test-e.ro

CAIET ELEV

Lecția 29

Programare orientată pe obiecte – noțiuni de bază. Clase predefinite

Lecția formalizează retroactiv ceea ce elevii practică de zece lecții: listele, șirurile și fișierul întors de open() sunt obiecte.

DISCIPLINA Informatică **CLASA** a IX-a · mate-info **LIMBAJ** Python 3

NUMELE ELEVULUI .....

CUM FOLOSEȘTI CAIETUL

Rezolvă exercițiile pe hârtie: urmărești execuția pas cu pas, completezi, asociezi și scrii singur soluțiile, apoi le verifici pe platformă. Ce ai voie să folosești: tot ce ai învățat până la lecția 28, plus noul construct class — if/elif/else, while, for cu range și for pe listă, break, continue, %, //, ord()/chr(), liste, indexare, len, metodele listei, def/return, import math, min/max/sum, slicing, atribuire simultană, open/with. Fără sorted() și fără global. Clasa Elev îți este dată gata scrisă: o instanțiezi și îi apelezi metoda, nu proiectezi clase noi. Problemele cu date calendaristice lucrează doar cu valori citite de la intrare. Fără diacritice în codul Python.

CUPRINS

1 Ne reamintim uneltele .....
2 Tot ce ai folosit până acum este un obiect
3 Ce știe să facă un text .....
4 Din text în listă, cu o singură metodă ....
5 Datele calendaristice sunt și ele obiecte .....
6 De unde vin clasele și cum arată ele pe dinăuntru
7 Cod ghidat · cinci probleme .....

SECȚIUNEA 1 · REACTUALIZARE: ȘIRURI, LISTE, FIȘIERE, SUBPROGRAME

1 Ne reamintim *uneltele*

Înainte de a da un nume lucrurilor pe care le folosim de zece lecții, trecem repede prin ele: caracterele unui șir, listele și metodele lor, modurile de deschidere a unui fișier și subprogramele. Un quiz scurt, o trasare și două exerciții de reîncălzire.

Un șir de caractere se parcurge cu un indice, de la `0` până la `len(s) - 1`, iar `s[i]` este caracterul de pe poziția `i`. O listă se construiește cu paranteze drepte și crește cu `append`. Un fișier se deschide cu `open(nume, mod)`, unde modul `"r"` citește, `"w"` scrie de la zero, iar `"a"` adaugă la sfârșit. Un subprogram se definește cu `def` și trimite un rezultat înapoi cu `return`.

Parcurgerea clasică a unui șir, cu indice:

```
s = "obiect"
i = 0
while i < len(s):
    print(s[i])
    i = i + 1
```

DE REȚINUT

Toate uneltele de mai sus au ceva în comun, iar lecția de azi va spune ce anume. Deocamdată reține doar mecanica: indice, `len`, `append`, mod de deschidere, `return`.

A EXERCIȚIUL A · GRILĂ

Ce știm deja

GRILĂ

Încercuiește răspunsul corect la fiecare întrebare.

1. Ce face metoda append pentru o listă?

   - `Șterge ultimul element din listă și îl întoarce`
   - `Adaugă un element nou la sfârșitul listei`
   - `Așază elementele listei în ordine crescătoare`
   - `Întoarce numărul de elemente pe care le are lista`

2. Ce se întâmplă cu conținutul unui fișier deschis cu open("date.txt", "w")?

   - `Conținutul vechi rămâne, iar scrierile noi se adaugă la sfârșit`

- Fișierul este deschis doar pentru citire, nu se poate scrie în el
- Conținutul vechi este șters, fișierul rămâne gol
- Se produce o eroare dacă fișierul există deja pe disc

3. Pentru s = "informatica", ce afișează print(s[3])?

- litera f, pentru că numărătoarea pozițiilor începe de la 1
- litera r, pentru că pozițiile se numără de la sfârșitul șirului
- litera o, pentru că numărătoarea pozițiilor începe de la 0
- orm, pentru că indicele întoarce trei caractere consecutive

4. Ce întoarce un subprogram care nu conține instrucțiunea return?

- valoarea 0, ca orice funcție matematică fără rezultat
- ultima valoare calculată în corpul subprogramului
- None, adică lipsa unei valori
- o eroare de execuție în momentul apelului

B

EXERCIȚIUL B · DRY-RUN

Numărăm literele e

DRY-RUN

Parcurge șirul literă cu literă. Coloana i este dată; tu completezi caracterul s[i] (valoare text, o singură literă) și valoarea contorului n după executarea pasului. Ultimul rând arată starea în care bucla se oprește; acolo s[i] nu mai există și se scrie —.

```
s = "obiect"
i = 0
n = 0
while i < len(s):
    if s[i] == "e":
        n = n + 1
    i = i + 1
print(n)
```

i	s[i]	n după pas
0	o	0
1		
2		
3		
4		
5		
6 — bucla se oprește	—	1

HINT *la rândurile pe rând: scrie mai întâi litera de pe poziția i din „obiect”, apoi întreabă-te dacă acea literă este e. Contorul crește doar atunci; altfel rămâne cât era.*

SPRIJIN · SCRIE CODUL

Câte cuvinte are propoziția?

SCRIERE COD

Se citește de la tastatură o propoziție scrisă pe un singur rând, în care cuvintele sunt separate prin exact un spațiu, fără spații la început sau la sfârșit. Afișează numărul de cuvinte din propoziție. Ex.: Pentru propoziția ana are mere există 2 spații, deci 3 cuvinte; se afișează 3.

HINT **Observație.** *Propoziția conține cel puțin un cuvânt.*

HINT **Atenție.** *Numărul de cuvinte este numărul de spații plus unu, nu numărul de spații.*

SOLUȚIA TA · PYTHON

SPRIJIN · SCRIE CODUL

Prima poziție a unui caracter

SCRIERE COD

Se citesc două rânduri: pe primul un text s fără spații, pe al doilea un singur caracter c. Afișează poziția primei apariții a lui c în s, pozițiile fiind numerotate de la 0, sau -1 dacă c nu apare în s. Ex.: Pentru s = programare și c = a, litera a apare prima dată pe poziția 5, deci se afișează 5.

HINT Atenție. *Dacă lași bucla să meargă până la capăt fără nicio precauție, vei reține ultima apariție, nu prima. Reține poziția doar dacă nu ai reținut deja una.*

HINT Curiozitate. *Convenția „-1 înseamnă nu am găsit” e veche de zeci de ani și o vei regăsi în aproape orice limbaj de programare.*

SOLUȚIA TA · PYTHON

SECȚIUNEA 2 · CLASĂ, OBIECT, MEMBRI

2 Tot ce ai folosit până acum este un *obiect*

Listele, șirurile de caractere, chiar și fișierul întors de `open()` — toate sunt obiecte, adică pachete care țin la un loc niște date și acțiunile care lucrează cu ele. În secțiunea asta le dăm numele corect: clasă, obiect, atribut, metodă.

O **clasă** este un tipar: descrie ce date va conține fiecare exemplar și ce poate face el. Un **obiect** este un exemplar concret construit după tipar; se mai numește **instanță** a clasei, iar operația prin care îl creăm se numește **instanțiere**. Din aceeași clasă se pot construi oricâte obiecte, fiecare cu datele lui.

Elementele pe care le conține un obiect se numesc **membri** și sunt de două feluri. **Atributele** sunt datele memorate în obiect — se citesc fără paranteze. **Metodele** sunt subprogramele care lucrează cu acele date — se apelează cu paranteze. Amândouă se accesează cu **operatorul punct**, scris între numele obiectului și numele membrului.

Nimic din toate astea nu e nou pentru tine. Când ai scris `note.append(9)`, ai apelat metoda `append` a obiectului `note`, care este o instanță a clasei `list`. Când ai scris `f = open("date.txt", "r")`, ai primit un obiect al clasei `TextIOWrapper`, iar `f.read()` și `f.close()` erau metodele lui.

Funcția `type()` îți arată clasa oricărei valori:

```
1  print(type([3, 5, 7]))
2  print(type("informatica"))
3  print(type(2026))
4  with open("date.txt", "w") as f:
5      print(type(f))
```

Programul afișează, pe rând: `<class 'list'>`, `<class 'str'>`, `<class 'int'>`, `<class '_io.TextIOWrapper'>`.

DE REȚINUT

Clasa decide ce metode ai voie să apelezi. `note.append(9)` funcționează pentru că `note` e o listă; `"text".append(9)` dă eroare, pentru că un obiect `str` nu are metoda `append`. Când primești eroarea „object has no attribute”, întrebarea corectă e „ce clasă are, de fapt, valoarea mea?”.

ATENȚIE

Atributul se scrie fără paranteze (`d.year`), metoda cu paranteze (`d.weekday()`). Dacă pui paranteze la un atribut, Python încearcă să-l apeleze ca pe un subprogram și programul cade.

UNDE SE FOLOSEȘTE ÎN VIAȚA REALĂ

Aplicația de mesaje de pe telefon ține fiecare conversație ca pe un obiect, cu atribute (numele contactului, ora ultimului mesaj) și metode (trimite, șterge, arhivează). Într-un joc, fiecare personaj e un obiect cu atributele „viață” și „poziție” și cu metodele „mergi” și „atacă” — de asta toate personajele se mișcă la fel, dar fiecare are viața lui. Iar în orice aplicație bancară, un cont e un obiect cu atributul „sold” și metodele „depune” și „retrage”, tocmai ca soldul să nu poată fi schimbat oricum, ci doar prin acele metode.

A

EXERCIȚIUL A · GRILĂ

Clasă și obiect

GRILĂ

Încercuiește răspunsul corect la fiecare întrebare.

1. Care este relația corectă între clasă și obiect?

   - `Obiectul e tiparul, iar clasa e o copie a lui creată în memorie`
   - `Clasa e tiparul, iar obiectul e un exemplar concret construit după el`
   - `Obiectul există doar în fișiere, iar clasa doar în memoria calculatorului`
   - `Clasa și obiectul sunt două nume diferite pentru exact același lucru`

2. În instrucțiunea note.append(9), ce este append?

   - `O funcție a limbajului, care nu are legătură cu vreo clasă anume`
   - `Numele unei biblioteci care trebuie importată înainte de folosire`
   - `Un atribut al obiectului note, care păstrează ultima valoare adăugată`
   - `O metodă a clasei list, apelată pe obiectul note`

3. Ce afișează print(type("informatica"))?

   - `11, pentru că type întoarce numărul de caractere ale textului`
   - `<class 'text'>, pentru că în Python textele se numesc text`
   - `informatica, pentru că type afișează conținutul variabilei`
   - `<class 'str'>, pentru că un text este un obiect al clasei str`

B

EXERCIȚIUL B · ASOCIERE

Expresie → clasa obiectului

ASOCIERE

Fiecare expresie din stânga construiește un obiect. Scrie în căsuță litera clasei căreia îi aparține obiectul respectiv. Fiecare literă se folosește o singură dată.

[3, 5, 7]

"informatica"

open("date.txt", "r")

date(2026, 9, 5)

A `date` B `list` C `TextIOWrapper` D `str`

C

EXERCIȚIUL C · COMPLETARE

Cum ajungem la un membru

COMPLETARE

Completează golurile folosind cuvintele din rezervă (două cuvinte rămân nefolosite).

`bibliotecă` `virgulă` `punct` `metodă` `atribut`

Ca să ajungem la un membru din interiorul unui obiect, scriem numele obiectului, apoi operatorul _____, apoi numele membrului. Dacă membrul este un _____, îl citim direct; dacă este o _____, adăugăm la final și parantezele rotunde.

3 Ce știe să facă un *text*

SECȚIUNEA 3 · CLASA STR ȘI METODELE EI

Clasa `str` vine cu un set de metode care caută, numără, înlocuiesc și schimbă mărimea literelor. Toate au aceeași particularitate, care surprinde pe toată lumea prima dată: nu modifică textul, ci construiesc unul nou.

Un obiect de clasă `str` are metode pe care le apelezi cu operatorul punct. Cele mai folosite sunt: `upper()` și `lower()` (toate literele mari, respectiv toate mici), `count(x)` (de câte ori apare fragmentul `x`), `find(x)` (poziția primei apariții a lui `x`, sau `-1` dacă nu apare) și `replace(a, b)` (înlocuiește **toate** aparițiile lui `a` cu `b`).

Toate pe același exemplu:

```
1  s = "informatica"
2  print(s.upper())
3  print(s.count("a"))
4  print(s.find("m"))
5  print(s.replace("a", "-"))
```

```
6  print(s)
```

Programul afișează, pe rând: `INFORMATICA`, `2`, `5`, `inform-tic-` și, pe ultimul rând, `informatica`.

ATENȚIE

Ultimul rând e cel important: după toate metodele apelate, `s` a rămas neschimbat. Șirurile din Python sunt **imutabile** — nu pot fi modificate. O metodă ca `upper()` nu schimbă șirul, ci **construiește altul** și ți-l dă. Dacă vrei să păstrezi rezultatul, trebuie să-l atribui: `s = s.upper()`. Scris singur pe un rând, `s.upper()` calculează un șir nou și îl aruncă imediat.

METODE CARE ÎNTORC, NU CARE MODIFICĂ

Regula clasei `str`: rezultatul metodei e o valoare nouă, pe care ori o folosești pe loc (`print(s.upper())`), ori o salvezi (`mare = s.upper()`). Metoda `find` întoarce un număr, metoda `count` întoarce un număr, celelalte întorc un șir nou.

UNDE SE FOLOSEȘTE ÎN VIAȚA REALĂ

Când te loghezi undeva, adresa de e-mail e trecută prin `lower()` înainte de comparare, ca `Ana@mail.ro` și `ana@mail.ro` să fie același cont. Filtrele de limbaj de pe platformele de chat folosesc `count` și `replace` ca să găsească și să acopere cuvintele interzise. Iar căutarea `Ctrl+F` din browser este, la bază, exact `find`: îți spune poziția primei apariții, apoi caută mai departe de acolo.

A

EXERCIȚIUL A · GRILĂ

Metodele clasei str

GRILĂ

Încercuiește răspunsul corect la fiecare întrebare.

1. Ce afișează secvența s = "clasa" / s.upper() / print(s)?

   - None, pentru că rezultatul metodei nu a fost atribuit nimănui
   - o eroare, pentru că metoda upper() trebuie apelată cu un argument
   - CLASA, pentru că upper() modifică direct șirul din variabila s
   - clasa, pentru că upper() întoarce un șir nou și nu îl schimbă pe s

2. Cât este "informatica".count("a")?

   - 3, pentru că se numără și litera căutată, dată ca argument
   - 1, pentru că metoda se oprește la prima apariție găsită
   - 11, pentru că count întoarce lungimea totală a șirului
   - 2, pentru că litera a apare pe pozițiile 6 și 10

3. Ce întoarce "informatica".find("z")?

- `0, pentru că numărătoarea pozițiilor începe de la zero`
- `11, adică poziția de imediat după ultimul caracter al șirului`
- `-1, valoarea convenită pentru „fragmentul nu a fost găsit”`
- `o eroare de execuție, pentru că șirul nu conține caracterul căutat`

4. Ce afișează secvența a = "Ana@Mail.ro" / print(a.lower()) / print(a)?

- `ana@mail.ro pe amândouă rândurile, pentru că lower() modifică șirul`
- `Ana@Mail.ro pe amândouă rândurile, pentru că lower() cere reatribuire ca să facă ceva`
- `ana@mail.ro pe primul rând și Ana@Mail.ro pe al doilea`
- `o eroare, pentru că lower() nu se poate aplica unui text care conține cifre sau semne`

B

EXERCIȚIUL B · SCRIE CODUL

Normalizarea unui nume

SCRIERE COD

Se citește de la tastatură un nume scris pe un singur rând, cu litere mari și mici. Afișează pe primul rând numele scris integral cu litere mari, iar pe al doilea rând de câte ori apare litera A în numele scris cu litere mari. Ex.: Pentru Popescu Ana se afișează POPESCU ANA pe primul rând și 2 pe al doilea.

HINT **Atenție.** *Numără literele după transformarea la majuscule, altfel un a mic nu va fi găsit. Salvează rezultatul lui upper() într-o variabilă — vei avea nevoie de el de două ori.*

HINT **Observație.** *Numele poate conține spații; ele nu se numără.*

SOLUȚIA TA · PYTHON

REMEDIERE · SCRIE CODUL

Un singur cuvânt, cu majuscule

SCRIERE COD

Se citește un cuvânt scris pe un singur rând, fără spații. Afișează pe primul rând cuvântul scris integral cu litere mari, iar pe al doilea rând de câte ori apare litera A în el. Ex.: Pentru cuvântul ana se afișează ANA pe primul rând și 2 pe al doilea.

HINT **Observație.** *Este varianta simplificată a exercițiului de mai sus: un singur cuvânt, fără spații de numărat. Dacă acolo te-ai împotmolit, începe de aici.*

HINT **Atenție.** *Metoda upper() nu schimbă variabila; îți dă un șir nou. Scrie cuvant = cuvant.upper(), apoi folosește cuvant de două ori: o dată pentru afișare, o dată pentru numărare.*

SOLUȚIA TA · PYTHON

PERFORMANȚĂ · SCRIE CODUL

A k-a apariție a unui caracter — exercițiu de performanță

SCRIERE COD

Se citesc trei rânduri: pe primul un text s fără spații, pe al doilea un caracter c, pe al treilea un număr natural k mai mare decât 0. Afișează poziția celei de-a k-a apariții a lui c în s, pozițiile fiind numerotate de la 0, sau -1 dacă c apare de mai puțin de k ori. Ex.: Pentru s = informatica, c = a și k = 2: litera a apare pe pozițiile 6 și 10, deci a doua apariție e pe poziția 10 și se afișează 10.

HINT Observație. *Metoda find acceptă un al doilea argument: s.find(c, start) caută începând de la poziția start. Folosind-o repetat, muți căutarea tot mai la dreapta.*

HINT Atenție. *Când find întoarce -1, nu mai există apariții și trebuie să oprești căutarea; dacă ai găsit mai puțin de k apariții, răspunsul final este -1, nu ultima poziție găsită.*

SOLUȚIA TA · PYTHON

4 SECȚIUNEA 4 · SPLIT(): O METODĂ CARE ÎNTOARCE UN OBIECT DE ALTĂ CLASĂ

Din text în listă, cu o singură *metodă*

`split()` este metoda care taie un text în bucăți și le pune într-o listă. E interesantă pentru că e o metodă a clasei `str`, dar rezultatul ei aparține clasei `list` — dovada că obiectele lucrează împreună.

Apelată fără argument, `split()` taie textul după spații și întoarce lista cuvintelor. Apelată cu un argument, taie după caracterul dat. Rezultatul e o listă de **șiruri**, chiar dacă bucățile arată ca niște numere — dacă vrei să le aduni, trebuie să le treci prin `int()`.

```
linie = "8 9 10 7"
note = linie.split()
print(note)
print(type(note))
print(len(note))
print("1,2,3".split(","))
```

Programul afișează, pe rând: `['8', '9', '10', '7']`, `<class 'list'>`, `4` și `['1', '2', '3']`.

Odată obținută lista, o parcurgi exact ca pe orice listă învățată la L17, cu indice și `len`:

```
suma = 0
i = 0
while i < len(note):
    suma = suma + int(note[i])
    i = i + 1
print(suma)
```

METODA CARE SCHIMBĂ CLASA

`split()` pornește de la un obiect `str` și îți întoarce un obiect `list`. Așa se leagă clasele între ele: rezultatul unei metode poate fi obiectul altei clase, cu alte metode disponibile mai departe.

ATENȚIE

Elementele listei întoarse sunt **text**, nu numere. `note[0] + note[1]` dă `"89"`, nu `17`. Conversia cu `int()` nu e opțională.

UNDE SE FOLOSEȘTE ÎN VIAȚA REALĂ

Orice fișier exportat din Excel (`.csv`) e citit exact așa: fiecare rând e un text, tăiat cu `split(",")` în celule. Catalogul electronic desface „Popescu Ana Maria” în nume și prenume cu `split()`. Iar când scrii un mesaj cu diacritice și hashtag-uri, aplicația îl taie în cuvinte tot cu `split()` ca să vadă care încep cu `#`.

A

EXERCIȚIUL A · GRILĂ

Ce întoarce split()

GRILĂ

Încercuiește răspunsul corect la fiecare întrebare.

1. Ce clasă are rezultatul expresiei "8 9 10".split()?

   - `int, pentru că toate bucățile din exemplu sunt numere`
   - `str, pentru că o metodă a clasei str întoarce tot un str`
   - `TextIOWrapper, pentru că textul ar putea proveni dintr-un fișier`
   - `list, pentru că metoda întoarce lista bucăților găsite`

2. Ce afișează print("1,2,3".split(","))?

   - `'1,2,3' — șirul rămâne neschimbat, virgula fiind ignorată`
   - `[1, 2, 3] — trei numere întregi, gata de adunat`
   - `['1,2,3'] — o listă cu un singur element, șirul întreg`
   - `['1', '2', '3'] — trei șiruri de caractere, nu trei numere`

B

EXERCIȚIUL B · DRY-RUN

Adunăm notele de pe un rând

DRY-RUN

După split(), lista cuvinte are 4 elemente. Coloana i este dată; tu completezi elementul cuvinte[i] (valoare text) și valoarea variabilei suma după executarea pasului (valoare numerică). Ultimul rând arată starea în care bucla se oprește.

```
1  linie = "8 9 10 7"
2  cuvinte = linie.split()
3  suma = 0
4  i = 0
5  while i < len(cuvinte):
6      suma = suma + int(cuvinte[i])
7      i = i + 1
8  print(suma)
```

i	cuvinte[i]	suma după pas
0	8	8
1		
2		
3		
4 — bucla se oprește	—	34

HINT *Scrie mai întâi bucata de text de pe poziția i din listă, apoi adaugă valoarea ei la suma de pe rândul de deasupra. Atenție: al treilea element este 10, nu 1 și 0.*

C

EXERCIȚIUL C · SCRIE CODUL

Notele de pe un rând

SCRIERE COD

Se citește un singur rând care conține numere naturale separate prin câte un spațiu — notele unui elev. Afișează pe primul rând câte note sunt, iar pe al doilea rând suma lor. Ex.: Pentru rândul 8 9 10 7 se afișează 4 pe primul rând și 34 pe al doilea.

HINT **Atenție.** *Elementele întoarse de split() sunt șiruri; fără int() adunarea le va lipi unul de altul în loc să le însumeze.*

HINT **Observație.** *Rândul conține cel puțin o notă.*

SOLUȚIA TA · PYTHON

SECȚIUNEA 5 · CLASA DATE DIN BIBLIOTECA DATETIME

5 Datele calendaristice sunt și ele *obiecte*

A calcula câte zile sunt între 15 martie 2010 și 5 septembrie 2026 pare o problemă de aritmetică cu ani bisecți și luni de lungimi diferite. Cu clasa `date` din biblioteca `datetime` e o scădere.

Clasa `date` nu vine de la sine: se aduce în program cu `from datetime import date`. Instanțierea se face dând anul, luna și ziua, în ordinea asta: `d = date(2026, 9, 5)`.

Obiectul obținut are **atribute** (`d.year`, `d.month`, `d.day` — fără paranteze) și **metode** (`d.weekday()`, care întoarce ziua săptămânii ca număr, `0` pentru luni și `6` pentru duminică; `d.strftime("%d.%m.%Y")`, care întoarce data ca text în formatul cerut).

```
from datetime import date
d = date(2026, 9, 5)
print(d.year)
print(d.month)
print(d.weekday())
print(d.strftime("%d.%m.%Y"))
```

Programul afișează, pe rând: `2026`, `9`, `5` și `05.09.2026`.

Scăderea a două obiecte `date` întoarce un obiect al unei alte clase, `timedelta`, care are atributul `days`:

```
from datetime import date
dif = date(2026, 9, 5) - date(2010, 3, 15)
print(dif.days)
```

Programul afișează `6018`. Anii bisecți sunt numărați corect de bibliotecă — tu nu te ocupi de ei.

Există și drumul invers: `from datetime import timedelta` îți dă clasa cu care aduni zile la o dată. `date(2026, 9, 5) + timedelta(days=30)` este `2026-10-05`.

> **DATA CURENTĂ**

Biblioteca mai are clasa `datetime`, cu metoda `datetime.now()`, care întoarce data și ora din acest moment. E foarte utilă în aplicații reale, dar **nu se poate folosi în probleme cu teste**: dă alt rezultat la fiecare rulare, deci nicio ieșire așteptată nu ar fi corectă mâine. De aceea toate problemele de aici lucrează cu date citite de la tastatură.

ATENȚIE

Ordinea argumentelor este `date(an, luna, zi)`, nu cea în care scriem data în română. Iar `d.year` se scrie fără paranteze: `d.year()` dă eroare, pentru că `year` e atribut, nu metodă.

UNDE SE FOLOSEȘTE ÎN VIAȚA REALĂ

Aplicația care îți spune „mai sunt 43 de zile până la vacanță” face exact scăderea de mai sus. Un magazin online calculează termenul de garanție adunând `timedelta(days=730)` la data cumpărării. Iar catalogul electronic verifică dacă ai împlinit 18 ani comparând data nașterii cu data de azi — fără să numere el lunile și anii bisecți.

A

EXERCIȚIUL A · GRILĂ

Obiecte date

GRILĂ

Încercuiește răspunsul corect la fiecare întrebare.

1. Ce clasă are rezultatul scăderii a două obiecte date?

   - `date, adică data aflată exact la mijlocul intervalului`
   - `int, adică direct numărul de zile dintre cele două date`
   - `timedelta, un obiect care are atributul days`
   - `str, adică diferența scrisă în formatul zz.ll.aaaa`

2. În expresia d.year, unde d este un obiect date, ce este year?

   - `O funcție a bibliotecii, independentă de obiectul pe care îl avem`
   - `O metodă a obiectului, care trebuie apelată obligatoriu cu d.year()`
   - `O variabilă globală creată automat la importul bibliotecii`
   - `Un atribut al obiectului — o dată memorată în el, citită fără paranteze`

B

EXERCIȚIUL B · ORDONARE

Pașii calculului cu date

ORDONARE

Numerotează pașii de la 1 la 5, în ordinea în care se scriu într-un program care calculează câte zile au trecut între două date citite de la tastatură. Pașii îți apar amestecați.

- [ ] Instanțiem câte un obiect date pentru fiecare dată citită.
- [ ] Importăm clasa date din biblioteca datetime.
- [ ] Afișăm atributul days al obiectului obținut.
- [ ] Citim ziua, luna și anul pentru fiecare dintre cele două date.
- [ ] Scădem cele două obiecte și obținem un obiect timedelta.

C

EXERCIȚIUL C · SCRIE CODUL

Câte zile au trecut

SCRIERE COD

Se citesc șase numere naturale, câte unul pe rând, în ordinea: ziua, luna și anul unei prime date, apoi ziua, luna și anul unei a doua date, garantat ulterioară sau egală cu prima. Afișează câte zile au trecut între cele două date. Ex.: Pentru 1 1 2010 și 1 1 2011 se afișează 365.

HINT **Observație.** *Anul 2010 nu e bisect; dacă intervalul ar conține un 29 februarie, biblioteca l-ar număra automat.*

HINT **Atenție.** *Constructorul primește argumentele în ordinea date(an, luna, zi), adică invers față de ordinea în care le citești. Rezultatul scăderii nu este un număr — este un obiect timedelta, din care trebuie să iei atributul days.*

SOLUȚIA TA · PYTHON

PERFORMANȚĂ · SCRIE CODUL

Câți ani împliniți avea — exercițiu de performanță

SCRIERE COD

Se citesc șase numere naturale, câte unul pe rând, în ordinea: ziua, luna și anul nașterii unei persoane, apoi ziua, luna și anul unei date de referință, garantat ulterioară nașterii. Afișează câți ani împliniți avea persoana la data de referință. Ex.: Pentru nașterea 5 9 2010 și data de referință 5 9 2026 se afișează 16; pentru referința 4 9 2026 se afișează 15, pentru că ziua de naștere nu venise încă.

HINT **Observație.** *Aici nu ajunge scăderea a două obiecte date: numărul de zile împărțit la 365 dă rezultate greșite din cauza anilor bisecți. Compară atributele: mai întâi lunile, iar dacă lunile sunt egale, zilele.*

HINT **Atenție.** *Diferența anilor se micșorează cu 1 dacă la data de referință ziua de naștere nu a trecut încă. Un elev născut pe 29 februarie 2008 împlinește 17 ani, nu 18, la 28 februarie 2026.*

SOLUȚIA TA · PYTHON

SECȚIUNEA 6 · BIBLIOTECI ȘI INTERIORUL UNEI CLASE

6 De unde vin clasele și cum arată ele pe *dinăuntru*

Clasele nu apar din senin: stau grupate în biblioteci, aduse în program cu `import`. Iar la final, ca lectură suplimentară, ridicăm capacul și ne uităm o dată la cum arată o clasă scrisă de mână — ca să înțelegi ce ai folosit toată lecția.

O **bibliotecă** (numită și modul) este un ansamblu de clase și funcții reutilizabile, grupate sub un nume și aduse în program cu `import`. Ai folosit deja `math` (funcții matematice, L21) și acum `datetime` (clase pentru date calendaristice). La lecția următoare vei folosi `tkinter`, biblioteca de clase pentru interfețe grafice. Bibliotecile există ca să nu rescrii de fiecare dată ce a scris deja altcineva mai bine.

> **EXTINDERE · PESTE PROGRAMĂ**
>
> Ce urmează, până la finalul secțiunii, **nu se evaluează** și nu este cerut de programa clasei a IX-a: la această clasă se studiază doar folosirea claselor gata făcute. Citește paragrafele următoare ca să vezi cum arată, pe dinăuntru, clasele pe care le-ai folosit toată lecția — atât.

O clasă scrisă de mână arată așa:

```
1  class Elev:
2      def __init__(self, nume, nota):
3          self.nume = nume
4          self.nota = nota
5
6      def promovat(self):
7          return self.nota >= 5
```

Trei lucruri de citit în ea. `class Elev:` deschide tiparul. `__init__` este **constructorul**: se execută automat la instanțiere și așază atributele în obiectul nou. `self` este obiectul care tocmai se construiește sau pe care se apelează metoda — Python îl trimite singur, tu nu îl scrii la apel.

Folosirea e cea cu care te-ai obișnuit deja:

```
1  e = Elev("Ana", 9)
2  print(e.nume)
3  print(e.promovat())
```

Programul afișează `Ana`, apoi `True`.

> **CONSTRUCTORUL __INIT__**

Când scrii `Elev("Ana", 9)`, Python creează un obiect gol și îi dă imediat `__init__`-ului, ca prim argument (`self`), împreună cu `"Ana"` și `9`. Constructorul le așază în `self.nume` și `self.nota` — de aceea, după instanțiere, obiectul „știe” cine e.

ATENȚIE

Fiecare metodă a clasei primește `self` ca prim parametru, dar la apel nu îl scrii: `e.promovat()`, nu `e.promovat(e)`. Iar la instanțiere nu se scrie `new` — în Python se scrie direct numele clasei, ca și cum ar fi o funcție.

AȘA ARATĂ LEGĂTURA CU TOT RESTUL LECȚIEI

Exact așa sunt scrise, undeva în Python, și `list`, și `str`, și `TextIOWrapper`, și `date`. Diferența e că pe acelea nu le-a scris nimeni din clasa ta — sunt **predefinite**, gata făcute, și de asta poți scrie `note.append(9)` fără să definești nimic.

UNDE SE FOLOSEȘTE ÎN VIAȚA REALĂ

Aplicația de pe telefon care îți numără pașii are undeva o clasă `Antrenament` cu atributele „durată” și „distanță” și cu metoda „calculează caloriile”. Un magazin online are o clasă `Produs` cu atributele „preț” și „stoc” și cu metoda „aplică reducerea”, ca reducerea să se calculeze la fel peste tot. Iar orice joc are o clasă pentru dușmani, instanțiată de o sută de ori: o singură descriere, o sută de obiecte diferite.

A

EXERCIȚIUL A · GRILĂ

Biblioteci și clase

GRILĂ

Încercuiește răspunsul corect la fiecare întrebare.

1. Ce este o bibliotecă în Python?

   - `Un obiect special care conține toate datele programului curent`
   - `O listă de variabile globale disponibile în orice program scris`
   - `Un fișier text în care programul își salvează rezultatele finale`
   - `Un ansamblu de clase și funcții reutilizabile, grupate sub un nume`

2. Ce face instrucțiunea from datetime import date?

   - `Transformă toate variabilele programului în obiecte de clasă date`
   - `Afișează lista tuturor claselor pe care le conține biblioteca datetime`
   - `Aduce în program clasa date din biblioteca datetime, ca să o putem instanția`
   - `Creează un obiect nou de clasă date, care conține data curentă`

3. Ce se întâmplă când scriem d = date(2026, 9, 5)?

- `Se produce o eroare, pentru că numele clasei trebuie scris cu literă mare`
- `Se instanțiază clasa date: se construiește un obiect nou, cu datele lui`
- `Se apelează o metodă a obiectului d, care întoarce ziua săptămânii`
- `Se creează o listă cu trei elemente: anul, luna și ziua`

B

EXERCIȚIUL B · SCRIE CODUL

Câte date cad în weekend

SCRIERE COD

Se citește un număr natural n, iar pe fiecare dintre următoarele n rânduri se află trei numere naturale separate prin câte un spațiu: ziua, luna și anul unei date calendaristice valide. Afișează câte dintre aceste date cad în weekend, adică sâmbătă sau duminică. Ex.: Pentru n = 3 și rândurile 5 9 2026, 7 9 2026, 6 9 2026 se afișează 2: 5 septembrie cade sâmbătă, 6 septembrie duminică, iar 7 septembrie luni.

HINT Observație. *Metoda weekday() numerotează zilele de la 0 (luni) la 6 (duminică), deci weekendul înseamnă exact valorile 5 și 6. Clasa date se aduce din bibliotecă cu from datetime import date.*

HINT Atenție. *Rândul citit este un singur text; îl tai cu split() în trei bucăți, iar fiecare bucată trebuie trecută prin int() înainte de a ajunge în constructor. Ordinea argumentelor este date(an, luna, zi), invers față de ordinea în care sunt scrise pe rând.*

SOLUȚIA TA · PYTHON

SECȚIUNEA 7 · COD GHIDAT

7 Scrie cod care *trece testele*

1

PROBLEMA 7.1 · COD GHIDAT

Inițialele

SCRIERE COD

Se citește un nume complet, scris pe un singur rând, cu cuvintele separate prin câte un spațiu. Afișează inițialele numelui, cu litere mari, scrise una după alta, fără spații între ele. Ex.: Pentru popescu ana maria se afișează PAM.

HINT **Observație.** *Numele conține cel puțin un cuvânt.*

HINT **Atenție.** *Prima literă a cuvântului de pe poziția i din listă se scrie parti[i][0]: primul indice alege cuvântul, al doilea alege litera din el.*

SOLUȚIA TA · PYTHON

2

PROBLEMA 7.2 · COD GHIDAT

Cuvintele care conțin un caracter

SCRIERE COD

Se citesc două rânduri: pe primul un text cu cuvintele separate prin câte un spațiu, pe al doilea un singur caracter c. Afișează câte cuvinte din text conțin caracterul c. Ex.: Pentru textul ana are mere și caracterul a, cuvintele ana și are conțin litera a, iar mere nu; se afișează 2.

HINT **Observație.** *Un cuvânt se numără o singură dată, indiferent de câte ori apare caracterul în el.*

HINT **Atenție.** *count întoarce numărul de apariții, nu True/False; compară-l cu 0 ca să afli dacă litera apare măcar o dată.*

SOLUȚIA TA · PYTHON

3

PROBLEMA 7.3 · COD GHIDAT

În ce zi a săptămânii cade

SCRIERE COD

Se citesc trei numere naturale, câte unul pe rând: ziua, luna și anul unei date calendaristice valide. Afișează numărul zilei din săptămână în care cade acea dată, unde 0 înseamnă luni, 1 marți, și așa mai departe până la 6, care înseamnă duminică. Ex.: Pentru 5, 9, 2026 — adică 5 septembrie 2026 — se afișează 5, pentru că data cade sâmbătă.

HINT Observație. *Numerotarea 0 = luni … 6 = duminică este convenția metodei weekday() din biblioteca datetime; alte limbaje pornesc de la duminică.*

HINT Atenție. *Ordinea argumentelor constructorului este date(an, luna, zi), invers față de ordinea în care se citesc datele.*

SOLUȚIA TA · PYTHON

PROBLEMA 7.4 · REPARĂ CODUL

4 Numele care refuză să devină majuscul

SCRIERE COD

Codul de mai jos ar trebui să citească un nume și să-l afișeze scris cu litere mari. Rulat cu intrarea ana, el afișează însă ana, iar cu intrarea Ana Maria afișează Ana Maria — adică exact ce s-a citit. Scrie versiunea corectă a programului, care afișează numele cu litere mari.

Programul de corectat:

```
1 nume = input()
2 nume.upper()
3 print(nume)
```

Ex.: Pentru intrarea ana, programul corect afișează ANA.

HINT **Observație.** *Greșeala verifică o afirmație din teoria secțiunii S3: metodele clasei str întorc un șir nou, nu îl modifică pe cel original.*

HINT **Atenție.** *Rândul nume.upper() nu e greșit din punct de vedere sintactic — el chiar calculează șirul cu majuscule. Problema e că nimeni nu păstrează rezultatul, așa că acesta se pierde imediat.*

PROGRAMUL CORECTAT · PYTHON

PROBLEMA 7.5 · PROVOCARE

Cenzura

SCRIERE COD

Se citesc două rânduri: pe primul un text, pe al doilea un cuvânt interzis. Afișează pe primul rând textul în care fiecare apariție a cuvântului interzis este înlocuită cu tot atâtea caractere * câte litere are cuvântul, iar pe al doilea rând de câte ori s-a făcut înlocuirea. Ex.: Pentru textul azi mergem la mare și cuvântul mare se afișează azi mergem la **** pe primul rând și 1 pe al doilea.

HINT **Observație.** *Înlocuirea se face oriunde apare secvența de litere, chiar dacă e în interiorul altui cuvânt: pentru textul ana are mere și cuvântul re, rezultatul este ana a** me**.*

HINT **Atenție.** *Numără aparițiile înainte de înlocuire; după replace cuvântul nu mai există în text și count ar da 0. Șirul de asteriscuri se construiește cu "*" * len(cuvant).*

SOLUȚIA TA · PYTHON