# FIŞA DISCIPLINEI

*Programarea calculatoarelor şi limbaje de programare*

Anul universitar 2026-2027

**1. Date despre program**

| | |
|---|---|
| 1.1. Instituția de învățământ 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 | Calculatoare şi Tehnologia Informaţiei |
| 1.5. Ciclul de studii | Licenţă |
| 1.6. Programul de studii / Calificarea | Ingineria Informaţiei (în limba engleză) |
| 1.7. Forma de învățământ | Cu frecvență |

**2. Date despre disciplină**

| | | | | | |
|---|---|---|---|---|---|
| 2.1. Denumirea disciplinei | **Programarea calculatoarelor si limbaje de programare** | | | Codul disciplinei | **MLE5171** |
| 2.2. Titularul activităților de curs | Asist. dr. Anamaria Briciu | | | | |
| 2.3. Titularul activităților de seminar | Asist. dr. Anamaria Briciu | | | | |
| 2.4. Anul de studiu | 1 | 2.5. Semestrul | 1 | 2.6. Tipul de evaluare | Colocviu |
| 2.7. Regimul disciplinei | Obligatoriu | | 2.8. Tipul disciplinei | | Disciplină fundamentală (DF) |

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

| | | | | | |
|---|---|---|---|---|---|
| 3.1. Număr de ore pe săptămână | 5 | din care: 3.2. curs | 2 | 3.3. seminar/ laborator/ proiect | 1 S<br>2 LP |
| 3.4. Total ore din planul de învățământ | 70 | din care: 3.5. curs | 28 | 3.6 seminar/laborator | 42 |
| **Distribuția fondului de timp pentru studiul individual (SI) și activități de autoinstruire (AI)** | | | | | **ore** |
| Studiul după manual, suport de curs, bibliografie și notițe (AI) | | | | | 20 |
| Documentare suplimentară în bibliotecă, pe platformele electronice de specialitate și pe teren | | | | | 4 |
| Pregătire seminare/ laboratoare/ proiecte, teme, referate, portofolii și eseuri | | | | | 34 |
| Tutoriat (consiliere profesională) | | | | | 9 |
| Examinări | | | | | 8 |
| Alte activități | | | | | 5 |
| **3.7. Total ore studiu individual (SI) și activități de autoinstruire (AI)** | | | | **80** | |
| **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 | Proiector |
| 5.2. de desfășurare a seminarului/ laboratorului | Calculatoare, limbajul şi mediul de programare Python |

**6.1. Competențele dobândite în urma absolvirii programului de studii** (se preiau din planul de învățământ)[1]

| Competențe profesionale | |
|---|---|
| **Codul competenței** | **Competență** |
| CP1 | creează softuri |
| CP3 | analizează specificații software |
| CP5 | proiectează sistemul informatic |
| **Competențe transversale** | |
| **Codul competenței** | **Competență** |
| CT2 | Soluționează probleme |
| CT3 | Gândește analitic |

**6.2. Rezultatele învățării specifice programului de studii** (se preiau din planul de învățământ)[2]

| Rezultatele învățării vizate prin disciplină | | |
|---|---|---|
| **Codul competenței** | **Cunoștințe și înțelegere (Knowledge and understanding)** | **Abilități academice specifice (Specific academic skills)** |
| CP1<br>CP3 | 1. Studentul/absolventul descrie, identifică, sumarizează concepte şi metode elementare privitoare la limbaje de programare, medii de programare, tehnici de programare, baze de date. inteligență artificială și inginerie software și modul lor de aplicare în probleme concrete | 1. Studentul/absolventul alege şi explică concepte proprii specifice proiectării algoritmilor, programării orientate pe obiecte, programării logice şi funcționale. Studentul/absolventul specifică cerințe, analizează, elaborează, dezvoltă şi testează programe în limbaje de programare de uz general (C, etc.) şi /sau obiect-orientate (C++, Java, etc.), aplicând elementele specifice ingineriei software. |
| CP5 | 2. Absolventul are cunoştințe adecvate legate de folosirea mediilor de dezvoltare integrate în scopul creării de aplicații complexe de dimensiuni mari. Absolventul are cunoştințe legate de programare, matematică, inginerie şi tehnologie si are abilitățile necesare pentru a le folosi in crearea de sisteme informatice complexe. | 2. Absolventul are abilitatea de a dezvolta, proiecta şi crea noi aplicații, sisteme sau produse folosind bunele practici din domeniu. |

**7. Rezultatele învățării specifice disciplinei**

| Cunoștințe și înțelegere (Knowledge and understanding) |
|---|
| 1. Studentul cunoaște conceptele de bază din domeniul ingineriei software (proiectare, implementare şi mentenanță). |
| 2. Studentul înțelege structura şi funcționalitatea aplicațiilor software la diferite niveluri de abstractizare, inclusiv arhitectură, module, clase şi funcții. |

[1] Se vor prelua din Planul de învățământ al programului de studii acele competențe profesionale şi/sau transversale la dezvoltarea cărora contribuie disciplina pentru care se elaborează fişa disciplinei. Pentru fiecare competență se va prelua întregul enunț, inclusiv codul competenței, cu formularea care apare în planul de învățământ, fără modificări. Dacă nu se preia nici o competență din oricare din cele două categorii, se şterge linia din tabel aferentă acelei categorii.

[2] Se menționează rezultatele învățării specifice programului de studiu la dezvoltarea cărora contribuie disciplina pentru care se elaborează fişa. Enunțurile, preluate fără modificări din Planul de învățământ în funcție de tipul disciplinei (DF/DS/DC) se trec în dreptul competenței asociate.

| |
|---|
| 3. Studentul cunoaşte elementele de bază din limbajul de programare Python şi este familiar cu uneltele folosite în dezvoltarea, execuţia, testarea şi depanarea programelor |
| **Abilități academice specifice (Specific academic skills)** |
| 1.Studentul poate proiecta și implementa aplicații software într-un limbaj de programare, respectând specificațiile date și aplicând principii de programare adecvate. |
| 2. Studentul poate analiza, interpreta și explica cod sursă existent, identificând structura, funcționalitatea și potențialele probleme ale acestuia. |
| 3. Studentul poate organiza codul sursă în componente reutilizabile și ușor de întreținut și poate elabora documentația tehnică corespunzătoare. |

**8. Conținuturi**

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

[3] De exemplu aspecte organizatorice, recomandări pentru studenți, aspecte specifice legate de curs/seminar cum ar fi invitarea unor practicieni în domeniu etc.

| | | |
|---|---|---|
| • Excepţii | | |
| 5. **Principii de proiectare software**<br>• Arhitectura stratificată: intefaţa utilizator, aplicaţie, domeniu, infrastructură<br>• Cum organizăm codul sursă: responsabilităţi, principiul responsabilităţii unice, separarea grijilor, dependenţe, cuplaj, coeziune | • Expunerea interactivă<br>• Explicaţia<br>• Conversaţia<br>• Exemple<br>• Demonstraţia didactică | |
| 6. **Programare orientată obiect**<br>• Tipuri abstracte de date<br>• Implementarea claselor în Python<br>• Obiecte şi clase: clase, obiecte, atribute, metode, scop şi namespace în Python | • Expunerea interactivă<br>• Explicaţia<br>• Conversaţia<br>• Exemple<br>• Demonstraţia didactică | |
| 7. **Aspecte legate de proiectare**<br>• Strategii de tip top-down şi bottom-up<br>• Elemente de interfaţă utilizator (UI) | • Expunerea interactivă<br>• Explicaţia<br>• Conversaţia<br>• Exemple<br>• Demonstraţia didactică | |
| **8. Testarea şi inspecţia programelor**<br>• Metode de testare: testare exhaustivă, testare de tip black box, testare de tip white box<br>• Testare automată, TDD<br>• Operaţii cu fişiere în Python | • Expunerea interactivă<br>• Explicaţia<br>• Conversaţia<br>• Exemple<br>• Demonstraţia didactică | |
| **9. Recursivitate**<br>• Noţiunea de recursivitate<br>• Recursivitate directă şi indirectă<br>• Exemple<br>**Complexitatea algoritmilor** | • Expunerea interactivă<br>• Explicaţia<br>• Conversaţia<br>• Exemple<br>• Demonstraţia didactică | |
| 10. **Algoritmi de căutare**<br>• Definirea problemei<br>• Metode de căutare: secvenţială, binară<br>• Complexitatea algoritmilor<br>**Algoritmi de sortare**<br>• Definirea problemei<br>• Metode de sortare: Bubble Sort, Selection Sort, Insertion Sort, Quick Sort | • Expunerea interactivă<br>• Explicaţia<br>• Conversaţia<br>• Exemple<br>• Demonstraţia didactică | |

| | | |
|---|---|---|
| ● Complexitatea algoritmilor | | |
| **11. Backtracking**<br>● Prezentarea generală a metodei<br>● Complexitatea algoritmilor<br>● Exemple | ● Expunerea interactivă<br>● Explicația<br>● Conversația<br>● Exemple<br>● Demonstrația didactică | |
| 12**. Divide et impera, Greedy**<br>● Prezentarea generală a metodei<br>● Complexitatea algoritmilor<br>● Exemple | ● Expunerea interactivă<br>● Explicația<br>● Conversația<br>● Exemple<br>● Demonstrația didactică | |
| **13. Programare dinamică**<br>● Descrierea metodei<br>● Exemple | ● Expunerea interactivă<br>● Explicația<br>● Conversația<br>● Exemple<br>● Demonstrația 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

6. The Python Tutorial - https://docs.python.org/3/tutorial/

| **8.2 Laborator** | **Metode de predare - învățare** | **Observații** |
|---|---|---|
| 1. Programe Python simple | ● Expunerea interactivă<br>● Explicația<br>● Conversația<br>● Demonstrația didactică | |
| 2. Programare procedurală | | |
| 3. Programare modulară | | |
| 4. Dezvoltare software bazată pe funcționalități | | |
| 5. Tipuri abstracte de date | | |
| 6. Principii de proiectare | | |
| 7. Programare orientată obiect | | |

| | | |
|---|---|---|
| 8. Proiectarea programelor. Arhitectura stratificată | | |
| 9. Inspecţie şi testare | | |
| 10. Recursivitate. Complexitatea algoritmilo | | |
| 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/


| **8.3 Seminar** | **Metode de predare - învăţare** | **Observaţii** |
|---|---|---|
| 1. Programe Python simple. Programare procedurală | • Expunerea interactivă<br>• Explicaţia<br>• Conversaţia<br>• Demonstraţia didactică | |
| 2. Programare modulară. Dezvoltare software bazată pe funcţionalităţi | | |
| 3. Tipuri abstracte de date. Principii de proiectare | | |
| 4. Programare orientată obiect. Proiectarea programelor. Arhitectura stratificată | | |
| 5. Inspecţie şi testare. Recursivitate. Complexitatea algoritmilor | | |
| 6. Algoritmi de căutare şi sortare. Metode de rezolvare a problemelor: Backtracking | | |
| 7. Metode de rezolvare a problemelor: Greedy | | |


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. Evaluare**

| Tip activitate | 9.1 Criterii de evaluare[4] | 9.2 Metode de evaluare[5] | 9.3 Pondere din nota finală |
|---|---|---|---|
| 9.4 Curs | Corectitutdinea şi completitudinea cunoştinţelor acumulate şi capacitatea de a proiecta şi a implementa corect programe Python | Examen scris | 40% |
| 9.5 Laborator | Abilitatea de a proiecta, implementa şi testa un program Python | Examen practic | 30% |
| | Corectitudinea temelor de laborator şi documentaţiă livrată în timpul semestrului | Program şi documentaţie | 30% |
| 9.6. Seminar | Activitate de seminar | Participare activă la discuțiile din timpul seminariilor (punerea de întrebări, răspuns la întrebările cadrului didactic, rezolvarea de probleme, etc) | Maximum 0.5 puncte bonus, adăugate notei finale |

9.7. Standard minim de promovare

- Fiecare student trebuie să demonstreze un nivel acceptabil de cunoștințe și înțelegere a domeniului, capacitatea de a prezenta cunoștințele într-o manieră coerentă și abilitatea de a stabili conexiuni și de a folosi aceste cunoștințe pentru a rezolva diverse probleme în Python.
- Fiecare student este obligat să participe la minimum 10 seminarii și 12 laboratoare.
- Trebuie obținută o notă minimă de 5 la activitatea de laborator, testul practic și examenul scris.

**10. Etichete ODD (Obiective de Dezvoltare Durabilă / Sustainable Development Goals)[6]**

[4] Criteriile de evaluare trebuie să reflecte direct rezultatele învățării vizate la nivel de program de studii, respectiv la nivel de disciplină. Mai concret, se evaluează achizițiile de învățare menționate în rezultatele anticipate ale învățării.

[5] Se recomandă stabilirea atât a metodelor de evaluare finală, cât și a strategiei de evaluare pe parcurs.

[6] Selectați o singură etichetă, cea care, în conformitate cu *Procedura de aplicare a etichetelor ODD în procesul academic*, se potrivește cel mai bine disciplinei. Dacă disciplina tratează tema dezvoltării durabile la modul general (de ex. prin prezentarea/introducerea cadrului general al dezvoltării durabile etc.) atunci se poate aloca eticheta generală de Dezvoltare Durabilă. Dacă niciuna dintre etichete nu descrie disciplina, selectați ultima opțiune: „*Nu se aplică nici o etichetă*”.

| | | |
|---|---|---|
|  | | Eticheta generală pentru Dezvoltare durabilă |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
|  |  |  |  |  |  |  |  |  |
| | | | | | | | | x |
|  |  |  |  |  |  |  |  | Nu se aplică nici o etichetă |
| | | | | | | | | |

Data completării:
19.10.2026

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