test-e.ro

CAIET ELEV

Lecția 9

Operații cu cifrele unui număr

Firul lecției de azi: orice număr întreg poate fi ***desfăcut în cifre*** *și apoi* ***construit la loc****, cu trei unelte simple. Cu ele afli produsul cifrelor, cea mai mare și cea mai mică cifră, numeri cifrele cu o proprietate, elimini cifre păstrând ordinea și verifici dacă un număr e palindrom.*

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

NUMELE ELEVULUI .....

CUM FOLOSEȘTI CAIETUL

Acest caiet adună exercițiile scrise din Lecția 9. Rezolvă-le pe hârtie — reactualizezi uneltele care desfac un număr în cifre, prezici produsul cifrelor (atenție la cifra `0`), trasezi execuția pas cu pas pentru cifra minimă și pentru oglindit, completezi cod și pseudocod, ordonezi liniile unui algoritm și prinzi capcana palindromului. Parcurgi cifrele **numai cu** `while` (`% 10` și `// 10`), fără funcții și fără parcurgerea numărului ca șir. Regula de aur: **rezultatul îl calculezi tu, înainte de a rula programul.** Verificarea automată a codului rămâne pe platformă.

CUPRINS

1 Uneltele pentru cifre .....
2 Produsul cifrelor .....
3 Cifra maximă și minimă .....
4 Numărarea și filtrarea cifrelor .....
5 Eliminarea cifrelor · ordinea .....
6 Palindrom și cifra de control .....
7 Verifică-ți codul .....

SECȚIUNEA 1 · RECAPITULARE

1 Uneltele cu care *desfacem* un număr

Începem o unitate nouă: prelucrarea datelor numerice. Toate operațiile din această lecție se sprijină pe **trei unelte** — două le știi de la structuri repetitive, a treia construiește un număr.

`n % 10`

Îți dă **ultima cifră** a numărului (restul împărțirii la 10).

`n // 10`

Îți dă **numărul fără ultima cifră** (câtul împărțirii întregi la 10).

`n * 10 + c`

Adaugă cifra **c la dreapta** lui `n`: din 5 și 3 → 53.

TIPARUL DE PARCURGERE A CIFRELOR

Cât timp mai există cifre (`while n > 0`), luăm ultima cifră cu `% 10`, o folosim, apoi o eliminăm cu `// 10`. Bucla se oprește singură când `n` devine `0` — **fără** `break`.

A EXERCIȚIUL A · QUIZ DE DIAGNOZĂ

Îți amintești uneltele?

GRILĂ

Pentru fiecare întrebare, încercuiește răspunsul corect. Sunt operațiile de bază pe care le vei folosi în toată lecția.

1. Cât este `537 % 10`? `5` `7` `53`
2. Cât este `537 // 10`? `7` `53` `537`
3. Ai `n = 5` și cifra `c = 3`. Ce expresie adaugă cifra la dreapta, obținând `53`?

`n + c` `n * 10 + c` `c * 10 + n`

4. De câte ori se execută corpul buclei `while n > 0` pentru `n = 537`? `2` `3` `537`

B EXERCIȚIUL B · REPARĂ CODUL

Numărarea cifrelor

CORECTARE

Programul ar trebui să citească un număr natural și să afișeze **câte cifre are**. Rulează, dar dă mereu un rezultat greșit (pentru `n = 537` ar trebui `3`). Fă un dry-run în minte, **încercuiește linia greșită**, scrie numărul ei și rescrie linia corectată.

```
1  n = int(input())
```

```
nr = 1
while n > 0:
    nr = nr + 1
    n = n // 10
print(nr)
```

→ Linia greșită: _____ corectată: _____

HINT *contorul numără câte cifre elimini. Cu ce valoare ar trebui să pornească înainte de a fi eliminat vreo cifră?*

C

EXERCIȚIUL C · DRY-RUN

Oglindește numărul 452 pas cu pas

DRY-RUN

Urmărim cum se construiește **oglinditul** (răsturnatul) lui `452`. La fiecare pas luăm ultima cifră a lui `n` și o lipim la dreapta lui `rasturnat`. Completează tabelul — primul rând e dat ca exemplu.

```
n = int(input())
rasturnat = 0
while n > 0:
    cifra = n % 10
    rasturnat = rasturnat * 10 + cifra
    n = n // 10
print(rasturnat)
```

Pas	n (rămas)	cifra	rasturnat
după trecerea 1	45	2	2
după trecerea 2			
după trecerea 3			

→ Programul afișează: _____

HINT *la fiecare trecere:* `cifra = n % 10`*, apoi* `rasturnat = rasturnat * 10 + cifra`*, apoi* `n = n // 10`*. Bucla se oprește când* `n` *devine* `0`*.*

SECȚIUNEA 2 · TEMĂ NOUĂ

2 Produsul cifrelor și *construirea* unui număr

La suma cifrelor porneai de la `0`. La **produsul cifrelor** pornești de la `1` — elementul neutru al înmulțirii. Apoi, la fiecare cifră, `produs = produs * cifra`.

> **CAPCANA CIFREI ZERO**
>
> Dacă o singură cifră este `0`, întreg produsul devine `0` (orice înmulțit cu zero dă zero). De aceea deosebim **produsul tuturor cifrelor** de **produsul cifrelor nenule**, unde sărim peste cifra `0`.

A treia unealtă construiește un număr adăugând o cifră. La dreapta e simplu; la stânga trebuie să știi cu ce putere a lui `10` înmulțești cifra nouă. Pornim de la `47` și adăugăm cifra `5`:

LA DREAPTA

```
n = 47
c = 5
n = n * 10 + c
= 47 * 10 + 5 = 475
```

Împingi tot numărul cu o poziție (× 10) și pui cifra pe locul unităților.

LA STÂNGA

```
n = 47 (2 cifre)
c = 5
n = c * 100 + n
= 5 * 100 + 47 = 547
```

Cifra nouă se înmulțește cu `10` la puterea numărului de cifre ale lui `n`.

A EXERCIȚIUL A · PREZICE REZULTATUL

Produsul tuturor cifrelor

COMPLETARE

Pentru fiecare număr, scrie **produsul tuturor cifrelor** lui (pornind de la 1, înmulțind fiecare cifră). Fii atent la cifra `0`.

NUMĂRUL	PRODUSUL TUTUROR CIFRELOR
`234`	_____
`6`	_____
`205`	_____
`720`	_____
`111`	_____

HINT *orice număr care are cel puțin o cifră `0` are produsul tuturor cifrelor egal cu `0`.*

B

EXERCIȚIUL B · PREZICE IEȘIREA

Ce afișează pentru n = 205?

GRILĂ

Codul calculează produsul **tuturor** cifrelor. Execută-l în minte pentru `n = 205` și încercuiește ce afișează.

```
n = int(input())
produs = 1
while n > 0:
    cifra = n % 10
    produs = produs * cifra
    n = n // 10
print(produs)
```

→ Afișează: `0` `10` `7`

De ce? _____

C

EXERCIȚIUL C · COMPLETEAZĂ CODUL

Produsul cifrelor nenule

COMPLETARE

Vrem produsul cifrelor **nenule** (sărim peste cifrele egale cu `0`). Ce condiție trebuie pusă în locul liber, astfel încât înmulțirea să se facă doar pentru cifrele diferite de zero? Încercuiește varianta corectă.

```
n = int(input())
produs = 1
while n > 0:
    cifra = n % 10
    ________
        produs = produs * cifra
    n = n // 10
print(produs)
```

_____ *a)* `if cifra == 0:`

_____ *b)* `if cifra != 0:`

_____ *c)* `while cifra != 0:`

HINT *vrei să înmulțești* **doar când** *cifra e diferită de 0 — deci ai nevoie de o decizie (`if`), nu de o buclă.*

SECȚIUNEA 3 · TEMĂ NOUĂ

3 Cifra *maximă* și cifra *minimă*

Pentru a găsi cea mai mare cifră, parcurgem cifrele și ținem minte, la fiecare pas, cea mai mare cifră văzută până atunci. Este tiparul **„campionului curent"**, pe care îl cunoști de la găsirea maximului dintr-un șir.

> **INIȚIALIZAREA CONTEAZĂ**
>
> Pentru cifra **maximă** pornim de la `maxim = 0`: orice cifră este cel puțin `0`, deci prima comparație funcționează. Pentru cifra **minimă**, `0` ar fi o alegere **greșită** (nicio cifră nu ar fi vreodată mai mică decât `0`) — pornim de la `9`, cea mai mare cifră posibilă.

A

EXERCIȚIUL A · DRY-RUN

Găsește cifra minimă a lui 6041

DRY-RUN

Pornim cu `minim = 9`. La fiecare pas: `cifra = n % 10`, iar dacă `cifra < minim`, atunci `minim = cifra`. Completează tabelul pentru `6041` — primul rând e exemplu. Atenție la cifra `0`, cea mai mică posibilă.

Pas	cifra	minim (după pas)
după trecerea 1	1	1
după trecerea 2		
după trecerea 3		
după trecerea 4		

→ Cifra minimă este: _____

HINT *parcurgi cifrele lui 6041 de la coadă spre cap: 1, 4, 0, 6. Minimul scade doar când întâlnești o cifră mai mică decât cea reținută.*

B

EXERCIȚIUL B · COMPLETEAZĂ PSEUDOCODUL

Algoritmul pentru cifra maximă

COMPLETARE

Completează cele **trei spații libere** din pseudocod, astfel încât să afle corect cea mai mare cifră a lui `n`.

```
citește n
maxim ← ______
cât timp n > 0 execută
    cifra ← n mod 10
    dacă cifra _____ maxim atunci
        maxim ← cifra
    n ← n _____ 10
scrie maxim
```

1. valoarea de pornire a maximului **2.** operatorul de comparație **3.** operația care elimină ultima cifră

HINT *maximul pornește de la cea mai mică valoare posibilă a unei cifre. Comparăm cifra curentă cu maximul reținut. Ultima cifră se elimină prin împărțire întreagă (`div`) la 10.*

SECȚIUNEA 4 · TEMĂ NOUĂ

4 Numărarea și *filtrarea* cifrelor

De multe ori nu ne interesează toate cifrele, ci doar acelea cu o anumită proprietate: cifrele pare, cifrele mai mari decât un prag, cifrele egale cu o valoare. Tiparul este mereu același: **parcurge** cifrele, **testează** proprietatea, **numără** câte o îndeplinesc.

> **NUMĂRĂ CU O DECIZIE ÎN CORPUL BUCLEI**
>
> În corpul buclei adăugăm un `if`: dacă cifra îndeplinește condiția, creștem un contor cu `1`. Restul cifrelor sunt ignorate — nu strică numărătoarea.

A

EXERCIȚIUL A · POTRIVEȘTE CONDIȚIA

Ce condiție traduce fiecare cerință?

ASOCIERE

Pentru fiecare cerință, alege din rezervă condiția Python care s-ar pune în `if ___:` (unde `cifra` este cifra curentă). Scrie litera condiției pe linie.

`A · cifra % 2 == 0` `B · cifra > 5` `C · cifra == 7` `D · cifra % 2 == 1`

CERINȚA (ÎN CUVINTE)	CONDIȚIE
Numără cifrele **pare**.	_____
Numără cifrele **mai mari decât 5**.	_____
Numără cifrele **egale cu 7**.	_____
Numără cifrele **impare**.	_____

HINT *o cifră este pară dacă restul ei la 2 este 0; impară dacă restul este 1.*

B

EXERCIȚIUL B · COMPLETEAZĂ CODUL

Numără aparițiile cifrei k

COMPLETARE

Programul citește un număr `n` și o cifră `k`, apoi numără de câte ori apare cifra `k` printre cifrele lui `n`. Completează cele **4 spații libere**.

```
n = int(input())
k = int(input())
contor = ______
while n ______ 0:
    cifra = n % 10
    if cifra ______ k:
        contor = contor + 1
    n = n ______ 10
print(contor)
```

1. valoarea de pornire a contorului **2.** condiția buclei **3.** testul de egalitate cu k **4.** eliminarea ultimei cifre

HINT *contorul pornește de la `0`; bucla merge cât timp `n > 0`; testezi `cifra == k`; elimini ultima cifră cu `// 10`.*

SECȚIUNEA 5 · TEMĂ NOUĂ

5 Eliminarea cifrelor, *păstrând ordinea*

Uneori vrem să păstrăm doar unele cifre — de exemplu, să eliminăm cifrele pare din `4729` și să rămânem cu `79`. Aici apare o dificultate reală: parcurgând cifrele cu `% 10`, le obținem **de la coadă spre cap**. Dacă nu suntem atenți, numărul reconstruit iese **oglindit**.

Ca să păstrăm ordinea originală, ținem minte o **putere a lui 10** care crește la fiecare cifră păstrată. Compară cele două metode, pentru cifrele pare din `29400` (rezultat corect: `2400`):

METODA GREȘITĂ

```
rez = rez * 10 + cifra
```

Oglindește cifrele păstrate **și** pierde zerourile de la coadă — rezultat greșit.

METODA CORECTĂ

```
rez = rez + cifra * putere
putere = putere * 10
```

Fiecare cifră intră pe poziția ei corectă → ordinea și zerourile se păstrează.

A

EXERCIȚIUL A · ORDONEAZĂ ALGORITMUL

Pune liniile în ordinea corectă

ORDONARE

Liniile de mai jos păstrează doar **cifrele pare** din `n`, în ordinea lor. Numerotează-le de la **1** la **10** în căsuțele din stânga. Respectă indentarea și, în corpul buclei, ordinea: *întâi* adaugi cifra pe poziția ei, *apoi* crești puterea.

- [ ] `    cifra = n % 10`
- [ ] `rezultat = 0`
- [ ] `print(rezultat)`
- [ ] `n = int(input())`
- [ ] `        putere = putere * 10`
- [ ] `    n = n // 10`
- [ ] `putere = 1`
- [ ] `    if cifra % 2 == 0:`
- [ ] `while n > 0:`
- [ ] `        rezultat = rezultat + cifra * putere`

HINT *întâi cele trei inițializări (`n`, `rezultat`, `putere`), apoi bucla. În `if`, adaugi cifra* **înainte** *de a crește puterea; `n = n // 10` este ultima linie din buclă.*

B

EXERCIȚIUL B · PREZICE REZULTATUL

Ce rămâne după eliminare?

COMPLETARE

Pentru fiecare rând, scrie ce număr rămâne după eliminarea cifrelor cerute, **păstrând ordinea** celorlalte. Dacă nu mai rămâne nicio cifră, răspunsul este `0`.

NUMĂRUL	ELIMINĂ	RĂMÂNE
`4729`	cifrele pare	_____
`29400`	cifrele impare	_____
`5050`	cifrele egale cu 0	_____
`2468`	cifrele pare	_____

HINT *la `2468` toate cifrele sunt pare — nu rămâne nimic, deci răspunsul este `0`.*

SECȚIUNEA 6 · TEMĂ NOUĂ

6 Palindrom și *cifra de control*

Un număr este **palindrom** dacă se citește la fel de la stânga la dreapta și invers: `121`, `4554`, `7`. Ca să verificăm, comparăm numărul cu **oglinditul** lui.

> **PĂSTREAZĂ O COPIE!**
>
> Construirea oglinditului „consumă" numărul (îl duce la `0` prin împărțiri repetate). Ca să-l putem compara la final cu originalul, trebuie să **păstrăm o copie** a numărului înainte de a începe bucla.

A

EXERCIȚIUL A · REPARĂ CODUL

Verificarea de palindrom minte

GRILĂ CORECTARE

Codul ar trebui să verifice dacă `n` este palindrom, dar spune că **orice** număr palindrom *nu* este palindrom. Citește-l cu atenție și încercuiește care este problema.

```
1  n = int(input())
2  oglindit = 0
3  while n > 0:
4      cifra = n % 10
5      oglindit = oglindit * 10 + cifra
```

```
    n = n // 10
if n == oglindit:
    print("palindrom")
else:
    print("nu este palindrom")
```

_____ *a)* `oglindit` pornește greșit — ar trebui de la `1`.

_____ *b)* după buclă `n` a ajuns `0`; trebuie păstrată o copie și comparat **copia** cu `oglindit`.

_____ *c)* condiția buclei ar trebui să fie `while n >= 0`.

→ Scrie liniile pe care le adaugi/schimbi ca să repari: _____

HINT *la linia 7, `n` nu mai este numărul original (a fost consumat de buclă). Salvează `copie = n` la început și compară `copie == oglindit`.*

B

EXERCIȚIUL B · CALCULEAZĂ CIFRA DE CONTROL

O cifră care detectează greșelile

COMPLETARE

Cifra de control se obține prin **însumare repetată**: aduni cifrele; dacă suma are mai multe cifre, aduni din nou cifrele sumei; repeți până rămâne o singură cifră. Pentru `85`: `8 + 5 = 13`, apoi `1 + 3 = 4` → cifra de control `4`. Completează tabelul.

NUMĂRUL	SUME INTERMEDIARE (SCRIE PAȘII)	CIFRA DE CONTROL
`85`	8 + 5 = 13 → 1 + 3 = 4	`4`
`777`	_____	_____
`1234`	_____	_____
`99`	_____	_____

HINT *repeți însumarea cifrelor* **cât timp** *rezultatul are mai mult de o cifră (adică e `>= 10`).*

SECȚIUNEA 7 · COD GHIDAT

7 Verifică-ți *codul*

Acum scrii tu programele. Fiecare problemă are 5 cazuri de test alese special. Parcurge cifrele **doar cu** `while` (`% 10` și `// 10`), fără funcții. Afișează **doar** ce se cere, pe o singură linie, fără text în plus. Primele trei probleme intră în progres (prag 4 din 5); ultimele două sunt opționale. Verificarea automată rămâne pe platformă.

1

PROBLEMA 7.1 · DE BAZĂ

Produsul cifrelor nenule

SCRIERE COD

Se citește un număr natural `n`. Afișează **produsul cifrelor nenule** (sari peste cifrele egale cu `0`). Ex.: `234` → `24`; `205` → `10`.

HINT *pornește produsul de la `1`. Parcurge cifrele cu `while` și înmulțește doar cifrele diferite de `0`.*

SOLUȚIA TA · PYTHON

2

PROBLEMA 7.2 · DE BAZĂ

Cifra minimă

SCRIERE COD

Se citește un număr natural nenul `n`. Afișează **cea mai mică cifră** a lui. Ex.: `3816` → `1`; `4071` → `0`.

HINT *pornește minimul de la `9` și, la fiecare cifră, dacă e mai mică decât minimul de până acum, actualizează-l. Nu sări peste cifra `0` — ea poate fi chiar minimul!*

SOLUȚIA TA · PYTHON

PROBLEMA 7.3 · DE BAZĂ

3 Elimină cifrele pare, păstrând ordinea

SCRIERE COD

Se citește un număr natural `n`. Afișează numărul rămas după **eliminarea cifrelor pare**, păstrând ordinea cifrelor impare. Dacă nu rămâne nicio cifră, afișează `0`. Ex.: `4729` → `79`; `2468` → `0`.

HINT *folosește o* **putere a lui 10** *care crește la fiecare cifră păstrată, ca să reconstruiești numărul în ordinea corectă (nu oglindit).*

SOLUȚIA TA · PYTHON

EXERCIȚII DIFERENȚIATE · OPȚIONALE

Următoarele două nu contează la deblocarea evaluării. Una e o provocare de extindere, cealaltă o recapitulare de remediere.

★

EXTINDERE · OPȚIONAL

Cifra de control

SCRIERE COD

Se citește un număr natural `n`. Calculează **cifra de control** prin însumare repetată: aduni cifrele; dacă suma are mai multe cifre, aduni din nou cifrele sumei; repeți până rămâne o singură cifră. Ex.: `85` → `4`; `777` → `3`.

HINT *ai nevoie de o buclă exterioară (repetă cât timp `n >= 10`) și una interioară (adună cifrele lui `n`, apoi pune `n` egal cu suma).*

SOLUȚIA TA · PYTHON

REMEDIERE · OPȚIONAL

Numărarea cifrelor

SCRIERE COD

Se citește un număr natural `n`. Afișează **câte cifre** are. (Numărul `0` are o cifră.) Ex.: `42` → `2`; `99999` → `5`.

HINT *un contor care crește cu `1` la fiecare cifră eliminată cu `n = n // 10`. Tratează separat cazul `n = 0`.*

SOLUȚIA TA · PYTHON