INFORMATICĂ · PYTHON · CLASA A IX-A · STRATEGII DE REZOLVARE A PROBLEMELOR · LECȚIA 16

# Generarea sistematică a elementelor unei liste

Durată: 150 min  Nivel: mate-info · Python  Înainte: lecțiile py9-15

**CE VEI ȘTI DUPĂ LECȚIE**

- Construiești o listă-rezultat ai cărei termeni respectă o regulă: o proprietate, o expresie matematică sau o relație de recurență.
- Alegi criteriul de oprire potrivit: primii n termeni, termenii până la o limită, primul termen care depășește o valoare.
- Calculezi un termen din cel precedent în loc să-l recalculezi de la zero și estimezi câte operații câștigi.
- Testezi generatorul pe cazurile limită: n = 0, n = 1, n = 2, limită mai mică decât primul termen.

## 1. Ce înseamnă „a genera” o listă

Până acum listele veneau din date citite. Acum le **producem noi**, după o regulă. Schema este mereu aceeași:

```
rez ← listă vidă
pentru fiecare candidat / cât timp nu am terminat:
    calculează sau verifică termenul
    dacă e bun: adaugă-l în rez
```

| Fel de regulă | Exemplu | Cum obții termenul |
|---|---|---|
| proprietate dată | multiplii lui 7 din [20, 60] | parcurgi candidații și îi **filtrezi**, sau sari direct la cei buni |
| expresie matematică | `1, 1·2, 1·2·3, …` | termenul curent **din cel precedent** |
| relație de recurență | Fibonacci: `f(n) = f(n−1) + f(n−2)` | din ultimii termeni deja generați |

Criteriul de oprire face parte din cerință: **primii n termeni** ( `for` ), **termenii ≤ o limită** sau **până când se îndeplinește o condiție** ( `while` ).

## 2. Secvențe cu o proprietate dată

**Primele n pătrate perfecte.** Nu le căutăm printre toate numerele: le generăm direct.

```python
n = 6
patrate = []
for k in range(1, n + 1):
    patrate.append(k * k)
print(patrate)                      # [1, 4, 9, 16, 25, 36]
```

**Multiplii lui k din intervalul [a, b].** Două variante cu același rezultat:

```python
a, b, k = 20, 60, 7
rez = []
for x in range(a, b + 1):            # varianta 1: verifici fiecare candidat
    if x % k == 0:
        rez.append(x)
prim = (a + k - 1) // k * k          # varianta 2: primul multiplu ≥ a, apoi pas k
print(rez, list(range(prim, b + 1, k)))
# [21, 28, 35, 42, 49, 56] [21, 28, 35, 42, 49, 56]
```

Varianta 1 face `b − a + 1 = 41` verificări; varianta 2 atinge doar cele 6 numere bune. Când poți **sări direct** la termenii buni, sari.

Uneori nu există scurtătură și filtrezi. Numerele de două cifre cu suma cifrelor 9:

```python
rez = []
for x in range(10, 100):
    if x // 10 + x % 10 == 9:
        rez.append(x)
print(rez)                # [18, 27, 36, 45, 54, 63, 72, 81, 90]
```

## 3. Termeni ai unei expresii: folosește termenul precedent

Vrem termenii `1, 1·2, 1·2·3, …, 1·2·…·n` și suma lor. Observația-cheie: **fiecare termen este precedentul înmulțit cu i**.

```python
n = 5
termeni = []
p = 1
for i in range(1, n + 1):
    p *= i                   # p devine 1·2·…·i, din valoarea anterioară
    termeni.append(p)
print(termeni, sum(termeni)) # [1, 2, 6, 24, 120] 153
```

Dacă ai calcula fiecare termen de la zero (o buclă interioară `for j in range(1, i + 1)`), ai face `1 + 2 + … + n = n(n + 1)/2` înmulțiri: 15 pentru `n = 5`, dar **500 500** pentru `n = 1000`. Cu termenul precedent faci doar `n`. Prima variantă este O(n²), a doua O(n).

> **SFAT**
>
> Aceeași idee merge pentru `1, 1/2, 1/4, …` (`termen /= 2`), `x, x², x³, …` (`termen *= x`) sau pentru semne alternante (`semn = -semn`). Întreabă-te mereu: **cu ce operație trec de la un termen la următorul?**

## 4. Șiruri recurente: Fibonacci

Șirul lui Fibonacci: primii doi termeni sunt 1, iar fiecare termen este suma celor doi dinainte. Îl regăsești în natură (spiralele florii-soarelui, ale conurilor de brad); raportul a doi termeni consecutivi se apropie de **proporția de aur**, ≈ 1,618. Proporția de aur `φ = (1 + √5) / 2 ≈ 1,618` apare când împarți un segment în două părți astfel încât întregul raportat la partea mare să fie egal cu partea mare raportată la cea mică. Un **dreptunghi de aur** (laturile în raportul `φ`) are o proprietate frumoasă:

dacă îi tai un pătrat, rămâne tot un dreptunghi de aur. Din acest motiv unii artiști și arhitecți au folosit-o ca ghid de compoziție (arhitectul Le Corbusier a construit pe ea sistemul de proporții *Modulor*). În natură, numărul spiralelor la floarea-soarelui sau la conurile de brad este de obicei doi termeni Fibonacci consecutivi. Atenție însă: multe afirmații „X are proporția de aur" sunt exagerate, de aceea se verifică prin măsurare (vezi fișa, P23). De exemplu, pentru spirale `34` și `55`, raportul `55 / 34 ≈ 1,6176` este foarte aproape de `φ`.

```
def fibonacci(n):
    f = []
    a, b = 1, 1                 # a = termenul curent, b = următorul
    for _ in range(n):
        f.append(a)
        a, b = b, a + b         # avansăm cu un pas în șir
    return f

print(fibonacci(10))   # [1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
print(fibonacci(0), fibonacci(1), fibonacci(2))   # [] [1] [1, 1]
```

**ATENȚIE — GREȘELI FRECVENTE**

Varianta „naivă" `f = [1, 1]` urmată de `while len(f) < n: f.append(f[-1] + f[-2])` greșește pentru `n = 0` și `n = 1`: întoarce tot `[1, 1]`. Testează **mereu** generatorul pentru lista vidă, un element și două elemente.

**Până la o limită** în loc de „primii n": schimbi doar condiția buclei.

```
def fib_pana_la(m):
    f = []
    a, b = 1, 1
    while a <= m:
        f.append(a)
        a, b = b, a + b
    return f

print(fib_pana_la(100))  # [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
print(fib_pana_la(0))    # []
```

**Recurență cu formulă.** Pentru `a(0) = 3` și `a(n) = 2·a(n − 1)`, termenii sunt `3, 6, 12, 24, …`, adică `a(n) = 3·2ⁿ`. Când găsești formula termenului general, poți calcula direct orice termen: `3 * 2 ** n`. Generarea pas cu pas rămâne utilă când vrei **toți** termenii sau când nu există formulă simplă (cum e la Fibonacci, pentru calcule exacte).

## 5. Până când se îndeplinește o condiție

Ai 1000 de lei într-un depozit cu dobândă de 5% pe an (dobânda se adaugă la sold). După câți ani soldul **depășește** 1500 de lei?

```python
sold = 1000.0
ani = 0
while sold <= 1500:
    sold = sold * 1.05
    ani += 1
print(ani, f"{sold:.2f}")     # 9 1551.33
```

Nu știi dinainte câți termeni generezi, deci folosești `while`. Dacă vrei și evoluția an cu an (pentru un grafic sau ca să cauți anul în care s-a trecut de 1200), **memorezi** fiecare sold într-o listă: `istoric.append(sold)`.

**LA BAC**

La generarea cu `while`, verifică trei lucruri: condiția se schimbă la fiecare pas (altfel bucla e infinită), primul termen este și el verificat (ce se întâmplă dacă soldul inițial deja depășește limita?) și numărătoarea începe de unde trebuie (aici `ani = 0`).

## 6. Cum testezi un generator

| Test | Ce prinde |
|---|---|
| `n = 0` | lista vidă: nu trebuie să apară niciun termen |
| `n = 1`, `n = 2` | inițializări de tipul `[1, 1]` care „scapă" termeni în plus |
| limita mai mică decât primul termen | `fib_pana_la(0)` trebuie să dea `[]` |
| limita egală cu un termen | `≤` sau `<`? (`fib_pana_la(89)` trebuie să-l conțină pe 89) |
| un caz verificat de mână | primii 5–6 termeni, scriși pe hârtie |
| un caz mare | rulează repede? (O(n) sau O(n²)?) |

## 7. Verifică-te

1. Scrie o singură instrucțiune care construiește lista multiplilor lui 5 din intervalul [1, 30].
2. Care este următorul termen după `p` în șirul `1, 1·3, 1·3·5, 1·3·5·7, …`, dacă `p` este al `k`-lea termen?
3. Ce afișează `fibonacci(7)`? Dar `fib_pana_la(1)`?
4. Pentru `a(0) = 5`, `a(n) = a(n − 1) + 4`, scrie primii 5 termeni și formula termenului general.
5. În programul cu dobânda, ce afișează dacă soldul inițial este 2000 de lei?
6. Câte înmulțiri face calculul „de la zero" al termenilor `1, 1·2, …, 1·2·…·10`? Dar varianta cu termenul precedent?

### Răspunsuri

1. `list(range(5, 31, 5))` → `[5, 10, 15, 20, 25, 30]`.
2. `p * (2 * k + 1)`: al `k`-lea termen se termină cu factorul `2k − 1`, următorul factor este `2k + 1`.

3. `[1, 1, 2, 3, 5, 8, 13]` și `[1, 1]`.
4. `5, 9, 13, 17, 21`; `a(n) = 5 + 4n` (progresie aritmetică).
5. `0 2000.00`: condiția este falsă de la început, bucla nu se execută deloc.
6. `1 + 2 + … + 10 = 55`, față de `10`.

## 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