

FIȘA DISCIPLINEI

1. Date despre program

1.1 Instituția de învățământ superior	Universitatea Tehnică din Cluj-Napoca
1.2 Facultatea	Facultatea de Științe (Centrul Universitar Nord din Baia Mare)
1.3 Departamentul	Departamentul de Matematică și Informatică
1.4 Domeniul de studii	Informatică
1.5 Ciclul de studii	Licență
1.6 Programul de studii / Calificarea	Informatică (Baia Mare, limba română) / Licențiat în informatică
1.7 Forma de învățământ	IF – învățământ cu frecvență

2. Date despre disciplină

2.1 Denumirea disciplinei	Tehnici avansate de programare	Codul disciplinei	22.10
2.2 Titularul de curs	Conf. univ. dr. Cosmin Nicolae SABO – cosmin.sabo@mi.utcluj.ro		
2.3 Titularul activităților de seminar / laborator / proiect / practică	Asistent universitar, post vacant		
2.4 Anul de studiu	II	2.5 Semestrul	2
		2.6 Tipul de evaluare	E
2.7 Regimul disciplinei	Categoriza formativă		
	Opționalitate		

3. Timpul total estimat

3.1 Număr de ore pe săptămână	4	din care:	3.2 Curs	2	3.3 Seminar	-	3.3 Laborator	2	3.3 Proiect	-	3.3 Practică	-
3.4 Număr de ore pe semestru	56	din care:	3.5 Curs	28	3.6 Seminar	-	3.6 Laborator	28	3.6 Proiect	-	3.3 Practică	-
3.7 Distribuția fondului de timp (ore pe semestru) pentru studiu individual și evaluare:												
(a) Evaluare												6
(b) Studiul după manual, suport de curs, bibliografie și notițe												20
(c) Documentare suplimentară în bibliotecă, pe platforme electronice de specialitate și pe teren												8
(d) Pregătire seminarii / laboratoare, teme, referate, portofolii și eseuri												30
(e) Tutoriat												3
(f) Alte activități												2
3.8 Total ore studiu individual și evaluare (suma (3.7(a))...3.7(f))								69				
3.9 Total ore pe semestru (3.4+3.8)								125				
3.10 Numărul de credite								5				

4. Precondiții (acolo unde este cazul)

4.1 de curriculum	Disciplinele de programare și de algoritmi din anul I.
4.2 de competențe	Programare în limbajul de lucru din anul I (funcții, liste, dicționare, fișiere, recursivitate), noțiuni de bază de algoritmi și de complexitate, utilizarea unui mediu de dezvoltare. Nu se presupune experiență anterioară de programare orientată pe obiecte: noțiunile necesare se introduc în săptămâna 2, cu corespondențe pentru studenții care le cunosc din alte limbaje.

5. Condiții (acolo unde este cazul)

5.1. de desfășurare a cursului	Sală de curs cu videoproiector și acces la internet; fiecare curs pornește de la o problemă concretă și ajunge la mecanismul de limbaj sau la structura de date care o rezolvă, cu cod scris în fața studenților; studenții pot urmări și rula exemplele pe laptopul personal. Suportul de curs, exemplele rezolvate, testele de autoevaluare și anunțurile se publică pe platforma Campus Virtual. Cursul este comun programelor Matematică informatică și Informatică (Tehnici avansate de programare) și Informatică economică (Structuri de date).
5.2. de desfășurare a seminarului / laboratorului / proiectului	Laborator cu calculatoare pe care sunt instalate interpretorul limbajului de lucru (Python, versiune curentă), un sistem de control al versiunilor și un editor de cod cu depanator; se recomandă lucrul pe laptopul personal, cu același mediu configurat în prima ședință. Activitatea este individuală, pe sarcini cu nivel de bază (obligatoriu) și extinderi (opționale, notate suplimentar); livrabilele se predau prin Campus Virtual, sunt verificate automat prin teste și primesc feedback la laborator. Proiectul final poate fi realizat, cu acordul titularului, și într-un alt limbaj studiat în planul de învățământ, cu aceleași cerințe de teste și documentație. Asistenții bazați pe inteligență artificială sunt permisi la laborator și la proiect, cu declararea lor în fiecare livrabil și cu obligația de a explica propriul cod; sunt interziși la teste, la proba practică și la examen. Prezența la laborator este obligatorie, conform regulamentului UTCN privind activitatea profesională a studenților.

6. Competențele specifice acumulate

Competențe profesionale	C1. Programarea în limbaje de nivel înalt: mecanismele avansate ale unui limbaj de nivel înalt: modelul de date și metodele speciale, iteratori și generatoare, închideri și decoratori, adnotări de tip, clase de date, protocoale, excepții, concurență introductivă. C2. Dezvoltarea și întreținerea aplicațiilor informatice: principii și șabloane de proiectare, testare automată, refactorizare, control al versiunilor, organizarea codului în module. C4. Utilizarea bazelor teoretice ale informaticii și a modelelor formale: tipuri abstracte de date, structuri de date fundamentale (secvențe, tabele de dispersie, structuri înlănțuite, heap-uri, arbori, grafuri) și analiza complexității lor.
Competențe transverale	CT1. Aplicarea regulilor de muncă organizată și eficientă, a unor atitudini responsabile față de domeniul didactic-științific, pentru valorificarea creativă a propriului potențial, cu respectarea principiilor și a normelor de etică profesională. CT3. Utilizarea unor metode și tehnici eficiente de învățare, informare, cercetare și dezvoltare a capacităților de valorificare a cunoștințelor, de adaptare la cerințele unei societăți dinamice și de comunicare în limba română și într-o limbă de circulație internațională.

7. Rezultatele așteptate ale învățării

Cunoștințe	RI1. Explică modelul de date al limbajului de lucru (obiecte, referințe, mutabilitate, egalitate și dispersie, metode speciale) și mecanismele de programare avansată: iteratori și generatoare, închideri și decoratori, adnotări de tip, clase de date, clase abstracte și protocoale, excepții și gestionarea resurselor. RI2. Descrie structurile de date fundamentale (secvențe, tabele de dispersie, structuri înlănțuite, stive, cozi, cozi de prioritate, arbori binari și de căutare, grafuri), operațiile lor, complexitatea acestora și criteriile de alegere între ele. RI3. Prezintă principiile de proiectare a codului și un set de șabloane de proiectare uzuale, rolul testării automate, al refactorizării și al măsurării performanței, precum și noțiunile introductive de concurență.
Abilități	RI4. Proiectează și implementează tipuri abstracte de date generice, cu invarianți verificați prin teste și cu adnotări de tip. RI5. Prelucreează eficient fluxuri și fișiere mari de date cu iteratori, generatoare și structuri de date potrivite, măsurând timpul și memoria consumate. RI6. Aplică închideri, decoratori, memoizare și șabloane de proiectare pentru a scrie cod reutilizabil și pentru a refactoriza cod existent. RI7. Scrie teste unitare, parametrizate și bazate pe proprietăți, tratează erorile prin excepții proprii, gestionează resursele și folosește controlul versiunilor.
Responsabilitate și autonomie	RI8. Realizează autonom un proiect software de dimensiune mică, planificat pe etape, cu respectarea termenelor, și îl prezintă și îl explică. RI9. Revizuieste codul colegilor și propriul cod pe criterii de corectitudine, lizibilitate și performanță; utilizează responsabil sursele de documentare și instrumentele de asistență, inclusiv cele bazate pe inteligență artificială, le declară și este capabil să explice fiecare linie din propriul cod.

8. Obiectivele disciplinei (reieșind din grila competențelor specifice acumulate)

8.1 Obiectivul general al disciplinei	Trecerea de la scrierea de scripturi la dezvoltarea de software corect, lizibil, eficient și testat: stăpânirea mecanismelor avansate ale limbajului de lucru și a structurilor de date fundamentale, aplicate pe probleme concrete, într-un mod accesibil studenților cu pregătire anterioară diferită.
8.2 Obiectivele specifice	O1. Înțelegerea modelului de date al limbajului și a programării orientate pe obiecte la nivelul necesar proiectării de tipuri proprii. O2. Stăpânirea iteratorilor, generatoarelor, închiderilor și decoratorilor pentru prelucrarea leneșă și reutilizabilă a datelor. O3. Alegerea, implementarea și măsurarea structurilor de date potrivite pentru o problemă dată. O4. Utilizarea adnotărilor de tip, a claselor de date, a claselor abstracte și a protocoalelor în proiectarea de tipuri abstracte de date generice. O5. Aplicarea principiilor și a șabloanelor de proiectare și refactorizarea codului existent. O6. Formarea deprinderilor de testare automată, tratare a erorilor, control al versiunilor și revizuire de cod, cu o introducere în concurență.

9. Conținuturi

9.1 Curs	Nr. ore	Metode de predare	Observații
1. Introducere: de la script la software; instrumentele de lucru (mediu, versionare, teste, stil); măsurarea timpului și a memoriei.	2	Prelegere interactivă construită pe o problemă concretă pe săptămână (întâi problema, apoi mecanismul care o rezolvă), cu cod scris în fața studenților, un singur concept nou pe săptămână și generalizarea după exemple; întrebări și sondaje; suport de curs, exemple și teste de autoevaluare pe Campus Virtual.	2 ore/săpt. Curs comun MI, INF, IE. Tema continuă la laborator. Teste scurte în săpt. 6 și 11. Se cere nucleul din suport; restul e opțional.
2. Clase și obiecte: attribute, metode, compoziție, moștenire; corespondența cu alte limbaje.	2		
3. Modelul de date: referințe și mutabilitate, egalitate și dispersie, metode speciale; secvențe proprii.	2		
4. Dicționare și mulțimi în profunzime: tabele de dispersie, obiecte hashabile, colecții specializate; alegerea structurii potrivite.	2		
5. Iteratori și generatoare: protocolul de iterare, evaluare leneșă, fluxuri de date mari; liste înlănțuite ca iterabile.	2		
6. Funcții ca obiecte: închideri, decoratori, funcții de ordin superior; memoizare și programare dinamică.	2		
7. Tipuri și contracte: adnotări de tip, clase de date, clase abstracte și protoale; tipuri abstracte de date generice (stivă, coadă, coadă de priorități, heap).	2		
8. Testare și calitate: teste unitare, parametrizate și bazate pe proprietăți, dubluri de test, acoperire.	2		
9. Excepții și robustețe: ierarhii de excepții proprii, gestionarea resurselor, jurnalizare.	2		
10. Recursivitate și arbori: arbori binari și de căutare, parcurgeri, arbori de expresii; recursiv și iterativ.	2		
11. Grafuri: reprezentări, parcurgeri, drumuri minime; modelarea problemelor prin grafuri.	2		
12. Principii și șabloane de proiectare I: principiile SOLID; strategie, observator, iterator, metodă șablon.	2		
13. Șabloane de proiectare II și refactorizare: fabrică, constructor, decorator, comandă, adaptor; mirosuri de cod.	2		
14. Concurență introductivă (fire de execuție, procese, programare asincronă); sinteză: arhitectura unei aplicații mici; pregătirea evaluării.	2		
Bibliografie			
1. Suportul de curs al disciplinei (nucleu, exemple rezolvate, aprofundări opționale), publicat pe Campus Virtual.			
2. Ramalho, L., Fluent Python: Clear, Concise, and Effective Programming, ed. a 2-a, O'Reilly, 2022; codul sursă gratuit, https://github.com/fluentpython/example-code-2e .			
3. Slatkin, B., Effective Python: 125 Specific Ways to Write Better Python, ed. a 3-a, Addison-Wesley, 2024.			
4. Beazley, D., Python Distilled, Addison-Wesley, 2021.			
5. Goodrich, M. T., Tamassia, R., Goldwasser, M. H., Data Structures and Algorithms in Python, Wiley, 2013.			
6. Miller, B. N., Ranum, D. L., Problem Solving with Algorithms and Data Structures using Python, ed. a 3-a, Runestone Academy; ediție interactivă gratuită, https://runestone.academy/ns/books/published/pythonds3/ .			
7. Wengrow, J., A Common-Sense Guide to Data Structures and Algorithms in Python, vol. 1, Pragmatic			

9.1 Curs	Nr. ore	Metode de predare	Observații
Bookshelf, 2023. 8. Ayeva, K., Kasampalis, S., Mastering Python Design Patterns, ed. a 3-a, Packt, 2024; sinteză gratuită a șabloanelor, cu exemple: https://refactoring.guru/design-patterns/python. 9. Percival, H., Gregory, B., Architecture Patterns with Python, O'Reilly, 2020; ediție online gratuită, https://www.cosmicpython.com. 10. Python Software Foundation, The Python Language Reference și The Python Standard Library, https://docs.python.org/3/.			

9.2 Seminar / laborator / proiect / practică	Nr. ore	Metode de predare	Observații
1. Mediul de lucru: versionare, mediu izolat, rularea testelor date, stil de cod; profilarea unui script dat.	2	Lucru individual pe calculator, pornind de la problema săptămânii începută la curs; sarcini pe două niveluri (de bază și extindere), verificate automat prin teste, cu feedback la laborator; fișe de laborator și soluții de referință pentru titularul de laborator; revizuirea codului între colegi; asistenți AI permiși cu declarare; predare prin Campus Virtual.	2 ore/săpt., pe subgrupe. Lucrări notate L1-L6 (săpt. 3, 5, 7, 9, 11, 12). Proiect final în săpt. 9-14. Sarcini pe două niveluri.
2. Modelarea unui domeniu cu clase și compoziție; corespondența cu limbajele cunoscute.	2		
3. O secvență proprie (vector, polinom sau matrice) cu metode speciale; egalitate și dispersie (L1).	2		
4. Analiza unui set de date cu dicționare și colecții specializate; alegerea structurii prin măsurare.	2		
5. Procesarea leneșă a unui fișier mare cu generatoare; listă înlănțuită iterabilă (L2).	2		
6. Decoratori de instrumentare (cronometrare, cache, reîncercare); memoizare pe o problemă recursivă; test scurt T1 (săpt. 1-5).	2		
7. Tipuri abstracte de date generice cu adnotări de tip: stivă, coadă, coadă de priorități; verificare statică (L3).	2		
8. Teste pentru structurile proprii: cazuri limită, teste parametrizate și bazate pe proprietăți; acoperire.	2		
9. Robustețe: excepții proprii, gestionarea resurselor, jurnalizare (L4); anunțarea temelor de proiect.	2		
10. Arbore binar de căutare și arbore de expresii, cu teste.	2		
11. Modelarea unei probleme reale prin grafuri (rute, dependențe) și rezolvarea ei (L5); test scurt T2 (săpt. 6-10).	2		
12. Refactorizarea unei aplicații date cu principii și șabloane de proiectare (L6).	2		
13. Proiect: revizuire între colegi, integrare, teste; experiment de concurență pe proiect.	2		
14. Prezentarea și evaluarea proiectului.	2		

Bibliografie 1. Îndrumarul de laborator al disciplinei (fișe de laborator, șabloane de cod, teste automate, temele proiectului), publicat pe Campus Virtual. 2. Okken, B., Python Testing with pytest, ed. a 2-a, Pragmatic Bookshelf, 2022. 3. Python Software Foundation, The Python Standard Library (modulele collections, itertools, functools, heapq, dataclasses, typing, unittest, logging, concurrent.futures, asyncio), https://docs.python.org/3/library/. 4. van Rossum, G., Warsaw, B., Coghlan, A., PEP 8: Style Guide for Python Code, https://peps.python.org/pep-0008/. 5. Chacon, S., Straub, B., Pro Git, ed. a 2-a, Apress, 2014; ediție online gratuită,			
---	--	--	--

9.2 Seminar / laborator / proiect / practică	Nr. ore	Metode de predare	Observații
https://git-scm.com/book/ro.			
6. Refactoring Guru, Design Patterns și Refactoring, cu exemple în Python, https://refactoring.guru.			

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

Disciplina corespunde cursurilor de programare de nivel avansat din anul II ale programelor de informatică din țară (programare avansată, metode avansate de programare, structuri de date) și din străinătate (advanced programming), precum și recomandărilor ACM/IEEE-CS Computer Science Curricula 2023 pentru ariile „Foundations of Programming Languages” (programare orientată pe obiecte, funcțională, sisteme de tipuri, concurență), „Algorithmic Foundations” (structuri de date fundamentale) și „Software Engineering” (calitatea codului, testare, refactorizare, șabloane de proiectare). Competențele formate (proiectarea de tipuri abstracte de date, prelucrarea eficientă a datelor, scrierea de cod testat și mentenabil, lucrul cu versionare și revizuire de cod) sunt cele cerute de angajatorii din industria software la stagiile de practică și la angajare, inclusiv de firmele din regiune cu care departamentul colaborează prin programele de practică și prin întâlnirile periodice cu mediul economic. Pentru programul Informatică, disciplina este complementară cu Programare orientată pe obiecte II și cu Dezvoltarea aplicațiilor web și mobile din același semestru (aceleași idei, văzute în limbaje diferite) și pregătește Programare funcțională, Programare paralelă, concurentă și distribuită, Inginerie software și Procese și dezvoltare software din anul III; ocupație vizată: programator de sistem informatic (COR 251204).

11. Evaluare

Tip activitate	11.1 Criterii de evaluare	11.2 Metode de evaluare (și forma evaluare: continuă/sumativă)	11.3 Pondere din nota finală
11.4 Curs	Cunoașterea și înțelegerea conceptelor (RI1-RI3): modelul de date, mecanismele de programare avansată, structurile de date și complexitatea lor, principiile și șabloanele de proiectare, testarea. Capacitatea de a rezolva individual, pe calculator, probleme noi: implementarea unui tip abstract de date cu teste date și prelucrarea eficientă a unui set de date, cu cod corect, lizibil și testat (RI4-RI7).	Evaluare continuă: două teste scurte pe Campus Virtual (T1 în săpt. 6, materia săpt. 1-5; T2 în săpt. 11, materia săpt. 6-10), corectate automat (10%). Evaluare sumativă, examen în sesiune: probă practică individuală pe calculator, 2 ore, două sarcini: implementarea unui tip abstract de date care trece testele date și prelucrarea eficientă a unui set de date, fără instrumente de asistență, verificată automat și citită de titular (30%); chestionar conceptual pe Campus Virtual, 30 de minute (10%).	50% (10% teste scurte, 30% probă practică, 10% chestionar conceptual)

11.5 Seminar/Laborator /Proiect / practică	Portofoliul de laborator (RI4-RI7, RI9): sarcinile săptămânale cu nivelul de bază trecut de verificarea automată și lucrările notate, cu cod corect, lizibil și testat; respectarea termenelor; declararea instrumentelor de asistență. Proiectul final (RI4-RI9): aplicație individuală mică, organizată pe module, cu tipuri abstracte de date proprii sau structuri alese justificat, teste automate, documentație și istoric de versiuni, funcțională în cerințele de bază, prezentată și explicată de student.	Evaluare continuă: portofoliul de laborator (lucrările L1-L6), predat prin Campus Virtual, verificat automat și notat pe rubrică de titularul de laborator (30%). Evaluare continuă: proiectul final, pe una dintre temele date, predat și prezentat la laborator în săpt. 14, notat pe rubrică; studentul explică și modifică la cerere propriul cod (20%).	50% (30% portofoliul de laborator, 20% proiectul final)
--	---	--	---

11.6 Standard minim de performanță

Nota finală $N = 0,10 \times TS + 0,30 \times PL + 0,20 \times PF + 0,30 \times EP + 0,10 \times EC$, unde TS = media testelor scurte, PL = portofoliul de laborator, PF = proiectul final, EP = proba practică, EC = chestionarul conceptual, toate notate de la 1 la 10. Evaluarea continuă (TS, PL, PF) reprezintă 60% din nota finală, evaluarea sumativă (EP, EC) 40%.

Condiții minime pentru nota 5: toate lucrările L1-L6 predate și $PL \geq 5$; proiectul final predat, funcțional în cerințele de bază, cu testele de bază trecute și explicat de student ($PF \geq 5$); ambele teste scurte susținute; la proba practică, studentul rezolvă singur, complet și corect, cel puțin una dintre sarcini și obține $EP \geq 5$; nota finală $N \geq 5$.

Livrabilele întârziate pierd un punct pe săptămână din nota livrabilului. Instrumentele de asistență se declară în fiecare livrabil și sunt interzise la teste și la proba practică; nedeclararea sau incapacitatea de a explica propriul cod se tratează ca abatere de la integritatea academică, conform regulamentelor UTCN. În sesiunea de restanțe se reface numai examenul (proba practică și chestionarul conceptual); testele, portofoliul și proiectul rămân valabile în anul universitar curent.

Data completării:	Titulari	grad didactic, titlu Prenume NUME	Semnătura
10.09.2026	Curs	Conf. univ. dr. Cosmin Nicolae SABO	
	Aplicații	Asistent universitar, post vacant	
		-	

Avizul Directorului departamentului care coordonează disciplina, doar dacă acesta diferă de departamentul organizator al programului de studii	Nu este cazul (disciplina este coordonată de departamentul organizator al programului de studii).
--	---

Data avizării în Consiliul Departamentului: 14.09.2026

Director Departament DMI

Conf. univ. dr. Cosmin Nicolae SABO

Data aprobării în Consiliul Facultății: 17.09.2026

Decan,

Conf. univ. dr. Monica Liliana MARIAN