test-e.ro

CAIET ELEV

Lecția 4

Structura repetitivă cu contor (for · range)

*A patra structură fundamentală: repetiția cu număr cunoscut de pași. Bucla **for**, generatorul **range(start, stop, pas)**, buclele imbricate, ramura **for…else** și echivalența dintre **for** și **while**.*

DISCIPLINA Informatică **CLASA** a IX-a · mate-info **LIMBAJ** Python 3

NUMELE ELEVULUI .....

CUM FOLOSEȘTI CAIETUL

Acest caiet adună exercițiile scrise din Lecția 4. Rezolvă-le pe hârtie — prezici ce afișează o buclă, completezi antete de **for**, desenezi scheme logice, urmărești bucle imbricate și scrii programe. Folosește ce s-a predat până acum: **for** cu **range()** și **while** (fără funcții **def/return**). **break** și **for…else** sunt material de extindere (opțional). Spațiile punctate și liniate sunt pentru scrisul tău; verificarea automată rămâne pe platformă.

CUPRINS

1 Recapitulare: de ce un contor?
2 Structura for și range()
3 for în practică
4 for…else și bucle imbricate
5 Transformări for ↔ while
6 Dry-Run Lab: execuție pe hârtie
7 Verifică-ți codul

SECȚIUNEA 1 · RECAPITULARE

1 De ce avem nevoie de un *contor*?

Cu `while` repeți **cât timp** o condiție e adevărată — potrivit când **nu știi dinainte** de câte ori. Dar foarte des știi exact: „afișează numerele de la 1 la 100", „repetă de `n` ori". Pentru aceste situații folosim o buclă **cu contor**. Ne reamintim mai întâi cum arată un contor scris cu `while`: **inițializare** → **condiție** → **corp** → **pas**.

A

EXERCIȚIUL A · RECAPITULARE

Ce-ți mai aduci aminte?

GRILĂ COMPLETARE

Încercuiește `True` / `False` sau completează pe linia punctată.

1. Într-o buclă cu contor, dacă uiți **pasul** , bucla rulează la nesfârșit.

True False

2. Contorul se pregătește (inițializează) _____ de intrarea în buclă.

3. Bucla `while` se folosește când **nu** cunoști dinainte numărul de repetări.

True False

4. Numărul de repetări ale unei bucle care merge de la `1` la `n` (pas 1) este _____

B

EXERCIȚIUL B · DRY-RUN

O buclă care numără

DRY-RUN

Pseudocodul pornește contorul `i` de la `s` și îl crește din 1 în 1 **cât timp** `i ≤ f`, afișând de fiecare dată valoarea lui `i`; la final afișează `gata`. Execută-l în minte pentru fiecare set de intrare și scrie ce afișează.

```
citește s, f
i ← s
cât timp i ≤ f execută
    scrie i
    i ← i + 1
scrie "gata"
```

s	f	Ce afișează?
1	4	
3	3	
5	2	

HINT *la ultimul rând, condiția `i ≤ f` este falsă din start — corpul rulează de zero ori.*

C

SCRIE SINGUR · CU WHILE

Numărătoare inversă

SCRIERE COD

Se citește un număr natural `n` (`n > 0`). Afișează, pe câte o linie, numerele de la `n` până la `1` descrescător. Exemplu: `n = 4` → `4 3 2 1`. Folosește o buclă **cu contor** (`while`): aici contorul nu crește, ci **scade**.

HINT *gândește-te de unde pornește contorul, când trebuie să se oprească și cum îl modifici la fiecare pas.*

SOLUȚIA TA · PYTHON

SECȚIUNEA 2 · STRUCTURA FOR

2 Structura *for* și *range()*

Bucla `for` parcurge o secvență de valori generată de `range`. Contorul pornește, se testează și avansează **automat** — nu mai ai grijă tu de cele trei elemente. `for i in range(...)` înseamnă „pentru fiecare valoare `i` din secvență, execută corpul".

CUM FUNCȚIONEAZĂ RANGE(START, STOP, PAS)

1. **start** — prima valoare (inclusiv). Dacă lipsește, este `0`.
2. **stop** — limita la care bucla se oprește. Valoarea `stop` **NU** este inclusă.
3. **pas** — cu cât avansează contorul. Dacă lipsește, este `1`; poate fi și negativ.

Regula esențială: **stop nu intră în șir**. Ca să ajungi până la `n` inclusiv, scrii `range(1, n + 1)`.

FORMA A · CRESCĂTOARE (PSEUDOCOD)

```
pentru i ← start, stop, pas execută
    instrucțiune
sfârșit pentru
```

FORMA A · PYTHON

```
for i in range(start, stop + 1, pas):
    instrucțiune
```

FORMA B · DESCRESCĂTOARE (PSEUDOCOD)

```
pentru i ← stop, start, -pas execută
    instrucțiune
sfârșit pentru
```

FORMA B · PYTHON

```
for i in range(stop, start - 1, -pas):
    instrucțiune
```

A EXERCIȚIUL A · GRILĂ

Ce valori produce range?

GRILĂ COMPLETARE

Scrie pe linia punctată **toate** valorile parcurse de contor, în ordine, separate prin spațiu. Atenție: `stop` nu intră!

1. `range(1, 6)` → _____
2. `range(0, 10, 2)` → _____
3. `range(5, 0, -1)` → _____
4. `range(3, 3)` → _____

HINT *ca să incluzi capătul `n` într-o buclă crescătoare, scrii `range(1, n + 1)`; într-una descrescătoare până la `1`, scrii `range(n, 0, -1)`.*

B

EXERCIȚIUL B · DRY-RUN

Citește două bucle for

DRY-RUN

Pentru fiecare buclă, scrie **exact ce afișează** (valorile separate prin spațiu). Urmărește: unde pornește contorul, unde se oprește (atenție la `stop`) și care e pasul.

BUCLA 1 · FORMA B (DESCRESCĂTOARE)

```
for i in range(10, 5, -1):
    print(i, end=" ")
```

BUCLA 2 · FORMA A CU PAS (MULTIPLI)

```
for i in range(0, 20, 5):
    print(i, end=" ")
```

BUCLA 1 AFIȘEAZĂ

BUCLA 2 AFIȘEAZĂ

C

EXERCIȚIUL C · CONCEPEREA SCHEMEI

Schema logică a buclei for

SCHEMĂ LOGICĂ

Un `for i in range(1, n + 1)` ascunde, de fapt, o **inițializare** (`i ← 1`), un **test** (`i ≤ n ?`) și un **pas** (`i ← i + 1`). De aceea îl desenăm în forma „desfășurată" (ca la `cât timp`). Rombul are **două ieșiri**: pe **DA** se execută corpul, apoi fluxul revine la romb; pe **NU** se afișează rezultatul și STOP. Folosește: **oval** (START/STOP), **paralelogram** (citire/scriere), **dreptunghi** (calcul), **romb** (condiție).

```
citește n
i ← 1
```

```
cât timp i ≤ n execută
    scrie i
    i ← i + 1
```

DESENEAZĂ SCHEMA AICI · NU UITA RAMURILE DA / NU ŞI ÎNTOARCEREA LA ROMB

D

EXERCIȚIUL D · COMPLETEAZĂ COD

Completează antetul buclei

COMPLETARE

Se citesc două numere întregi `a` și `b` (cu `a ≤ b`). Programul numără câte numere **pare** sunt în intervalul `[a, b]` (capete incluse). Lipsește exact **antetul buclei**. Completează-l ca bucla să parcurgă toate numerele de la `a` până la `b` inclusiv.

```
a = int(input())
b = int(input())
nr = 0

.....................................................
    if i % 2 == 0:
        nr = nr + 1
print(nr)
```

ANTETUL CARE LIPSEȘTE

SECȚIUNEA 3 · FOR ÎN PRACTICĂ

3 *for* în practică

Punem `for` și `range` la treabă pe probleme uzuale, toate cu o variabilă care se schimbă la fiecare pas (o sumă, un produs, un contor). Studiază exemplul comentat, apoi rezolvă.

EXEMPLU COMENTAT · PRODUSUL 1·2·…·N (FACTORIAL)

```
n = int(input())
p = 1               # element neutru la înmulțire
for i in range(1, n + 1):   # i ia pe rând 1, 2, …, n
    p = p * i         # înmulțesc cu valoarea curentă
print(p)
```

La sumă pornești acumulatorul cu `0`; la produs, cu `1`. Reține că `range(1, n + 1)` parcurge exact `n` valori.

A

EXERCIȚIUL A · DRY-RUN

Câți divizori are n?

DRY-RUN

Bucla parcurge toate numerele `d` de la `1` la `n` și, dacă `d` îl divide pe `n` (`n % d == 0`), crește contorul `nr`. Pentru fiecare `n`, scrie câți divizori are (valoarea finală a lui `nr`).

```
nr = 0
for d in range(1, n + 1):
    if n % d == 0:
        nr = nr + 1
print(nr)
```

n	Câți divizori?
6	
7	
12	

B

EXERCIȚIUL B · CAZ-LIMITĂ

Dar dacă a > b?

GRILĂ

Programul de la Secțiunea 2·D numără parele din `[a, b]` cu `for i in range(a, b + 1)`. Ce se întâmplă dacă cineva îl rulează cu `a = 8` și `b = 3`? Bifează varianta corectă.

- *a)* Programul dă eroare fiindcă `a > b` .
- *b)* `range(8, 4)` e gol, bucla nu rulează deloc, se afișează `0` .
- *c)* Numără parele de la 3 la 8, în ordine inversă.

HINT *dacă `start ≥ stop` la pas pozitiv, `range` produce un șir* **gol** *— corpul rulează de zero ori.*

C

EXERCIȚIUL C · ORDONARE

Ordonează algoritmul factorialului

ORDONARE

Instrucțiunile de mai jos formează, amestecate, programul care calculează factorialul lui `n` cu `for`. Numerotează-le de la **1** la **4** în ordinea corectă. O instrucțiune este o **capcană** (nu face parte din algoritm) — taie-o.

- `for i in range(1, n + 1):`
- `print(p)`
- `p = 1`
- `p = p * i`
- `p = 0` ← *capcană?*

D

SCRIE SINGUR · CU FOR

Suma numerelor pare din [a, b]

SCRIERE COD

Se citesc două numere întregi `a` și `b` (`a ≤ b`). Calculează și afișează **suma numerelor pare** din intervalul `[a, b]` (capete incluse). Exemplu: `a = 3`, `b = 8` → `4 + 6 + 8 = 18`.

HINT *parcurge tot intervalul cu* `range(a, b + 1)` *și adună într-un acumulator pornit de la* `0` *doar numerele cu* `i % 2 == 0`.

SOLUȚIA TA · PYTHON

4 SECȚIUNEA 4 · FOR…ELSE ȘI IMBRICATE

for…else și bucle imbricate

Buclele imbricate (o buclă înăuntrul alteia) sunt conținut de bază; `break` și ramura `for…else` sunt **material de extindere** (opțional) — le parcurgi ca să înțelegi cum funcționează.

CELE DOUĂ „EXTRA" ALE LUI FOR — EXTINDERE · OPȚIONAL

break oprește bucla imediat — valorile rămase din `range` nu mai sunt parcurse. Ramura **for… else** se execută **doar dacă bucla s-a terminat normal** (a parcurs tot `range`), **nu** și dacă a fost oprită de un `break`. Perfectă pentru o căutare: „dacă am găsit, ies cu `break`; dacă am terminat fără să găsesc, ramura `else` îmi spune".

BUCLE IMBRICATE

Pentru fiecare valoare a contorului din bucla **exterioară**, bucla **interioară** se rotește complet, de la început până la sfârșit. Astfel, o buclă interioară care rulează de 4 ori, pusă într-una exterioară de 3 ori, execută corpul de `3 × 4 = 12` ori.

A

EXERCIȚIUL A · PREZICE OUTPUT-UL · OPȚIONAL

Ce afișează bucla cu break?

DRY-RUN

Prezice exact ce afișează programul (numerele separate prin spațiu). Ține minte: `break` oprește bucla pe loc.

```
for i in range(1, 6):
    if i == 3:
        break
    print(i, end=" ")
```

CE AFIȘEAZĂ PROGRAMUL

B

EXERCIȚIUL B · GRILĂ · OPȚIONAL

Când rulează else?

GRILĂ

Programul verifică dacă `n` este prim. Pentru fiecare valoare a lui `n`, bifează dacă ramura `else` a buclei (linia `print("e prim")`) **se execută** sau **nu**.

```
n = int(input())
for d in range(2, n):
    if n % d == 0:
        print("are divizor:", d)
        break
else:
    print("e prim")
```

1. `n = 12` : else se execută? DA NU
2. `n = 7` : else se execută? DA NU

HINT *`else` rulează doar dacă bucla NU a fost oprită de `break`. La `n = 12`, `d = 2` îl divide → `break`.*

C

EXERCIȚIUL C · DRY-RUN

Citește o buclă imbricată

DRY-RUN

Bucla afișează produsul `i · j` pentru fiecare pereche. Urmărește ordinea: `i` ia prima valoare, apoi `j` parcurge tot intervalul; apoi `i` trece la valoarea următoare. Scrie **exact ce se afișează**, în ordine.

```
for i in range(1, 4):
    for j in range(1, 4):
        print(i * j, end=" ")
```

CE AFIȘEAZĂ PROGRAMUL (ÎN ORDINE)

HINT *sunt* `3 × 3 = 9` *valori afișate. Pentru* `i = 1`*:* `1 2 3`*; pentru* `i = 2`*: …*

5 SECȚIUNEA 5 · FOR ↔ WHILE

Transformări *for ↔ while*

Un `for` cu contor și un `while` scris cu contor fac exact același lucru — diferă doar forma. Rețeta conversiei `for → while`: **inițializezi** contorul înainte, pui **condiția** în `while` și adaugi **pasul** la finalul corpului.

FOR (TEST AUTOMAT)

```
for i in range(1, n + 1):
    print(i)
```

WHILE ECHIVALENT

```
i = 1
while i <= n:
    print(i)
    i = i + 1
```

CONVERSIA INVERSĂ NU MERGE MEREU

Drumul `while → for` se poate **doar** dacă bucla `while` are contor regulat (inițializare, condiție `i ≤ stop`, pas `i ← i + pas`). O buclă care se oprește la o condiție necunoscută dinainte (ex.: citește numere **până când** apare `0`) **nu** se poate scrie ca `for` — nu există un `range` care s-o descrie.

A

EXERCIȚIUL A · CONVERSIE

Completează while-ul echivalent

COMPLETARE

Bucla `for` din stânga afișează numerele de la 1 la 10. În dreapta e aceeași buclă rescrisă cu `while`, dar lipsește exact o linie (**pasul**). Completează linia care lipsește ca cele două bucle să facă același lucru.

FOR (DAT)

```
for i in range(1, 11):
    print(i)
```

WHILE (DE COMPLETAT)

```
i = 1
while i <= 10:
    print(i)

    ....................
```

LINIA CARE LIPSEȘTE

B

EXERCIȚIUL B · GRILĂ

Care NU se poate rescrie ca for?

GRILĂ

Dintre buclele `while` de mai jos, bifează-o pe cea care **nu** poate fi rescrisă ca buclă `for` cu `range`.

a) `i = 1; while i <= 100: print(i); i = i + 1`

b) `i = 10; while i >= 1: print(i); i = i - 1`

c) `x = int(input()); while x != 0: print(x); x = int(input())`

HINT *întreabă-te: știu* **dinainte** *de câte ori se repetă? Dacă nu (bucla depinde de ce introduce utilizatorul), nu există un* `range` *potrivit.*

SECȚIUNEA 6 · EXECUȚIE PE HÂRTIE

6 *Dry-Run* Lab

A face **dry-run** înseamnă să execuți un program „în minte" sau pe hârtie, pas cu pas. La teză nu ai calculator — trebuie să știi exact ce face o buclă.

TREI PAȘI PENTRU UN DRY-RUN CORECT

1. **Scrie valorile lui range.** Notează ce numere produce: start, start+pas, … până înainte de stop (care nu intră).
2. **Fă un tabel.** O coloană pentru contor (`i`) și câte una pentru fiecare variabilă care se schimbă (sumă, contor, produs…).
3. **Completează rând cu rând.** Ultimul rând îți dă rezultatul afișat.

A

EXERCIȚIUL A · TABEL PAS-CU-PAS

Câte numere impare sunt de la 1 la 7

DRY-RUN

Bucla parcurge numerele de la 1 la 7 și numără câte sunt impare (`i % 2 != 0`). Completează, pentru fiecare `i`, cât devine contorul `c` **după** ce a fost procesat acel `i`.

```
c = 0
for i in range(1, 8):
    if i % 2 != 0:
        c = c + 1
print(c)
```

i	1	2	3	4	5	6	7
c							

→ Programul afișează: _____

B

EXERCIȚIUL B · PREDICT OUTPUT

Ce afișează bucla?

DRY-RUN

Fără să rulezi codul, scrie exact ce afișează. Atenție la pasul negativ și la `stop` (care nu intră).

```python
for i in range(20, 5, -3):
    print(i, end=" ")
```

VALORILE AFIȘATE, ÎN ORDINE

HINT *pornești din 20 și scazi 3 de fiecare dată; te oprești înainte să treci de 5 (5 nu intră).*

C

EXERCIȚIUL C · CAPCANE

Trei capcane frecvente

GRILĂ COMPLETARE

Cele mai dese greșeli la `for` vin din neatenția la `range`. Răspunde scurt.

1. Câte valori produce `range(1, 5)` ? _____
2. `range(0, 5)` include valoarea `5` . `True` `False`
3. Ca să parcurgi numerele de la `1` la `n` inclusiv, scrii `range(1,` _____ `)`

7 SECȚIUNEA 7 · VERIFICĂ-ȚI CODUL

Verifică-ți *codul*

Scrie soluția pe hârtie cu `for` (și `range`), apoi testeaz-o pe platformă. Gândește-te la cazurile-limită.

1

PROBLEMA 1

Suma numerelor de la 1 la n

SCRIERE COD

Citește un număr natural `n` (`n > 0`) și calculează **suma** tuturor numerelor de la `1` la `n`. Afișează un singur număr. Exemplu: `n = 5` → `15`.

SOLUȚIA TA · PYTHON

2

PROBLEMA 2

Câți multipli de 3

SCRIERE COD

Citește un număr natural `n` (`n > 0`) și afișează **câte numere** de la `1` la `n` sunt divizibile cu `3`. Exemplu: `n = 10` → `3` (numerele 3, 6, 9).

HINT *parcurge intervalul și crește un contor separat doar când* `i % 3 == 0`.

SOLUȚIA TA · PYTHON

3

PROBLEMA 3 · PROVOCARE

Triunghi de asteriscuri

SCRIERE COD

Se citește un număr natural `n` (`n > 0`). Folosind **două bucle imbricate**, afișează un triunghi: pe linia `k` se află exact `k` asteriscuri, separate prin câte un spațiu (`k` de la `1` la `n`).

EXEMPLU · N = 4

HINT *bucla exterioară trece prin linii (`k` de la 1 la `n`); cea interioară afișează cele `k` asteriscuri ale liniei curente. Trece pe rând nou după fiecare linie.*

SOLUȚIA TA · PYTHON