SYLLABUS

1. Information about the study program

1.1 Higher education institution	Babeș-Bolyai University
1.2 Faculty	Faculty of Psychology and Educational Sciences
1.3 Department	Department of Psychology
1.4 Field of study	Psychology - Cognitive Sciences
1.5 Study cycle	Bachelor level
1.6 Study program / Qualification	Psychologist

2. Information about the course

2.1 Title of the course	Algorithms and Programming						
2.2 Teacher in charge of the lecture	Conf. dr. Camelia Chira						
2.3 Teacher in charge of the seminar	Conf. dr. Camelia Chira						
2.4 Study year	1	2.5 Semester	1	2.6. Examination type	C	2.7 Course type	C

3. Estimated total time (number of hours of teaching activities per semester)

3.1 Number of hours per week	4	out of which: 3.2 lecture	2	3.3 seminar / laboratory	1+1
3.4 Total number of hours in the curriculum	56	out of which: 3.5 lecture	28	3.6 seminar / laboratory	28
Distribution of the allocated amount of time:					hours
Individual study (textbook, course support, bibliography, and notes)					16
Supplementary documentation at the library using specialized electronic platforms in the field					16
Preparing for seminars / laboratories, homework, papers, portfolios, and essays					16
Tutoring					15
Exams					6
Other activities: research activities					

3.7 Total number of hours of individual study	69
3.8 Total number of hours per semester	125
3.9 Number of credits (ECTS)	5

4. Prerequisites (if applicable)

4.1 Curriculum	•
4.2 Competencies	-

5. Requirements (if applicable)

5.1 For the lecture	• Classroom with at least 180 seats, computer and video projector / Online course conducted through the MS Teams platform.
5.2 For the seminar / laboratory	• Room with at least 50 seats, computer and video projector / Online seminar conducted through the MS Teams platform.

6. Specific skills acquired

Professional skills	**Knowledge and understanding** • Definition and description of programming paradigms and of language specific mechanisms, as well as identification of syntactic and semantic differences. • Description of existing software applications, on different levels of abstraction (architecture, classes, methods) using adequate basic knowledge. **Explanation and interpretation** • Elaboration of adequate source code and testing of components in a well-known programming language, based on given specifications. **Instrumental - applicative** • Testing applications based on testing plans. **Attitude**
Transversal skills	• 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. • 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 course (based on the grid of acquired competencies)

7.1 General objective	• To know the basic concepts of software engineering (design, implementation and maintenance) and to learn Python programming language
7.2 Specific objectives	• 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 Lecture	Teaching strategies	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	Lecture, demonstrative example, synthesis of knowledge, guided discovery	
2. Procedural programming • Compound types: list, tuple, dictionary • Functions: test cases, definition, variable scope, calling, parameter passing • Test-driven development (TDD), refactoring	Lecture, demonstrative example, synthesis of knowledge, guided discovery	
3. Modular programming • What is a module: Python module definition, variable scope in a module, packages, standard module libraries, deployment Eclipse + PyDev	Lecture, demonstrative example, synthesis of knowledge, guided discovery	
4. User defined types • How to define new data types: encapsulation, data hiding in Python, guidelines • Introduction to object-oriented programming • Exceptions	Lecture, demonstrative example, synthesis of knowledge, guided discovery	

5. Object-oriented programming • Abstract data types • Implementation of classes in Python • Objects and classes: classes, objects, fields, methods, Python scope and namespace	Lecture, demonstrative example, synthesis of knowledge, guided discovery	
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	Lecture, demonstrative example, synthesis of knowledge, guided discovery	
7. Program testing and inspection • Testing methods: exhaustive testing, black box testing, white box testing • Automated testing, TDD • File operations in Python	Lecture, demonstrative example, synthesis of knowledge, guided discovery	
8. Recursion • Notion of recursion • Direct and indirect recursion • Examples • Computational complexity	Lecture, demonstrative example, synthesis of knowledge, guided discovery	
9. Search algorithms • Problem definition • Search methods: sequential, binary • Complexity of algorithms	Lecture, demonstrative example, synthesis of knowledge, guided discovery	
10. Sorting algorithms • Problem definition • Sort methods: Bubble Sort, Selection Sort, Insertion Sort, Quick Sort • Complexity of algorithms	Lecture, demonstrative example, synthesis of knowledge, guided discovery	
11. Problem solving methods (I) • General presentation of the Backtracking, Divide & Conquer methods • Algorithms and complexity • Examples	Lecture, demonstrative example, synthesis of knowledge, guided discovery	
12. Problem solving methods (II)	Lecture, demonstrative	

• General presentation of the Greedy and Dynamic Programming methods • Algorithms and complexity • Examples	example, synthesis of knowledge, guided discovery	
13. Revision • Revision of most important topics covered by the course	Lecture, demonstrative example, synthesis of knowledge, guided discovery	
14. Evaluation	Lecture, demonstrative example, synthesis of knowledge, guided discovery	

Mandatory references:

- M. Frentiu, H.F. Pop, Fundamentals of Programming, Cluj University Press, 2006
- The Python Programming Language - https://www.python.org/
- The Python Tutorial - https://docs.python.org/3/tutorial/

Optional references:

- M.L. Hetland, Beginning Python: From Novice to Professional, Apress, 2005.
- K. Beck, Test Driven Development: By Example. Addison-Wesley Longman, 2002. http://en.wikipedia.org/wiki/Test-driven_development
- M. Fowler, Refactoring. Improving the Design of Existing Code, Addison-Wesley, 1999. http://refactoring.com/catalog/index.html
- The Python Standard Library - https://docs.python.org/3/library/index.html

8.2 Seminar / laboratory	Teaching strategies	Remarks
1. Simple Python programs	Exposure, conversation	
2. Procedural Programming	Presentation, knowledge synthesis, conceptual clarification, practical activities	
3. Modular Programming	Presentation, knowledge synthesis, conceptual clarification, group activities, guided discovery, practical activities	
4. Abstract data types. Object-oriented programming.	Presentation, knowledge synthesis, conceptual clarification, group activities, guided discovery, practical activities	
5. Program design. Layered architecture	Presentation, knowledge synthesis, conceptual clarification, group activities, guided discovery, practical	

	activities	
6. Search and sorting algorithms	Presentation, knowledge synthesis, conceptual clarification, group activities, guided discovery, practical activities	
7. Practical test	Presentation, knowledge synthesis, conceptual clarification, group activities, Guided discovery, practical activities	

Mandatory references:

- M. Frentiu, H.F. Pop, Fundamentals of Programming, Cluj University Press, 2006
- The Python Programming Language - https://www.python.org/
- The Python Tutorial - https://docs.python.org/3/tutorial/

Optional references:

- M.L. Hetland, Beginning Python: From Novice to Professional, Apress, 2005.
- K. Beck, Test Driven Development: By Example. Addison-Wesley Longman, 2002. http://en.wikipedia.org/wiki/Test-driven_development
- M. Fowler, Refactoring. Improving the Design of Existing Code, Addison-Wesley, 1999. http://refactoring.com/catalog/index.html
- The Python Standard Library - https://docs.python.org/3/library/index.html

9. Correlations between the content of the course and the expectations of the representatives of the epistemic community, professional associations and representative employers in the field related to 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 Weight in the final grade
10.4 Lecture	The correctness and completeness of the accumulated knowledge and the capacity to design and implement correct	Written exam	40%

	Python programs		
10.5 Seminar / laboratory	Be able to design, implement and test a Python program	Practical exam	30%
	Correctness of assignments and documentation	Assignments	30%
10.6 Minimum passing score			
The final grade consists of: a. score obtained in the written exam in proportion of 40% b. practical exam and assignments 60% The minimum passing score is 5.			

Date 18/11/2021

Signature of the teacher in charge of the lecture

Conf. univ. dr. Camelia Chira

[illegible]

Signature of the teacher in charge of the seminar

Conf. univ. dr. Camelia Chira

[illegible]

Approval date in the department

Signature of the Head of the department /director