

FIȘA DISCIPLINEI

Algoritmi și Programare

Anul universitar 2025-2026

1. Date despre program

1.1. Instituția de învățământ superior	Universitatea Babeș-Bolyai
1.2. Facultatea	Facultatea de Matematică și Informatică
1.3. Departamentul	Departamentul de Informatică
1.4. Domeniul de studii	Matematică
1.5. Ciclu de studii	Licență
1.6. Programul de studii / Calificarea	Matematică – limba de studiu română
1.7. Forma de învățământ	Cu frecvență

2. Date despre disciplină

2.1. Denumirea disciplinei		Algoritmi și Programare				Codul disciplinei		MLR5115			
2.2. Titularul activităților de curs			Lect. Dr. Găceanu Radu Dan								
2.3. Titularul activităților de seminar			Lect. Dr. Găceanu Radu Dan								
2.4. Anul de studiu		1	2.5. Semestrul		1	2.6. Tipul de evaluare		C	2.7. Regimul disciplinei		Obligativu

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

3.1. Număr de ore pe săptămână	6	din care: 3.2. curs	2	3.3. seminar/ laborator/proiect	2 sem 2 lab
3.4. Total ore din planul de învățământ	84	din care: 3.5. curs	28	3.6 seminar/laborator/proiect	56
Distribuția fondului de timp pentru studiul individual (SI) și activități de autoinstruire (AI)					ore
Studiul după manual, suport de curs, bibliografie și notițe (AI)					14
Documentare suplimentară în bibliotecă, pe platformele electronice de specialitate și pe teren					12
Pregătire seminare/ laboratoare/ proiecte, teme, referate, portofolii și eseuri					14
Tutoriat (consiliere profesională)					8
Examinări					18
Alte activități					
3.7. Total ore studiu individual (SI) și activități de autoinstruire (AI)				66	
3.8. Total ore pe semestru				150	
3.9. Numărul de credite				6	

4. Precondiții (acolo unde este cazul)

4.1. de curriculum	
4.2. de competențe	

5. Condiții (acolo unde este cazul)

5.1. de desfășurare a cursului	Sală cu proiector
5.2. de desfășurare a seminarului/ laboratorului	Laboratoare echipate cu medii de dezvoltare Python; sala cu proiector (seminar)

6.1. Competențele specifice acumulate¹

Competențe profesionale/esențiale	elaborarea și analiza unor algoritmi pentru rezolvarea problemelor
Competențe transversale	- aplicarea regulilor de muncă riguroasă și eficientă, manifestarea unor atitudini responsabile față de domeniul didactico-științific, pentru valorificarea optimă și creativă a propriului potențial în situații specifice, cu respectarea principiilor și a normelor de etică profesională - desfășurarea eficientă și eficace a activităților organizate în echipă - utilizarea eficientă a surselor informaționale și a resurselor de comunicare și formare profesională asistată, atât în limba română, cât și într-o limbă de circulație internațională

6.2. Rezultatele învățării

Cunoștințe	Absolventul/a este familiarizat/ă cu metodele de prelucrare a datelor și cu instrumentele de vizualizare a rezultatelor obținute.
Aptitudini	Absolventul/a este capabil/ă să analizeze literatura de specialitate și să utilizeze instrumente de sprijinire a cercetării.
Responsabilități și autonomie	Absolventul/a are capacitatea de a interpreta articole sau cărți din literatura de specialitate și de a încorpora idei și rezultate din literatura de specialitate în prezentările lor scrise și orale.

7. Obiectivele disciplinei (reieșind din grila competențelor acumulate)

7.1 Obiectivul general al disciplinei	Să cunoască conștepele de baza ale ingineriei software (proiectare, implementare și întreținere) și să învețe limbajul de programare Python.
---------------------------------------	--

¹ Se poate opta pentru competențe sau pentru rezultatele învățării, respectiv pentru ambele. În cazul în care se alege o singură variantă, se va șterge tabelul aferent celeilalte opțiuni, iar opțiunea păstrată va fi numerotată cu 6.

7.2 Obiectivele specifice	• Să cunoască conceptele de baza ale programarii • Să cunoască conceptele de baza ale ingineriei software • Să folosească instrumente de baza pentru construirea programelor • Să învețe limbajul Python si instrumente de dezvoltare pentru programarea, executia si depanarea programelor Python. • Să-și însușeasca un stil de programare conform celor mai bune recomandări practice.
----------------------------------	---

8. Conținuturi

8.1 Curs	Metode de predare	Observații
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; Exemple: calculator	Expunere interactivă; Explicație; Conversație; Exemple; Demonstrație didactică	
2. Programare procedurală Tipuri structurate: liste, tuple, dicționare; Funcții: cazuri de testare, definire, variabile, apel, transmiterea parametrilor, funcții anonime; Cum scriem funcții: programare dirijată de teste, refactorizări	Expunere interactivă; Explicație; Conversație; Exemple; Demonstrație didactică	
3. Programare modulară Ce este un modul: modul Python, domeniul variabilelor, pachete, module standard, distribuie module; Cum organizăm codul sursă: responsabilități, single responsibility principle, separation of concerns, dependency, coupling, cohesion; Eclipse+PyDev	Expunere interactivă; Explicație; Conversație; Exemple; Demonstrație didactică	
4. Tipuri definite de utilizator Cum definim tipuri noi; Incapsulare, ascunderea informației, tipuri abstracte de date; Introducere programarea orientată obiect –clasa implementarea unui TAD	Expunere interactivă; Explicație; Conversație; Exemple; Demonstrație didactică	
5-6. Programarea orientată obiect Mecanismele programării orientate obiect; Dezideratele proiectării obiectuale: Cuplare redusă; Coeziune ridicată; Abstractizare corespunzătoare; Complexitate gestionabilă; Principii și șabloane de proiectare: SOLID; DDD: entity, value object, repository, service; IoC, app coordinator, dependency injection; GOF: singleton, strategy; Operații CRUD; Arhitectura stratificată.	Expunere interactivă; Explicație; Conversație; Exemple; Demonstrație didactică	
7. Proiectarea programelor Diagrame UML; Șabloane Gasp; Excepții	Expunere interactivă; Explicație; Conversație; Exemple; Demonstrație didactică	
8. Persistența și procesarea datelor DTO, Filter, lambda	Expunere interactivă; Explicație; Conversație; Exemple; Demonstrație didactică	
9. Testarea și inspectarea programelor Black box testing, white box testing; Unit testing, integration testing; Program inspection: coding style, refactoring	Expunere interactivă; Explicație; Conversație; Exemple; Demonstrație didactică	
10-11. Complexitatea Algoritmilor, recursivitate, căutare, sortare	Expunere interactivă; Explicație; Conversație; Exemple;	

Recursivitate directa si indirecta; Căutare secvențială; Căutare binară 7 Complexitatea algoritmilor; BubbleSort; SelectionSort; InsertionSort; QuickSort; MergeSort; Cmplexitatea algoritmilor	Demonstrație didactică	
12. Sisteme de versionare a codului Introducere in Git	Expunere interactivă; Explicație; Conversație; Exemple; Demonstrație didactică	
13. Metode de rezolvare a problemelor Metoda divizarii; Backtracking; Greedy; Programare dinamică	Expunere interactivă; Explicație; Conversație; Exemple; Demonstrație didactică	
14. Colocviu - Evaluare		
Bibliografie		
1. Kent Beck. Test Driven Development: By Example. Addison-Wesley Longman, 2002. See also Testdriven development. http://en.wikipedia.org/wiki/Test-driven_development 2. Martin Fowler. Refactoring. Improving the Design of Existing Code. Addison-Wesley, 1999. See also http://refactoring.com/catalog/index.html 3. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. Introduction to Algorithms 3rd ed., 2009. 4. Craig Larman. Applying UML and Patterns. An Introduction to Object Oriented Analysis and Design, 2004. 5. The Python language reference. http://docs.python.org/py3k/reference/index.html 6. The Python standard library. http://docs.python.org/py3k/library/index.html		
8.2 Seminar / laborator	Metode de predare	Observații
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; Exemple: calculator	Expunere interactivă; Explicație; Conversație; Exemple; Demonstrație didactică	
2. Programare procedurală Tipuri structurate: liste, tuple, dicționare; Funcții: cazuri de testare, definire, variabile, apel, transmiterea parametrilor, funcții anonime; Cum scriem funcții: programare dirijată de teste, refactorizări	Expunere interactivă; Explicație; Conversație; Exemple; Demonstrație didactică	
3. Programare modulară Ce este un modul: modul Python, domeniul variabilelor, pachete, module standard, distribuie module; Cum organizăm codul sursă: responsabilități, single responsibility principle, separation of concerns, dependency, coupling, cohesion; Eclipse+PyDev	Expunere interactivă; Explicație; Conversație; Exemple; Demonstrație didactică	
4. Tipuri definite de utilizator Cum definim tipuri noi; Incapsulare, ascunderea informației, tipuri abstracte de date; Introducere programarea orientată obiect –clasa implementarea unui TAD	Expunere interactivă; Explicație; Conversație; Exemple; Demonstrație didactică	
5-6. Programarea orientată obiect Mecanismele programării orientate obiect; Dezideratele proiectării obiectuale: Cuplare redusă; Coeziune ridicată; Abstractizare corespunzătoare; Complexitate gestionabilă; Principii și șabloane de proiectare: SOLID; DDD: entity, value object, repository, service; IoC, app coordinator, dependency injection; GOF: singleton, strategy; Operații CRUD; Arhitectura stratificată.	Expunere interactivă; Explicație; Conversație; Exemple; Demonstrație didactică	
7. Proiectarea programelor Diagrame UML; Șabloane Grasp; Excepții	Expunere interactivă; Explicație; Conversație; Exemple;	

	Demonstrație didactică	
8. Persistența și procesarea datelor DTO, Filter, lambda	Expunere interactivă; Explicație; Conversație; Exemple; Demonstrație didactică	
9. Testarea și inspectarea programelor Black box testing, white box testing; Unit testing, integration testing; Program inspection: coding style, refactoring	Expunere interactivă; Explicație; Conversație; Exemple; Demonstrație didactică	
10-11. Complexitatea Algoritmilor, recursivitate, căutare, sortare Recursivitate directă și indirectă; Căutare secvențială; Căutare binară și Complexitatea algoritmilor; BubbleSort; SelectionSort; InsertionSort; QuickSort; MergeSort; Complexitatea algoritmilor	Expunere interactivă; Explicație; Conversație; Exemple; Demonstrație didactică	
12. Sisteme de versionare a codului Introducere în Git	Expunere interactivă; Explicație; Conversație; Exemple; Demonstrație didactică	
13-14. Pregătire examene.	Expunere interactivă; Explicație; Conversație; Exemple; Demonstrație didactică	
Bibliografie		
1. Kent Beck. Test Driven Development: By Example. Addison-Wesley Longman, 2002. See also Testdriven development. http://en.wikipedia.org/wiki/Test-driven_development 2. Martin Fowler. Refactoring. Improving the Design of Existing Code. Addison-Wesley, 1999. See also http://refactoring.com/catalog/index.html 3. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. Introduction to Algorithms 3rd ed., 2009. 4. Craig Larman. Applying UML and Patterns. An Introduction to Object Oriented Analysis and Design, 2004. 5. The Python language reference. http://docs.python.org/py3k/reference/index.html 6. The Python standard library. http://docs.python.org/py3k/library/index.html		

9. Coroborarea conținuturilor disciplinei cu așteptările reprezentanților comunității epistemice, asociațiilor profesionale și angajatori reprezentativi din domeniul aferent programului

- Cursul respectă recomandările IEEE și ACM legate de Curricula pentru specializarea Informatică.
- Cursul face parte din programul de studiu de la majoritatea universităților importante din România și din străinătate.
- Conținutul cursului este considerat de companiile soft ca fiind important pentru un nivel mediu de cunoștințe în 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 și completitudinea cunoștințelor acumulate. Capacitatea de a proiecta și implementa programe scrise în limbajul Python	Examen scris	20%

10.5 Seminar/laborator	Abilitatea de a scrie și depana un program Python	Examen practic	70%
	Programele scrise în timpul semestrului	Documentație; teme	10%
10.6 Standard minim de performanță			
Nota finala minim 5.			

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

Nu se aplică.

Data completării:
15.04.2025

Semnătura titularului de curs
Lect. Dr. Găceanu Radu Dan

Semnătura titularului de seminar
Lect. Dr. Găceanu Radu Dan

Data avizării în departament:
...

Semnătura directorului de departament
Conf.dr. Adrian STERCA

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