INFORMATICĂ · PYTHON · CLASA A IX-A · PRELUCRĂRI ALE DATELOR NUMERICE · LECȚIA 4

Cifrele unui număr

Durată: 100 min Nivel: mate-info · Python Înainte: lecțiile py9-02, py9-03

CE VEI ȘTI DUPĂ LECȚIE

- Accesezi orice cifră a unui număr cu // și %, fără să-l transformi în text.
- Adaugi o cifră la dreapta (n · 10 + c) sau la stânga (c · p + n) unui număr.
- Construiești un număr nou din cifrele altuia: oglindit, cu cifre eliminate sau înlocuite, păstrând ordinea.
- Alegi argumentat între varianta aritmetică și cea cu str().

1. Acces la o cifră

Două operații fac aproape toată treaba: `n % 10` dă **ultima cifră**, iar `n // 10` **taie** ultima cifră.

```
n = 58274
print(n % 10)          # 4      ultima cifră
print(n // 10)         # 5827   fără ultima cifră
print(n // 100 % 10)   # 2      a treia cifră de la dreapta
print(divmod(n, 10))   # (5827, 4)  ambele deodată
```

Regula generală: cifra de pe poziția `k` numărată de la dreapta, începând cu 0, este `n // 10**k % 10`. Pentru `k = 4`: `58274 // 10000 % 10 = 5`.

2. Parcurgerea cifrelor

Șablonul îl știi din gimnaziu: iei ultima cifră, o prelucrezi, o tai, până nu mai rămâne nimic.

```
while n > 0:
    c = n % 10         # cifra curentă (de la dreapta la stânga)
    ...                # o prelucrezi
    n //= 10           # o tai
```

ATENȚIE — GREȘELI FRECVENTE

Cazul `n = 0`. Numărul 0 are o cifră, dar `while n > 0` nu face niciun pas, deci un program care numără cifrele afișează 0. Dacă `n` poate fi 0, folosește varianta „execută întâi, verifică apoi”:

```
n = int(input())
nr = 0
cmax = 0
while True:
    c = n % 10
    nr += 1
    if c > cmax:
        cmax = c
    n //= 10
    if n == 0:
        break
print(nr, cmax)        # 58274 → 5 8     0 → 1 0     7 → 1 7
```

Bucla **distruge** numărul: la final `n` este 0. Dacă mai ai nevoie de el, lucrează pe o copie: `m = n`, apoi parcurgi `m`.

Câți pași face bucla? Câte cifre are `n`, adică aproximativ $\log_{10} n$: pentru un număr de 18 cifre, 18 pași. Parcurgerea cifrelor este foarte rapidă.

3. Adăugarea unei cifre

Unde	Formula	Exemplu (`n = 427`, `c = 9`)
la dreapta	`n = n * 10 + c`	`4279`
la stânga	`n = c * p + n`, unde `p = 10**(numărul de cifre ale lui n)`	`9 * 1000 + 427 = 9427`

Adăugarea **la dreapta** nu are nevoie de nimic în plus. Pentru adăugarea **la stânga** trebuie să știi pe ce poziție pui cifra, adică puterea lui 10 potrivită. Când construiești un număr cifră cu cifră spre stânga, ții `p` într-o variabilă și îl înmulțești cu 10 după fiecare cifră adăugată.

4. Construirea unui număr nou

Oglinditul. Iei cifrele de la dreapta și le adaugi tot **la dreapta** rezultatului: ordinea se inversează.

```
r = 0
while n > 0:
    r = r * 10 + n % 10
    n //= 10
print(r)               # 58274 → 47285
```

Păstrarea ordinii: eliminarea cifrelor impare. Iei cifrele tot de la dreapta, dar le adaugi **la stânga** rezultatului, cu `p`: ordinea se păstrează.

```python
r = 0
p = 1
while n > 0:
    c = n % 10
    if c % 2 == 0:      # păstrăm doar cifrele pare
        r = c * p + r
        p *= 10
    n //= 10
print(r)                # 2735048 → 2048     4130 → 40
```

La `4130` rezultatul corect este `40`: cifrele pare sunt 4 și 0, **în această ordine**, iar zeroul de la final contează. O idee „oglindesc de două ori" ar pierde zeroul (vezi secțiunea 5).

Înlocuirea unei cifre folosește aceeași schemă: în loc să sari peste cifră, adaugi altă cifră. De exemplu, `if c == 7: c = 1`, apoi `r = c * p + r` și `p *= 10` pentru **fiecare** cifră. Pentru `17270` se obține `11210`.

> **DE ȘTIUT**
>
> Dacă nu rămâne nicio cifră (de exemplu, `1357` nu are cifre pare), rezultatul este `0`. Nu poți deosebi „rezultatul este 0" de „nu există cifre". Când enunțul cere un mesaj separat, numără cifrele păstrate într-o variabilă.

5. Oglinditul oglinditului nu dă mereu numărul inițial

`n`	oglindit	oglinditul oglinditului
58274	47285	58274
120	21	**12**
1200	21	**12**

Zerourile de la finalul lui `n` ajung la începutul oglinditului, unde nu mai contează. De aceea un singur test „frumos" (`58274`) nu dovedește că `ogl(ogl(n)) == n`; testul `120` arată contrariul.

6. Varianta cu str() și când e potrivită

Python poate transforma un număr în text, iar textul se parcurge caracter cu caracter.

Ce vrei	Aritmetic	Cu `str()`
numărul de cifre	bucla din secțiunea 2	`len(str(n))`
suma cifrelor	`s += n % 10` în buclă	`sum(int(c) for c in str(n))`
cifra maximă	comparații în buclă	`int(max(str(n)))`
oglinditul	`r = r * 10 + n % 10`	`int(str(n)[::-1])`

Pentru `58274` variantele dau aceleași rezultate: `5`, `26`, `8`, `47285`. Aici `[::-1]` înseamnă „textul citit de la coadă la cap" (`"123"[::-1]` este `"321"`); felierea (*slicing*) o studiem pe larg la liste, în lecția py9-15.

Folosești `str()` când	Folosești varianta aritmetică când
vrei o verificare rapidă sau un test pentru varianta ta	enunțul cere algoritmul (Bac, pseudocod)
lucrezi cu numărul ca text oricum (de exemplu, un cod)	construiești un număr nou, cu `p`
contează doar claritatea codului, nu algoritmul (de exemplu, `len(str(n))`)	vrei să știi exact ce se întâmplă pas cu pas

ATENȚIE — GREȘELI FRECVENTE

`str()` are capcane: `str(-305)` are 4 caractere (`"-305"`), iar `int(str(-305)[::-1])` dă `ValueError`, pentru că textul inversat este `"503-"`.

LA BAC

La Bac nu poți scrie `str(n)[::-1]` în pseudocod. Trebuie să știi pe de rost cele trei mișcări: `n % 10`, `n // 10`, `r * 10 + c` (și `c * p + r`).

7. Verifică-te

1. Ce dau `9081 % 100`, `9081 // 100` și `9081 // 10 % 10`?
2. Ce afișează programul de numărare a cifrelor cu `while n > 0` pentru `n = 0`? Cum îl repari?
3. Adaugă cifra 3 la stânga lui 50 și la dreapta lui 50.
4. Ce obții dacă, în programul de eliminare a cifrelor impare, scrii `r = r * 10 + c` în loc de `r = c * p + r`? Încearcă pe `2735048`.
5. Dă un număr de trei cifre pentru care oglinditul oglinditului nu este numărul inițial.
6. De ce `int(str(n)[::-1])` nu merge pentru numere negative?

Răspunsuri

1. `81`, `90`, `8`.
2. Afișează `0`, dar 0 are o cifră. Fie tratezi separat `n == 0`, fie folosești `while True` cu `break` după tăierea cifrei.
3. `350` și `503`.
4. `8402`: cifrele pare apar în ordine inversă (oglinditul lui `2048`).
5. Orice număr cu ultima cifră 0, de exemplu `350` (oglindit `53`, apoi `35`).
6. `str(-305)` este `"-305"`, inversat devine `"503-"`, iar `int("503-")` dă `ValueError`.

Probleme de lucru în clasă (pbinfo)

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

- [ ] **#66** **numaruldecifre** — ușoară
  https://www.pbinfo.ro/probleme/66/numaruldecifre
- [ ] **#69** **oglindit** — ușoară
  https://www.pbinfo.ro/probleme/69/oglindit

- [ ] #360 **cifra** — ușoară
  https://www.pbinfo.ro/probleme/360/cifra

Temă pentru acasă (pbinfo)

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

- [ ] #68 **ciframaxima** — ușoară
  https://www.pbinfo.ro/probleme/68/ciframaxima
- [ ] #3932 **stergezerouri** — ușoară
  https://www.pbinfo.ro/probleme/3932/stergezerouri
- [ ] #368 **nrpal** — ușoară
  https://www.pbinfo.ro/probleme/368/nrpal
- [ ] #1059 **schimbcifre** — medie
  https://www.pbinfo.ro/probleme/1059/schimbcifre
