test-e.ro

CAIET ELEV

Lecția 31

Recapitulare integrată

Recapitulare integrată a blocului L22–L30, înaintea Evaluării sumative 4.

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, ordonezi și scrii singur soluțiile, apoi le verifici pe platformă. Ce ai voie să folosești: secvențial, if/elif/else, while, for cu range și for pe listă, break/continue, %, //, ord()/chr(), liste (literal, indexare, len, slicing, metode), min/max/sum, import math, def/return, open + with, class, tkinter, măsurarea timpului. Fără atribuiri simultane: interschimbările se fac cu aux, iar Fibonacci cu a treia variabilă c. Paritatea se scrie % 2 != 0. Cifrele se parcurg doar cu while. Sortările se scriu de mână: sort() nu se folosește în soluții. Fișierele se scriu cu „w” și se recitesc în același program. Intrarea obișnuită: n pe prima linie, cele n valori pe a doua, citite cu input().split(). Fără diacritice în codul Python.

CUPRINS

1 Reactualizare generală .....

2 Generarea sistematică și șirul lui Fibonacci

3 Sortarea prin selecția minimului .....

4 Sortarea cu listă de frecvențe .....

5 Metoda bulelor și eficiența celor trei metode

6 Fișiere text, obiecte și ferestre Tkinter

7 Cod ghidat · trei probleme .....

SECȚIUNEA 1 · REACTUALIZARE GENERALĂ

1 Reactualizare *generală*

Înainte de test, trecem prin tot ce am construit de la generarea sistematică până la ferestrele Tkinter. Fiecare exercițiu de aici atinge câte o unealtă pe care o vei folosi în problemele din lecție și în evaluare.

Toate problemele din lecția aceasta pornesc de la o listă. O citim dintr-o singură linie: `input()` întoarce un obiect de tip `str`, metoda lui `split()` îl taie la spații și întoarce o listă de șiruri, iar fiecare șir se transformă în număr cu `int`.

Aceeași idee în Python (cu sintaxă reală):

```
1  n = int(input())
2  v = input().split()
3  lista = []
4  for x in v:
5      lista.append(int(x))
```

Două tipare se repetă în tot blocul L22–L30. **Căutarea poziției minimului**: presupui că minimul e pe prima poziție și îl actualizezi ori de câte ori găsești ceva mai mic — e inima sortării prin selecție. **Numărarea aparițiilor**: o listă `frecv` în care `frecv[c]` spune de câte ori a apărut valoarea `c` — e inima sortării cu listă de frecvențe. Și un al treilea, mai vechi: **interschimbarea prin `aux`**, pe care o folosesc toate sortările.

> **DE REȚINUT**
>
> `a, b = b, a` nu apare în lecțiile noastre. Interschimbarea se scrie în trei pași, cu o variabilă auxiliară: `aux = a`, `a = b`, `b = aux`. La Fibonacci, același motiv ne cere o a treia variabilă, `c`, care păstrează suma înainte ca `a` să fie suprascris.

A EXERCIȚIUL A · GRILĂ

Întrebări de încălzire

GRILĂ

Încercuiește răspunsul corect la fiecare întrebare.

1. Considerând F1 = F2 = 1, care este al 7-lea termen al șirului lui Fibonacci? (13) (11) (8) (21)

2. După prima trecere a sortării prin selecția minimului, ce știm sigur despre listă?

   (Ultimele două elemente ale listei au ajuns deja pe pozițiile lor finale)

- Elementul cel mai mare a urcat pe ultima poziție a listei
- Pe prima poziție se află cel mai mic element din întreaga listă
- Lista este deja sortată dacă nu s-a făcut nicio interschimbare

3. Care mod de deschidere șterge conținutul vechi al fișierului <code>note.txt</code>?

- <code>open('note.txt', 'a')</code>
- <code>open('note.txt', 'w')</code>
- <code>open('note.txt', 'r')</code>
- <code>with open('note.txt') as f</code>

4. Ce reprezintă instrucțiunea <code>lista.append(7)</code> privită prin ochii programării orientate pe obiecte?

- Crearea unei noi clase numite append, pornind de la clasa list
- O funcție din biblioteca math care adună 7 la fiecare element
- Apelul metodei append a obiectului lista, prin operatorul punct
- Instanțierea unui obiect nou de tip list, cu valoarea inițială 7

B

EXERCIȚIUL B · DRY-RUN

Termenii lui Fibonacci, cu variabila de rezervă

DRY-RUN

Urmărește bucla pas cu pas. Valoarea lui k la intrarea în buclă este dată; tu completezi c, a și b după ce corpul buclei s-a executat. Pe ultimul rând condiția k <= 8 devine falsă și nimic nu se mai modifică.

```
a = 1
b = 1
k = 3
while k <= 8:
    c = a + b
    a = b
    b = c
    k = k + 1
print(b)
```

k	c = a + b	a	b
3	2	1	2
4			
5			
6			
7			
8			
9 — 9 <= 8 fals, bucla se oprește	—	13	21

HINT *La fiecare pas: întâi c = a + b, apoi a primește vechiul b, iar b primește c. Verifică celulele marcate.*

SPRIJIN · SCRIE CODUL

Unde e cel mai mic?

SCRIERE COD

Se citește un număr natural n (n ≥ 1), apoi, pe linia următoare, n numere naturale separate prin spațiu. Afișează, pe același rând, cea mai mică valoare și poziția pe care apare prima dată, numărând pozițiile de la 1. Ex.: Pentru n = 5 și valorile 7 3 9 3 8, minimul este 3, iar prima lui apariție e pe poziția 2 ⇒ se afișează 3 2.

HINT **Atenție.** *În Python pozițiile din listă încep de la 0. Tu lucrezi cu indicii listei, dar la afișare aduni 1.*

SOLUȚIA TA · PYTHON

SPRIJIN · SCRIE CODUL

Câte apariții are fiecare cod?

SCRIERE COD

Se citește un număr natural n, apoi, pe linia următoare, n coduri, fiecare între 1 și 5. Afișează, pe un singur rând, de câte ori apare codul 1, de câte ori apare codul 2, și așa mai departe până la codul 5. Ex.: Pentru codurile 3 1 3 5 2 3 1: codul 1 apare de 2 ori, codul 2 o dată, codul 3 de 3 ori, codul 4 niciodată, codul 5 o dată ⇒ se afișează 2 1 3 0 1.

HINT **Observație.** *Un cod care nu apare deloc se afișează cu 0 — nu se sare.*

SOLUȚIA TA · PYTHON

SECȚIUNEA 2 · GENERAREA SISTEMATICĂ ȘI ȘIRUL LUI FIBONACCI

2 Generarea sistematică și șirul lui *Fibonacci*

Unele liste nu se citesc: se construiesc. Parcurgi un interval și păstrezi doar ce respectă o regulă, sau calculezi fiecare termen din cei de dinainte.

Generarea sistematică (L22) înseamnă să iei, pe rând, toate valorile dintr-un interval și să le păstrezi doar pe cele care respectă o proprietate. Proprietatea merită scrisă ca subprogram separat, care întoarce `True` sau `False`: programul principal rămâne scurt, iar regula poate fi testată singură.

> **REGULA CALENDARULUI GREGORIAN**
>
> Un an este bisect dacă se împarte exact la 4, **cu excepția** anilor care se împart la 100 — aceștia sunt bisecți doar dacă se împart și la 400. Deci 2024 e bisect, 1900 nu, 2000 da. Regula a fost stabilită în 1582, prin bula papală *Inter gravissimas* a papei Grigore al XIII-lea, ca să corecteze decalajul acumulat de calendarul iulian.

Aceeași regulă în Python, ca subprogram:

```
1  def bisect(an):
2      if an % 400 == 0:
3          return True
4      if an % 100 == 0:
5          return False
6      return an % 4 == 0
```

Ordinea testelor contează. Dacă ai întreba întâi „se împarte la 4?”, anul 1900 ar trece drept bisect. Verifici de la regula cea mai strictă (400) spre cea mai largă (4).

Șirul lui Fibonacci (L23) e un șir **recurent**: primii doi termeni sunt dați (1 și 1), iar fiecare termen următor e suma celor doi dinainte: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144. Îl generăm cu trei variabile: `a` și `b` țin ultimii doi termeni, `c` primește suma, apoi „glisăm fereastra” cu un pas.

> **ORDINEA ATRIBUIRILOR**
>
> După `c = a + b` scrii întâi `a = b`, apoi `b = c`. Invers (`b = c` întâi) ai pierde vechiul `b` înainte să-l copiezi în `a`, și șirul s-ar strica de la al patrulea termen.

> **UNDE SE FOLOSEȘTE ÎN VIAȚA REALĂ**
>
> Regula gregoriană rulează în calendarul fiecărui telefon și în programele băncilor care calculează dobânda pe zile (un an bisect are 366 de zile). Șirul lui Fibonacci stă pe cărțile de **Planning Poker**: echipele de programatori care lucrează după metoda Scrum își estimează

sarcinile numai cu valori din șir (1, 2, 3, 5, 8, 13, 21), pentru că diferența tot mai mare dintre cărți spune cinstit că o sarcină mare e și mai greu de estimat precis.

A

EXERCIȚIUL A · GRILĂ

Reguli și termeni

GRILĂ

Încercuiește răspunsul corect la fiecare întrebare.

1. Care este primul an bisect de după 2096? `2108` `2100` `2104` `2102`
2. Câți ani bisecți există între 1901 și 2000, inclusiv capetele? `25` `26` `23` `24`
3. De ce, la generarea șirului lui Fibonacci, folosim o a treia variabilă <code>c</code>?

`Pentru că șirul lui Fibonacci are nevoie de trei termeni inițiali dați`

`Pentru că Python nu permite adunarea directă a două variabile de tip int`

`Ca să păstrăm suma înainte ca a să fie suprascris, fără atribuire simultană`

`Ca să numărăm câți termeni au fost generați până în acel moment`

B

EXERCIȚIUL B · COMPLETARE

Ce fel de șir e Fibonacci?

COMPLETARE

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

`trei` `aleator` `recurent` `doi` `adunarea` `înmulțirea`

Șirul lui Fibonacci este un șir _____: fiecare termen, începând cu al treilea, se obține prin _____ celor _____ termeni dinaintea lui.

C

EXERCIȚIUL C · SCRIE CODUL

Anii bisecți dintr-un interval

SCRIERE COD

Se citesc două numere naturale a și b (a ≤ b), fiecare pe câte o linie. Afișează, pe un singur rând și separați prin spațiu, toți anii bisecți din intervalul [a, b], după regula calendarului gregorian. Dacă în interval nu există niciun an bisect, afișează 0. Scrie regula ca subprogram bisect(an), care întoarce True sau False. Ex.: Pentru a = 1896 și b = 1904 se afișează 1896 1904: 1900 se împarte la 100, dar nu și la 400, deci nu e bisect.

HINT **Atenție.** *Testează divizibilitatea în ordinea 400 → 100 → 4. Dacă verifici întâi „se împarte la 4?”, anii 1900 și 2100 ies greșit bisecți.*

HINT **Curiozitate.** *Între 2096 și 2104 vor trece opt ani fără niciun an bisect, pentru că 2100 se împarte la 100, dar nu și la 400. Ultima „pauză” de acest fel a fost între 1896 și 1904.*

SOLUȚIA TA · PYTHON

D

EXERCIȚIUL D · SCRIE CODUL

Planning Poker pe scala Fibonacci

SCRIERE COD

O echipă de programatori își estimează sarcinile pe scala Fibonacci: 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, … (fiecare valoare e suma celor două dinainte). Se citește un număr natural n, apoi n estimări brute (numere naturale ≥ 1). Pentru fiecare estimare afișează cea mai mică valoare de pe scală care este mai mare sau egală cu ea. Rezultatele se afișează pe un singur rând, separate prin spațiu. Ex.: Pentru estimările 4 8 1 14 22 se afișează 5 8 1 21 34: 4 urcă la 5, 8 e deja pe scală, 14 urcă la 21, 22 urcă la 34.

HINT **Observație.** *Scala pornește de la 1, 2 (fără al doilea 1 al șirului), fiindcă o carte cu aceeași valoare de două ori n-ar avea sens în joc. Multe pachete comerciale rotunjesc capătul scalei (20, 40, 100); aici folosim șirul pur.*

SOLUȚIA TA · PYTHON

SECȚIUNEA 3 · SORTAREA PRIN SELECȚIA MINIMULUI

3 Sortarea prin selecția *minimului*

Cauți cel mai mic element rămas și îl aduci în față. Repeți. După fiecare pas, o poziție în plus e definitivă — iar asta face metoda perfectă când îți trebuie doar primii câțiva.

SORTAREA PRIN SELECȚIA MINIMULUI

Pentru fiecare poziție i, de la prima la penultima: găsești poziția `pmin` a celui mai mic element din zona `i … n-1`, apoi interschimbi `lista[i]` cu `lista[pmin]`. După pasul i, primele i + 1 poziții conțin cele mai mici valori, în ordine, și nu se mai mișcă.

Aceeași idee în Python, ca subprogram care modifică lista primită:

```
1  def sortare_selectie(lista):
2      n = len(lista)
3      for i in range(n - 1):
4          pmin = i
5          for j in range(i + 1, n):
6              if lista[j] < lista[pmin]:
7                  pmin = j
8          aux = lista[i]
9          lista[i] = lista[pmin]
10         lista[pmin] = aux
```

Numărul de comparații nu depinde de cum arată lista: la pasul 0 se fac n−1 comparații, la pasul 1 n−2, …, la ultimul una. Total: n·(n−1)/2 — pentru 10 elemente, 45, fie că lista e deja sortată, fie că e inversă (am numărat prin rulare). Avantajul ei: face cel mult n−1 interschimbări și, mai ales, **poți să te oprești după k pași** dacă îți trebuie doar primele k valori.

> **UNDE SE FOLOSEȘTE ÎN VIAȚA REALĂ**
>
> Cronometrajul electronic de la o finală de atletism produce opt timpi, iar crainicul are nevoie de ei în ordine. La o finală cu sute de concurenți (maratonul unui oraș), organizatorii anunță mai întâi **podiumul** — primii trei — adică doar primii trei pași ai selecției. Același tipar îl folosește un magazin online când îți arată „cele mai ieftine 5 oferte” dintr-o mie.

A

EXERCIȚIUL A · ORDONARE

Un pas al sortării prin selecție

ORDONARE

Pașii de mai jos descriu sortarea prin selecția minimului, dar sunt amestecați. Numerotează-i de la 1 la 5, în ordinea corectă.

- [ ] Parcurgi elementele de la poziția i + 1 până la ultima poziție a listei.
- [ ] Pentru poziția curentă i, presupui că minimul se află chiar pe poziția i (pmin = i).
- [ ] Interschimbi lista[i] cu lista[pmin], folosind o variabilă aux.
- [ ] Când găsești un element mai mic decât lista[pmin], actualizezi pmin cu poziția lui.
- [ ] Treci la poziția următoare și repeți, până ajungi la penultima poziție.

B

EXERCIȚIUL B · DRY-RUN

Selecția pe lista [7, 3, 9, 1, 5]

DRY-RUN

Pentru fiecare pas i (dat), completează poziția minimului găsit în zona nesortată (numărând de la 0) și cum arată lista după interschimbare. Scrie lista ca numere separate prin spațiu.

```
lista = [7, 3, 9, 1, 5]
n = len(lista)
for i in range(n - 1):
    pmin = i
    for j in range(i + 1, n):
        if lista[j] < lista[pmin]:
            pmin = j
```

```
    aux = lista[i]
    lista[i] = lista[pmin]
    lista[pmin] = aux
print(lista)
```

i	pmin	lista după interschimbare
0	3	1 3 9 7 5
1		
2		
3		
4 — range(4) s-a terminat, bucla se oprește	—	1 3 5 7 9

HINT *La fiecare pas i caută minimul doar în zona i … 4 (pozițiile numărate de la 0), apoi interschimbă-l cu lista[i]. Verifică celulele marcate.*

C

EXERCIȚIUL C · SCRIE CODUL

Clasamentul finalei de 100 m

SCRIERE COD

Se citește un număr natural n (numărul finaliștilor), apoi n timpi, exprimați în sutimi de secundă (de exemplu 979 înseamnă 9,79 s). Afișează timpii în ordine crescătoare, pe un singur rând, separați prin spațiu. Ordonează lista cu sortarea prin selecția minimului, scrisă ca subprogram. Ex.: Timpii finalei olimpice de la Paris 2024 (bărbați, 100 m), citiți într-o ordine amestecată: 985 979 988 981 986 979 991 982 ⇒ se afișează 979 979 981 982 985 986 988 991.

HINT **Curiozitate.** *La Paris, primii doi au terminat amândoi în 9,79 s. Aurul s-a decis la miimi de secundă — 9,784 față de 9,789 —, o diferență pe care cronometrajul oficial o vede, dar lista noastră în sutimi nu.*

HINT **Observație.** *Metoda sort() a listelor există, dar aici scrii algoritmul de mână: la evaluare ți se va cere să-l recunoști și să-l aplici.*

SOLUȚIA TA · PYTHON

PERFORMANȚĂ · SCRIE CODUL

Doar podiumul — exercițiu de performanță

SCRIERE COD

Se citesc numărul n de concurenți, apoi un număr natural k (1 ≤ k ≤ n), apoi cei n timpi, în sutimi de secundă. Afișează, în ordine crescătoare, doar cei mai buni k timpi. Folosește sortarea prin selecția minimului, dar execută doar primii k pași — nu sorta toată lista. Ex.: Pentru n = 8, k = 3 și timpii 985 979 988 981 986 979 991 982 se afișează 979 979 981 — podiumul, obținut după trei pași de selecție.

HINT Atenție. *Căutarea minimului, la fiecare pas i, se face tot pe toată zona rămasă (până la n − 1), nu doar până la k. Se oprește numărul de pași, nu lungimea căutării.*

HINT Curiozitate. *Pentru un maraton cu 1000 de alergători, sortarea completă prin selecție face 499 500 de comparații; podiumul, doar 999 + 998 + 997 = 2994.*

SOLUȚIA TA · PYTHON

SECȚIUNEA 4 · SORTAREA CU LISTĂ DE FRECVENȚE

4 Sortarea cu listă de *frecvențe*

Când valorile sunt întregi mici, nu le mai compari între ele: le numeri. Așa se ordonează pacienții după gravitate sau cererile de reparație după tipul defecțiunii.

SORTAREA CU LISTĂ DE FRECVENȚE

Construiești o listă `frecv` cu câte o căsuță pentru fiecare valoare posibilă. Parcurgi datele o singură dată și, pentru fiecare valoare `c`, crești `frecv[c]` cu 1. Apoi parcurgi valorile posibile în ordine crescătoare și scrii fiecare valoare `c` de exact `frecv[c]` ori.

Aceeași idee în Python, pentru coduri de la 1 la 5:

```
frecv = [0] * 6
for x in v:
    c = int(x)
    frecv[c] = frecv[c] + 1
for cod in range(1, 6):
    for k in range(frecv[cod]):
        print(cod, end=' ')
```

Condițiile de aplicabilitate (L25) sunt stricte: valorile trebuie să fie **numere întregi** (nu poți avea căsuța `frecv[9.79]`) și dintr-un **interval mic, cunoscut dinainte** (lista are câte o căsuță pentru fiecare valoare posibilă, deci pentru valori până la un miliard ar trebui un miliard de căsuțe). Când condițiile sunt îndeplinite, metoda face aproximativ n + k pași (n date, k valori posibile), mult mai puțin decât cele n·(n−1)/2 comparații ale selecției.

MĂRIMEA LISTEI

Pentru valori de la 1 la k, lista are nevoie de k + 1 căsuțe, pentru că indicii încep de la 0: `frecv = [0] * (k + 1)`. Căsuța 0 rămâne nefolosită. Cu o căsuță mai puțin, prima valoare maximă întâlnită oprește programul cu `IndexError`.

UNDE SE FOLOSEȘTE ÎN VIAȚA REALĂ

În camera de gardă, fiecare pacient primește la triaj un cod de gravitate pe o scară cu 5 niveluri, în care **codul 1 înseamnă cazul cel mai grav** (așa lucrează UPU-urile din România și, în Marea Britanie, sistemul Manchester Triage System) — cinci valori întregi, exact cazul ideal. Un service de lifturi primește zilnic cereri cu codul defecțiunii (1–8) și vrea să știe ce defect apare cel mai des: asta e **analiza Pareto**, folosită în managementul calității, care ordonează categoriile după frecvență ca să se repare întâi ce strică cel mai mult.

A

EXERCIȚIUL A · GRILĂ

Când se poate folosi?

GRILĂ

Încercuiește răspunsul corect la fiecare întrebare.

1. Pentru care set de date este potrivită sortarea cu listă de frecvențe?

   - Salariile brute, cu zecimale, ale angajaților unei companii
   - Notele de la 1 la 10 ale celor 300 de elevi dintr-un liceu
   - Numele elevilor dintr-o clasă, pentru ordinea din catalog
   - Distanțele în kilometri, cu zecimale, dintre orașele țării

2. Codurile de defect ale unui lift iau valori de la 1 la 8. Câte elemente trebuie să aibă lista <code>frecv</code> ca să putem scrie direct <code>frecv[c]</code>?

   7 | 9 | 8 | 10

B

EXERCIȚIUL B · DRY-RUN

Construirea listei de frecvențe

DRY-RUN

Codurile de triaj sosesc pe rând (coloana c, dată). După fiecare cod, completează valorile frecv[1] … frecv[5]. Toate valorile sunt numerice.

```
coduri = [3, 1, 3, 5, 2, 3, 1]
frecv = [0] * 6
for c in coduri:
    frecv[c] = frecv[c] + 1
print(frecv[1:])
```

c	frecv[1]	frecv[2]	frecv[3]	frecv[4]	frecv[5]
3	0	0	1	0	0
1					
3					
5					
2					
3					
1					
– (lista coduri s-a epuizat, bucla se oprește)	2	1	3	0	1

HINT *La fiecare cod c crește doar căsuța frecv[c] cu 1; toate celelalte rămân cum erau pe rândul de deasupra.*

REMEDIERE · REPARĂ CODUL

Ordinea de triaj

SCRIERE COD

Programul de mai jos ar trebui să citească n coduri de triaj (de la 1 la 5) și să le afișeze în ordine crescătoare, cu sortarea cu listă de frecvențe. Pentru codurile 3 1 3 5 2 3 1 se oprește însă cu eroarea IndexError: list index out of range. Corectează-l.

Programul de corectat:

```
n = int(input())
v = input().split()
frecv = [0] * 5
for x in v:
    c = int(x)
    frecv[c] = frecv[c] + 1
for cod in range(1, 5):
    for k in range(frecv[cod]):
        print(cod, end=' ')
```

Ex.: Pentru n = 7 și codurile 3 1 3 5 2 3 1, programul corect afișează 1 1 2 3 3 3 5.

HINT Atenție. *Programul are două greșeli legate de același lucru. Dacă o repari doar pe prima, eroarea dispare, dar pacienții cu codul 5 dispar și ei din listă — fără niciun mesaj. Verifică și bucla de afișare.*

HINT Observație. *Remedierea verifică regula din teorie: pentru valori de la 1 la k, lista are k + 1 căsuțe, iar afișarea parcurge valorile de la 1 la k inclusiv.*

PROGRAMUL CORECTAT · PYTHON

C

EXERCIȚIUL C · SCRIE CODUL

Analiza Pareto a defecțiunilor de lift

SCRIERE COD

Un service de lifturi primește n cereri de mentenanță, fiecare cu codul defecțiunii (număr între 1 și 8). Afișează, câte unul pe linie, fiecare cod care apare măcar o dată, urmat de un spațiu și de numărul de cereri cu acel cod. Codurile se afișează de la cel mai frecvent la cel mai rar; la frecvență egală, codul mai mic se afișează primul. Ex.: Pentru cererile 3 7 3 1 3 7 5 3 8 7: codul 3 apare de 4 ori, codul 7 de 3 ori, codurile 1, 5 și 8 câte o dată ⇒ se afișează pe linii 3 4, 7 3, 1 1, 5 1, 8 1.

HINT Observație. *Problema are două etape: întâi numeri cu lista de frecvențe, apoi afișezi codurile prezente în ordinea cerută. Cea mai simplă cale refolosește lista: pentru fiecare frecvență f, de la n în jos până la 1, parcurgi codurile de la 1 la 8 și le afișezi pe cele cu frecv[c] == f. Dacă preferi selecția (căutând maximul), atenție: interschimbările strică ordinea codurilor cu aceeași frecvență, așa că la egalitate trebuie să compari și codurile.*

HINT Curiozitate. *Principiul lui Pareto („80% din efecte vin din 20% din cauze”) poartă numele economistului Vilfredo Pareto; în controlul calității l-a popularizat inginerul Joseph Juran.*

SOLUȚIA TA · PYTHON

5 SECȚIUNEA 5 · METODA BULELOR ȘI EFICIENȚA CELOR TREI METODE

Metoda bulelor și eficiența celor trei *metode*

A treia metodă compară doar vecini. Pe o listă deja sortată se oprește după o singură trecere; pe una inversă, e cea mai lentă. Comparăm cele trei metode prin numărul real de pași.

METODA BULELOR (BUBBLE SORT) CU FANION

Parcurgi lista și compari fiecare element cu vecinul lui din dreapta; dacă sunt în ordine greșită, îi interschimbi. Valorile mari „urcă” spre final ca bulele. Repeți trecerile cât timp ultima trecere a făcut măcar o interschimbare; variabila `sortat` (fanionul) îți spune când să te oprești.

Aceeași idee în Python:

```
def bule(lista):
    n = len(lista)
    sortat = False
    while not sortat:
        sortat = True
```

```
        for i in range(n - 1):
            if lista[i] > lista[i + 1]:
                aux = lista[i]
                lista[i] = lista[i + 1]
                lista[i + 1] = aux
                sortat = False
```

Am numărat, rulând cod real, câte comparații face fiecare metodă pe 10 elemente (bulele — în varianta de mai sus, cu fanion și treceri complete):

LISTA DE 10 ELEMENTE	SELECȚIA MINIMULUI	METODA BULELOR CU FANION
deja sortată	45	9
inversă (10, 9, …, 1)	45	90

Pentru 1000 de elemente: selecția face mereu 499 500 de comparații; bulele fac 999 pe o listă sortată, dar 999 000 pe una inversă. **Notația O** descrie cum crește numărul de pași când crește n, în cazul cel mai nefavorabil: selecția și bulele sunt **O(n²)** — dublezi n, pașii cresc de aproximativ 4 ori —, iar lista de frecvențe e **O(n + k)**. Alegerea metodei depinde deci de date: cât de mari sunt valorile, cât de „dezordonată” e lista, câte rezultate îți trebuie.

> **UNDE SE FOLOSEȘTE ÎN VIAȚA REALĂ**
>
> Numărul de interschimbări pe care îl face metoda bulelor pe un clasament spune cât de departe e acel clasament de ordinea corectă. Statisticienii îl numesc **distanța Kendall tau** (după Maurice Kendall, 1938) și îl folosesc ca să compare două clasamente: cât de bun a fost un pronostic sportiv față de clasamentul final, sau cât de mult diferă ordinea rezultatelor dată de două motoare de căutare pentru aceeași întrebare.

A

EXERCIȚIUL A · ASOCIERE

Fiecare metodă, cu proprietatea ei

ASOCIERE

Fiecărei metode sau notații din stânga îi corespunde o singură proprietate din dreapta. Scrie în căsuță litera proprietății potrivite. Fiecare literă se folosește o singură dată.

- ☐ Selecția minimului
- ☐ Lista de frecvențe
- ☐ Metoda bulelor cu fanion

Notația $O(n^2)$

A Nu compară elementele între ele; cere valori întregi dintr-un interval mic

B Se oprește după o singură trecere dacă lista e deja sortată

C Dacă dublezi n, numărul de pași crește de aproximativ 4 ori

D După pasul i, primele poziții sunt definitive — te poți opri după k pași

B

EXERCIȚIUL B · GRILĂ

Pași și creștere

GRILĂ

Încercuiește răspunsul corect la fiecare întrebare.

1.

Sortarea prin selecție face 499 500 de comparații pe o listă de 1000 de elemente. Câte face pe o listă de 2000 de elemente?

3 998 000 | 999 000 | 4 000 000 | 1 999 000

2. Metoda bulelor cu fanion primește o listă de 100 de elemente, deja sortată. Ce se întâmplă?

Face 4950 de comparații, exact cât ar face selecția minimului

Nu face nicio comparație, pentru că lista este deja sortată

Face 100 de treceri complete, cu câte 99 de comparații fiecare

Face o singură trecere, cu 99 de comparații, și se oprește

C

EXERCIȚIUL C · DRY-RUN

Trecerile metodei bulelor pe [4, 2, 5, 1, 3]

DRY-RUN

Numărul trecerii e dat. După fiecare trecere completă, scrie cum arată lista (numere separate prin spațiu), câte interschimbări s-au făcut în trecerea aceea și dacă fanionul sortat a rămas DA sau a devenit NU.

```
lista = [4, 2, 5, 1, 3]
n = len(lista)
sortat = False
while not sortat:
    sortat = True
    for i in range(n - 1):
```

```
        if lista[i] > lista[i + 1]:
            aux = lista[i]
            lista[i] = lista[i + 1]
            lista[i + 1] = aux
            sortat = False
print(lista)
```

trecerea	lista după trecere	interschimbări	sortat la final de trecere
1	2 4 1 3 5	3	NU
2			
3			
4			

HINT *Într-o trecere compari, de la stânga la dreapta, fiecare pereche de vecini și îi interschimbi dacă stânga e mai mare. Fanionul rămâne DA doar dacă trecerea n-a făcut nicio interschimbare.*

D

EXERCIȚIUL D · SCRIE CODUL

Distanța Kendall tau dintre pronostic și clasament

SCRIERE COD

Un comentator sportiv a făcut un pronostic pentru n echipe, numerotându-le de la 1 (locul 1 prezis) la n. După campionat, se citește clasamentul final, scris cu numerele din pronostic: primul număr citit e echipa care a ieșit de fapt pe locul 1, al doilea cea de pe locul 2 și așa mai departe. Afișează distanța Kendall tau dintre pronostic și clasament: numărul de interschimbări pe care le face metoda bulelor ca să aducă lista citită în ordinea 1, 2, …, n. Scrie numărarea ca subprogram care întoarce rezultatul. Ex.: Pentru n = 4 și clasamentul 2 1 4 3, metoda bulelor face 2 interschimbări (2↔1 și 4↔3) ⇒ se afișează 2. Un pronostic perfect are distanța 0.

HINT **Observație.** *Distanța maximă apare când clasamentul e exact invers pronosticului: pentru 5 echipe, 5 4 3 2 1 dă 10 = 5·4/2.*

SOLUȚIA TA · PYTHON

SECȚIUNEA 6 · FIȘIERE TEXT, OBIECTE ȘI FERESTRE TKINTER

6 Fișiere text, obiecte și ferestre *Tkinter*

Datele reale stau în fișiere, iar rezultatele ajung la oameni prin ferestre. Între ele, tot ce folosești sunt obiecte: șiruri, liste, fișiere, butoane.

Un **fișier text** păstrează date pe disc după ce programul s-a oprit. Funcția `open(nume, mod)` îl deschide și întoarce un obiect al clasei `TextIOWrapper` (L28): cu modul `'w'` scrii de la zero (ce era înainte se șterge), cu `'a'` adaugi la final, cu `'r'` citești. Construcția `with` închide fișierul automat, chiar dacă apare o eroare. Fără `with`, închiderea o scrii tu: `f = open(...)`, apoi, la final, `f.close()`. Un fișier deschis pentru citire se poate parcurge linie cu linie cu `for`; fiecare linie vine ca `str`, cu tot cu caracterul de sfârșit de linie.

Scriem, apoi recitim (așa lucrează și problemele din lecție, în sistemul de fișiere al browserului):

```
1 with open('meteo.txt', 'w') as f:
2     f.write('-3\n5\n2\n')
3 temp = []
4 with open('meteo.txt', 'r') as f:
5     for linie in f:
6         temp.append(int(linie))
```

`int(linie)` ignoră singur caracterul de sfârșit de linie. Pentru text, folosește metoda `strip()` ca să-l scoți.

> **TOTUL E UN OBIECT**

O **clasă** e un tipar (`str`, `list`, `TextIOWrapper`, `Tk`); un **obiect** e o instanță concretă a ei, cu date proprii și cu **metode** apelate prin operatorul punct (L29). `"popescu ana".upper()` întoarce `"POPESCU ANA"`; `"Ana,9,10".split(',')` întoarce lista `['Ana', '9', '10']`; `lista.append(7)` modifică obiectul `lista`; `f.write(text)` scrie în obiectul-fișier. O **bibliotecă** (math, datetime, tkinter) e o colecție de clase și funcții gata făcute.

FORMATUL CSV

„Comma-Separated Values”: fiecare linie e o înregistrare, iar câmpurile ei sunt separate prin virgulă — `Popescu Ana,9,10,8`. Formatul e descris în documentul standard RFC 4180 și îl exportă orice program de calcul tabelar. Îl citești cu `linie.strip().split(',')`.

Tkinter (L30) transformă programul într-o aplicație cu fereastră. Codul de mai jos îl rulezi în IDE-ul de pe calculator: browserul nu poate deschide ferestre Python adevărate. În ultima problemă din S7 vei lucra totuși cu o fereastră, verificată cu un simulator.

```
import tkinter as tk
from tkinter import messagebox

def adauga():
    cos.insert(tk.END, produs.get())

fereastra = tk.Tk()
produs = tk.Entry(fereastra)
produs.pack()
buton = tk.Button(fereastra, text='Adauga', command=adauga)
buton.pack()
cos = tk.Listbox(fereastra)
cos.pack()
fereastra.mainloop()
```

Clasa `Listbox` are metodele `insert` (adaugă un rând), `delete` (șterge) și `curselection()` (întoarce pozițiile rândurilor selectate). `messagebox.showinfo(titlu, mesaj)` deschide o casetă de mesaj. `command=adauga` leagă butonul de subprogram — atenție, **fără paranteze**: îi dai numele subprogramului, nu îl apelezi.

MODUL 'W' NU IARTĂ

Dacă deschizi cu `'w'` un fișier în care voiai să adaugi, tot ce era scris dispare fără avertisment. Pentru un registru care crește zi de zi, modul corect e `'a'`.

UNDE SE FOLOSEȘTE ÎN VIAȚA REALĂ

Stațiile meteo automate scriu la intervale fixe câte o linie într-un fișier text, iar meteorologii calculează din el **amplitudinea termică** (diferența dintre maxima și minima unei perioade). Catalogul electronic al școlii exportă notele în format CSV, pe care secretariatul le importă apoi în alt program. Iar aplicațiile de casă de marcat sau de rezervări sunt, în fond, ferestre cu butoane legate de subprograme — exact ce ai construit cu Tkinter.

A

EXERCIȚIUL A · ORDONARE

Construiești fereastra coșului de cumpărături

ORDONARE

Pașii de construire a unei ferestre Tkinter sunt amestecați. Numerotează-i de la 1 la 6, în ordinea în care trebuie să apară în program.

- [ ] Creează componentele (Entry, Button cu command=adauga, Listbox), cu fereastra drept părinte.
- [ ] Creează fereastra principală: fereastra = tk.Tk().
- [ ] Scrie subprogramul care va fi apelat la apăsarea butonului.
- [ ] Importă biblioteca: import tkinter as tk.
- [ ] Pornește bucla de evenimente: fereastra.mainloop().
- [ ] Așază fiecare componentă în fereastră cu pack().

B

EXERCIȚIUL B · GRILĂ

Fișiere, șiruri, liste vizuale

GRILĂ

Încercuiește răspunsul corect la fiecare întrebare.

1. Ce face instrucțiunea <code>f = open('note.txt', 'a')</code>?

   `Deschide fișierul pentru citire, de la începutul primei lui linii`

   `Afișează pe ecran conținutul fișierului, linie cu linie`

   `Deschide fișierul pentru scriere la final, păstrând ce era deja scris`

   `Șterge tot conținutul fișierului și îl pregătește pentru scriere`

2. Ce întoarce expresia <code>"12,9,10".split(',')</code>?

   <code>['12,9,10']</code> <code>[12, 9, 10]</code> <code>['12', '9', '10']</code>

   <code>['12', ',', '9', ',', '10']</code>

3. Ce metodă a unui obiect <code>Listbox</code> întoarce pozițiile rândurilor selectate de utilizator?

<code>curselection()</code> <code>insert()</code> <code>delete()</code> <code>get()</code>

C

EXERCIȚIUL C · DRY-RUN

Maxima dintr-un fișier, linie cu linie

DRY-RUN

Programul scrie trei temperaturi într-un fișier, apoi le citește înapoi. Linia citită este dată; completează valoarea lui t și valoarea lui tmax după ce corpul buclei s-a executat. Valorile sunt numerice.

```
with open('temp.txt', 'w') as f:
    f.write('-3\n5\n2\n')
tmax = -100
with open('temp.txt', 'r') as f:
    for linie in f:
        t = int(linie)
        if t > tmax:
            tmax = t
print(tmax)
```

linia citită	t	tmax
-3	-3	-3
5		
2		
(sfârșitul fișierului — bucla se oprește)	—	5

HINT *t primește valoarea liniei citite, iar tmax se schimbă doar când t este mai mare decât tmax-ul de până atunci.*

D

EXERCIȚIUL D · SCRIE CODUL

Amplitudinea termică a stației meteo

SCRIERE COD

Se citește un număr natural n, apoi n temperaturi înregistrate de o stație meteo, exprimate în zecimi de grad Celsius (numere întregi, pot fi negative). Scrie temperaturile în fișierul meteo.txt, câte una pe linie. Apoi deschide fișierul pentru citire, încarcă temperaturile într-o listă și afișează, pe un singur

rând, temperatura maximă, temperatura minimă și amplitudinea termică (diferența dintre ele). Ex.: Recordurile absolute ale României sunt 44,5 °C (Ion Sion, 1951) și −38,5 °C (Bod, 1942). Pentru n = 2 și valorile 445 -385 se afișează 445 -385 830 — o amplitudine de 83 de grade.

HINT **Atenție.** *Programul trebuie să treacă prin fișier: întâi îl scrie complet (modul 'w'), apoi îl redeschide cu 'r'. Dacă folosești 'a', fișierul păstrează valorile de la rularea precedentă și rezultatele se amestecă.*

HINT **Observație.** *Stațiile meteo lucrează în zecimi de grad tocmai ca să poată păstra numere întregi în fișiere.*

SOLUȚIA TA · PYTHON

E

EXERCIȚIUL E · SCRIE CODUL

Catalogul exportat în CSV

SCRIERE COD

Se citește un număr natural n, apoi n linii în format CSV, fiecare de forma Nume Prenume,nota1,nota2,… (numele, apoi una sau mai multe note întregi). Scrie liniile, neschimbate, în fișierul catalog.csv. Apoi redeschide fișierul, citește-l linie cu linie și afișează, câte unul pe linie și cu litere mari, numele elevilor a căror medie este sub 5. Dacă nu există niciunul, afișează NIMENI. Ex.: Pentru liniile Popescu Ana,9,10,8, Ionescu Dan,4,5,3 și Marin Ioana,5,4,6: media lui Ionescu Dan e 4, a celorlalți 9 și 5 ⇒ se afișează IONESCU DAN. Media 5 nu e sub 5.

HINT **Atenție.** *Numele poate avea spații în plus la margini (lupu dan ,1,8). Folosește strip() pe nume înainte de upper(). Ca să eviți împărțirea cu zecimale, compară suma notelor cu 5 înmulțit cu numărul de note.*

SOLUȚIA TA · PYTHON

SECȚIUNEA 7 · PROBLEME INTEGRATE

7 Probleme *integrate*

Acum le pui împreună: un fișier, o listă, o metodă de sortare aleasă cu motiv și subprograme care își împart munca. Așa arată programele adevărate.

O problemă integrată se rezolvă **modular** (CS 2.3, 3.3): o împarți în subprobleme, iar fiecare devine un subprogram cu un singur rol — unul salvează datele în fișier, unul le citește înapoi într-o listă, unul le ordonează. Avantajul se vede la testare: dacă rezultatul e greșit, verifici fiecare modul separat.

Schema tipică a unei aplicații din lecție are trei subprograme: `salveaza(n, nume_fisier)` scrie în fișier cele n linii citite de la tastatură; `citeste(nume_fisier)` deschide fișierul și întoarce lista înregistrărilor; un al treilea subprogram ordonează lista cu metoda potrivită datelor. Programul principal doar le apelează, în ordine.

CUM ALEGI METODA DE SORTARE

Valori întregi dintr-un interval mic (coduri 1–5)? Lista de frecvențe. Ai nevoie doar de primele k valori? Selecția, oprită după k pași. Lista e deja sortată sau aproape — adică niciun element nu trebuie să se deplaseze mult spre stânga — ori vrei ca programul să-și dea seama singur că

a terminat? Bulele cu fanion. Atenție: lista 2, 3, …, 10, 1 pare aproape sortată, dar valoarea 1 coboară o singură poziție la fiecare trecere, așa că bulele fac 90 de comparații, cât pe lista inversă.

UNDE SE FOLOSEȘTE ÎN VIAȚA REALĂ

Registrul de triaj al unei UPU face exact ce faci tu în prima problemă: citește pacienții sosiți, îi ordonează după gravitate fără să schimbe ordinea sosirii la același cod, și afișează lista pe ecranul din camera de gardă (acolo, într-o fereastră ca cea din Tkinter). Senzorii meteo trimit uneori valori absurde — 9999 sau −9999 când se defectează o clipă —, iar **mediana** (valoarea din mijlocul listei sortate) nu se lasă păcălită de ele, spre deosebire de medie; de aceea se folosește la filtrarea citirilor automate.

1

PROBLEMA 7.1 · COD GHIDAT

Registrul de triaj

SCRIERE COD

Se citește un număr natural n, apoi n linii, fiecare de forma Nume cod (numele pacientului, fără spații, și codul de triaj, între 1 și 5; codul 1 este cel mai grav), în ordinea sosirii. Scrie cele n linii în fișierul triaj.txt. Apoi citește fișierul înapoi și afișează pacienții în ordinea în care vor fi consultați: întâi toți cei cu codul 1, apoi cei cu codul 2 și așa mai departe; pacienții cu același cod rămân în ordinea sosirii. Pe fiecare linie afișezi numele, un spațiu și codul. Organizează programul în cel puțin trei subprograme: salvarea în fișier, citirea din fișier, ordonarea. Ex.: Pentru sosirile Ana 3, Dan 1, Ion 3, Ela 5, Radu 2 se afișează, pe linii: Dan 1, Radu 2, Ana 3, Ion 3, Ela 5. Ana și Ion au același cod, iar Ana a venit prima.

HINT **Atenție.** *Lista de frecvențe spune câți pacienți are fiecare cod, dar nu și cine sunt. Pentru fiecare cod prezent, parcurge lista pacienților în ordinea sosirii și afișează-i pe cei cu acel cod.*

SOLUȚIA TA · PYTHON

2

PROBLEMA 7.2 · COD GHIDAT

Mediana unui senzor meteo

SCRIERE COD

Se citește un număr natural impar n, apoi n citiri ale unui senzor de temperatură, în zecimi de grad (numere întregi; un senzor defect poate trimite 9999 sau −9999). Scrie citirile în fișierul senzor.txt, câte una pe linie, apoi citește-le înapoi într-o listă, ordonează lista cu metoda bulelor cu fanion (scrisă ca subprogram) și afișează mediana — valoarea aflată pe poziția din mijloc a listei ordonate. Ex.: Pentru n = 5 și citirile 182 185 9999 181 184, lista ordonată este 181 182 184 185 9999, iar mediana este 184. Media ar fi fost 2146,2 — adică 214,6 °C, un rezultat absurd provocat de o singură citire greșită.

HINT **Observație.** *Pentru n impar, poziția din mijloc (numărând de la 0) este n // 2.*

SOLUȚIA TA · PYTHON

3

PROBLEMA 7.3 · COD GHIDAT

Registrul de triaj, cu fereastră

SCRIERE COD

Transformă registrul de triaj într-o aplicație cu fereastră. Programul citește, ca la problema „Registrul de triaj”, un număr natural n și n linii de forma Nume cod, le salvează în fișierul triaj.txt, le citește înapoi și le ordonează după codul de triaj (codul 1 este cel mai grav; la același cod rămâne ordinea sosirii). Partea de fereastră este deja scrisă în editor: un Listbox care afișează pacienții în ordinea consultării, câte unul pe rând, sub forma Nume cod, și un buton Consulta. Completează cele trei subprograme și subprogramul consulta: la fiecare click, primul pacient iese din coadă și din Listbox, iar o fereastră-mesaj cu titlul Triaj anunță Urmeaza: Nume cod. Dacă nu mai are cine să intre, fereastra-mesaj anunță Nu mai sunt pacienti. Ex.: Pentru sosirile Ana 3, Dan 1, Ion 3, Ela 5, Radu 2, Listbox-ul arată la pornire Dan 1, Radu 2, Ana 3, Ion 3, Ela 5. După primul click pe Consulta apare fereastra-mesaj Urmeaza: Dan 1, iar în Listbox rămân Radu 2, Ana 3, Ion 3, Ela 5.

HINT Cum se verifică. *Pe pagină nu se pot deschide ferestre, așa că verificatorul folosește o fereastră simulată și joacă rolul asistentei de la triaj: după ce programul a citit pacienții, apasă butonul Consulta de câteva ori și citește ce arată fereastra. În rezultatele verificării, conținutul Listbox-ului apare între paranteze drepte — [Radu 2, Ana 3, Ion 3, Ela 5] —, iar fereastra-mesaj apare scrisă așa: [showinfo] Triaj: Urmeaza: Dan 1. Dacă uiți pack() sau mainloop(), verificatorul nu „vede” lista sau butonul — exact ca un utilizator adevărat.*

HINT Observație. *Pacienții așteaptă într-o coadă: primul intrat e primul care iese. Metoda pop(0) a listei scoate și întoarce primul element, iar delete(0) șterge primul rând din Listbox — cele două trebuie să rămână mereu în pas. Ca să afli dacă mai sunt pacienți, verifică lungimea listei coada.*

SOLUȚIA TA · PYTHON