COURSE OUTLINE

1. Study programme information

1.1 Higher education institution	Universitatea de Vest din Timişoara
1.2 Faculty / Department	Chimie, Biologie, Geografie / Departamentul de Geografie
1.3 Sub-department	Geografie
1.4 Field of study	Geography
1.5 Level of study	Master's degree
1.6 Study programme / Qualification	Geographic Information Systems

2. Course information

2.1 Course title				**Introduction to programming**			
2.2 Course convenor/ Lecturer				Lect. Dr. Dornik Andrei			
2.3 Teaching assistant				Lect. Dr. Dornik Andrei			
2.4 Year of study	1	2.5 Semester	1	2.6 Type of assessment	E	2.7 Course type	DS/ DO

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

3.1 Number of hours per week	3	of which: 3.2 lecture	1	3.3 seminar/laboratory	2
3.4 Total hours in the curriculum	42	of which: 3.5 lecture	14	3.6 seminar/laboratory	28
Time distribution:					**hours**
Studying textbooks, course materials, bibliography and notes					35
Further research in libraries, on electronic platforms and in the field					35
Preparing seminars/ laboratories, homework, research papers, portfolios and essays					20
Tutoring					9
Examinations					9
Other activities ……					
3.7 Total hours of individual study	**108**				
3.8 Total hours per semester	**150**				
3.9 Number of credits	**6**				

4. Prerequisites (if applicable)

4.1 based on curriculum	Basics in informatics; Geographic Information Systems; Geoinformatics
4.2 based on competencies	Basic skills of computer use; analytical spirit and the ability to break down problems into sub-problems

5. Conditions (if applicable)

5.1 for the course	• Computer / laptop with audio-video system for the teacher and students • internet access; access to the Elearning UVT platform; • video projector
5.2 for the seminar/laboratory	• complete fulfilment of tasks of laboratory work and projects • Computer / laptop with audio-video system for the teacher and students; • internet access; access to the Elearning UVT platform; • video projector

6. Objectives of the discipline - expected learning outcomes to the formation of which contribute to the completion and promotion of the discipline

Knowledges	• Basic knowledge of computer science and mathematics • Concepts related to the structure and operation of a computing system • Concepts and methodologies regarding the analysis, design and implementation of computer applications • Understanding the operation of an algorithm
Skills	• Create algorithms in pseudocode • Analysis of the complexity and correctness of a code • Implementation and testing of a program • The ability to identify algorithms and data structures appropriate to a particular problem, to apply the principles of computer application development, and to implement algorithms in a programming language • The ability to use programming environments/tools/platforms specific to each stage in the development of program • The ability to use file systems, to manage processes specific to a computing system, to ensure effective communication between software components
Responsibility and autonomy	• Development of a critical and analytical spirit among students; appreciating the advantages of using algorithmic thinking • The ability to solve specific tasks autonomously • The ability to identify/select appropriate solutions and generate innovative ideas • The ability to correctly/effectively identify and plan tasks specific to a particular project • The application of effective and responsible work strategies, based on the principles, norms and values of the code of professional ethics • Application of effective work techniques in a multidisciplinary team, ethical attitude, respect for diversity and multiculturalism, acceptance of diversity of opinion • Self-assessment of the need for continuous professional training for the purpose of insertion and adaptability to the requirements of the labor market • Capitalizing on the results obtained to analyses, studies and geographical projects

7. Content

7.1 Lecture	**Teaching methods**	**Observations**
1. Introduction to algorithms. The notion of algorithm. The objectives of programming. Properties of algorithms. Data and data classifications. Simple processing. Structured processing (sequential, decision, cycle).	Lecture, Interactive presentations, heuristic conversation, problematization and hands-on examples	2 hours
2. Description of algorithms and variables. Pseudocode. Description of fundamental processing and structured data. Examples of simple algorithms (calculations of sums and finite products, approximation of infinite sums, operations on whole numbers, operations on tables). Successive refinement technique and decomposition of an algorithm into subalgorithms.		2 hours
3. Lists and dictionaries. Definition and access to elements. Simple operations on lists. Lists. Elementary sorting methods. The problem and method of inserting, selecting and exchanging neighboring elements (for each method: variants of the algorithm, correctness verification, complexity analysis).		3 hours
4. Functions. Local variables. Parameter specification. Calling functions. Returning the results.		2 hours
5. Working with files. Exception handling.		1 hours
6. Implementation of recursive functions. Implementation of recursive		2 hours

algorithms (generation of permutations, generation of subsets).		
7. Algorithm debugging. The stages of verifying the correctness of the algorithms. Elements of formal analysis of correctness: preconditions, postconditions, invariants, termination functions. Analysis of the complexity of algorithms. Purpose of the analysis. Analyzed resources. Estimation of execution time (best case, worst case, average case). Examples: finite sums, product of two matrices, minimum determination, sequential search.		2 hours

Bibliography

- Elkner J., Downey A.B., Meyers C., How to think like a computer scientist. Learning with Python, Green Tea Press, 2002
- Lutz, M., Learning Python, 3rd edition, O. Reilly, 2007
- Swaroop C. H., Zimmerhoff J., A Byte of Python, 2017, ISBN 1977878490
- Tanimoto S., Introduction to Python for Artificial Intelligence, IEEE Computer Society
- Additional references and course presentations are posted on Elearning UVT Platform (https://elearning.e-uvt.ro/)

7.2 Seminar / laboratory	**Teaching methods**	**Observations**
1. Introduction to Python. Installation. Familiarization with the interface. The read-evaluate-print cycle. Evaluation of expressions. Simple mathematical operations.	Hands-on exercises, case studies, scientific explanation and demonstration.	2 hours
2. Specifying variables (numeric, logical, character strings). Rules for constructing expressions. Explicit display of results. Specifying conditional and repetitive processing.		2 hours
3. Lists and dictionaries. Definition and access to elements. Simple operations on lists.		2 hours
4. Definition of functions. Local variables. Parameter specification. Calling functions. Returning the results.		2 hours
5. Working with files. Exception handling.		2 hours
6. Processing of lists. Implementation of search and sorting algorithms		2 hours
7. Implementation of recursive functions. Implementation of recursive algorithms (generation of permutations, generation of subsets).		2 hours
8. Implementation of algorithms based on the division technique (binary search, simple algorithms from computational geometry)		2 hours
9. Implementation of quick sorting and sorting by interclassing		2 hours
10. Other data types (tuples). The difference between modifiable and non-modifiable types. Implementation of heuristic algorithms.		2 hours
11. Peculiarities of working with multidimensional lists. Implementation of optimization algorithms based on dynamic programming.		2 hours
12. Modules and packages. Creation and use. Import and reload functions. Namespaces.		2 hours
13. Implementation of some algorithms based on the return search technique. Implementation of some algorithms based on the branch and bound technique.		2 hours
14. Evaluation, Feedback		2 hours

Bibliography

- Downey A.B., How to think like a computer scientist. Learning with Python, Green Tea Press, 2002
- Lutz, M., Learning Python, 3rd edition, O. Reilly, 2007
- Swaroop C. H., Zimmerhoff J., A Byte of Python, 2017, ISBN 1977878490
- Tanimoto S., Introduction to Python for Artificial Intelligence, IEEE Computer Society
- Additional references and course presentations are posted on Elearning UVT Platform (https://elearning.e-uvt.ro/)

8. Corroborating course content with the expectations held by the representatives of the epistemic community, professional associations and typical employers in the field of the study programme

The content of the discipline was developed in accordance with the curriculum and meets the didactic and scientific requirements corresponding to similar specializations in other university centers. Introduction to programming facilitates the acquisition of basic knowledge in carrying out a research project, both from a theoretical point of view and from the point of view of working methods in the field, developing students' analytical thinking, the ability to problematize, to manage a scientific approach, of a database and its operation. The software used in the practical applications are among the most modern and frequently used in specialized institutions. Such applied training makes students compatible with the job market in the field of geographic information systems, or research activity.

9. Assessment

Type of activity	9.1 Assessment criteria	9.2 Assessment methods	9.3 Weight in the final mark
9.4 Lecture	Understanding and assimilation of theoretical knowledge	Test	20%
9.5 Seminar / laboratory	Develop a program in Python	Continuous formative evaluation – presentation and feedback during the semester	20%
		Presentation of the final Python program/code	40%
	Test programming skills	Practical test	20%
9.6 Minimum performance standard			
• Minimum mark 5 at course evaluation. • Minimum mark 5 at practical activities.			

Date

11.09.2023

Course convenor's signature

Lect. Dr. Andrei Dornik

Date of approval in the department

Head of department's signature