FIŞA DISCIPLINEI

1. Date despre program

1.1 Instituţia de învăţământ superior	**Universitatea Babeş-Bolyai Cluj-Napoca**
1.2 Facultatea	**Facultatea de Biologie şi Geologie**
1.3 Departamentul	**Departamentul de Biologie Moleculară și Biotehnologie**
1.4 Domeniul de studii	**Biologie**
1.5 Ciclul de studii	**Master**
1.6 Programul de studiu / Calificarea	**Bioinformatică aplicată în științele vieții**

2. Date despre disciplină

2.1 Denumirea disciplinei (ro) (en)	**Algoritmi şi Programare** **Algorithms and Programming**						
2.2 Titularul activităţilor de curs	**Prof. dr. Camelia Chira**						
2.3 Titularul activităţilor de seminar	**Prof. dr. Camelia Chira**						
2.4 Anul de studiu	**1**	2.5 Semestrul	**1**	2.6. Tipul de evaluare	**C**	2.7 Regimul disciplinei	**Optional**
2.8 Codul disciplinei	**MLE5115**						

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

3.1 Număr de ore pe săptămână	6	Din care: 3.2 curs	2	3.3 seminar/laborator	2 sem 2 lab
3.4 Total ore din planul de învăţământ	84	Din care: 3.5 curs	28	3.6 seminar/laborator	56
Distribuţia fondului de timp:					ore
Studiul după manual, suport de curs, bibliografie şi notiţe					14
Documentare suplimentară în bibliotecă, pe platformele electronice de specialitate şi pe teren					12
Pregătire seminarii/laboratoare, teme, referate, portofolii şi eseuri					14
Tutoriat					8
Examinări					18
Alte activităţi:					

3.7 Total ore studiu individual	66
3.8 Total ore pe semestru	150
3.9 Numărul de credite	6

4. Precondiţii (acolo unde este cazul)

4.1 de curriculum	•
4.2 de competenţe	•

5. Condiţii (acolo unde este cazul)

5.1 De desfăşurare a cursului	• Videoproiector • Platformă de comunicare online
5.2 De desfăşurare a seminarului/laboratorului	• Calculatoare, medii de dezvoltare şi implementare pentru limbajul Python

6. Competenţele specifice acumulate

Competenţe profesionale	C1.1 Definirea şi descrierea paradigmelor de programare şi mecanismelor specifice limbajului, precum şi identificarea diferenţelor sintactice şi semantice. C1.2 Descrierea aplicaţiilor software existente, pe diferite nivele de abstractizare (arhitectură, clase, metode) folosind cunoştinţe de bază adecvate. C1.3 Elaborarea unui cod sursă adecvat şi testarea componentelor într-un limbaj de programare bine-cunoscut, pe baza unor specificaţii date. C1.4 Testarea aplicaţiilor pe baza unor planuri de testare. C1.5 Dezvoltarea unor unităţi de programe şi documentaţia corespunzătoare.
Competenţe transversale	CT1. Aplicarea regulilor de muncă organizată şi eficientă, a unor atitudini responsabile faţă de domeniul didactic-ştiinţific, pentru valorificarea creativă a propriului potenţial, cu respectarea principiilor şi a normelor de etică profesională CT2. Utilizarea unor metode şi tehnici eficiente de învăţare, informare, cercetare şi dezvoltare a capacităţilor de valorificare a cunoştinţelor, de adaptare la cerinţele unei societăţi dinamice și de comunicare în limba română și într-o limbă de circulație internațională

7. Obiectivele disciplinei (reieşind din grila competenţelor acumulate)

7.1 Obiectivul general al disciplinei	• Învățarea conceptelor de bază din ingineria software (proiectare, implementare şi mentenanţă) şi învăţarea limbajului de programare Python
7.2 Obiectivele specifice	• Cunoaşterea conceptelor cheie din programare • Cunoaşterea conceptelor de bază din ingineria software • Înţelegerea uneltelor software de bază folosite în dezvoltarea de programe • Învățarea limbajului de programare Python şi a uneltelor pentru dezvolare, rulare, testare și depanare programe • Însuşirea şi îmbunătăţirea unui stil de programare pe baza celor mai bune recomandări pactice

8. Conţinuturi

8.1 Curs	Metode de predare	Observaţii
1. Introducere în procese de dezvoltare software • Ce este programarea: algoritm, program, elemente de bază Python, interpretor Python, roluri în ingineria software • Cum scriem programe: enunț problemă, cerințe, proces de dezvoltare dirijat de funcționalități	Expunerea interactivă Explicarea Prezentarea Exemple practice	
2. Programare procedurală • Tipuri structurate: liste, tuple, dictionare • Functii: cazuri de testare, definire, variabile, apel, transmiterea parametrilor • Programare dirijata de teste, refactorizari		
3. Programare modulară • Ce este un modul: definire modul Python, domeniul variabilelor, pachete, module standard, distribuire module • Eclipse+PyDev		
4. Tipuri definite de utilizator		

• Cum definim tipuri noi: incapsulare, ascunderea informatiei în Python • Introducere în programarea orientată obiect • Excepţii		
5. Programare orientată obiect • Tipuri abstracte de date • Implementarea claselor în Python • Obiecte şi clase		
6. Principii de proiecte de software • Arhitectura stratificată: UI layer, application layer, domain layer, infrastructure layer • Cum să organizăm codul sursă: responsabilităţi, single responsibility principle, separation of concerns, dependency, coupling, cohesion		
7. Testarea şi inspecţia programelor • Metode de testare: testare exhaustivă, testare black box, testare white box • Testare automată • Operaţii pe fişiere în Python		
8. Recursivitate • Noţiunea de recursivitate • Recursivitate directă şi indirectă • Exemple • Complexitate		
9. Algoritmi de căutare • Definirea problemei • Metode de căutare: secvenţială, binară • Complexitatea algoritmilor		
10. Algoritmi de sortare • Definirea problemei • Metode de sortare: Bubble Sort, Selection Sort, Insertion Sort, Quick Sort		
11. Metode de rezolvare a problemelor (I) • Prezentare generală a metodelor Backtracking, Divide et impera • Algoritmi şi complexitate • Exemple		
12. Metode de rezolvare a problemelor (II) • Prezentare generală a metodelor Greedy, Programare dinamică • Algoritmi şi complexitate • Exemple		
13. Recapitulare • Recapitularea celor mai importante concepte prezentate de curs		
14. Evaluare		

Bibliografie

1. M.L. Hetland, Beginning Python: From Novice to Professional, Apress, 2005.
2. M. Frentiu, H.F. Pop, Fundamentals of Programming, Cluj University Press, 2006.
3. K. Beck, Test Driven Development: By Example. Addison-Wesley Longman, 2002.http://en.wikipedia.org/wiki/Test-driven_development
4. M. Fowler, Refactoring. Improving the Design of Existing Code, Addison-Wesley, 1999.http://refactoring.com/catalog/index.html
5. The Python Programming Language - https://www.python.org/
6. The Python Standard Library - https://docs.python.org/3/library/index.html

7. The Python Tutorial - https://docs.python.org/3/tutorial/		
8.2 Seminar / laborator	Metode de predare	Observaţii
1. Programe Python simple	Expunerea interactivă Explicarea Conversaţia Demonstraţia didactică	
2. Programare procedurală		
3. Programare modulară		
4. Dezvoltare software dirijată de funcţionalităţi		
5. Tipuri abstracte de date		
6. Principii de proiectare		
7. Programare orientată obiect		
8. Proiectare. Arhitectura stratificată		
9. Inspecţie şi testare		
10. Recursivitate. Complexitatea algoritmilor		
11. Algoritmi de căutare şi testare		
12. Metode de rezolvare a problemelor: Backtracking		
13. Metode de rezolvare a problemelor: Greedy		
14. Test practic		

Bibliografie

1. M.L. Hetland, Beginning Python: From Novice to Professional, Apress, 2005.
2. M. Frentiu, H.F. Pop, Fundamentals of Programming, Cluj University Press, 2006.
3. K. Beck, Test Driven Development: By Example. Addison-Wesley Longman, 2002.http://en.wikipedia.org/wiki/Test-driven_development
4. M. Fowler, Refactoring. Improving the Design of Existing Code, Addison-Wesley, 1999.http://refactoring.com/catalog/index.html
5. The Python Programming Language - https://www.python.org/
6. The Python Standard Library - https://docs.python.org/3/library/index.html
7. The Python Tutorial - https://docs.python.org/3/tutorial/

9. Coroborarea conţinuturilor disciplinei cu aşteptările reprezentanţilor comunităţii epistemice, asociaţiilor profesionale şi angajatori reprezentativi din domeniul aferent programului

- Cursul respectă recomandările IEEE şi ACM pentru studii în informatică.
- Acest curs există în programul de studiu al tuturor universităților importante din Romania și străinătate.
- Conținutul acestui curs este considerat de către companiile de IT ca și important pentru abilităţi medii de programare.

10. Evaluare

Tip activitate	10.1 Criterii de evaluare	10.2 metode de evaluare	10.3 Pondere din nota finală
10.4 Curs	Corectitudinea şi completitudinea cunoştinţelor acumulate şi capacitatea de proiecta şi implementa corect programe Python	Examen scris	40%
10.5 Seminar/laborator	Abilitatea de a proiecta, implementa şi testa un program Python	Examen practic	30%
	Corectitudinea temelor de laborator şi documentaţiei livrate în timpul semestrului	Programe şi documentaţie	30%

10.6 Standard minim de performanţă
Fiecare student trebuie să obţină minim 5 pentru examenul scris, pentru examenul practic şi pentru nota finală. Pentru a obține nota minimă 5 studentul trebuie să demonstreze însușirea conceptelor de bază în algoritmi şi programare.

Data completării	**Semnătura titularului de curs**	**Semnătura titularului de seminar**
12.01.2023	**Prof. univ. dr. Camelia Chira**	**Prof. univ. dr. Camelia Chira**

Data avizării în departament	**Semnătura directorului de departament**
16.01.2023.....	**Prof. dr. Laura Dioșan**