering**

2. Information regarding the discipline

2.1 Name of the discipline (en) (ro)			**Computer programming and programming languages Programarea calculatoarelor si limbaje de programare**				
2.2 Course coordinator			**Prof. dr. Camelia Chira**				
2.3 Seminar coordinator			**Prof. dr. Camelia Chira**				
2.4. Year of study	**1**	2.5 Semester	**1**	2.6. Type of evaluation	**C**	2.7 Type of discipline	**Compulsory DF**
2.8 Code of the discipline		MLE5171					

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	1 S S LP
3.4 Total hours in the curriculum	70	Of which: 3.5 course	28	3.6 seminar/laboratory	42
Time allotment:					hours
Learning using manual, course support, bibliography, course notes					18
Additional documentation (in libraries, on electronic platforms, field documentation)					18
Preparation for seminars/labs, homework, papers, portfolios and essays					14
Tutorship					12
Evaluations					18
Other activities:					

3.7 Total individual study hours	80
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	• Projector
5.2. for the seminar /lab activities	• Computers, Python programming language and environment

6. Specific competencies acquired

Professional competencies	C1.1 Recognizing and describing specific concepts to calculability, complexity, programming paradigms and modeling of computing and communication systems C1.2 Using specific theories and tools (algorithms, schemes, models, protocols, etc.) for explaining the structure and the functioning of hardware, software and communication systems C2.1 Describing the structure and operation of hardware, software and communication components C2.3 Construction of hardware and software components of computing systems using design methods, languages, algorithms, data structures, protocols and technologies C2.5 Implementation of hardware, software and communication components
Transversal competencies	CT1 Honorable, responsible, ethical behavior, in the spirit of the law, to ensure the professional reputation CT3 Demonstrating initiative and pro-active behavior for updating professional, economical and organizational culture knowledge

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

7.1 General objective of the discipline	• To know the basic concepts of software engineering (design, implementation and maintenance) and to learn Python programming language
7.2 Specific objective of the discipline	• To know the key concepts of programming • To know 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	• 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, packages, standard module libraries • Eclipse + PyDev	• 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. Object-oriented programming • Abstract data types • Implementation of classes in Python • Objects and classes: classes, objects, fields, methods, Python scope and namespace	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration	
6. 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	
7. 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	
8. Recursion • Notion of recursion • Direct and indirect recursion • Examples • Computational complexity	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration	
9. Search algorithms • Problem definition • Search methods: sequential, binary • Complexity of algorithms	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration	
10. Sorting algorithms • Problem definition • Sort methods: Bubble Sort, Selection Sort, Insertion Sort, Quick Sort	• Interactive exposure • Explanation • Conversation • Examples	

• Complexity of algorithms	• Didactical demonstration	
11. Problem solving methods (I) • General presentation of the Backtracking, Divide & Conquer methods • Algorithms and complexity • Examples	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration	
12. Problem solving methods (II) • General presentation of the Greedy and Dynamic Programming methods • Algorithms and complexity • Examples	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration	
13. Revision • Revision of most important topics covered by the course	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration	
14. Evaluation		

Bibliography

1. M.L. Hetland, Beginning Python: From Novice to Professional, Apress, Third Edition, 2017.
2. M. Frentiu, H.F. Pop, Fundamentals of Programming, Cluj University Press, 2006.
3. K. Beck, Test Driven Development: By Example. Addison-Wesley Longman, 2002. http://en.wikipedia.org/wiki/Test-driven_development
4. M. Fowler, Refactoring. Improving the Design of Existing Code, Addison-Wesley, 1999. http://refactoring.com/catalog/index.html
5. The Python Programming Language - https://www.python.org/
6. The Python Standard Library - https://docs.python.org/3/library/index.html
7. The Python Tutorial - https://docs.python.org/3/tutorial/

8.3 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.L. Hetland, Beginning Python: From Novice to Professional, Apress, Third Edition, 2017.
2. M. Frentiu, H.F. Pop, Fundamentals of Programming, Cluj University Press, 2006.
3. K. Beck, Test Driven Development: By Example. Addison-Wesley Longman, 2002. http://en.wikipedia.org/wiki/Test-driven_development
4. M. Fowler, Refactoring. Improving the Design of Existing Code, Addison-Wesley, 1999. http://refactoring.com/catalog/index.html

5. The Python Programming Language - https://www.python.org/
6. The Python Standard Library - https://docs.python.org/3/library/index.html
7. The Python Tutorial - https://docs.python.org/3/tutorial/

8.2 Seminar	Teaching 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.L. Hetland, Beginning Python: From Novice to Professional, Apress, Third Edition, 2017.
2. M. Frentiu, H.F. Pop, Fundamentals of Programming, Cluj University Press, 2006.
3. K. Beck, Test Driven Development: By Example. Addison-Wesley Longman, 2002. http://en.wikipedia.org/wiki/Test-driven_development
4. M. Fowler, Refactoring. Improving the Design of Existing Code, Addison-Wesley, 1999. http://refactoring.com/catalog/index.html
5. The Python Programming Language - https://www.python.org/
6. The Python Standard Library - https://docs.python.org/3/library/index.html
7. 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

Type of activity	10.1 Evaluation criteria	10.2 Evaluation methods	10.3 Share in the 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 Seminar/lab activities	Be able to design, implement and test a Python program	Practical exam	30%
	Correctness of laboratory	Program and documentation	30%

	assignments and documentation delivered during the semester		

10.6Minimum performance standards

- Each student has to demonstrate an acceptable level of knowledge and comprehension of the domain, and the ability to coherently express the knowledge and use it to solve problems.
- A minimum grade of 5 should be obtained for the practical exam, the written exam and the final grade.

Date	Signature of course coordinator	Signature of seminar coordinator
9.05.2022	Prof. dr. Camelia Chira	Prof. dr. Camelia Chira

Date of approval

24.05.2022

Signature of the head of department

Prof. dr. Laura Dioşan