

SYLLABUS/FIŞA DISCIPLINEI

1. Date despre program

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

2. Date despre disciplină

| | | | | | | | |
|--|----------------------|---------------|----------|-----------------------|----------|--|----------|
| 2.1 Denumirea disciplinei | Programming I | | | | | | |
| 2.2 Titularul activităților de curs | Adrian Spătaru | | | | | | |
| 2.3 Titularul activităților de seminar | Adrian Spătaru | | | | | | |
| 2.4 Anul de studiu | 1 | 2.5 Semestrul | 1 | 2.6 Tipul de evaluare | E | 2.7 Regimul disciplinei: M(andatory)/ E(lective)/ F(acultative) | M |

3. Timpul total estimat (ore pe semestru al activităților didactice)

| | | | | | |
|---|------------|--------------------|----|-----------------------|-----|
| 3.1 Număr de ore pe săptămână | 4 | din care: 3.2 curs | 2 | 3.3 seminar/laborator | 2 |
| 3.4 Total ore din planul de învățământ | 56 | din care: 3.5 curs | 28 | 3.6 seminar/laborator | 28 |
| Distribuția fondului de timp: | | | | | ore |
| Studiul după manual, suport de curs, bibliografie și notițe | | | | | 15 |
| Documentare suplimentară în bibliotecă, pe platformele electronice de specialitate / pe teren | | | | | 14 |
| Pregătire seminare / laboratoare, teme, referate, portofolii și eseuri | | | | | 50 |
| Tutoriat | | | | | 25 |
| Examinări | | | | | 20 |
| Alte activități | | | | | |
| 3.7 Total ore studiu individual | 124 | | | | |
| 3.8 Total ore pe semestru | 180 | | | | |
| 3.9 Numărul de credite | 6 | | | | |

4. Precondiții (acolo unde este cazul)

| | |
|-------------------|---|
| 4.1 de curriculum | <ul style="list-style-type: none"> • basic mathematical and logical operations |
| 4.2 de competențe | <ul style="list-style-type: none"> • medium level of computer usage |

5. Condiții (acolo unde este cazul)

| | |
|--|---|
| 5.1 de desfășurare a cursului | <ul style="list-style-type: none"> • Lecture hall with whiteboard and video projector |
| 5.2 de desfășurare a seminarului / laboratorului | <ul style="list-style-type: none"> • Face to face: Lab room with computers with Python installed and access to the internet. |

6. Obiectivele disciplinei - rezultate așteptate ale învățării la formarea cărora contribuie parcurgerea și promovarea disciplinei

| | |
|-------------------------------|---|
| Cunoștințe | C1 - fundamente informatică & matematică C2 - structura și funcționarea unui sistem de calcul C3 - analiză/proiectare/implementare sisteme informative |
| Abilități | A2 - utilizare sis. de fișiere, gestionare procese A3 - identificare, implementare alg. A4 - utilizare medii/instrumente/platforme de programare |
| Responsabilitate și autonomie | R1 - rezolvare autonomă sarcini R2 - identificare soluții, idei inovative R3 - planificare eficientă sarcini R4 - gestiune eficientă resurse R5 - asumare sarcini, respectare etică R6 - adaptare la noi cerințe |

7. Conținuturi

| 7.1 Curs | Metode de predare | Detalii/Observații |
|---|---------------------------------------|--------------------|
| 1. What is computer Science? Hardware basics. Programming languages. Python | Lectures, illustration, demonstration | 2h |

| | | |
|--|---------------------------------------|----|
| programs. Data types in Python, Operators, Expressions, Assignments, Conditional Statements. Elements of Program: names, expressions. Interpreted languages vs compiled languages | | |
| 2-3. Software development process. Loops: for, while, break, continue. Data structures and sequences: list, tuples, dictionaries, set. Idea of aliasing, idea of mutability, idea of cloning. | Lectures, illustration, demonstration | 4h |
| 4. Unstructured programming -> Procedural Programming (decomposition, abstraction). Functions. Function parameters. Functions call. Function context call. Local variables. Global Variables | Lectures, illustration, demonstration | 2h |
| 5. Modules. Program documentation. Metaprogramming - python feature. 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 |
| 6. Testing and debugging the programs. Exceptions. Testing, inspection and debugging programs. Black box vs white box testing. Unit testing, integration testing. Automate testing, PyUnit. Debugging. Exceptions. Assert, code coverage | Lectures, illustration, demonstration | 2h |
| 7. Study case (live design and coding based on problems proposed by the audience). Project description announcement | Lectures, illustration, demonstration | 2h |
| 8-9. Streams. Files. Streams definitions. Files a type of stream. | Lectures, illustration, demonstration | 4h |

| 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. Object serialization. CRUD operations on data entity. | | |
|--|---|--------------------------------------|
| 10. Abstract data types. Classes. Objects. Modular Programming -> Abstract Data Types. What is an Abstract data type? Classes. Objects. Constructors. Member methods. Data and Methods visibility (public & private). Information hiding. Operator overloads | Lectures, illustration, demonstration | 2h |
| 11. Interaction between objects. Inheritance. Dependence. Aggregation. Composition. <u>Program to an Interface, not to an Implementation. Hollywood Principle. Favor composition over inheritance</u> | Lectures, illustration, demonstration | 2h |
| 12. Refactoring. Polymorphism | Lectures, illustration, demonstration | 2h |
| 13. Study case (adapting previous study case with files and object-oriented concepts) Recap | Lectures, illustration, demonstration | 2h |
| 14. Project presentations | Lectures, illustration, demonstration | 2h |
| Recommended bibliography / Bibliografie | | |
| John Zelle, Python Programming: An Introduction to Computer Science MIT, Introduction to computer 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/ învățare | Remarks, details / Observații |
| Lab 1: Simple Expressions. Conditional Statements | Problem solving, questioning, dialogue | 2h |
| Lab 2-3: Loops. Lists. Matrices. Tuples. Sets. Maps | Problem solving, questioning, dialogue | 4h |

| | | |
|---|--|----|
| Lab 4: Functions | Problem solving, questioning, dialogue | 2h |
| Lab 5: Lab Test | Problem solving, questioning, dialogue | 2h |
| Lab 6: Strings | Evaluation | 2h |
| Lab 7: Debug, Exceptions, Assertions | Problem solving, questioning, dialogue | 4h |
| Lab 8-9: Files | Problem solving, questioning, dialogue | 4h |
| Lab 10: Lab Test | Problem solving, questioning, dialogue | 2h |
| Lab 11-12: Classes and objects. | Problem solving, questioning, dialogue | 4h |
| Lab 13: Inheritance and composition. Polymorphism | Problem solving, questioning, dialogue | 2h |
| Lab 14: Recap | Problem solving, questioning, dialogue | 2h |

Recommended bibliography / Bibliografie

John Zelle, Python Programming: An Introduction to Computer Science

MIT, Introduction to computer 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. 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 angajatorii reprezentativi din domeniul aferent programului

The course content covers commonly used practices used for development of software applications for medium to large size industries projects. A high mark on this grade (>=80%) demonstrates the independence to solve a problem in an efficient and extensible manner, as well as to test its behaviour and justify the implementation process. These skills can help a student pass most interviews for a summer internship on the local and national software development labor market.

9. 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 |
|------------------------------|--|--|---|
| 9.4. Lecture / Curs | <i>Theoretical Examination or Project</i> | Quiz with multiple choices Problem modeling and solving | 30% |
| 9.5. Seminar/ lab | Laboratory tests | Quiz with multiple choices, Problem solving | 70% |

9.6 Standard minim de performanță

10.6. Minimal knowledge for passing / Standard minim de performanță

Knowledge of basic principles of programming.

Knowing the structure of a computer program.

The final mark is calculated as the weighted mean of the marks presented in section 9. The subject is passed if the final mark is greater or equal to 5 and the exam mark (9.4) is greater or equal to 5.

Important!!

To be eligible for the written exam (final exam, in the exam sessions A-I, B-I, C) the student must have grades for the Lab Tests (9.5) which average to a mark ≥ 5 at the end of the semester. Tests can only be taken once.

If a student does not meet the above requirement, it means he/she has not met the minimum required activity for this course. Conform with university regulations, this means that this subject must be re-contracted next year. If the above requirements are met but the final exam is failed, the student should only apply for re-examination.

Data completării

18.09.2023

Titular de disciplină

Data avizării în departament

Director de departament