test-e.ro

CAIET ELEV

Lecția 33

Prezentarea proiectelor. Evaluare finală. Bilanț al anului școlar

Ultima lecție a anului nu mai adaugă nicio unealtă: elevul are deja trusa completă — liste, subprograme, fișiere text, șiruri de caractere (split, upper, find), cele trei metode de sortare, interfața Tkinter — și arată că știe să o folosească într-o aplicație adevărată și să explice ce a construit.

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: Nu apare niciun construct nou: lecția e un plafon, nu o temă (tot ce e permis până la L30 — liste, def/return, split/upper/find, open/with, tkinter). Permis nu înseamnă obligatoriu. Fără diacritice în cod, fără atribuiri simultane (interschimbări prin aux), fără sort()/sorted() acolo unde sortarea e chiar exercițiul, fără global, fără clase definite de elev; listele se citesc cu n pe prima linie și câte o valoare pe linie. Codul Tkinter apare doar ca bloc static: Pyodide nu poate deschide ferestre.

CUPRINS

1 Trusa anului: tot ce știi să folosești .....

2 Mediu: arhitectura aplicației, explicată modul cu modul

3 Digitalizare: testele pe care le arăți la demonstrație

4 Sănătate: cum răspunzi la întrebări despre algoritm

5 Patrimoniu: interfața pe care o arăți și logica pe care o testezi

6 Bilanț: argumentezi, prezinți, te autoevaluezi

7 Cod ghidat · cinci probleme .....

SECȚIUNEA 1 · TRUSA ANULUI

1 Trusa anului: tot ce știi să *folosești*

Astăzi nu înveți nimic nou. Verifici, pe rând, că fiecare unealtă din trusa ta — lista, subprogramul, fișierul, șirul de caractere, sortarea — îți stă la îndemână înainte să-ți prezinți proiectul.

Pe parcursul anului ai strâns o trusă întreagă. O **listă** păstrează oricâte valori sub un singur nume: `v.append(x)` adaugă la final, `len(v)` spune câte sunt, `v[i]` dă elementul de pe poziția `i` (numerotarea începe de la 0). Un **subprogram** (`def`) primește date prin parametri și trimite înapoi un rezultat cu `return`. Un **fișier text** se deschide cu `open(nume, mod)`, iar construcția `with` îl închide automat. Un **șir de caractere** se desparte în bucăți cu `split(separator)` și se compară fără grija literelor mari și mici după `upper()`. Pentru **sortare** ai trei metode: selecția minimului, metoda bulelor și lista de frecvențe.

Citirea unei liste o facem mereu la fel: întâi `n`, apoi `n` valori, câte una pe linie.

Aceeași idee în Python (cu sintaxă reală):

```
n = int(input())
v = []
for i in range(n):
    v.append(int(input()))
```

DE REȚINUT

Aproape orice aplicație din proiectul final are trei etaje: **citirea datelor** (tastatură sau fișier) → **prelucrarea** (subprograme care primesc lista și întorc un rezultat) → **afișarea** (consolă sau fereastră Tkinter). Dacă poți spune într-o frază ce face fiecare etaj, poți prezenta proiectul.

A

EXERCIȚIUL A · GRILĂ

Ce știu deja

GRILĂ

Încercuiește răspunsul corect la fiecare întrebare.

1. Ce afișează print(len([4, 0, 7]))? `3` `11` `0` `2`

2. Ce valoare are "Peles;Sinaia;1883".split(";")?

   `['Peles', 'Sinaia', 1883]` `['Peles;Sinaia;1883']`
   `['Peles', ';', 'Sinaia', ';', '1883']` `['Peles', 'Sinaia', '1883']`

3. Câte comparații între elemente face sortarea prin selecția minimului pe o listă cu 5 elemente?

20 | 10 | 25 | 4

4. Ce face modul 'a' din open('jurnal.txt', 'a')?

- Creează automat o copie de rezervă a fișierului înainte de a-l modifica
- Deschide fișierul doar pentru citire, iar orice write produce o eroare
- Șterge tot conținutul vechi al fișierului și scrie de la prima linie
- Adaugă ce scrii la sfârșitul fișierului, fără să șteargă ce era deja

B

EXERCIȚIUL B · DRY-RUN

Câte valori trec de prag?

DRY-RUN

Urmărește bucla pas cu pas. Coloanele i și v[i] sunt date; tu completezi dacă valoarea trece de prag (scrie DA sau NU) și cât valorează contorul cnt după pasul respectiv. Atenție la semnul comparației: >, nu >=.

```
1 v = [12, 55, 0, 61, 50]
2 prag = 50
3 cnt = 0
4 for i in range(len(v)):
5     if v[i] > prag:
6         cnt = cnt + 1
7 print(cnt)
```

pas	i	v[i]	v[i] > prag (completează: DA/NU)	cnt
înainte de buclă	—	—	—	0
1	0	12		
2	1	55		
3	2	0		
4	3	61		
5	4	50		
bucla se oprește (range(5) s-a epuizat)	—	—	—	2

HINT *Atenție la semnul comparației: >, nu >=. Contorul cnt crește doar pe rândurile cu DA.*

SPRIJIN · SCRIE CODUL

Totalul unei liste

SCRIERE COD

Se citește un număr natural n, apoi n numere naturale, câte unul pe linie. Memorează numerele într-o listă, apoi afișează suma lor. Ex.: pentru n = 3 și valorile 10, 0, 25 se afișează 35.

HINT *Poate fi totalul litrilor de apă consumați într-o săptămână sau al minutelor de mișcare — orice jurnal începe cu o astfel de citire.*

SOLUȚIA TA · PYTHON

SPRIJIN · SCRIE CODUL

Câmpurile unei linii de catalog

SCRIERE COD

Se citește o linie de forma Nume;An, cu numele unui obiectiv turistic și anul construirii lui, apoi, pe a doua linie, anul curent. Afișează numele obiectivului pe primul rând și vechimea lui (în ani) pe al doilea rând. Ex.: pentru liniile Castelul Peles;1883 și 2027 se afișează Castelul Peles, iar pe rândul următor 144.

HINT **Atenție.** *După split(";"), anul este tot un șir de caractere ('1883'). Trebuie convertit cu int() înainte de scădere.*

SOLUȚIA TA · PYTHON

SECȚIUNEA 2 · MEDIU

2 Mediu: arhitectura aplicației, explicată modul cu *modul*

Primul lucru pe care îl arăți la prezentare nu e codul, ci harta lui: ce module are aplicația și ce face fiecare. Exersăm pe o aplicație de mediu care urmărește poluarea aerului.

Juriul nu are timp să citească tot programul tău. Are nevoie de o hartă: câte module are aplicația, ce primește fiecare și ce dă înapoi. Regula practică este **„un modul = o frază”**. Dacă pentru un modul îți trebuie trei fraze, probabil face prea multe lucruri și merită împărțit.

Să luăm o aplicație de mediu: jurnalul poluării aerului dintr-un oraș. Stațiile de măsurare raportează zilnic concentrația de particule PM10, în micrograme pe metru cub (µg/m³). Aplicația are patru module:

- **citirea** — preia valorile zilnice (de la tastatură sau dintr-un fișier) și le pune într-o listă;
- **prelucrarea** — subprogramul `zile_depasire(v, prag)` numără zilele cu valoare peste prag;
- **afișarea** — scrie numărul de zile și verdictul;
- **interfața** — un buton Tkinter care, la apăsare, cheamă prelucrarea și afișează rezultatul într-o fereastră.

Modulul de prelucrare, scris ca subprogram:

```
1  def zile_depasire(v, prag):
2      cnt = 0
3      for i in range(len(v)):
4          if v[i] > prag:
5              cnt = cnt + 1
6      return cnt
```

VALOAREA-LIMITĂ PM10 (DIRECTIVA 2008/50/CE)

Directiva europeană privind calitatea aerului fixează pentru PM10 o valoare-limită zilnică de 50 µg/m³, care nu poate fi depășită în mai mult de **35 de zile** într-un an calendaristic. Deci aplicația are două praguri: 50 pentru fiecare zi și 35 pentru numărul de zile.

ATENȚIE LA „PESTE”

Limita e depășită când valoarea e **mai mare decât** 50. O zi cu exact 50 µg/m³ este încă în limită, deci comparația corectă e `v[i] > prag`, nu `v[i] >= prag`. La prezentare, un astfel de detaliu arată că ai citit regula, nu doar ai scris cod.

UNDE SE FOLOSEȘTE ÎN VIAȚA REALĂ

> Rețeaua Națională de Monitorizare a Calității Aerului publică pe calitateaer.ro măsurătorile stațiilor, inclusiv ale celor din Ploiești, iar autoritățile de mediu numără exact zilele de depășire ca să decidă dacă un oraș are nevoie de un plan de calitate a aerului. Aplicațiile meteo de pe telefon afișează același tip de indicator, iar școlile îl pot folosi ca să decidă dacă ora de sport se ține afară.

A

EXERCIȚIUL A · ASOCIERE

Fiecare modul la locul lui

ASOCIERE

Leagă fiecare bucată de cod din stânga de responsabilitatea ei din dreapta. Scrie în căsuța din fața fiecărui element din stânga litera variantei potrivite. Fiecare literă se folosește o singură dată.

☐	v.append(int(input())) într-un for
☐	def zile_depasire(v, prag):
☐	print(k) urmat de if k > 35:
☐	Button(fereastra, text="Analizeaza", command=analiza)

A `preia valorile zilnice și le pune în listă`

B `numără zilele în care poluarea a trecut de prag`

C `leagă apăsarea unui buton de un subprogram`

D `comunică utilizatorului rezultatul și verdictul`

B

EXERCIȚIUL B · SCRIE CODUL

Zilele cu depășire PM10

SCRIERE COD

Se citește un număr natural n (n ≥ 1), apoi n valori zilnice ale concentrației PM10, în µg/m³, câte una pe linie. Scrie subprogramul zile_depasire(v, prag), care întoarce numărul de valori din lista v strict mai mari decât prag. Folosește-l cu pragul 50 și afișează pe primul rând numărul de zile cu depășire. Pe al doilea rând afișează DEPASIRE dacă sunt mai mult de 35 de astfel de zile, sau IN LIMITA în caz contrar. Ex.: pentru n = 7 și valorile 12, 55, 0, 61, 50, 48, 73, zilele cu depășire sunt cele cu 55, 61 și 73. Se afișează 3, apoi IN LIMITA.

HINT **Atenție.** *Ziua cu exact 50 µg/m³ nu e depășire. La fel, exact 35 de zile de depășire înseamnă încă IN LIMITA: regula spune „mai mult de 35”.*

HINT **Curiozitate.** *Directiva 2008/50/CE a fost revizuită în 2024, cu praguri mai stricte care se aplică din 2030. Aplicația ta s-ar adapta schimbând un singur număr, fiindcă pragul e parametru, nu e scris în interiorul subprogramului.*

SOLUȚIA TA · PYTHON

SECȚIUNEA 3 · DIGITALIZARE

3 Digitalizare: testele pe care le arăți la *demonstrație*

O demonstrație live convinge doar dacă arăți și cazurile în care programul ar putea greși. Exersăm pe un catalog electronic care calculează media și nota finală a unui elev.

Regulamentul școlar (ROFUIP) fixează o regulă precisă: la fiecare disciplină se încheie **o singură medie anuală**, obținută prin **rotunjirea mediei aritmetice a notelor la cel mai apropiat întreg**; la exact 50 de sutimi, rotunjirea se face **în favoarea elevului** (în sus). Catalogul nostru electronic afișează, pe lângă notă, și media cu **două zecimale exacte** — se păstrează primele două zecimale, fără rotunjire, aceeași regulă (trunchierea) după care se calculează media generală anuală. Așadar aplicația face doi pași:

1. calculează **media aritmetică** a notelor, cu **două zecimale exacte**, pentru afișare;
2. calculează nota finală: media **rotunjită la cel mai apropiat întreg**, la 50 de sutimi în sus.

Pentru notele 9, 8, 8: media e 25 / 3 = 8,333…, adică **8,33**, iar nota finală e **8**. Pentru notele 8 și 9: media e **8,50**, iar nota e **9**.

MEDIA ÎN SUTIMI, FĂRĂ NUMERE CU VIRGULĂ

Ca să nu depinzi de rotunjirile calculatorului, lucrezi cu numere întregi. `m = s * 100 // n` este media exprimată în sutimi (pentru 8,33 → `m = 833`). Nota finală este `(m + 50) // 100`: adaugi 50 de sutimi și păstrezi partea întreagă. Pentru `m = 850` obții `900 // 100 = 9`, iar pentru `m = 849` obții `899 // 100 = 8`.

Aceeași idee în Python (cu sintaxă reală):

```
m = s * 100 // n
nota = (m + 50) // 100
```

CAPCANA ROUND()

Funcția `round()` din Python **nu** rotunjește „în favoarea elevului”. La exact jumătate, ea alege numărul **par** (rotunjirea bancherului): `round(8.5)` dă `8`, iar `round(9.5)` dă `10`. Un program care folosește `round()` greșește tocmai la mediile cu 50 de sutimi. Și la toate celelalte teste pare corect!

Ce teste arăți la demonstrație. Un set bun de teste are câte un caz din fiecare categorie: un caz obișnuit (9, 8, 8 → 8,33 și 8), un **caz-limită** (8, 9 → 8,50 și 9, adică exact la jumătate), un caz extrem (10, 10, 10 → 10,00 și 10), un caz cu zecimala a doua zero (media 8,08, care trebuie afișată cu `0` în față) și un caz cu multe note. Dacă toate testele tale sunt „obișnuite”, demonstrația nu dovedește nimic.

UNDE SE FOLOSEȘTE ÎN VIAȚA REALĂ

Cataloagele electronice folosite în școli calculează automat mediile după regulile din regulamentul școlar, iar o singură greșeală de rotunjire ar schimba mediile a mii de elevi. Aceeași problemă apare la casele de marcat (rotunjirea bonului la bani) și în aplicațiile bancare, unde se lucrează intern cu bani întregi (sutimi) exact ca să nu apară erori de rotunjire.

A

EXERCIȚIUL A · GRILĂ

Alege testul potrivit

GRILĂ

Încercuiește răspunsul corect la fiecare întrebare.

1. Un elev are notele 8 și 9. Ce notă finală trebuie să afișeze catalogul? `8.5` `9` `8` `8.50`

2. Un program folosește round(s / n) pentru nota finală. Care set de note îi dezvăluie greșeala?

   `7, 9, 10, 6, 8, 5, 9` `9, 8, 8` `8, 9` `10, 10`

3. Notele unui elev sunt 7, 7, 8. Care este media cu două zecimale exacte?

`7.333` `7.34` `7.30` `7.33`

B

EXERCIȚIUL B · DRY-RUN

Media în sutimi, pas cu pas

DRY-RUN

Urmărește programul instrucțiune cu instrucțiune. Instrucțiunea și variabila modificată sunt date; tu completezi valoarea pe care o are variabila după execuție.

```
1  note = [9, 10, 7, 8, 9]
2  s = 0
3  for i in range(len(note)):
4      s = s + note[i]
5  m = s * 100 // len(note)
6  nota = (m + 50) // 100
```

pas	instrucțiunea	variabila	valoarea
1	`s = 0`	`s`	`0`
2	`s = s + note[0]`	`s`	
3	`s = s + note[1]`	`s`	
4	`s = s + note[2]`	`s`	
5	`s = s + note[3]`	`s`	
6	`s = s + note[4]`	`s`	
7	`bucla se oprește: range(5) s-a epuizat`	—	—
8	`m = s * 100 // len(note)`	`m`	
9	`nota = (m + 50) // 100`	`nota`	

HINT *În buclă, s crește cu fiecare notă; apoi m = s * 100 // 5 dă media în sutimi, iar nota = (m + 50) // 100 dă nota finală.*

C

EXERCIȚIUL C · SCRIE CODUL

Media anuală în catalog

SCRIERE COD

Se citește un număr natural n (n ≥ 1), apoi n note întregi de la 1 la 10, câte una pe linie. Afișează pe primul rând media cu două zecimale exacte, în forma 8.33 (cu punct și exact două cifre după el). Pe al doilea rând afișează nota finală, rotunjită după regula școlii: la 50 de sutimi sau mai mult, în sus. Ex.: pentru notele 9, 8, 8 se afișează 8.33, apoi 8. Pentru notele 8, 9 se afișează 8.50, apoi 9.

HINT **Atenție.** *Nu folosi round(): pentru 8,50 ar da 8. Lucrează în sutimi: m = s * 100 // n.*

HINT **Atenție.** *Când a doua parte are o singură cifră (media 8,08 → m % 100 este 8), trebuie afișat 8.08, nu 8.8. Adaugă un 0 în fața zecimalelor mai mici decât 10.*

SOLUȚIA TA · PYTHON

4 SECȚIUNEA 4 · SĂNĂTATE

Sănătate: cum răspunzi la întrebări despre *algoritm*

După demonstrație vin întrebările: „De ce ai ales metoda asta? Cât de repede merge? Ce se întâmplă dacă…?”. Exersăm răspunsurile pe un jurnal de activitate fizică.

Organizația Mondială a Sănătății (OMS), în ghidul său privind activitatea fizică și comportamentul sedentar (2020), recomandă copiilor și adolescenților de 5–17 ani **în medie cel puțin 60 de minute pe zi** de activitate fizică moderată sau intensă. Cuvântul important este „în medie”: o zi cu 120 de minute poate compensa o zi cu 0. Un jurnal de mișcare pe 7 zile, cu minutele 30, 90, 0, 75, 60, 45, 120, are totalul 420 = 60 · 7. Recomandarea e atinsă, deși doar în 4 zile s-au adunat cel puțin 60 de minute.

La întrebări, juriul verifică de obicei trei lucruri:

1. **De ce funcționează?** Arăți unde e regula în cod: `total >= 60 * n` compară totalul cu 60 · n, adică media cu 60, fără să împarți.
2. **Cât de repede merge?** Numeri câte comparații face programul. O singură parcurgere a listei face câte o comparație pe zi, deci pentru 365 de zile face 365 de comparații: complexitate **O(n)**.
3. **De ce nu altfel?** Dacă întrebarea propune altă metodă (de exemplu sortarea), explici ce ar strica sau ce ar costa în plus.

> **CEA MAI LUNGĂ SERIE (SECVENȚA MAXIMĂ)**
>
> Ca să afli cea mai lungă serie de zile **consecutive** cu cel puțin 60 de minute, ții minte două numere: seria curentă și seria maximă. Parcurgi zilele în ordinea din jurnal. Dacă ziua e activă, seria curentă crește cu 1 și, dacă a depășit-o pe cea maximă, o înlocuiește. Dacă ziua nu e activă, seria curentă revine la 0. O singură parcurgere, deci O(n).

Aceeași idee în Python (cu sintaxă reală):

```
maxim = 0
serie = 0
for i in range(n):
    if v[i] >= 60:
        serie = serie + 1
        if serie > maxim:
            maxim = serie
    else:
        serie = 0
```

> **NU SORTA!**
>
> Sortarea ar pune toate zilele active una lângă alta și ar „inventa” o serie care nu a existat în calendar. Când ordinea datelor are un înțeles (zile, ore, citiri ale unui contor), algoritmul trebuie să o păstreze.

> **UNDE SE FOLOSEȘTE ÎN VIAȚA REALĂ**

> Aplicațiile de fitness și ceasurile inteligente afișează exact „seria de zile active” ca să te motiveze să nu o întrerupi. Aplicațiile de învățat limbi străine numără la fel zilele consecutive de exercițiu. Medicii de familie folosesc jurnale de activitate ca să vadă dacă un pacient respectă recomandarea săptămânală, nu doar dacă a avut o zi bună.

A

EXERCIȚIUL A · GRILĂ

Exersează răspunsurile la întrebări

GRILĂ

Încercuiește răspunsul corect la fiecare întrebare.

1.

Un coleg întreabă: „De ce nu sortezi mai întâi minutele, ca să găsești mai ușor cea mai lungă serie?”. Care este cel mai bun răspuns?

- `Sortarea strică ordinea zilelor, iar o serie înseamnă zile consecutive în calendar`
- `Sortarea cu listă de frecvențe nu merge pentru valori mai mari de 60 de minute`
- `Sortarea ar fi mai rapidă, dar ar trebui să scriu în plus metoda bulelor de mână`
- `Seria se poate afla și după sortare; ar dura doar puțin mai mult timp de execuție`

2. Câte comparații v[i] >= 60 face programul care numără zilele active, pentru un jurnal de 365 de zile?

`730` `60` `133225` `365`

3. Ce ordin de complexitate are algoritmul pentru cea mai lungă serie de zile active?

`O(1)` `O(n²)` `O(log n)` `O(n)`

B

EXERCIȚIUL B · ORDONARE

Pașii algoritmului „cea mai lungă serie”

ORDONARE

Numerotează pașii de la 1 la 5, în ordinea corectă. Pașii îți apar amestecați.

- [ ] Pornești cu seria curentă și seria maximă egale cu 0
- [ ] Parcurgi zilele în ordinea în care apar în jurnal
- [ ] Altfel, seria curentă revine la 0
- [ ] Dacă ziua are cel puțin 60 de minute, crești seria curentă și, dacă a depășit maximul, actualizezi maximul
- [ ] După ultima zi, afișezi seria maximă

C

EXERCIȚIUL C · SCRIE CODUL

Recomandarea OMS: zile active și media săptămânii

SCRIERE COD

Se citește un număr natural n (n ≥ 1), apoi n numere naturale, câte unul pe linie: minutele de activitate fizică din fiecare zi. Afișează pe primul rând numărul de zile cu cel puțin 60 de minute. Pe al doilea rând afișează RECOMANDARE ATINSA dacă media zilnică este cel puțin 60 de minute, sau SUB RECOMANDARE în caz contrar. Ex.: pentru n = 7 și minutele 30, 90, 0, 75, 60, 45, 120, zilele cu cel puțin 60 de minute sunt 4 (90, 75, 60, 120), iar totalul 420 înseamnă exact 60 pe zi în medie. Se afișează 4, apoi RECOMANDARE ATINSA.

HINT *Poți compara media cu 60 fără împărțire: media ≥ 60 înseamnă total ≥ 60 · n.*

HINT **Atenție.** *Aici „cel puțin 60” înseamnă >= 60. O zi cu exact 60 de minute se numără.*

SOLUȚIA TA · PYTHON

★

PERFORMANȚĂ · SCRIE CODUL

Cea mai lungă serie de zile active

SCRIERE COD

Se citește un număr natural n (n ≥ 1), apoi n numere naturale, câte unul pe linie: minutele de mișcare din fiecare zi. O zi este activă dacă are cel puțin 60 de minute. Afișează pe primul rând lungimea celei mai lungi serii de zile active consecutive. Dacă există cel puțin o zi activă, afișează pe al doilea rând numărul zilei în care începe această serie. Zilele se numerotează de la 1, iar dacă sunt mai

multe serii de lungime maximă, se alege prima. Dacă nu există nicio zi activă, se afișează doar 0. Ex.: pentru n = 7 și minutele 30, 90, 0, 75, 60, 45, 120, seriile active sunt ziua 2 (lungime 1), zilele 4–5 (lungime 2) și ziua 7 (lungime 1). Se afișează 2, apoi 4.

HINT **Atenție.** *Lista are indici de la 0, iar zilele se numără de la 1. Dacă seria de lungime serie se termină la indicele i, ea începe la indicele i - serie + 1, adică în ziua i - serie + 2.*

HINT **Curiozitate.** *Problema e o generalizare a numărării: în loc de „câte zile active?”, întrebi „câte zile active una după alta?”. Aceeași schemă se folosește pentru cea mai lungă secvență de temperaturi în creștere sau de zile fără ploaie.*

SOLUȚIA TA · PYTHON

SECȚIUNEA 5 · PATRIMONIU

5 Patrimoniu: interfața pe care o arăți și logica pe care o *testezi*

Fereastra e partea pe care o vede juriul, dar punctele vin din ce se întâmplă în spatele ei. Exersăm pe un catalog de obiective turistice din Prahova, cu logica separată de interfață.

Un catalog de patrimoniu păstrează fiecare obiectiv pe o linie, cu câmpurile separate prin `;`: `Castelul Peles;Sinaia;1883`. Metoda `split(";")` desparte linia în trei bucăți — nume, oraș, an — iar anul se transformă în număr cu `int()`. Pentru obiectivele dintr-un oraș, ordonate cronologic,

aplici selecția minimului pe lista anilor. Ca numele să rămână alături de anul lui, interschimbi **în același timp** și elementele din lista numelor.

Aplicația arătată la prezentare are o fereastră Tkinter. Totuși, subprogramul care filtrează și ordonează obiectivele **nu conține nicio instrucțiune Tkinter**. El primește date și întoarce un rezultat, deci poate fi testat singur, fără fereastră. Fereastra doar citește ce a scris utilizatorul (`get()`) și cheamă subprogramul la apăsarea butonului (`command=`).

Codul complet al aplicației (de copiat și rulat în IDLE sau Thonny; în pagină nu rulează, pentru că Pyodide nu poate deschide ferestre). Partea de logică nu conține nicio instrucțiune Tkinter, iar rezultatul ei este un text gata de afișat:

```
import tkinter as tk
from tkinter import messagebox

# --- logica: fara nicio instructiune Tkinter ---
nume = ["Manastirea Sinaia", "Castelul Peles", "Muzeul Ceasului"]
orase = ["Sinaia", "Sinaia", "Ploiesti"]
ani = [1695, 1883, 1963]

def obiective_din(oras):
    rezultat = ""
    for i in range(len(nume)):
        if orase[i].upper() == oras.upper():
            rezultat = rezultat + str(ani[i]) + " " + nume[i] + "\n"
    if rezultat == "":
        rezultat = "NICIUNUL"
    return rezultat

# --- interfata: doar preia textul si afiseaza rezultatul ---
def cauta():
    oras = caseta.get()
    rezultat = obiective_din(oras)
    messagebox.showinfo("Rezultat", rezultat)

fereastra = tk.Tk()
fereastra.title("Patrimoniul Prahovei")
tk.Label(fereastra, text="Orasul:").pack()
caseta = tk.Entry(fereastra)
caseta.pack()
tk.Button(fereastra, text="Cauta", command=cauta).pack()
fereastra.mainloop()
```

Listele sunt deja în ordine cronologică; dacă le citești dintr-un fișier, le ordonezi întâi prin selecția minimului, exact ca în problema de mai jos.

DE REȚINUT

Separarea logicii de interfață are un nume: **logica nu știe că există o fereastră**. Tot ce se poate calcula (filtrare, sortare, numărare) stă în subprograme fără Tkinter, testate cu date fixe. Tkinter doar preia datele și afișează rezultatul. La prezentare poți spune: „butonul cheamă funcția `obiective_din`, pe care am testat-o separat cu aceste cinci cazuri”.

CRITERIILE „USER-FRIENDLY”

O interfață prietenoasă (1) nu cere utilizatorului să scrie exact ca în baza de date — compari textele după `upper()`, deci „sinaia” și „Sinaia” dau același rezultat; (2) răspunde clar și când nu găsește nimic (`NICIUNUL`, nu o fereastră goală); (3) are etichete care spun ce trebuie scris în fiecare casetă.

UNDE SE FOLOSEȘTE ÎN VIAȚA REALĂ

Lista Monumentelor Istorice a României, gestionată de Institutul Național al Patrimoniului, e un catalog în care fiecare monument are un cod, o localitate și o datare, iar căutarea pe localitate e prima operație pe care o folosesc ghizii și turiștii. Aplicațiile de turism și hărțile online filtrează la fel obiectivele după oraș și le ordonează. Muzeele din Prahova folosesc cataloage digitale pentru inventarul obiectelor expuse.

A

EXERCIȚIUL A · COMPLETARE

Piesele unei ferestre Tkinter

COMPLETARE

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

`Entry` `Button` `Listbox` `Label` `pack()` `Tk()`

Fereastra principală a aplicației se creează cu _____. Un text fix, precum „Orașul:”, se afișează cu un _____. Utilizatorul scrie numele orașului într-un _____, iar apăsarea unui _____ cheamă subprogramul dat la parametrul command.

B

EXERCIȚIUL B · GRILĂ

Logică sau interfață?

GRILĂ

Încercuiește răspunsul corect la fiecare întrebare.

1. De ce subprogramul care filtrează obiectivele nu conține nicio instrucțiune Tkinter?

   `Ca să poată fi testat separat, fără fereastră, cu date de intrare fixe`

- Pentru că o fereastră Tkinter se poate deschide doar dintr-un fișier text
- Pentru că într-o fereastră Tkinter se pot afișa doar numere, nu texte
- Pentru că Tkinter nu poate apela subprograme scrise de utilizator

2. Utilizatorul caută „sinaia”, dar în catalog scrie „Sinaia”. Ce face o aplicație prietenoasă?

- Afișează un mesaj că orașul trebuie scris exact ca în catalog, cu majusculă
- Compară textele după ce le transformă pe amândouă cu upper()
- Caută doar după prima literă a orașului, ca să evite problemele de scriere
- Șterge din catalog orașele scrise cu literă mare, ca să nu mai apară diferențe

C

EXERCIȚIUL C · SCRIE CODUL

Obiectivele unui oraș, în ordine cronologică

SCRIERE COD

Se citește un număr natural n, apoi n linii de forma Nume;Oras;An, reprezentând obiective turistice din Prahova, apoi, pe ultima linie, numele unui oraș. Afișează obiectivele din orașul dat, câte unul pe rând, în forma An Nume, în ordinea crescătoare a anilor. Ordonarea o faci tu, prin selecția minimului, nu cu sort(). Dacă orașul nu are niciun obiectiv în catalog, afișează NICIUNUL. Anii obiectivelor din același oraș sunt diferiți. Ex.: pentru catalogul cu cele 7 obiective din teste și orașul Sinaia se afișează 1695 Manastirea Sinaia, 1883 Castelul Peles, 1902 Castelul Pelisor, pe rânduri separate.

HINT **Atenție.** *Când interschimbi doi ani, interschimbă și numele de pe aceleași poziții, altfel anul 1695 ar ajunge lângă „Castelul Peles”. Folosește o variabilă auxiliară pentru fiecare interschimbare.*

HINT *Anul este anul finalizării construcției (castele, mănăstirea) sau al deschiderii muzeului. Numele sunt scrise fără diacritice, ca în multe baze de date mai vechi.*

SOLUȚIA TA · PYTHON

SECȚIUNEA 6 · BILANȚ

6 Bilanț: argumentezi, prezinți, te *autoevaluezi*

Ai 7–10 minute să arăți ce ați construit și de ce contează. Pui ordine în prezentare și exersezi coevaluarea, cu regula folosită de jurii ca să fie corecte.

O prezentare de proiect bună urmează o ordine firească, de la „de ce” la „cum”:

1. **Problema reală și tema transversală.** Spui cine are problema și de ce contează: „O gospodărie din Ploiești nu știe în ce zile consumă cel mai mult”. Tema (mediu, digitalizare, sănătate, patrimoniu) se argumentează prin **oamenii care ar folosi aplicația**, nu prin numărul de linii de cod.
2. **Arhitectura.** Modulele, câte o frază fiecare, și cine le-a scris.
3. **Demonstrația live.** Un scenariu pregătit dinainte, cu date realiste.
4. **Testele.** Cazurile-limită pe care le-ați verificat: lista goală, o singură valoare, valorile egale, zero.
5. **Ce ați îmbunătăți.** Onestitatea despre limite e un semn de maturitate, nu de slăbiciune.
6. **Întrebările.** Asculți întreaga întrebare, răspunzi arătând locul din cod sau din date.

Coevaluarea. Fiecare echipă primește note de la celelalte echipe. Ca nota să fie argumentată, nu dată „din ochi”, fiecare echipă evaluatoare completează o **grilă cu criterii** (o găsești mai jos) și trimite, pe lângă notă, un feedback scurt despre claritatea și independența modulelor. Ca o singură notă exagerată (prea mare din simpatie, prea mică din rivalitate) să nu strice rezultatul, se folosește o regulă din sport.

MEDIA FĂRĂ EXTREME (MEDIA TRUNCHIATĂ)

Din toate notele primite se elimină **o singură** notă maximă și **o singură** notă minimă, iar media se face din notele rămase. Pentru notele 8, 9, 7, 10, 6 se elimină 10 și 6, iar media lui 8, 9, 7 este 8,00. Dacă mai multe note sunt egale cu maximul, se elimină doar una dintre ele.

CAPCANA NOTELOR EGALE

Dacă lucrezi cu **pozițiile** maximului și minimului, pentru notele 7, 7, 7 le poți găsi pe aceeași poziție — și atunci ai elimina o singură notă în loc de două. Caută întâi poziția maximului, apoi minimul printre **celelalte** poziții. Varianta pe **valori**, `s - max(v) - min(v)`, este și ea corectă: scade o dată valoarea maximă și o dată valoarea minimă, chiar dacă sunt egale.

UNDE SE FOLOSEȘTE ÎN VIAȚA REALĂ

La săriturile cu schiurile, din cele cinci note de stil ale arbitrilor se elimină cea mai mare și cea mai mică. La concursurile de proiecte școlare și la jurizarea olimpiadelor cu mai mulți evaluatori se aplică reguli asemănătoare. Statisticienii folosesc media trunchiată ca să reducă efectul valorilor aberante, de exemplu la măsurătorile de poluare cu un senzor defect.

A

EXERCIȚIUL A · ORDONARE

Ordinea unei prezentări de proiect

ORDONARE

Numerotează pașii de la 1 la 6, în ordinea corectă. Pașii îți apar amestecați.

- [ ] Arăți arhitectura: modulele aplicației și cine a scris fiecare modul
- [ ] Spui ce ai îmbunătăți într-o versiune următoare
- [ ] Răspunzi la întrebările colegilor și ale profesorului
- [ ] Faci demonstrația live, pe un scenariu pregătit dinainte
- [ ] Arăți testele, inclusiv cazurile-limită verificate
- [ ] Prezinți problema reală și tema transversală aleasă

B

EXERCIȚIUL B · SCRIE CODUL

Nota de coevaluare, fără extreme

SCRIERE COD

Se citește un număr natural n (n ≥ 3), apoi n note întregi de la 0 la 10, câte una pe linie, primite de o echipă de la celelalte echipe. Elimină o singură apariție a notei maxime (prima) și o singură apariție a notei minime, aflată pe altă poziție decât maximul. Afișează media notelor rămase cu două zecimale exacte, în forma 8.00. Ex.: pentru notele 8, 9, 7, 10, 6 se elimină 10 și 6. Media notelor 8, 9, 7 este 24 / 3 = 8, deci se afișează 8.00.

HINT **Atenție.** *Pentru notele 7, 7, 7, maximul și minimul trebuie să fie două poziții diferite. Rămâne o singură notă de 7, iar media este 7.00.*

HINT *Folosește aceeași tehnică din S3: media în sutimi m = s * 100 // (n - 2) și un 0 în fața zecimalelor mai mici decât 10.*

SOLUȚIA TA · PYTHON

7 SECȚIUNEA 7 · COD GHIDAT
Scrie cod care *trece testele*

Cinci probleme. Scrie soluția pe hârtie, urmărește-o mental pe exemplul dat și abia apoi tastează-o pe platformă, unde e verificată automat pe 5 cazuri de test.

1 PROBLEMA 7.1 · COD GHIDAT
Zilele cu consum de apă peste medie

SCRIERE COD

Se citește un număr natural n (n ≥ 1), apoi n numere naturale, câte unul pe linie: litrii de apă consumați de o gospodărie în fiecare zi. Afișează câte zile au avut un consum strict mai mare decât media consumului pe cele n zile. Ex.: pentru n = 4 și consumurile 120, 80, 0, 200, media este 400 / 4 = 100 de litri. Zilele cu mai mult de 100 de litri sunt cele cu 120 și 200, deci se afișează 2.

HINT *Nu ai nevoie de împărțire: v[i] > total / n este echivalent cu v[i] · n > total, iar așa lucrezi doar cu numere întregi.*

HINT **Atenție.** *Media se poate calcula abia după ce ai citit toate valorile. De aceea le păstrezi într-o listă și o parcurgi a doua oară.*

SOLUȚIA TA · PYTHON

2

PROBLEMA 7.2 · REPARĂ CODUL

Repară nota finală din catalog

SCRIERE COD

Programul de mai jos ar trebui să citească n (n ≥ 1) și n note de la 1 la 10, câte una pe linie, și să afișeze nota finală rotunjită după regula școlii (la 50 de sutimi sau mai mult, în sus). Programul greșește la unele medii. Găsește cauza și scrie programul corect. n = int(input()) s = 0

for i in range(n):

```
    s = s + int(input())
print(round(s / n))
```

Ex.: pentru notele 8 și 9, media e 8,50 și nota corectă este 9. Programul de mai sus afișează 8.

HINT *Programul stricat trece testele obișnuite (9, 8, 8 → 8; 10 → 10) și chiar pe unele medii cu 50 de sutimi (9,50 → 10). Greșește doar când partea întreagă e pară și media are exact 50 de sutimi. Verifică exact afirmația din teoria S3 despre round().*

HINT **Atenție.** *Nu înlocui round() cu int(): int(8.5) taie zecimalele și dă tot 8. Lucrează în sutimi.*

PROGRAMUL CORECTAT · PYTHON

3

PROBLEMA 7.3 · COD GHIDAT

Ziua cea mai puțin activă

SCRIERE COD

Se citește un număr natural n (n ≥ 1), apoi n numere naturale, câte unul pe linie: minutele de mișcare din fiecare zi. Afișează pe primul rând numărul zilei cu cele mai puține minute (zilele se numerotează de la 1; dacă sunt mai multe, prima dintre ele), iar pe al doilea rând numărul de minute din acea zi. Ex.: pentru n = 7 și minutele 30, 90, 0, 75, 60, 45, 120, cea mai puțin activă e ziua 3, cu 0 minute. Se afișează 3, apoi 0.

HINT **Atenție.** *Păstrează poziția minimului, nu doar valoarea lui, și adaugă 1 la afișare: indicii listei încep de la 0, zilele de la 1.*

SOLUȚIA TA · PYTHON

PROBLEMA 7.4 · PROVOCARE

Căutare prietenoasă în catalogul de patrimoniu

SCRIERE COD

Se citește un număr natural n, apoi n linii, fiecare cu numele unui obiectiv turistic, apoi un cuvânt de căutat. Afișează, în ordinea din catalog, câte unul pe rând, toate numele care conțin cuvântul căutat, indiferent de literele mari sau mici. Dacă niciun nume nu îl conține, afișează NICIUNUL. Ex.: pentru catalogul Castelul Peles, Castelul Pelisor, Manastirea Sinaia, Muzeul Ceasului, Castelul Iulia Hasdeu și cuvântul castel se afișează Castelul Peles, Castelul Pelisor și Castelul Iulia Hasdeu, pe rânduri separate.

HINT *Transformă cu upper() și numele, și cuvântul căutat, apoi folosește find(): dacă întoarce -1, cuvântul nu apare în nume.*

HINT **Atenție.** *Afișezi numele original, așa cum e în catalog, nu varianta cu majuscule folosită la comparare.*

SOLUȚIA TA · PYTHON

5

PROBLEMA 7.5 · COD GHIDAT

Distribuția notelor dintr-un catalog

SCRIERE COD

Se citește un număr natural n, apoi n note întregi de la 1 la 10, câte una pe linie. Construiește lista de frecvențe a notelor și afișează pe un singur rând, separate prin spațiu, cele 10 frecvențe: de câte ori apare nota 1, nota 2, …, nota 10. Ex.: pentru n = 4 și notele 10, 9, 9, 5 se afișează 0 0 0 0 1 0 0 0 2 1.

HINT **Atenție.** *Lista de frecvențe are 11 poziții ([0] * 11), ca nota să fie chiar indicele: f[nota]. Poziția 0 rămâne nefolosită.*

SOLUȚIA TA · PYTHON