SYLLABUS / FIȘA DISCIPLINEI

1. Information on the study programme / Date despre programul de studii

1.1. Institution / Instituția de învățământ superior	Universitatea de Vest din Timișoara
1.2. Faculty / Facultatea	Matematică și Informatică
1.3. Department / Departamentul	Computer Science (Informatică)
1.4. Study program field	Computer Science (Informatică)
1.5. Study cycle/ Ciclul de studii	Bachelor / licență
1.6. Study programme / Programul de studii / calificarea*	Computer Science / Informatică în limba engleză / Database administration / *Administrator baze de date - 252101; Computer network administration / Administrator de retea de calculatoare - 252301; Analyst / Analist - 251201; Research assistant in computer science / Asistent de cercetare în informatica - 214918; Teacher in secondary schools / Profesor în învatamântul gimnazial - 233002; Programmer / Programator - 251202; Software systems designers / Proiectant sisteme informatice - 251101*

2. Information on the course / Date despre disciplină

2.1. Title of the course / Denumirea disciplinei	Programming I							
2.2. Teacher in charge of the course / Titularul activităților de curs	Cosmin Bonchis							
2.3. Teacher in charge of the seminar / Titularul activităților de seminar	Teodora Selea, Adrian Spataru, Cosmin Bonchis							
2.4. Study year / Anul de studii	1	2.5. Semester / Semestrul	1	2.6. Examination type / Tipul de evaluare: E(xam)/C(olloquim)	E	2.7. Course type / Regimul disciplinei: M(andatory)/ E(lective)/ F(acultative)	M	

3. Estimated study time (number of hours per semester) /Timpul total estimat (ore pe semestru al activităților didactice)

3.1. Attendance hours per week / Număr de ore pe săptămână	4	out of which din care: 3.2 lecture/ curs	2	3.3. seminar/laborator	2
3.4. Attendance hours per semester / Total ore din planul de învățământ	56	out of which: 3.5 lecture / curs	28	3.6. seminar/laborator	28

Distribution of the allocated amount of time / Distribuția fondului de timp*	**hours/ ore**
Individual study /Studiu după manual, suport de curs, bibliografie și notițe	15
Supplementary documentation at library or using electronic repositories / Documentare suplimentară în bibliotecă, pe platformele electronice de specialitate	20
Preparing for laboratories, homework, reports etc. /Pregătire seminarii/laboratoare, teme, referate, portofolii și eseuri	60
Exams / Examinări	20

Tutoring / Tutorat	25
3.7. Total number of hours of individual study / Total ore studiu individual	140
3.8. Total number of hours per semester / Total ore pe semestru	196
3.9. Number of credits (ECTS) / Număr de credite	6

4. Prerequisites (if it is the case) / Precondiții (acolo unde e cazul)

4.1. curriculum / de curriculum	
4.2. skills / de competențe	

5. Requirements (if it is the case) / Condiții (acolo unde e cazul)

5.1. for the lecture / de desfășurare a cursului	• Cursuri online folosind Google Meet/Teams/Webex (sau alte tooluri potrivite pentru cursuri online) și Google Classroom • sau Sală de curs cu tablă și videoproiector
5.2. for the seminar, laboratory / de desfășurare a seminarului/laboratorului	• Laboratoare online folosind Google Meet/Teams/Webex (sau alte tooluri potrivite pentru cursuri online) și Google Classroom • sau Sala de laborator dotată corespunzător (calculatoare cu software instalat Python)

6. Acquired skills / Competențe specifice acumulate

Professional skills / Competențe profesionale	Learning the correct terminology and correct understanding of concepts and their utility Understanding the programming specific mechanisms Understanding the characteristics of language and the applications. The ability to analyze specific situations and to interpret / explain the correct meaning of a sequence of code / program Developing the capacity to design and carry out end-cycle development of complex applications, small / medium

Transversal skills / Competențe transversale	Responsiveness / attention, interest shown by individual search questions and answers Helping colleagues in all occasions except examination activities

7. Objectives of the course / Obiectivele disciplinei (reieșind din grila competențelor specifice acumulate)

7.1. General objective / Obiectivul general al disciplinei	The course is intended to familiarize the students with a programming language (the Pyhon language) and to form analitical thinking for problems solving. We intend to acquire the basic elements of the Python language and learning the minimum experience in their use. Students should improve their knowledge and skills needed to use an integrated application development environment.
7.2. Specfic objectives / Obiectivele specifice	It is an introductionary course that wants to discuss the paradigms that are encontered in programming languages (fuctional, object oriented programmig) and develop the ability of extract models from the proposed problem (identify usfull data structures, interactions between data structures and storage type).

8. Content / Conținuturi*

8.1. Lecture / Curs	**Teaching strategies / Metode de predare**	**Remarks, details / Observații**
1. What is computer Science? Hardware basics. Programming languages. Python programs. Data types in Python, Operators, Expressions, Assignments, Conditional Statements. Elements of Program: names, expressions. Interpreted languages vs compiled languages	Lectures, illustration, demonstration	2h
2.Software development process. Lopps: for, while, break, continue. Data structurse and sequences: list, tuples, dictionarys, set. Idea of aliasing, idea of mutability, idea of cloning.	Lectures, illustration, demonstration	4h
3.Unstructured programming -> Procedural Programming (decomposition, abstraction). Functions. Functions parameters. Functions call. Function context call. Local variables. Global Variables	Lectures, illustration, demonstration	2h

4. Modules. Program documentation. Metaprogramming - python featur. Procedural Programming -> Modular Programming. Modules in Python. Comments. Methods comments	Lectures, illustration, demonstration	2h
5. Testing and debugging the programs. Exceptions. Testing, inspection and debugging programs. Black box vs white box testing. Unit testing, integration testing. Automate testing, PyUnit. Debuging. Exceptions. Assert, code coverage	Lectures, illustration, demonstration	2h
6.Strings. Regular expressions. String tokens . Python string operation. Searching a substring into a string – brute force algorithms. String matching a pattern. Introduce regex – usage examples -> extract tokens from a string	Lectures, illustration, demonstration	2h
7.Study case: Identification of anagram words from a dictionary	Lectures, illustration, demonstration	2h
8. Numpy, aggregate information, image processing example	Lectures, illustration, demonstration	2h
9. Abstract data types. Classes. Objects. Modular Programming -> Abstract Data Types. What is an Abstract data type. Data type definition. Classes. Objects. Constructors. Member methods. Data and Methods visibility (public & private). Information hiding. Operator overloads	Lectures, illustration, demonstration	2h
10 Streams. Files. Streams definitions. Files a type of stream. Text files reading and writing. Binary files reading and writing. Files and operating system (checking file existence, getting directory listing, creating directories). Structure information in text files CSV, JSON, XML. Objects serialization. CRUD operations on data entity.	Lectures, illustration, demonstration	2h
11.Interaction between objects. Inheritance. Dependence. Aggregation. Composition.	Lectures, illustration, demonstration	2h

Program to an Interface, not to an Implementation. Hollywood Principle. Favor composition over inheritance		
12.Refactoring. SOLID - Single responsibility principle. Open/closed principle. Liskov substitution principle. Interface separation principle. Dependency inversion principle. GRASP - High Cohesion, Low Coupling, Polymorphism, Information Expert, Creator, Pure Fabrication, Controller, Indirection.	Lectures, illustration, demonstration	2h
13.Recap	Lectures, illustration, demonstration	2h

Recommended bibliography / Bibliografie

John Zelle, Python Programming: An Introduction to Computer Science
MIT, Introduction to comuter science - course
Mark Lutz - Learning Python, 5th Edition Powerful Object-Oriented Programming
Mark Summerfield - Programming in Python 3 (Second Edition) A Complete Introduction to the Python Language

8.2. Seminar, lab / Seminar, laborator	**Teaching/learning strategies / Metode de predare/ invățare**	**Remarks, details / Observații**
Simple Expresions. Conditonal Statements	Problem solving, questioning, dialogue	2h
Loops. List. Tuples. Sets. Maps	Problem solving, questioning, dialogue	4h
Functions	Problem solving, questioning, dialogue	2h
Strings	Problem solving, questioning, dialogue	2h
Testing	Problem solving, questioning, dialogue	2h
Test case	Problem solving, questioning, dialogue	4h
Classes	Problem solving, questioning, dialogue	2h
Files	Problem solving, questioning, dialogue	2h
Inheritance and composition	Problem solving, questioning, dialogue	4h
Test case	Problem solving, questioning, dialogue	4h

Recommended bibliography / Bibliografie

John Zelle, Python Programming: An Introduction to Computer Science
MIT, Introduction to comuter science - course
Mark Lutz - Learning Python, 5th Edition Powerful Object-Oriented Programming

Mark Summerfield - Programming in Python 3 (Second Edition) A Complete Introduction to the Python Language

9. Correlations between the content of the course and the requirements of the IT field / Coroborarea conținuturilor disciplinei cu așteptările reprezentanților comunității epistemice, asociațiilor profesionale și angajatorilor reprezentativi din domeniul aferent programului

10. Evaluation / Evaluare*

Activity / Tip de activitate	10.1. Evaluation criteria / Criterii de evaluare**	10.2. Evaluation methods / Metode de evaluare***	10.3. Weight in the averaged mark / Pondere din nota finală
10.4. Lecture / Curs	*Theoretical Examination*	Quiz with multiple choices, Problem resolving	30%
10.5. Seminar/ lab	Laboratoy tests	Quiz with multiple choices, Problem solving	70%
10.6. Minimal knowledge for passing / Standard minim de performanță			
Knowledge of basic principles of programming. *Knowing the structure of a computer program. Getting an average over 5 for all tests. Minimum 5 per examination task.*			

Date/ Data completării

20.09.2020

Signature (lecture) / Semnătura titularului de curs

Signature (seminar) Semnătura titularului de seminar

Signature (director of the department)
Semnătura directorului de departament