

FIȘA DISCIPLINEI

1. Date despre program

1.1 Instituția de învățământ superior	UNIVERSITATEA DIN ORADEA
1.2 Facultatea	INGINERIE ELECTRICĂ ȘI TEHNOLOGIA INFORMAȚIEI
1.3 Departamentul	CALCULATOARE ȘI TEHNOLOGIA INFORMAȚIEI
1.4 Domeniul de studii	CALCULATOARE ȘI TEHNOLOGIA INFORMAȚIEI
1.5 Ciclu de studii	LICENȚĂ
1.6 Programul de studii/Calificarea	TEHNOLOGIA INFORMAȚIEI / INGINER

2. Date despre disciplină

2.1 Denumirea disciplinei	PROGRAMARE FUNCȚIONALĂ						
2.2 Titularul activităților de curs	ș.l. dr. inf. Costea Felicia Mirabela						
2.3 Titularul activităților de seminar /laborator/proiect	ș.l. dr. inf. Costea Felicia Mirabela						
2.4 Anul de studiu	II	2.5 Semestrul	I	2.6 Tipul de evaluare	Ex	2.7 Regimul disciplinei	I

(I) Impusă; (O) Opțională; (F) Facultativă

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/proiect	0/2/0
3.4 Total ore din planul de învățământ	56	din care: 3.5 curs	28	3.6 seminar/laborator/proiect	0/28/0
Distribuția fondului de timp					ore
Studiul după manual, suport de curs, bibliografie și notițe					14
Documentare suplimentară în bibliotecă, pe platformele electronice de specialitate și pe teren					10
Pregătire seminarii/laboratoare, teme, referate, portofolii și eseuri					28
Tutoriat					7
Examinări					10
Alte activități.....					
3.7 Total ore studiu individual	69				
3.9 Total ore pe semestru	125				
3.10 Numărul de credite	5				

4. Precondiții (acolo unde este cazul)

4.1 de curriculum	Noțiunile de bază aferente disciplinelor din anul I
4.2 de competențe	Noțiuni fundamentale de algoritmi și structuri de date de bază. Recursivitate (aplicată pe exemple simple). Noțiuni de bază ale limbajului Python: tipuri de date primitive (int, float, string, bool), structuri de control (if, for, while), funcții simple, lucrul cu liste și dicționare.

5. Condiții (acolo unde este cazul)

5.1. de desfășurare a cursului	Cursul se desfășoară față în față. Cursul se desfășoară cu tehnicile disponibile: Laptop, Videoproiector, Tablă inteligentă
5.2. de desfășurare a seminarului/laboratorului/proiectului	Prezența obligatorie la laborator 100% Laboratorul se desfășoară față în față. Recuperarea laboratoarelor se face conform Regulament privind activitatea profesională a studenților în baza Sistemului European de Credite Transferabile (ECTS) Nivelul maxim admis al absențelor motivate sau nemotivate la activitățile practice (laboratoare, stagii, proiecte etc.) care pot fi recuperate este de până la 30% din totalul activităților. Lucrările de laborator se realizează utilizând mijloacele de lucru existente în laborator:

6.1. Competențele specifice acumulate	
Competențe profesionale	C2 - Proiectarea componentelor hardware, software și de comunicații C3 . Soluționarea problemelor folosind instrumentele științei și ingineriei calculatoarelor
Competențe transversale	- Comportarea onorabilă, responsabilă, etică, în spiritul legii pentru a asigura reputația profesiei - Descrierea clară și concisă în scris, în limba română a rezultatelor din domeniul de activitate - Demonstrarea spiritului de inițiativă și acțiune pentru actualizarea cunoștințelor profesionale
6.2. Rezultatele învățării	
Cunoștințe	Studentul/absolventul descrie, identifică și rezumă concepte și metode fundamentale ale programării funcționale (funcții pure, recursivitate, imutabilitate, funcții de ordin superior) și modul lor de aplicare în rezolvarea problemelor concrete de prelucrare a datelor și proiectare software.
Aptitudini	Studentul utilizează principii și tehnici de programare funcțională în Python pentru a proiecta și implementa componente software robuste și eficiente, demonstrând capacitatea de a integra metode moderne în soluționarea problemelor de inginerie a calculatoarelor.
Responsabilitate și autonomie	- Studentul documentează clar, concis și structurat soluțiile dezvoltate, redactând rapoarte și cod sursă bine comentat în limba română, respectând normele de comunicare profesională și etică în domeniu. - Studentul își manifestă spiritul de inițiativă prin actualizarea constantă a cunoștințelor în programarea funcțională și prin explorarea resurselor moderne (biblioteci, framework-uri, practici de testare), demonstrând capacitatea de autoinstruire și adaptare la cerințele pieței IT.

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

7.1 Obiectivul general al disciplinei	Scopul disciplinei este de a dezvolta capacitatea studenților de a gândi și a implementa soluții software robuste, utilizând principiile programării funcționale. Se urmărește însușirea unor tehnici moderne care reduc erorile, cresc lizibilitatea și predictibilitatea codului și facilitează dezvoltarea de aplicații scalabile și testabile..
7.2 Obiectivele specifice	- Înțelegerea principiilor fundamentale ale programării funcționale: funcții pure, imutabilitate, recursivitate. - Aplicarea funcțiilor de ordin superior (map, filter, reduce) și a expresiilor lambda pentru rezolvarea de probleme. - Utilizarea tipurilor de date compuse și a structurilor recursive (liste, arbori) în rezolvarea funcțională a problemelor. - Familiarizarea cu tehnici de evaluare leneșă și generatoare pentru lucrul cu date infinite sau mari. - Exersarea programării funcțional-reactive și a paradigmatelor concurente moderne în Python. - Învățarea metodelor de testare bazată pe proprietăți și verificare formală a corectitudinii programelor. - Dezvoltarea unui mini-proiect aplicat care să combine conceptele teoretice cu instrumente moderne din ecosistemul Python.

8. Conținuturi*

8.1 Curs	Metode de predare	Nr. Ore / Observații
1. Introducere în programarea funcțională.	Videoproiector, slide-uri și	2
2. Fundamente ale limbajelor funcționale Funcții pure, imutabilitate, recursivitate.		2

Tipuri de date de bază și compuse (liste, tupluri, structuri de date algebrice).	predare interactivă la tablă	
3. Funcții de ordin superior - Map, filter, reduce, fold, comprehensiuni. - Funcții anonime, closures, partial application.		2
4. Structuri avansate - Arbore binar, arbori de căutare, grafuri funcționale. - Lazy evaluation și colecții infinite.		2
5. Programare funcțional-reactivă - Streamuri de date, observabile, event-driven programming. - Exemple cu RxJS, Akka Streams, Elixir.		2
6. Tipuri și polimorfism Tipuri generice, inferență de tip, monade și functuri.		2
7. Calcul Lambda și fundamente teoretice - Noțiuni: category theory pentru programatori. - Legătura cu designul limbajelor actuale.		2
8. Programare funcțională în ecosisteme moderne Funcțional în Python (tool-uri precum functools, itertools).		2
9. Programare funcțională în ecosisteme moderne JavaScript/TypeScript: Redux, React hooks ca paradigme funcționale.		2
10. Programare funcțională în ecosisteme moderne Rust: ownership + functional patterns.		2
11. Aplicații practice Dezvoltarea unor aplicații robuste, testabile și scalabile..		2
12. Aplicații practice Microservicii funcționale cu Elixir/Phoenix.		2
13. Aplicații practice Data processing funcțional cu Spark (Scala, PySpark).		2
14. Aplicații practice.		2
Bibliografie - Steven Lott– <i>Functional Python Programming (3rd Edition)</i>, Packt, 2022 - Graham Huton, Programming in Haskell, http://www.cs.nott.ac.uk/~gmh/ - Venkat Subramaniam – <i>Programming Groovy & Functional Programming with Java</i> (O’Reilly, actualizat 2020) - David Mertz – <i>Functional programming in Python</i>, O’Reilly Media, 2015 - Ioan Alfred Letia, Liviu A. Negrescu – <i>Programare Funcțională. Vol. 1</i> , Ed. Albastra, 2018 - Rúnar Bjarnason – <i>Functional Programming Principles in Scala</i>, Coursera MOOC (actualizat periodic) - Eric Normand – <i>Grokking Simplicity: Taming complex software with functional thinking</i>, Manning, 2021 - Harold Abelson, Gerald Jay Sussman – <i>Structure and Interpretation of Computer Programs</i>, MIT Press, 2022 edition http://www.haskell.org/haskellwiki/Haskell_in_education https://www.python.org/ https://www.codecademy.com/language/python https://www.scala-lang.org/ https://elixir-lang.org/ https://doc.rust-lang.org/book/ https://rxjs.dev/ https://redux.js.org/ https://www.coursera.org/specializations/scala		
8.2 Seminar	Metode de predare	Nr. Ore / Observații
8.3 Laborator		

Introducere în programarea funcțională în Python – Folosirea lambda, funcții pure, principiul imutabilității. – Primele exerciții cu funcții simple și testarea lor în REPL / Jupyter Notebook.	Studiu experimental, programare , dezbateri.	2
Recursivitate în Python – Implementarea recursivă a factorialului, Fibonacci, căutării într-o listă. – Analiza avantajelor și limitărilor față de abordarea iterativă.	Studiu experimental, programare , dezbateri.	2
Funcții de ordin superior – Utilizarea map(), filter(), reduce() (din functools). – Scrierea unor funcții proprii de ordin superior.	Studiu experimental, programare , dezbateri.	2
Lucrul cu liste și colecții funcționale – List comprehensions și generator expressions. – Manipularea colecțiilor (liste, seturi, dicționare) în stil funcțional.	Studiu experimental, programare , dezbateri.	2
Pattern matching și structuri de date (Python 3.10+) – Folosirea instrucțiunii match-case pentru modelarea structurată a datelor. – Exerciții cu tuple și dicționare.	Studiu experimental, programare , dezbateri.	2
Arbori și structuri recursive – Definirea unei structuri de arbore binar cu dataclasses. – Funcții recursive pentru traversare (in-order, pre-order, post-order).	Studiu experimental, programare , dezbateri.	2
Evaluare leneșă și generatoare – Crearea de secvențe infinite cu yield. – Exemple: generarea numerelor prime, șirului Fibonacci.	Studiu experimental, programare , dezbateri.	2
Funcții ca obiecte și compunere – Funcții returnate din funcții. – Scrierea unui mecanism simplu de compunere a funcțiilor (compose).	Studiu experimental, programare , dezbateri.	2
Decoratori și funcționalitate avansată – Crearea de decoratori simpli pentru logare, memorare (functools.lru_cache). – Exemple practice de decoratori funcționali.	Studiu experimental, programare , dezbateri.	2
Prelucrarea funcțională a datelor – Procesarea unui fișier CSV folosind map, filter, reduce. – Exerciții de analiză funcțională pe dataset-uri mici (de ex. Pandas cu stil funcțional).	Studiu experimental, programare , dezbateri.	2
Programare funcțional-reactivă în Python – Introducere în RxPY (ReactiveX pentru Python). – Exemple cu fluxuri de evenimente și observabile.	Studiu experimental, programare , dezbateri.	2
Testare bazată pe proprietăți – Introducere în hypothesis pentru property-based testing. – Exerciții de validare automată a funcțiilor.	Studiu experimental, programare , dezbateri.	2
Mini-proiect – Construirea unei aplicații simple (ex. analiză de text, filtru de date, procesare streaming). – Aplicarea principiilor funcționale pe un caz concret.	Studiu experimental, programare , dezbateri.	2
Evaluare finală laborator – Test practic + prezentarea mini-proiectului.	Prezentarea proiectelor, întrebări	2
8.4 Proiect		
Bibliografie 1. David Mertz – <i>Functional Programming in Python</i> , O'Reilly Media, 2015. 2. Steven F. Lott & Dusty Phillips – <i>Functional Python Programming</i> , 2nd Edition, Packt Publishing, 2019. 3. Chris Meyers – <i>Functional Programming in Python</i> , Manning Publications, 2021. 4. Luciano Ramalho – <i>Fluent Python</i> , 2nd Edition, O'Reilly Media, 2022 5. Micha Gorelick, Ian Ozsvald – <i>High Performance Python</i> , 2nd Edition, O'Reilly Media, 2020 6. Julien Danjou – <i>Serious Python: Black-Belt Advice on Deployment, Scalability, Testing, and More</i> , No Starch Press, 2018		

7. David Beazley & Brian K. Jones – *Python Cookbook*, 3rd Edition, O'Reilly Media, 2013.
8. Hypothesis library – Documentație oficială pentru property-based testing:
<https://hypothesis.readthedocs.io/>
9. RxPY (ReactiveX for Python) – Documentație oficială: <https://rxpy.readthedocs.io/>
10. Documentația oficială Python – capitolele despre functools, itertools, collections:
<https://docs.python.org/3/library/>

* Se va detalia conținutul, respectiv numărul de ore alocat fiecărui curs/seminar/laborator/proiect pe durata celor 14 săptămâni ale fiecărui semestru al anului universitar.

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

Disciplina oferă cunoștințe teoretice și practice direct aplicabile în industria de calculatoare și în domeniul serviciilor de tehnologia informației. În sprijinul obiectivelor de business ale firmelor IT de a dezvolta produse software robuste și minimizarea erorilor, acest curs se concentrează pe corectitudinea dezvoltării programelor. În cadrul cursului sunt prezentate metode formale pe baza principiului inducției matematice pentru verificarea corectitudinii programelor. Conținutul disciplinei este în concordanță cu cursuri similare ale altor universități din țară.

10. Evaluare

Tip activitate	10.1 Criterii de evaluare	10.2 Metode de evaluare	10.3 Pondere din nota finală
10.4 Curs	Înțelegerea principiilor programării funcționale, explicarea corectitudinii programelor, aplicarea tehnicilor de raționament formal.	Evaluarea se face față în față Examen scris	40%
10.5 Seminar			
10.6 Laborator	Implementarea corectă a funcțiilor și algoritmilor în Python, folosirea funcțiilor de ordin superior, lucrul cu structuri recursive și generatoare.. Dezvoltarea unui mini-proiect aplicat (ex: analiză de date, procesare stream, aplicație educațională).	Test practic + testare cu pytest/hypothesis cod+documentație+ prezentare finală	30% 30%
Evaluare continuă (60%) și Evaluare sumativă (40%) Se va calcula o notă folosind media aritmetică ponderată ($4 \cdot \text{nota_curs} + 3 \cdot \text{nota_mini-proiect} + 3 \cdot \text{nota_test-laborator}$)/10.			
10.7 Proiect			
10.8 Standard minim de performanță Studentul trebuie să demonstreze capacitatea de: • a scrie cod corect și lizibil în Python folosind concepte funcționale (funcții pure, map/filter/reduce, recursivitate); • a rezolva o problemă simplă printr-o abordare funcțională; • a susține un mini-proiect care aplică cel puțin două concepte majore (funcții de ordin superior, colecții funcționale, evaluare leneșă, testare automată). • Obține minim nota 5 atât la curs cât și la laborator și mini-proiect • Prezența la laborator 100%			

Data completării
02.09.2025

Semnătura titularului de curs
Ș.l.dr.inf. CosteaMirabela
mira_costea@uoradea.ro

Semnătura titularului de laborator
Ș.l.dr.inf.. CosteaMirabela

Data avizării în departament
.09.2025

Semnătura directorului de departament
Conf. dr. inf. Moisi Elisa
emoisi@uoradea.ro

Data avizării în consiliul facultății
.09.2025

Semnătură Decan
Conf.dr.ing. Eugen GERGELY
egergely@uoradea.ro