SYLLABUS

1. Information regarding the programme

1.1 Higher education institution	**Babeş-Bolyai University of Cluj-Napoca**
1.2 Faculty	**Faculty of Mathematics and Computer Science**
1.3 Department	**Departament of Computer Science**
1.4 Field of study	**Computer Science**
1.5 Study cycle	**Bachelor**
1.6 Study programme / Qualification	**Computer Science**

2. Information regarding the discipline

2.1 Name of the discipline	**Fundamentals of Programming**						
2.2 Course coordinator	**Lect. PhD Czibula Istvan Gergely**						
2.3 Seminar coordinator	**Lect. PhD Czibula Gabriela Gergely** **Lect. PhD Diosan Laura** **Assist. PhD Molnar Arthur**						
2.4. Year of study	**1**	2.5 Semester	**1**	2.6. Type of evaluation	**E**	2.7 Type of discipline	**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	2 sem 2 lab
3.4 Total hours in the curriculum	84	Of which: 3.5 course	28	3.6 seminar/laboratory	56
Time allotment:					hours
Learning using manual, course support, bibliography, course notes					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	Class room with projector
5.2. for the seminar /lab	Laboratory with computers; Python programming language environment

activities	

6. Specific competencies acquired

Professional competencies	• Understanding the concepts of programming and software engineering. • Good programming skills in high-level languages. • Learn Python programming language.
Transversal competencies	• The ability to apply the acquired concepts, principles and techniques in solving real world problems. • Responsible execution of lab assignments. • Application of efficient and rigorous working rules. • Manifest responsible attitudes toward the scientific and didactic fields. • Respecting the professional and ethical principles.

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 of software systems) 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 (design, implementation and maintenance of software systems) • To understand the basic software tools • To learn Python programming language, and to get used to Python programming, running, testing, and debugging programs. • To acquire and improve the programming style.

8. Content

8.1 Course	Teaching methods	Remarks
1. Introduction to software development processes • What is programming: Algorithm, Program, Basic Elements Of Python, Python Interpreter, Basic roles in software engineering • How to write programs: Problem Statement, Requirements, Feature Driven Development Process • Example: calculator, iteration modeling	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration	
2. Procedural programming • Structured types: Lists, Tuples, Dictionaries • What is a function: Test cases, Definition, Variable scope, Calls • Passing parameters • Anonymous functions • How to write functions: Apply test-driven	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration	

development (TDD) steps, Refactorings		
3. Modular programming • What is a module: Python module definition, variable scope in a module, packages, standard module libraries, deployment • How to organize the source code: responsibilities, single responsibility principle, separation of concerns, dependency, coupling, cohesion • Common layers in an information system logical architecture • Eclipse+PyDev	• Interactive exposure • Explanation • Conversation • Didactical demonstration	
1. User defined types • How to define new data types: encapsulation, information hiding (data hiding in Python), guidelines, abstract data types	• Interactive exposure • Explanation • Conversation • Didactical demonstration	
5. Deployment principles • Problem statement: a program for managing information (CRUD operations) • Layered architecture: UI layer, Application layer, Domain layer, Infrastructure layer • GRASP patterns • Example of application development: entity, validator, repository, controller • Principles: Information Expert, Low Coupling, High Cohesion, Protected Variation, Single responsibility, Dependency Injection	• Interactive exposure • Explanation • Conversation • Didactical demonstration	
6. Object based programming • Objects and classes: classes, objects, fields, methods, special class methods (operator overloading), Python scope and namespace • UML Diagrams: class diagrams, relationships, associations, invariants • Inheritance: UML generalization, code reuse, overriding, inheritance in Python • Exceptions • Example: working with files in Python, repository implementation using files	• Interactive exposure • Explanation • Conversation • Didactical demonstration	
7. Program design • Top down and bottom up strategies: top down design, bottom up design, bottom up programming style, mixed approach • Organizing the UI • Class invariants	• Interactive exposure • Explanation • Conversation • Didactical demonstration	
8. Program testing and inspection • Testing methods: exhaustive testing, black box testing, white box testing • Testing levels: unit testing, integration testing • Automated testing, TDD • Program inspection: coding style, refactoring	• Interactive exposure • Explanation • Conversation • Didactical demonstration	
9. Recursion	• Interactive exposure	

• Notion of recursion • Direct and indirect recursion • Examples Algorithms complexity • Definition of complexity • Complexity as running time • Complexity as amount of required supplementary memory	• Explanation • Conversation • Didactical demonstration	
10. Algorithms complexity • Empiric analysis and asymptotic analysis • Asymptotic notation: big-o, little-o, big-omega, little-omega, theta; properties • Examples of magnitude orders • Comparison of algorithms from an efficiency point of view • Structural complexity	• Interactive exposure • Explanation • Conversation • Didactical demonstration	
11. Backtracking method • General presentation of the Backtracking method • Backtracking algorithm/subalgorithm and complexity • Extensions of the Backtracking method • Examples	• Interactive exposure • Explanation • Conversation • Didactical demonstration	
12. Division method • General presentation • Description of the subalgorithm • Examples Search algorithms and their complexity • specification of the search problem • search methods • sequential traversal • binary search • complexity of search algorithms	• Interactive exposure • Explanation • Conversation • Didactical demonstration	
13 Sort algorithms and their complexity • Secification of the sort problem • Srt methods: BubbleSort, SelectionSort, InsertionSort, QuickSort, MergeSort • Cmplexity of sort algorithms	• Interactive exposure • Explanation • Conversation • Didactical demonstration	
14. Revision	• Interactive exposure • Conversation	

Bibliography

1. Kent Beck.*Test Driven Development: By Example. Addison-Wesley Longman, 2002*. See also Test-driven development. http://en.wikipedia.org/wiki/Test-driven_development
2. Martin Fowler. *Refactoring. Improving the Design of Existing Code*. Addison-Wesley, 1999. See also http://refactoring.com/catalog/index.html
3. Frentiu, M., H.F. Pop, Serban G., Programming Fundamentals, Cluj University Press, 2006
4. *The Python language reference*. http://docs.python.org/py3k/reference/index.html
5. *The Python standard library*. http://docs.python.org/py3k/library/index.html
6. *The Python tutorial*. http://docs.python.org/tutorial/index.html

8.2 Seminar	Teaching methods	Remarks
		The seminar is structured as 2 hours

		classes every week
1. Python programs	● Interactive exposure ● Explanation ● Conversation ● Didactical demonstation	
2. Procedural programming	● Interactive exposure ● Explanation ● Conversation ● Didactical demonstation	
3. Modular programming	● Interactive exposure ● Explanation ● Conversation ● Didactical demonstation	
4. User defined types	● Interactive exposure ● Explanation ● Conversation ● Didactical demonstation	
5. Deployment principles	● Interactive exposure ● Explanation ● Conversation ● Didactical demonstation	
6. Object based programming	● Interactive exposure ● Explanation ● Conversation ● Didactical demonstation	
7. Programs design	● Interactive exposure ● Explanation ● Conversation ● Didactical demonstation	
8. Program testing and inspection	● Interactive exposure ● Explanation ● Conversation ● Didactical demonstation	
9. Recursion. Algorithms complexity	● Interactive exposure ● Explanation ● Conversation ● Didactical demonstation	
10. Algorithms complexity	● Interactive exposure ● Explanation ● Conversation ● Didactical demonstation	
11. Backtracking	● Interactive exposure	

	• Explanation • Conversation • Didactical demonstation	
12. Division method. Search algorithms	• Interactive exposure • Explanation • Conversation • Didactical demonstation	
13. Preparation for the practical test	• Interactive exposure • Explanation • Conversation • Didactical demonstation	
14: Preparation for the written exam	• Interactive exposure • Explanation • Conversation • Didactical demonstation	
8.3 Laboratory	Teaching methods	Remarks
		• The lab is structured as 2 hours classes every week. • The lab documents are due one week after the lab theme has been given and the lab programs are due two weeks later.
1. Simple Python program	• Lab assignment • Explanation • Conversation	
2. Feature driven software development process	• Lab assignment • Explanation • Conversation	
3. Feature driven software development process	• Lab assignment • Explanation • Conversation	
4. Feature driven software development process	• Lab assignment • Explanation • Conversation	
5. Layered architecture	• Lab assignment • Explanation • Conversation	
6. Layered architecture	• Lab assignment • Explanation • Conversation	
7. Layered architecture	• Lab assignment • Explanation • Conversation	
8. Text files	• Lab assignment • Explanation	

	• Conversation	
9. Testing	• Lab assignment • Explanation • Conversation	
10. Algorithms complexity	• Lab assignment • Explanation • Conversation	
11. Backtracking method	• Lab assignment • Explanation • Conversation	
12. Lab delivery time (see remark above)	• Lab assignment • Explanation • Conversation	
13. Lab delivery time (see remark above)	• Lab assignment • Explanation • Conversation	
14. Practical test simulation	• Lab assignment • Explanation • Conversation	

Bibliography

1. Kent Beck.*Test Driven Development: By Example. Addison-Wesley Longman, 2002*. See also Test-driven development. http://en.wikipedia.org/wiki/Test-driven_development
2. Martin Fowler. *Refactoring. Improving the Design of Existing Code*. Addison-Wesley, 1999. See also http://refactoring.com/catalog/index.html
3. Frentiu, M., H.F. Pop, Serban G., Programming Fundamentals, Cluj University Press, 2006
4. *The Python language reference*. http://docs.python.org/py3k/reference/index.html
5. *The Python standard library*. http://docs.python.org/py3k/library/index.html
6. *The Python tutorial*. http://docs.python.org/tutorial/index.html

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 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 (in the regular session)	40%
10.5 Seminar/Lab activities	• Be able to design, test and debug a Python program	Practical evaluation (in the regular session)	30%

	• Correctness of Python programs and lab documentations	-documentation -portofolio -continuous observations	30%
10.6 Minimum performance standards			
• Each student has to prove that (s)he acquired an acceptable level of knowledge and understanding of the, that (s)he is capable of stating these knowledge in a coherent form, that (s)he has the ability to establish certain connections and to use the knowledge in solving different problems in Python programming language. • Successful passing of the exam is conditioned by a minimum grade of 5 at the lab activity, practical test and written exam.			

Date Signature of course coordinator Signature of seminar coordinator

30.04.2013 Lect. dr. Istvan Gergely Czibula Lect. dr. Istvan Gergely Czibula

Date of approval Signature of the head of department

Prof. dr. Bazil Pârv