test-e.ro

CAIET ELEV

Lecția 22

Metode de generare sistematică. Secvențe cu proprietăți date

Listele nu se primesc doar de-a gata: ele se construiesc.

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: secvențial, if/elif/else, while, for cu range, %, //, ord()/chr(), liste, indexare, len, for pe listă, metodele listei, def/return, import math, min/max/sum. Fără dicționare, fără fișiere, fără clase. Slicing-ul este permis de la această lecție, dar rămâne la nivel de curiozitate: toate soluțiile compară vecinii cu v[i] și v[i + 1]. Fără atribuiri simultane. Cifrele se parcurg doar cu while, cu % 10 și // 10 — nicio conversie la șir, niciun break în buclele de cifre. Paritatea se scrie n % 2 != 0. Fără diacritice în codul Python și în răspunsurile rebusului.

CUPRINS

1 Ne reamintim: lista, parcurgerea, cel mai bun de până acum

2 Cum se construiește o listă după o regulă

3 Când fiecare termen se naște din cel dinaintea lui

4 Perechile de vecini dintr-un șir .....

5 Grupuri de elemente alăturate care au aceeași proprietate

6 Secvențe în care fiecare termen se raportează la cel dinaintea lui

7 Cod ghidat · cinci probleme .....

SECȚIUNEA 1 · REACTUALIZARE: LISTE, PARCURGERE, CEL MAI BUN DE PÂNĂ ACUM

1 Ne reamintim: lista, parcurgerea, cel mai bun de până *acum*

Înainte de a construi liste noi, ne asigurăm că le știm citi pe cele existente: cum se parcurg cu `for`, cum se adaugă un element cu `append`, cum se ține minte cea mai bună valoare întâlnită până într-un anumit moment.

O listă este o înșiruire ordonată de valori, fiecare cu poziția ei. Poziția se numește **indice** și se numără de la 0: în `v = [4, 7, 10, 3]`, elementul `v[0]` este 4, iar ultimul este `v[3]`, adică `v[len(v) - 1]`. Funcția `len(v)` spune câte elemente are lista, iar metoda `v.append(x)` adaugă o valoare nouă la sfârșit, mărind lista cu o poziție.

Lista se parcurge în două feluri, ambele învățate: cu indice, `for i in range(len(v))`, când ai nevoie și de poziție; sau direct pe valori, `for x in v`, când ai nevoie doar de conținut.

Aceeași idee în Python (cu sintaxă reală):

```
v = [4, 7, 10, 3]
for i in range(len(v)):
    print(i, v[i])
```

Al doilea lucru pe care îl reluăm este **tiparul „cel mai bun de până acum”**. Nu putem afla maximul dintr-o singură privire, așa cum facem noi când ne uităm pe o listă scrisă pe hârtie: programul vede un element o dată, apoi trece mai departe. Deci ținem minte, într-o variabilă, cea mai bună valoare întâlnită până în acel moment, și o înlocuim doar când găsim ceva mai bun.

```
mx = v[0]
for i in range(len(v)):
    if v[i] > mx:
        mx = v[i]
print(mx)
```

> **DE REȚINUT**
>
> Variabila care ține „cel mai bun de până acum” se inițializează cu **primul element**, nu cu 0. Dacă toate valorile ar fi negative, pornirea de la 0 ar da un răspuns greșit.

A

EXERCIȚIUL A · GRILĂ

Ce ne amintim despre liste

GRILĂ

Încercuiește răspunsul corect la fiecare întrebare.

1. Ce valoare are len(v) pentru v = [4, 7, 10, 3]?

   - `10, pentru că este cel mai mare element din listă`
   - `24, pentru că este suma tuturor elementelor listei`
   - `4, pentru că lista are patru elemente`
   - `3, pentru că indicii merg de la 0 la 3`

2. Pentru v = [5, 2, 9], ce afișează print(v[0] + v[2])?

   - `11, adică suma ultimelor două elemente ale listei`
   - `16, adică suma tuturor elementelor listei`
   - `7, adică suma primelor două elemente ale listei`
   - `14, adică suma primului și a ultimului element`

3. Care instrucțiune adaugă valoarea 8 la sfârșitul listei v?

   - `v.add(8), metoda standard de adăugare din Python`
   - `v.insert(8), care pune valoarea pe ultima poziție`
   - `v.append(8)`
   - `v = v + 8, care lipește valoarea la capătul listei`

4. Ce afișează print(max(v) - min(v)) pentru v = [3, 1, 4]?

   - `8, pentru că este suma tuturor elementelor listei`
   - `1, pentru că este cel mai mic element al listei`
   - `4, pentru că este cel mai mare element al listei`
   - `3, adică diferența dintre cel mai mare și cel mai mic element`

B

EXERCIȚIUL B · DRY-RUN

Suma și maximul, pas cu pas

DRY-RUN

Urmărește bucla rând cu rând. Coloanele i și v[i] sunt date; tu completezi coloanele s (suma acumulată) și mx (cel mai mare element de până acum), după executarea pasului. Ambele coloane conțin valori numerice. Ultimul rând arată starea în care bucla se oprește.

```
1  v = [4, 7, 10, 3]
2  s = 0
3  mx = v[0]
```

```
for i in range(len(v)):
    s = s + v[i]
    if v[i] > mx:
        mx = v[i]
print(s, mx)
```

i	v[i]	s (completeaza)	mx (completeaza)
— (înainte de buclă)	—	0	4
0	4		
1	7		
2	10		
3	3		
4 — bucla se oprește	—		

HINT *La fiecare pas s crește cu v[i], iar mx se schimbă doar când v[i] este mai mare decât mx. Ia rândurile pe rând, de sus în jos.*

SPRIJIN · SCRIE CODUL

Suma și cel mai mare element

SCRIERE COD

Se citește un număr natural n (n ≥ 1), apoi n numere naturale, fiecare de pe câte un rând. Afișează pe același rând, separate printr-un spațiu, suma celor n numere și cel mai mare dintre ele. Cerință: calculează ambele valori cu bucla proprie, fără sum și fără max. Ex.: pentru n = 4 și numerele 4, 7, 10, 3: suma este 4 + 7 + 10 + 3 = 24, iar cel mai mare element este 10, deci se afișează 24 10.

HINT **Atenție.** *Variabila în care ții maximul se pornește de la v[0], nu de la 0.*

SOLUȚIA TA · PYTHON

SPRIJIN · SCRIE CODUL

Câte valori sunt pare?

SCRIERE COD

Se citește un număr natural n (n ≥ 1), apoi n numere naturale, fiecare de pe câte un rând. Afișează câte dintre ele sunt pare. Ex.: pentru n = 5 și numerele 23, 45, 6, 22, 8: pare sunt 6, 22 și 8, deci se afișează 3.

HINT *Zero este număr par, deci intră în numărătoare.*

SOLUȚIA TA · PYTHON

SECȚIUNEA 2 · GENERAREA SISTEMATICĂ: DOMENIU, REGULĂ, COLECTARE

2 Cum se construiește o listă după o *regulă*

Nu toate listele vin de la tastatură. Multe se **construiesc**: alegem un domeniu de valori, îl parcurgem în ordine, verificăm pentru fiecare valoare o proprietate și păstrăm doar ce trece verificarea.

A genera **sistematic** înseamnă a trece prin toate valorile posibile dintr-un domeniu, în ordine, exact o dată fiecare — fără să sari, fără să repeți, fără să ghicești. Cuvântul e important: o căutare la întâmplare ar putea rata valori sau le-ar putea găsi de două ori, iar noi vrem certitudinea că am văzut tot.

Orice generare sistematică are trei părți, mereu aceleași:

> **TIPARUL GENERĂRII**
>
> **(1) Domeniul** — de la ce valoare până la ce valoare mergem. **(2) Proprietatea** — condiția pe care valoarea trebuie s-o îndeplinească pentru a fi păstrată. **(3) Colectarea** — lista în care punem, cu `append`, doar valorile care au trecut verificarea.

Aceeași idee în Python (cu sintaxă reală):

```
a = int(input())
b = int(input())
rezultat = []
for x in range(a, b + 1):
    if x % 7 == 0:
        rezultat.append(x)
print(len(rezultat))
```

Domeniul e `range(a, b + 1)` — atenție la `+ 1`, altfel capătul `b` nu e verificat. Proprietatea e `x % 7 == 0`. Colectarea e `rezultat.append(x)`. Schimbând doar linia cu proprietatea, același program generează cu totul altceva: multipli, pătrate perfecte, numere palindrome, numere cu suma cifrelor dată.

Un exemplu de proprietate mai interesantă este **palindromul**: un număr care se citește la fel de la stânga la dreapta și invers, ca 121 sau 3443. Ca să-l verificăm, construim oglinda numărului, cifră cu cifră, cu împărțiri repetate la 10, și o comparăm cu numărul de la care am pornit.

```
n = x
inv = 0
while n > 0:
    inv = inv * 10 + n % 10
```

```
5       n = n // 10
```

ATENȚIE

Bucla de mai sus **distruge** valoarea pe care o parcurge: la final `n` ajunge 0. De aceea copiem întâi `x` în `n` și lucrăm pe copie — altfel n-am mai avea cu ce compara oglinda.

UNDE SE FOLOSEȘTE ÎN VIAȚA REALĂ

(1) Filtrele dintr-un magazin online — „arată-mi toate produsele între 100 și 300 de lei, disponibile în stoc” — sunt exact o generare sistematică peste un domeniu, cu o proprietate de verificat. (2) Un program care caută toate zilele de 13 care cad într-o zi de vineri într-un an dat parcurge sistematic toate datele calendaristice și le păstrează pe cele care trec testul. (3) Generatoarele de coduri de reducere produc sistematic toate combinațiile posibile de litere și cifre de o anumită lungime, apoi le păstrează pe cele care nu au fost încă folosite.

A

EXERCIȚIUL A · SCRIE CODUL

Numerele palindrome dintr-un interval

SCRIERE COD

Se citesc două numere naturale a și b (1 ≤ a ≤ b), fiecare de pe câte un rând. Generează, în ordine crescătoare, toate numerele palindrome din intervalul [a, b] și adaugă-le într-o listă. Afișează pe primul rând câte sunt, iar pe al doilea rând valorile, separate prin câte un spațiu. Se garantează că există cel puțin un astfel de număr. Ex.: pentru a = 10 și b = 40, numerele palindrome sunt 11, 22 și 33, deci se afișează 3 pe primul rând și 11 22 33 pe al doilea.

HINT Atenție. *Un număr care se termină în 0, cum e 100, nu este palindrom: oglinda lui este 001, adică 1, iar 1 nu este egal cu 100.*

HINT Curiozitate. *Anul 2002 a fost palindrom. Următorul an palindrom este 2112 — deci nimeni dintre noi nu-l va prinde pe următorul după acela.*

SOLUȚIA TA · PYTHON

B

EXERCIȚIUL B · ORDONARE

Pașii unei generări sistematice

ORDONARE

Numerotează pașii de la 1 la 6, în ordinea în care se execută într-un program de generare sistematică. Pașii îți apar amestecați.

- [ ] Stabilim domeniul: de la ce valoare până la ce valoare mergem
- [ ] Pregătim o listă goală, în care vom colecta rezultatele
- [ ] După terminarea parcurgerii, afișăm câte valori am colectat și care sunt
- [ ] Pentru valoarea curentă verificăm dacă îndeplinește proprietatea cerută
- [ ] Dacă o îndeplinește, o adăugăm în listă cu append
- [ ] Parcurgem domeniul în ordine, valoare cu valoare

SECȚIUNEA 3 · SECVENȚE CU PAS CONSTANT: PROGRESII ȘI SERII

3 Când fiecare termen se naște din cel dinaintea *lui*

Unele secvențe nu se obțin filtrând un domeniu, ci construind termen după termen, după o regulă fixă: pornim de la o valoare și adunăm mereu același pas.

O **progresie aritmetică** este o secvență în care fiecare termen se obține din cel dinaintea lui adunând o valoare constantă, numită **rație** și notată de obicei cu `r`. Dacă primul termen este `a`, atunci al doilea este `a + r`, al treilea `a + 2r`, și așa mai departe. Pentru `a = 5` și `r = 3`, primii cinci termeni sunt `5, 8, 11, 14, 17`.

REGULA DE CONSTRUCȚIE

`t(0) = a`, iar `t(i) = t(i - 1) + r`. În program nu ai nevoie de toți termenii anteriori: ții o singură variabilă `t`, o adaugi în listă, apoi o „urci” cu `r`, și tot așa de n ori.

Aceeași idee în Python (cu sintaxă reală):

```
t = a
for i in range(n):
    v.append(t)
    t = t + r
```

Ordinea celor două linii din buclă contează. Dacă adaugi întâi `r` și abia apoi pui valoarea în listă, primul termen `a` nu mai apare niciodată în rezultat — secvența începe de la al doilea.

Odată ce ai secvența într-o listă, poți afla suma ei fie cu bucla proprie, fie direct cu `sum(v)`, funcția predefinită învățată la lecția trecută. La progresia aritmetică suma se poate calcula și cu o formulă, fără să generezi termenii: `S = n * (2 * a + (n - 1) * r) / 2`. Pentru `a = 5`, `r = 3`, `n = 5` formula dă `5 * (10 + 12) / 2 = 55` — exact cât obținem adunând `5 + 8 + 11 + 14 + 17`.

ATENȚIE

Rația poate fi și 0. Atunci toți termenii sunt egali cu `a`, iar secvența are n termeni identici — cazul e legal și trebuie să funcționeze, deci nu trata `r = 0` ca pe o eroare.

UNDE SE FOLOSEȘTE ÎN VIAȚA REALĂ

(1) Ratele fixe la un împrumut sau la o rată de telefon formează exact o progresie: aceeași sumă, lună de lună, iar suma progresiei e cât plătești în total. (2) Locurile dintr-o sală de spectacol numerotate din 2 în 2 pe un rând, sau stațiile unui autobuz care vine din 12 în 12 minute, sunt secvențe cu pas constant. (3) Marcajele de pe rigla unui grafic (0, 50, 100, 150 …) sunt generate de programul de desenare exact cu bucla de mai sus.

A

EXERCIȚIUL A · SCRIE CODUL

Termenii unei progresii aritmetice

SCRIERE COD

Se citesc trei numere naturale, fiecare de pe câte un rând: a (primul termen), r (rația) și n (câți termeni se generează, n ≥ 1). Generează primii n termeni ai progresiei într-o listă. Afișează pe primul rând termenii, separați prin câte un spațiu, iar pe al doilea rând suma lor. Cerință: scrie generarea într-un subprogram care primește a, r și n și întoarce lista. Ex.: pentru a = 5, r = 3 și n = 5 se generează 5 8 11 14 17, iar suma lor este 55, deci se afișează 5 8 11 14 17 pe primul rând și 55 pe al doilea.

HINT *Rația poate fi 0: pentru a = 7, r = 0 și n = 4 se obțin patru termeni egali cu 7.*

HINT **Atenție.** *În buclă, adaugi întâi valoarea curentă în listă și abia apoi o mărești cu r. În ordinea inversă, primul termen se pierde.*

SOLUȚIA TA · PYTHON

B

EXERCIȚIUL B · DRY-RUN

Cum crește suma unei progresii

DRY-RUN

Urmărește generarea progresiei cu a = 5 și r = 3. Coloanele i și t la intrarea în pas sunt date; tu completezi coloanele S (suma acumulată după pas) și t după pas. Ambele coloane conțin valori numerice. Ultimul rând arată starea în care bucla se oprește.

```
a = 5
r = 3
n = 5
t = a
S = 0
for i in range(n):
    S = S + t
    t = t + r
print(S, t)
```

i	t la intrarea în pas	S (completeaza)	t după pas (completeaza)
— (înainte de buclă)	5	0	—
0	5		
1	8		
2	11		
3	14		
4	17		
5 — bucla se oprește	—		

HINT *La fiecare pas se adună întâi t la S, și abia apoi t crește cu r. Coloana t după pas e mereu cu 3 mai mare decât t de la intrare.*

SECȚIUNEA 4 · COMPARAREA ELEMENTELOR ALĂTURATE

4 Perechile de vecini dintr-un *șir*

Multe întrebări despre o listă nu se pun despre un element singur, ci despre doi termeni alăturați: cu cât a crescut temperatura de ieri până azi, unde a fost cea mai mare săritură, dacă șirul urcă sau coboară.

Într-un șir cu `n` elemente, perechile de termeni aflați pe poziții consecutive sunt `(v[0], v[1])`, `(v[1], v[2])`, …, `(v[n-2], v[n-1])`. Sunt exact **n − 1** perechi, nu n: ultimul element nu are pe cine să urmeze după el.

> **ATENȚIE**
> Bucla care compară vecini merge până la `n - 2` inclusiv, adică se scrie `for i in range(n - 1)`. Dacă scrii `range(n)`, la ultimul pas `v[i + 1]` iese din listă, iar programul se oprește cu eroare de indice.

Aceeași idee în Python (cu sintaxă reală):

```
for i in range(n - 1):
    d = v[i] - v[i + 1]
    if d < 0:
```

```
        d = -d
    print(v[i], v[i + 1], d)
```

Cu perechile de vecini putem răspunde la întrebări de trei feluri. **Măsurăm** — cât de mare e diferența dintre doi termeni alăturați, și unde e cea mai mare. **Comparăm** — dacă fiecare termen e mai mare decât cel dinaintea lui, șirul e strict crescător; dacă e mai mare sau egal, e nedescrescător. **Numărăm** — câte perechi de vecini au aceeași paritate, câte au sumă pară, și așa mai departe.

DE REȚINUT

Diferența „în valoare absolută” înseamnă că nu ne interesează dacă șirul a urcat sau a coborât, ci **cu cât** s-a schimbat. O calculăm scăzând și, dacă rezultatul a ieșit negativ, schimbându-i semnul.

DE REȚINUT

Python are și o scriere prescurtată pentru „lista fără primul element”, `v[1:]`, și pentru „lista fără ultimul element”, `v[:-1]`. Ele permit compararea vecinilor fără indici. O menționăm ca pe o curiozitate: în această lecție lucrăm cu `v[i]` și `v[i + 1]`, pentru că indicii arată limpede *ce* se compară cu *ce*.

UNDE SE FOLOSEȘTE ÎN VIAȚA REALĂ

(1) Aplicația de vreme care spune „azi e cu 6 grade mai cald decât ieri” compară exact doi termeni alăturați dintr-un șir de temperaturi. (2) Un ceas inteligent care detectează o creștere bruscă a pulsului se uită la diferența dintre două măsurători consecutive, nu la valoarea absolută. (3) Programele care desenează graficul unei acțiuni la bursă colorează segmentul dintre două zile în verde sau roșu tocmai comparând valoarea de azi cu cea de ieri.

A

EXERCIȚIUL A · SCRIE CODUL

Cea mai mare săritură dintre doi vecini

SCRIERE COD

Se citește un număr natural n (n ≥ 2), apoi n numere naturale, fiecare de pe câte un rând. Afișează pe același rând, separate prin câte un spațiu: cea mai mare diferență în valoare absolută dintre doi termeni aflați pe poziții consecutive, apoi cei doi termeni care o realizează, în ordinea în care apar în șir. Dacă mai multe perechi dau aceeași diferență maximă, se afișează prima dintre ele. Ex.: pentru n = 5 și numerele 3, 8, 5, 2, 6, diferențele dintre vecini sunt 5, 3, 3 și 4, deci cea mai mare este 5 și se obține între 3 și 8. Se afișează 5 3 8.

HINT *Dacă toate valorile sunt egale, toate diferențele sunt 0, iar răspunsul este 0 urmat de primii doi termeni.*

HINT **Atenție.** *Bucla merge până la n - 2, adică for i in range(n - 1). Reține poziția i la care s-a găsit maximul, ca să poți afișa la final ambii termeni.*

SOLUȚIA TA · PYTHON

B

EXERCIȚIUL B · ASOCIERE

Ce fel de șir este?

ASOCIERE

Fiecărui șir din stânga îi corespunde o singură descriere. Compară fiecare termen cu cel dinaintea lui, apoi scrie în căsuță litera descrierii potrivite. Fiecare literă se folosește o singură dată.

☐	1 3 6 10
☐	2 5 5 9
☐	9 7 4 1
☐	3 8 5 2

A Nedescrescător, dar nu strict crescător: există doi vecini egali

B Strict crescător: fiecare termen este mai mare decât cel dinaintea lui

C Fără monotonie: șirul și urcă, și coboară

D Strict descrescător: fiecare termen este mai mic decât cel dinaintea lui

SECȚIUNEA 5 · CEA MAI LUNGĂ SECVENȚĂ DE ELEMENTE CU O PROPRIETATE

5 Grupuri de elemente alăturate care au aceeași *proprietate*

Aici ajungem la algoritmul central al lecției: nu ne mai interesează *câte* elemente au o proprietate, ci **cât de lung** este cel mai lung grup de elemente alăturate care o au toate.

Într-un șir, o **secvență** este un grup de termeni aflați pe poziții consecutive — nu oricare termeni aleși de ici-colo, ci o bucată continuă din șir. Întrebarea lecției este: care e cea mai lungă bucată continuă în care **toate** elementele au o anumită proprietate?

Pentru șirul `23 45 6 22 8 5 6 8 0 10` și proprietatea „este par”, bucățile continue de numere pare sunt `6 22 8` (3 termeni) și `6 8 0 10` (4 termeni). Răspunsul este 4.

Algoritmul ține două contoare deodată, și tocmai asta e ideea lui:

> **CONTORUL CURENT ȘI RECORDUL**
>
> `l` este lungimea secvenței în care ne aflăm **chiar acum**. `lmax` este cea mai lungă secvență găsită **până acum**. Cât timp elementul curent are proprietatea, `l` crește cu 1. Când dăm peste un element care nu o are, secvența s-a rupt: comparăm `l` cu `lmax` și repornim `l` de la 0.

Aceeași idee în Python (cu sintaxă reală):

```
lmax = 0
l = 0
for i in range(n):
    if v[i] % 2 == 0:
        l = l + 1
    else:
        if l > lmax:
            lmax = l
        l = 0
if l > lmax:
    lmax = l
print(lmax)
```

Ultimele două rânduri, cele de **după** buclă, sunt cele mai importante din toată lecția.

> **ATENȚIE**

Comparația cu `lmax` se face în ramura `else`, adică doar când secvența **se rupe**. Dacă șirul se termină chiar în interiorul unei secvențe — cum se întâmplă în exemplul de mai sus, care se încheie cu `10` — ramura `else` nu se mai execută niciodată pentru ultima secvență, iar aceasta rămâne necomparată. De aceea, după buclă, mai facem o dată comparația. Fără ea, programul ar răspunde 3 în loc de 4.

Schimbând o singură linie — condiția din `if` — același algoritm rezolvă zeci de probleme diferite: cea mai lungă secvență de numere divizibile cu 5, de numere strict pozitive, de note peste 7, de zile fără ploaie.

UNDE SE FOLOSEȘTE ÎN VIAȚA REALĂ

(1) „Seria” din Duolingo sau din aplicațiile de fitness — câte zile la rând ai avut activitate — e exact contorul `l`, iar recordul tău personal e `lmax`. (2) Un program de analiză a calității aerului caută cea mai lungă perioadă continuă în care valoarea a depășit pragul de alertă. (3) Programele care comprimă imagini caută șiruri lungi de pixeli identici alăturați și le înlocuiesc cu „această culoare, de 40 de ori” — cu cât secvențele sunt mai lungi, cu atât fișierul e mai mic.

A

EXERCIȚIUL A · DRY-RUN

Cum evoluează l și lmax

DRY-RUN

Urmărește algoritmul pe șirul dat. Coloanele i și v[i] sunt date; tu completezi coloanele l și lmax cu valorile lor după executarea pasului. Ambele coloane conțin valori numerice. Ultimul rând, cel de după buclă, arată de ce e nevoie de comparația finală.

```
v = [23, 45, 6, 22, 8, 5, 6, 8, 0, 10]
lmax = 0
l = 0
for i in range(len(v)):
    if v[i] % 2 == 0:
        l = l + 1
    else:
        if l > lmax:
            lmax = l
        l = 0
if l > lmax:
    lmax = l
print(lmax)
```

i	v[i]	l (completeaza)	lmax (completeaza)
– (înainte de buclă)	–	0	0
0	23		
1	45		
2	6		
3	22		
4	8		
5	5		
6	6		
7	8		
8	0		
9	10		
după buclă (comparația finală)	–		

HINT *Cât timp v[i] e par, l crește cu 1. Când v[i] e impar, întâi se compară l cu lmax, apoi l se face 0. Ultimul rând e comparația de după buclă.*

B

EXERCIȚIUL B · SCRIE CODUL

Cea mai lungă secvență de numere pare

SCRIERE COD

Se citește un număr natural n (n ≥ 1), apoi n numere naturale, fiecare de pe câte un rând. Afișează lungimea celei mai lungi secvențe de numere pare aflate pe poziții consecutive în șir. Dacă în șir nu există niciun număr par, se afișează 0. Ex.: pentru n = 10 și numerele 23, 45, 6, 22, 8, 5, 6, 8, 0, 10, secvențele de numere pare sunt 6 22 8 (3 termeni) și 6 8 0 10 (4 termeni), deci se afișează 4.

HINT **Atenție.** *Nu uita comparația de după buclă. Dacă șirul se termină chiar în mijlocul unei secvențe, aceasta nu a fost încă pusă față în față cu recordul.*

HINT *Zero este număr par, deci face parte din secvențe.*

SOLUȚIA TA · PYTHON

PERFORMANȚĂ · SCRIE CODUL

Unde începe cea mai lungă secvență?

SCRIERE COD

Se citește un număr natural n (n ≥ 1), apoi n numere naturale, fiecare de pe câte un rând. Afișează pe același rând, separate printr-un spațiu, lungimea celei mai lungi secvențe de numere pare aflate pe poziții consecutive și poziția, numerotată de la 1, la care începe prima secvență de lungime maximă. Dacă în șir nu există niciun număr par, se afișează 0 0. Ex.: pentru n = 10 și numerele 23, 45, 6, 22, 8, 5, 6, 8, 0, 10, cea mai lungă secvență de numere pare este 6 8 0 10, are 4 termeni și începe pe poziția 7, deci se afișează 4 7.

HINT **Atenție.** *Nu e suficient să numeri: trebuie să și reții unde a început secvența curentă. O variabilă în plus, actualizată exact în momentul în care l trece de la 0 la 1, rezolvă problema.*

HINT *„Prima secvență de lungime maximă” înseamnă că, la egalitate, se păstrează cea găsită mai devreme — deci compari cu >, nu cu >=.*

SOLUȚIA TA · PYTHON

SECȚIUNEA 6 · CÂND PROPRIETATEA DEPINDE DE ELEMENTUL PRECEDENT

6 Secvențe în care fiecare termen se raportează la cel dinaintea *lui*

Uneori proprietatea nu se poate verifica privind un singur element. „Este par” se vede dintr-o privire; „este mai mare decât cel dinaintea lui” are nevoie de doi termeni deodată.

Există două feluri de proprietăți, iar algoritmul se schimbă în funcție de care dintre ele avem. Proprietățile **locale** se testează pe un singur element: par, divizibil cu 5, mai mare decât 100. Proprietățile **relative** se testează pe o pereche: „nu e mai mic decât precedentul”, „e egal cu precedentul”, „diferă de precedentul cu cel mult 2”.

Când proprietatea este relativă, algoritmul din secțiunea anterioară se modifică în trei locuri:

> **ALGORITMUL CU ELEMENT PRECEDENT**
>
> **(1)** Bucla pornește de la poziția 1, nu de la 0 — primul element nu are un predecesor cu care să fie comparat. **(2)** Contorul `l` pornește de la **1**, nu de la 0 — un singur element este deja o secvență de lungime 1. **(3)** La rupere, `l` se reia tot de la **1**, nu de la 0 — pentru că elementul care a rupt secvența începe deja o secvență nouă.

Aceeași idee în Python (cu sintaxă reală):

```
1  lmax = 0
2  l = 1
3  for i in range(1, n):
4      if v[i - 1] <= v[i]:
```

```
        l = l + 1
    else:
        if l > lmax:
            lmax = l
        l = 1
if l > lmax:
    lmax = l
print(lmax)
```

Pentru șirul `3 8 5 2 6 8 9 1 2` și proprietatea „nu este mai mic decât precedentul”, secvențele nedescrescătoare sunt `3 8`, apoi `5`, apoi `2 6 8 9`, apoi `1 2`. Cea mai lungă are 4 termeni.

ATENȚIE

Confuzia cea mai frecventă este reinițializarea cu 0 în loc de 1. Cu `l = 0` la rupere, un șir strict descrescător ca `5 4 3 2 1` ar da răspunsul 0, deși răspunsul corect este 1 — fiecare element formează, singur, o secvență nedescrescătoare de lungime 1.

AȘA ARATĂ REZULTATUL CORECT

Pentru `5 4 3 2 1` răspunsul este `1`. Pentru `2 2 2 2` răspunsul este `4`, pentru că `2 <= 2` este adevărat, deci egalitatea nu rupe o secvență nedescrescătoare.

UNDE SE FOLOSEȘTE ÎN VIAȚA REALĂ

(1) Un jurnal al temperaturilor medii anuale poate fi analizat ca să afli cea mai lungă perioadă de ani în care temperatura a crescut fără întrerupere. (2) O aplicație de învățare care îți arată „cea mai lungă serie de teste cu punctaj în creștere” folosește exact acest algoritm. (3) Sistemele care verifică senzorii unei mașini caută secvențe lungi de măsurători identice — un semn că senzorul s-a blocat și nu mai măsoară nimic.

A

EXERCIȚIUL A · SCRIE CODUL

Cea mai lungă secvență nedescrescătoare

SCRIERE COD

Se citește un număr natural n (n ≥ 1), apoi n numere naturale, fiecare de pe câte un rând. Afișează lungimea celei mai lungi secvențe de termeni aflați pe poziții consecutive în care fiecare termen este mai mare sau egal decât cel dinaintea lui. Cerință: scrie prelucrarea într-un subprogram care primește lista și întoarce lungimea căutată. Ex.: pentru n = 9 și numerele 3, 8, 5, 2, 6, 8, 9, 1, 2, secvențele nedescrescătoare sunt 3 8, 5, 2 6 8 9 și 1 2, deci cea mai lungă are 4 termeni și se afișează 4.

HINT **Atenție.** *Contorul pornește de la 1 și, la fiecare rupere, se reia tot de la 1. Cu 0 în loc de 1, un șir strict descrescător ar da răspunsul 0 în loc de 1.*

HINT *Doi termeni egali nu rup o secvență nedescrescătoare, pentru că 2 ≤ 2 este adevărat.*

SOLUȚIA TA · PYTHON

EXERCIȚIUL B · COMPLETARE

B *Regula secvențelor cu element precedent*

COMPLETARE

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

`precedent` `1` `curentă` `0` `oprește` `începe`

Când proprietatea depinde de elementul _____, comparăm perechea formată din valoarea anterioară și cea _____. Contorul de lungime pornește de la _____ și, la fiecare rupere, se reia de la aceeași valoare, pentru că elementul care a rupt secvența _____ una nouă.

SECȚIUNEA 7 · COD GHIDAT

7 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

Numerele cu suma cifrelor dată

SCRIERE COD

Se citesc trei numere naturale, fiecare de pe câte un rând: a, b (1 ≤ a ≤ b) și s. Generează toate numerele din intervalul [a, b] a căror sumă a cifrelor este egală cu s și adaugă-le într-o listă. Afișează pe primul rând câte sunt, iar pe al doilea rând valorile, în ordine crescătoare, separate prin câte un spațiu. Se garantează că există cel puțin un astfel de număr. Ex.: pentru a = 10, b = 30 și s = 5, numerele cu suma cifrelor 5 sunt 14 (1 + 4 = 5) și 23 (2 + 3 = 5), deci se afișează 2 pe primul rând și 14 23 pe al doilea.

HINT **Atenție.** *Bucla care adună cifrele distruge numărul pe care îl parcurge. Copiază valoarea curentă într-o a doua variabilă înainte de a intra în buclă.*

HINT *Cifrele se parcurg cu while, prin % 10 și // 10 — nu prin transformarea numărului în șir de caractere.*

SOLUȚIA TA · PYTHON

2

PROBLEMA 7.2 · COD GHIDAT

Perechi de vecini cu aceeași paritate

SCRIERE COD

Se citește un număr natural n (n ≥ 2), apoi n numere naturale, fiecare de pe câte un rând. Afișează câte perechi de termeni aflați pe poziții consecutive au aceeași paritate, adică sunt amândoi pari sau amândoi impari. Ex.: pentru n = 5 și numerele 3, 8, 5, 2, 6, perechile de vecini sunt (3, 8), (8, 5), (5,

2) și (2, 6); doar ultima are ambii termeni de aceeași paritate, deci se afișează 1.

HINT *Doi termeni au aceeași paritate exact atunci când resturile lor la împărțirea cu 2 sunt egale.*

HINT **Atenție.** *Într-un șir cu n termeni există exact n − 1 perechi de vecini, deci bucla merge până la n - 2 inclusiv.*

SOLUȚIA TA · PYTHON

3 PROBLEMA 7.3 · COD GHIDAT

Cea mai lungă secvență de multipli de 5

SCRIERE COD

Se citește un număr natural n (n ≥ 1), apoi n numere naturale, fiecare de pe câte un rând. Afișează lungimea celei mai lungi secvențe de numere divizibile cu 5 aflate pe poziții consecutive în șir. Dacă în șir nu există niciun număr divizibil cu 5, se afișează 0. Ex.: pentru n = 8 și numerele 5, 10, 3, 15, 20, 25, 7, 30, secvențele de multipli de 5 sunt 5 10 (2 termeni), 15 20 25 (3 termeni) și 30 (1 termen), deci se afișează 3.

HINT **Atenție.** *Șirul se poate termina chiar în interiorul unei secvențe. Fă comparația finală și după terminarea buclei.*

HINT *Zero este divizibil cu 5, pentru că 0 : 5 dă restul 0.*

SOLUȚIA TA · PYTHON

4

PROBLEMA 7.4 · REPARĂ CODUL

Programul care uită ultima secvență

SCRIERE COD

Un coleg a scris un program care ar trebui să afișeze lungimea celei mai lungi secvențe de numere mai mari decât 0 aflate pe poziții consecutive. Programul lui:

```
n = int(input())
v = []
for i in range(n):
    x = int(input())
    v.append(x)
lmax = 0
l = 0
for i in range(n):
    if v[i] > 0:
        l = l + 1
    else:
        if l > lmax:
            lmax = l
        l = 0
print(lmax)
```

Pentru n = 7 și numerele 0, 4, 5, 0, 1, 2, 3 programul afișează 2, deși cea mai lungă secvență de numere mai mari decât 0 este 1 2 3 și are 3 termeni. Corectează programul.

Ex.: pentru n = 5 și numerele 1, 2, 3, 4, 5, programul stricat afișează 0, deși toate cele cinci numere sunt mai mari decât 0 și formează o singură secvență de lungime 5.

HINT **Atenție.** *Nu adăuga o a doua buclă. Greșeala e de un singur rând, iar el lipsește după buclă, nu în interiorul ei.*

HINT *Programul stricat dă răspunsuri corecte pe șirurile care se termină cu 0 — de aceea greșeala poate trece neobservată dacă nu testezi și șiruri care se termină cu o valoare mai mare decât 0.*

PROGRAMUL CORECTAT · PYTHON

PROBLEMA 7.5 · PROVOCARE

Cea mai lungă coborâre și unde începe

SCRIERE COD

Se citește un număr natural n (n ≥ 1), apoi n numere naturale, fiecare de pe câte un rând. Afișează pe același rând, separate printr-un spațiu, lungimea celei mai lungi secvențe de termeni aflați pe poziții consecutive care sunt strict descrescători și poziția, numerotată de la 1, la care începe prima secvență de lungime maximă. Cerință: scrie prelucrarea într-un subprogram care primește lista și afișează cele două valori. Ex.: pentru n = 5 și numerele 3, 8, 5, 2, 6, cea mai lungă secvență strict descrescătoare este 8 5 2, are 3 termeni și începe pe poziția 2, deci se afișează 3 2.

HINT **Atenție.** *Contorul pornește de la 1, iar variabila care reține începutul secvenței curente se actualizează exact în momentul ruperii, nu la fiecare pas. Pentru un șir cu un singur element răspunsul este 1 1.*

HINT *Doi termeni egali rup o secvență strict descrescătoare, spre deosebire de cea nedescrescătoare din secțiunea 6, unde egalitatea era permisă.*

SOLUȚIA TA · PYTHON