FIȘĂ DE LUCRU · PYTHON · CLASA A IX-A · FIȘA 16 · LECȚIA PY9-16

Generarea sistematică a elementelor unei liste

Durată: 150 min Nivel: mate-info · Python

CE VEI ȘTI DUPĂ LECȚIE

- Prezici termenii generați de o regulă, de o expresie sau de o recurență.
- Repari generatoare cu greșeli de oprire, de inițializare sau de actualizare.
- Scrii generatoare complete, cu criteriul de oprire cerut, și le testezi pe cazurile limită.

Ce trebuie să știi (o pagină)

Situația	Schema în Python	Atenție
progresie aritmetică	`list(range(prim, ultim + 1, pas))`	capătul din dreapta nu e inclus
termeni cu o proprietate	`for x in range(a, b + 1): if conditie: rez.append(x)`	dacă poți sări direct la termenii buni, sari
primii `n` termeni	`for _ in range(n):` `rez.append(t)`, apoi actualizezi `t`	`n = 0` trebuie să dea `[]`
termenii `≤ m`	`while t <= m:` `rez.append(t)`, apoi actualizezi `t`	`<` sau `≤`? testează cu `m` egal cu un termen
până la o condiție	`while not gata:` … `pasi += 1`	condiția trebuie să se schimbe, altfel buclă infinită
termen din precedentul	`p *= i`, `t /= 2`, `semn = -semn`	inițializezi **o dată**, înainte de buclă
Fibonacci	`a, b = 1, 1` · `a, b = b, a + b`	atribuirea simultană; `a = b; b = a + b` e greșit
recurență cu formulă	`a(n) = a(0) + n·r`, `a(n) = a(0)·qⁿ`	verifică formula pe primii termeni generați

Test obligatoriu	De ce
`n = 0`, `n = 1`, `n = 2`	inițializările „scapă” termeni în plus
limita sub primul termen	lista trebuie să fie vidă
limita egală cu un termen	prinde confuzia `<` / `≤`
primii 4–5 termeni, scriși de mână **înainte** de cod	confirmă regula și operația de trecere la termenul următor

Nivelul 1 · Prezice

★ citește și prezice

P1. Ce afișează `t = list(range(3, 30, 6))` și `print(t, len(t))`?

P2. Ce afișează? Compară cu P1: de ce iese aceeași listă?

```
r = []
for x in range(1, 31):
    if x % 2 == 1 and x % 3 == 0:
        r.append(x)
print(r)
```

P3. Ce termeni se afișează?

```
a = 1
for i in range(5):
    print(a, end=" ")
    a = 3 * a - 1
```

P4. Ce valoare are `s` la final? Scrie suma calculată ca expresie matematică.

```
s = 0
semn = 1
for k in range(1, 7):
    s += semn * k
    semn = -semn
print(s)
```

P5. Aceeași regulă ca la Fibonacci, alt început. Ce afișează?

```
a, b = 2, 1
for _ in range(7):
    print(a, end=" ")
    a, b = b, a + b
```

P6. Ce afișează `print(t, len(t), x)`?

```
x = 1
t = []
while x < 100:
    t.append(x)
    x *= 3
```

P7. Ce listă se generează și câte elemente are?

```
t = []
for i in range(1, 5):
    for _ in range(i):
        t.append(i)
print(t, len(t))
```

P8. Cele două bucle par să facă același lucru. Ce afișează fiecare?

```python
a, b = 1, 1
for _ in range(6):
    print(a, end=" ")
    a, b = b, a + b
print()
a, b = 1, 1
for _ in range(6):
    print(a, end=" ")
    a = b
    b = a + b
```

Nivelul 2 · Găsește și repară

★★ găsește și repară

P9. Programul trebuie să afișeze termenii Fibonacci **mai mici sau egali** cu `m`. Pentru `m = 21` afișează `1 1 2 3 5 8 13`. Repară-l modificând un singur caracter.

```python
m = int(input())
f = []
a, b = 1, 1
while a < m:
    f.append(a)
    a, b = b, a + b
print(*f)
```

P10. Programul trebuie să calculeze `1 + 1·2 + 1·2·3 + … + 1·2·…·n`. Pentru `n = 4` afișează `10` în loc de `33`. Ce calculează, de fapt? Repară-l.

```python
n = int(input())
s = 0
for i in range(1, n + 1):
    p = 1
    p *= i
    s += p
print(s)
```

P11. Programul trebuie să genereze multiplii lui `k` din `[a, b]`. Pentru `20 60 7` nu se mai oprește. De ce? Repară-l în două feluri.

```python
a, b, k = map(int, input().split())
rez = []
x = a
while x <= b:
    if x % k == 0:
        rez.append(x)
        x += k
print(rez)
```

P12. Programul trebuie să genereze pătratele perfecte `≤ n`. Pentru `n = 49` lipsește 49 (și la fel pentru `n = 50`). Repară-l. Ce test ai adăuga?

```python
n = int(input())
rez = []
for k in range(1, int(n ** 0.5)):
    rez.append(k * k)
print(rez)
```

P13. *Rânduri amestecate.* Programul afișează lista primilor `n` termeni ai șirului `a(0) = 2`, `a(k) = 2·a(k − 1) + 1`. Un rând este în plus.

```
A)      a = 2 * a + 1
B)  print(t)
C)  a = 2
D)      t.append(a)
E)  n = int(input())
F)  for _ in range(n):
G)  t = []
H)      a = 2
```

Ce afișează pentru `n = 5`? Dar pentru `n = 0`?

P14. După câți ani se **dublează** cel puțin suma depusă, la o dobândă anuală dată (în procente)? Pentru `1000` lei și `10%` programul afișează `9`, dar răspunsul corect este `8`. Găsește greșeala. Ce afișează programul reparat pentru `500` lei și `100%`?

```python
sold = float(input())
dob = float(input())
tinta = 2 * sold
ani = 1
while sold < tinta:
    sold = sold * (1 + dob / 100)
    ani += 1
print(ani)
```

Nivelul 3 · Scrie cod

★★ scrie cod

Pentru fiecare program, verifică **toate** testele înainte să spui că e gata.

P15. Progresia aritmetică. Citește `a1`, rația `r` și `n` (pe un rând). Afișează primii `n` termeni, iar pe rândul următor suma lor calculată în două feluri: cu `sum` și cu formula `n(2a1 + (n − 1)r)/2`.

intrare	ieșire
`3 4 5`	`3 7 11 15 19` / `55 55`
`10 -3 4`	`10 7 4 1` / `22 22`
`7 0 3`	`7 7 7` / `21 21`
`5 2 0`	(rând gol) / `0 0`

P16. Suma cifrelor dată. Citește `a`, `b`, `s`. Afișează numerele din `[a, b]` cu suma cifrelor egală cu `s`, sau `niciunul`.

intrare	ieșire
`1 50 5`	`5 14 23 32 41 50`
`100 200 1`	`100`
`10 20 30`	`niciunul`
`0 9 0`	`0`

P17. Seria alternantă. Calculează `1 − 1/2 + 1/3 − 1/4 + … ± 1/n` și afișează rezultatul cu 4 zecimale. Folosește o variabilă `semn`, nu puteri ale lui −1.

intrare	ieșire
`1`	`1.0000`
`2`	`0.5000`
`4`	`0.5833`
`1000`	`0.6926`

P18. Fibonacci într-un interval. Citește `a ≤ b`. Afișează termenii șirului Fibonacci (1, 1, 2, 3, …) din `[a, b]`, în ordine, sau `niciunul`.

intrare	ieșire
`5 60`	`5 8 13 21 34 55`
`1 1`	`1 1`
`14 20`	`niciunul`
`89 89`	`89`

P19. Șirul lui Collatz. Pornind de la `n ≥ 1`: dacă termenul e par, următorul este jumătatea lui; dacă e impar, următorul este `3·t + 1`. Te oprești la 1. Construiește lista termenilor (cu `n` și 1 incluși) și afișează lungimea ei și termenul maxim.

intrare	ieșire
`6`	`9 16`
`1`	`1 1`
`7`	`17 52`
`27`	`112 9232`

P20. Colonia de bacterii. O colonie se dublează la fiecare 20 de minute, dar după fiecare dublare 100 de bacterii sunt eliminate: `p ← 2p − 100`. Citește `p` și `M`. Afișează după câte minute colonia **depășește** `M` și, pe rândul următor, lista populațiilor (inclusiv cea inițială). Dacă `p > M` de la început, răspunsul e `0`. Dacă `p ≤ 100`, colonia nu poate crește: afișează `nu crește` (altfel bucla n-ar mai avea sfârșit!).

intrare	ieșire
`150 1000`	`100` / `150 200 300 500 900 1700`
`100 1000`	`nu crește`
`2000 1000`	`0` / `2000`
`101 102`	`40` / `101 102 104`

Nivelul 4 · Pune mintea la contribuție

★★★ pune mintea la contribuție

P21. Fibonacci de la coadă la cap. Se dă un termen `t` al șirului Fibonacci. Afișează termenii de la `t` **descrescător** până la primul 1, de exemplu `55 34 21 13 8 5 3 2 1 1`. Metoda 1: pentru fiecare termen afișat, regenerezi șirul de la început până la cel care îl precede. Metoda 2: generezi o singură dată, crescător, perechea `(precedent, t)`, apoi mergi înapoi **prin diferență**: termenul dinaintea perechii `(a, b)` este `b − a`. Implementează metoda 2 fără liste. Numără pașii (adunări + scăderi) ai fiecărei metode pentru `t = 55` și `t = 6765`. Care este ordinul de complexitate al fiecăreia, în funcție de poziția `k` a lui `t` în șir?

P22. Găsește formula. Șirul `a(0) = 3`, `a(n) = 2·a(n − 1) + 1` începe cu `3, 7, 15, 31, …`. Adună 1 la fiecare termen și privește rezultatul. Găsește formula termenului general și verific-o cu un program pentru `n = 0 … 14`.

P23. Proporția de aur. Generează rapoartele `f(k+1) / f(k)` ale termenilor Fibonacci consecutivi până când două rapoarte consecutive diferă cu mai puțin de `0.000001`. La ce termeni te oprești? Compară ultimul raport cu `(1 + 5 ** 0.5) / 2`. b) **Documentare.** Caută în surse de încredere (manual, enciclopedie, articol științific) câte un exemplu de proporție de aur din natură, din artă sau arhitectură și din știință; pentru fiecare, notează sursa și dacă are măsurători. c) Calculează raportul laturilor pentru: foaia A4 (`297 × 210` mm), un card bancar (`85.6 × 53.98` mm), un ecran 16:9 și o pereche de spirale `34` și `55`. Care sunt apropiate de `φ` (diferență sub `0.02`) și care nu? Ce concluzie tragi despre afirmația „se vede peste tot”?

P24. Problemă inversă. Un șir respectă regula lui Fibonacci (`t(k) = t(k−1) + t(k−2)`), dar nu începe cu `1, 1`. Știi că al treilea termen este 7 și al șaselea este 31. Care sunt primii doi termeni? Scrie primii șase termeni. (Indiciu: notează primii doi termeni cu `x` și `y` și exprimă-i pe ceilalți.)

P25. Mini-proiect: planul de alergare. În prima săptămână alergi `start` km. În fiecare săptămână următoare, distanța crește cu 10%, rotunjită la o zecimală: `km = round(km * 1.1, 1)`. Planul se încheie cu prima săptămână în care distanța este `≥ tinta`. Scrie funcția `plan(start, tinta)` care întoarce lista distanțelor și un program care afișează numărul de săptămâni și totalul de kilometri cu o zecimală. Teste: `plan(2, 10)` are 18 săptămâni și 91.1 km în total, ultima săptămână are 10.1 km; `plan(9.5, 10)` → `[9.5, 10.5]`; `plan(5, 5)` → `[5]`; `plan(12, 10)` → `[12]`.

Probleme de lucru în clasă (pbinfo)

Le rezolvăm împreună în ședință, în ordinea de mai jos.

- [ ] #255 **fibonacci** — ușoară
  https://www.pbinfo.ro/probleme/255/fibonacci

- [ ] #423 **fibonacci1** — ușoară
  https://www.pbinfo.ro/probleme/423/fibonacci1
- [ ] #493 **constr** — ușoară
  https://www.pbinfo.ro/probleme/493/constr

Temă pentru acasă (pbinfo)

Trimite soluțiile pe pbinfo și notează punctajul obținut. Bifează ce ai terminat.

- [ ] #424 **fibonacci-generalizat** — ușoară
  https://www.pbinfo.ro/probleme/424/fibonacci-generalizat
- [ ] #494 **constr1** — ușoară
  https://www.pbinfo.ro/probleme/494/constr1
- [ ] #257 **fibosum** — medie
  https://www.pbinfo.ro/probleme/257/fibosum
