FIŞA DISCIPLINEI

Algoritmi şi programare

Anul universitar 2025-2026

1. Date despre program

1.1. Instituţia de învăţământ superior	Universitatea Babeş-Bolyai Cluj-Napoca
1.2. Facultatea	Facultatea de Matematică şi Informatică
1.3. Departamentul	Departamentul de Informatică
1.4. Domeniul de studii	Matematică
1.5. Ciclul de studii	Licenţă
1.6. Programul de studii / Calificarea	Matematică Informatică (în limba engleză)
1.7. Forma de învăţământ	Cu frecvenţă

2. Date despre disciplină

2.1. Denumirea disciplinei	**Algoritmi şi programare**					Codul disciplinei	**MLE5115**
2.2. Titularul activităţilor de curs	Asist. dr. Anamaria Briciu						
2.3. Titularul activităţilor de seminar	Asist. dr. Anamaria Briciu						
2.4. Anul de studiu	1	2.5. Semestrul	1	2.6. Tipul de evaluare	C	2.7. Regimul disciplinei	Obligatorie DF

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/proiect	**2 S 2 LP**
3.4. Total ore din planul de învăţământ	84	din care: 3.5. curs	28	3.6 seminar/laborator/proiect	**56**
Distribuţia fondului de timp pentru studiul individual (SI) şi activităţi de autoinstruire (AI)					**ore**
Studiul după manual, suport de curs, bibliografie şi notiţe (AI)					14
Documentare suplimentară în bibliotecă, pe platformele electronice de specialitate şi pe teren					12
Pregătire seminare/ laboratoare/ proiecte, teme, referate, portofolii şi eseuri					14
Tutoriat (consiliere profesională)					8
Examinări					18
Alte activităţi					
3.7. Total ore studiu individual (SI) şi activităţi de autoinstruire (AI)				**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	Proiector
5.2. de desfăşurare a seminarului/ laboratorului	Calculatoare, limbajul şi mediul de programare Python

6. Competenţele specifice acumulate[1]

[1] Se poate opta pentru competenţe sau pentru rezultatele învăţării, respectiv pentru ambele. În cazul în care se alege o singură variantă, se va şterge tabelul aferent celeilalte opţiuni, iar opţiunea păstrată va fi numerotată cu 6.

Competenţe profesionale/esenţiale	C1.1 Definirea şi descrierea paradigmelor de programare şi a mecanismelor specifice limbajului Python, precum şi identificarea diferenţelor sintactice şi semantice. C1.2 Descrierea aplicaţiilor software existente la diferite niveluri de abstractizare (arhitectură, clase, metode) utilizând cunoştinţele de bază adecvate. C1.3 Elaborarea de cod sursă 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 unităţilor de programe şi a documentaţiei corespunzătoare.
Competenţe transversale	TC1 Aplicarea unor reguli de lucru eficiente şi riguroase, manifestarea unor atitudini responsabile faţă de domeniile ştiinţific şi didactic, valorificând potenţialul individual şi respectând principiile profesionale şi etice. TC2 Utilizarea unor metode şi tehnici eficiente pentru învăţare, informare, cercetare şi dezvoltarea abilităţilor de valorificare a cunoştinţelor, pentru adaptarea la nevoile unei societăţi dinamice şi pentru comunicarea într-o limbă străină.

7. Obiectivele disciplinei (reieşind din grila competenţelor acumulate)

7.1 Obiectivul general al disciplinei	● Cunoaşterea conceptelor de bază din domeniul ingineriei software (proiectare, implementare şi mentenanţă) şi învăţarea limbajului de programare Python
7.2 Obiectivele specifice	● Cunoaşterea unor concepte cheie de programare ● Cunoaşterea unor concepte de bază de inginerie software ● Înţelegerea uneltelor software de bază folosite în dezvoltarea de programe ● Învăţarea limbajului de programare Python şi familiarizarea cu uneltele folosite în dezvoltarea, execuţia, testarea şi depanarea programelor ● Asimilarea şi îmbunătăţirea unui stil de programare pe baza unor recomandări practice

8. Conţinuturi

8.1 Curs	Metode de predare	Observaţii
1. Introducere în procesele de dezvoltare software ● Ce este programarea: algoritm, program, elemente de bază ale limbajului Python, interpretorul Python, roluri de bază în ingineria software ● Cum scriem programe: enunţul problemei, cerinţe, procese de dezvoltare ghidate de funcţionalităţi	● Expunerea interactivă ● Explicaţia ● Conversaţia ● Exemple ● Demonstraţia didactică	
2. **Programare procedurală** ● Tipuri de date: liste, tuple, dicţionar ● Funcţii: cazuri de test, definiţie, scopul variabilelor, transmiterea parametrilor ● Dezvoltare ghidată de testare, refactorizare	● Expunerea interactivă ● Explicaţia ● Conversaţia ● Exemple ● Demonstraţia didactică	
3. Programare modulară ● Ce este un modul: definirea unui modul Python, scopul variabilelo într-un modul, pachete, librării standard ● PyCharm	● Expunerea interactivă ● Explicaţia ● Conversaţia ● Exemple ● Demonstraţia didactică	
4. **Tipuri definite de utilizator**	● Expunerea interactivă	

● Cum definim tipuri noi de date: încapsulare, ascunderea datelor în Python ● Introducere în programarea orientată obiect ● Excepţii	● Explicaţia ● Conversaţia ● Exemple ● Demonstraţia didactică	
5. Principii de proiectare software ● Arhitectura stratificată: intefaţa utilizator, aplicaţie, domeniu, infrastructură ● Cum organizăm codul sursă: responsabilităţi, principiul responsabilităţii unice, separarea grijilor, dependenţe, cuplaj, coeziune	● Expunerea interactivă ● Explicaţia ● Conversaţia ● Exemple ● Demonstraţia didactică	
6. **Programare orientată obiect** ● Tipuri abstracte de date ● Implementarea claselor în Python ● Obiecte şi clase: clase, obiecte, atribute, metode, scop şi namespace în Python	● Expunerea interactivă ● Explicaţia ● Conversaţia ● Exemple ● Demonstraţia didactică	
7. **Aspecte legate de proiectare** ● Strategii de tip top-down şi bottom-up ● Elemente de interfaţă utilizator (UI)	● Expunerea interactivă ● Explicaţia ● Conversaţia ● Exemple ● Demonstraţia didactică	
8. T**estarea şi inspecţia programelor** ● Metode de testare: testare exhaustivă, testare de tip black box, testare de tip white box ● Testare automată, TDD ● Operaţii cu fişiere în Python	● Expunerea interactivă ● Explicaţia ● Conversaţia ● Exemple ● Demonstraţia didactică	
9. Recursivitate ● Noţiunea de recursivitate ● Recursivitate directă şi indirectă ● Exemple **Complexitatea algoritmilor**	● Expunerea interactivă ● Explicaţia ● Conversaţia ● Exemple ● Demonstraţia didactică	
10. **Algoritmi de căutare** ● Definirea problemei ● Metode de căutare: secvenţială, binară ● Complexitatea algoritmilor **Algoritmi de sortare** ● Definirea problemei ● Metode de sortare: Bubble Sort, Selection Sort, Insertion Sort, Quick Sort ● Complexitatea algoritmilor	● Expunerea interactivă ● Explicaţia ● Conversaţia ● Exemple ● Demonstraţia didactică	
11. Backtracking ● Prezentarea generală a metodei ● Complexitatea algoritmilor ● Exemple	● Expunerea interactivă ● Explicaţia ● Conversaţia ● Exemple ● Demonstraţia didactică	
12**. Divide et impera, Greedy** ● Prezentarea generală a metodei ● Complexitatea algoritmilor ● Exemple	● Expunerea interactivă ● Explicaţia ● Conversaţia ● Exemple ● Demonstraţia didactică	
13. Programare dinamică ● Descrierea metodei ● Exemple	● Expunerea interactivă ● Explicaţia ● Conversaţia ● Exemple ● Demonstraţia didactică	
14. **Recapitulare**		
Bibliografie 1. M. Frentiu, H.F. Pop, Fundamentals of Programming, Cluj University Press, 2006.		

2. K. Beck, Test Driven Development: By Example. Addison-Wesley Longman, 2002.
http://en.wikipedia.org/wiki/Test-driven_development
3. M. Fowler, Refactoring. Improving the Design of Existing Code, Addison-Wesley, 1999.
http://refactoring.com/catalog/index.html
4. The Python Programming Language - https://www.python.org/
5. The Python Standard Library - https://docs.python.org/3/library/index.html
1. The Python Tutorial - https://docs.python.org/3/tutorial/

8.2 Seminar/Laborator	Metode de predare	Observaţii
1. Programe Python simple	● Expunerea interactivă ● Explicaţia ● Conversaţia ● Demonstraţia didactică	
2. Programare procedurală		
3. Programare modulară		
4. Dezvoltare software bazată pe funcţionalităţi		
5. Tipuri abstracte de date		
6. Principii de proiectare		
7. Programare orientată obiect		
2. Proiectarea programelor. Arhitectura stratificată		
3. Inspecţie şi testare		
10. Recursivitate. Complexitatea algoritmilor		
11. Algoritmi de căutare şi sortare		
12. Metode de rezolvare a problemelor: Backtracking		
13. Metode de rezolvare a problemelor: Greedy		
14. Test practic		

Bibliografie

1. M. Frentiu, H.F. Pop, Fundamentals of Programming, Cluj University Press, 2006.
2. K. Beck, Test Driven Development: By Example. Addison-Wesley Longman, 2002.
http://en.wikipedia.org/wiki/Test-driven_development
3. M. Fowler, Refactoring. Improving the Design of Existing Code, Addison-Wesley, 1999.
http://refactoring.com/catalog/index.html
4. The Python Programming Language - https://www.python.org/
5. The Python Standard Library - https://docs.python.org/3/library/index.html
6. 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 curiculare IEEE şi ACM pentru studii în informatică
- Cursul există în programele de studiu ale marilor universităţi din România şi străinătate
- Conţinutul cursului este considerat de companiile software ca fiind imporant 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	Corectitutdinea şi completitudinea cunoştinţelor acumulate şi capacitatea de a proiecta şi	Examen scris	40%	

	implementa corect programe Python			
10.5 Laborator	Abilitatea de a proiecta, implementa şi testa un programe Python	Examen practic	30%	
	Corectitudinea temelor de laborator şi documentaţiă livrată în timpul semestrului	Program şi documentaţie	30%	
10.6. Seminar	Activitate de seminar	Participare activă la discuţiile din timpul seminariilor (punerea de întrebări, răspuns la întrebările cadrului didactic, rezolvarea de probleme, etc)	Maximum 0.5 puncte bonus, adăugate notei finale	
10.6 Standard minim de performanţă				
• Fiecare student trebuie să demonstreze un nivel acceptabil de cunoştinţe şi înţelegere a domeniului, capacitatea de a prezenta cunoştinţele într-o manieră coerentă şi abilitatea de a stabili conexiuni şi de a folosi aceste cunoştinţe pentru a rezolva diverse probleme în Python. • Fiecare student este obligat să participe la minimum 10 seminarii şi 12 laboratoare. • Trebuie obţinută o notă minimă de 5 la activitatea de laborator, testul practic şi examenul scris.				

11. Etichete ODD (Obiective de Dezvoltare Durabilă / Sustainable Development Goals)[2]

Nu se aplică.

Data completării:
22.09.2025

Semnătura titularului de curs

Asist. dr. Anamaria Briciu

Semnătura titularului de seminar

Asist. dr. Anamaria Briciu

Data avizării în departament:
...

Semnătura directorului de departament

Conf.dr. Adrian STERCA

[2] Păstraţi doar etichetele care, în conformitate cu *Procedura de aplicare a etichetelor ODD în procesul academic*, se potrivesc disciplinei şi ştergeţi-le pe celelalte, inclusiv eticheta generală pentru *Dezvoltare durabilă* - dacă nu se aplică. Dacă nicio etichetă nu descrie disciplina, ştergeţi-le pe toate şi scrieţi "*Nu se aplică.*".