FIȘĂ DE LUCRU · PYTHON · CLASA A IX-A · FIȘA 2 · LECȚIA PY9-02

De la pseudocod (sau C++) la Python

Durată: 100 min Nivel: mate-info · Python

CE VEI ȘTI DUPĂ LECȚIE

- Prezici ce afișează un program Python scurt, cu atenție la //, %, range() și tipul lui input().
- Repari programele scrise „ca în C++" care nu merg în Python.
- Scrii programe complete cu citire de pe un rând, decizii, repetiții și afișare formatată.

Ce trebuie să știi (o pagină)

Regula	Exemplu	Rezultat
`input()` dă **text**; convertești cu `int()` / `float()`	`int("7") + 1`	`8`
mai multe numere pe un rând	`a, b = map(int, input().split())`	
număr necunoscut de valori (până la EOF)	`v = list(map(int, sys.stdin.read().split()))`	cu `import sys`
afișare pe un rând / pe rânduri	`print(*v)` / `print(*v, sep="\n")`	
stil PEP 8	`suma_pare`, `a = b + 1`, 4 spații	nume mici cu `_`
`/` real, `//` cât întreg, `%` rest, `**` putere	`7/2`, `7//2`, `7%2`, `2**10`	`3.5`, `3`, `1`, `1024`
`//` rotunjește în jos, `%` are semnul împărțitorului	`-7//2`, `-7%2`	`-4`, `1`
`range(a, b, pas)` nu include `b`	`range(2, 9, 3)`	2, 5, 8
blocurile = indentare (4 spații) după `:`	`if x > 0:` ↵ `····print(x)`	
`elif` = „altfel, dacă"; comparații înlănțuite	`1 <= x <= 9`	
`and`, `or`, `not`	`not (x > 0 and y > 0)`	
`print(…, sep="…", end="…")`, f-string	`f"{x:.2f}"`, `f"{n:4}"`	2 zecimale, lățime 4
`divmod(a, b)` dă câtul și restul	`divmod(135, 60)`	`(2, 15)`
nu există `i++` și nici `do-while`	`i += 1`; `while True: … break`	

SFAT

La nivelul 1 scrie răspunsul pe hârtie, apoi rulează codul. Unde ai greșit, notează **de ce**: aici se învață, nu la cele ghicite.

Nivelul 1 · Prezice

★ citește și prezice

P1. Ce afișează?

```
print(7 // 2, 7 % 2, 7 / 2, 2 ** 3 ** 2, -8 // 3, -8 % 3)
```

Indiciu: `**` se evaluează de la dreapta la stânga.

P2. Text sau număr?

```
x = "4"
y = 4
print(x * 3, y * 3, x + str(y), int(x) + y)
```

P3. Ce valori afișează bucla?

```
for i in range(10, 2, -3):
    print(i, end=" ")
```

P4. Două `if`-uri separate sau un singur lanț? Ce afișează pentru `a = 5`?

```
if a > 3: print("A", end="")
if a > 4: print("B", end="")
elif a > 2: print("C", end="")
else: print("D", end="")
```

P5. Ce afișează pentru `n = 1234`? Ce face programul, în cuvinte?

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

P6. Ce afișează? Atenție la ordinea lui `continue` și `break`.

```
s = 0
for i in range(1, 11):
    if i % 3 == 0: continue
    if i > 8: break
    s += i
print(s)
```

P7. Adevărat sau fals? `print(1 < 3 < 2, 2 < 3 < 4, not 0, 7 == 7.0)`

P8. De câte ori se execută `k += 1`?

```python
k = 0
for i in range(4):
    for j in range(i):
        k += 1
```

Nivelul 2 · Găsește și repară

★★ găsește și repară

P9. Programul ar trebui să afișeze dublul numărului citit. Pentru `7` afișează `77`. De ce? Repară-l.

```python
x = input()
print(x * 2)
```

P10. *Găsește și repară stilul.* Programul de mai jos calculează suma numerelor **pare** din `[a, b]` și funcționează corect (`3` și `10` dau `28`). Stilul lui însă încalcă PEP 8. Găsește cel puțin **cinci** abateri (nume, spații, indentare, instrucțiuni pe același rând, parantezele de la `print`) și rescrie programul îngrijit, cu nume care spun ce fac. Rulează ambele variante pe `3, 10` și pe `3, 3` și verifică dacă dau aceleași rezultate.

```python
def F(X,Y):
  R=0
  for I in range(X,Y+1):
      if I%2==0:R+=I
  return R
a=int(input());b=int(input())
print( F(a,b) )
```

P11. *Citirea datelor.* Programul trebuie să citească **oricâte** numere întregi, puse pe oricâte rânduri (de exemplu `4 7` și pe rândul următor `1`), și să afișeze suma și cel mai mare număr, adică `12 7`. Dacă nu există niciun număr, afișează `nimic`. Găsește **trei** probleme și repară programul.

```python
s = 0
mx = 0
while True:
    x = int(input())
    s += x
    if x > mx:
        mx = x
print(s, mx)
```

P12. Un elev a scris Python „ca în C++". Găsește **trei** greșeli și scrie varianta corectă.

```python
x = int(input())
i = 0
if x > 0 and < 10:
    i++
else if x == 10:
    i = 2
print(i)
```

P13. *Rânduri amestecate.* Pune rândurile în ordine ca programul să afișeze **câte numere din 1…n sunt divizibile cu 3 sau cu 5**. Un rând este în plus. (Indentarea o stabilești tu.)

```
A)  c += 1
B)  print(c)
C)  for i in range(1, n + 1):
D)  n = int(input())
E)  if i % 3 == 0 or i % 5 == 0:
F)  c = 0
G)  if i % 3 == 0 and i % 5 == 0:
```

Pentru `n = 20` ce afișează?

P14. Programul nu se mai oprește. Găsește cauza și repar-o.

```python
n = int(input())
c = 0
while n > 0:
    c += 1
    n // 10
print(c)
```

Nivelul 3 · Scrie cod

★★ scrie cod

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

P15. Biletul de tren. Primii 50 km costă 0,50 lei/km, iar fiecare km în plus costă 0,30 lei. Citește distanța și afișează prețul cu două zecimale.

intrare	ieșire
`30`	`15.00`
`50`	`25.00`
`120`	`46.00`

P16. Săptămâna meteo. Citește de pe **un singur rând** temperaturile a 7 zile. Afișează media cu o zecimală și, pe rândul următor, câte zile au fost **peste medie**. Test: `3 5 -1 0 8 2 4` → `3.0` și `3`.

P17. Codul PIN. Un PIN de 4 cifre este valid dacă nu are două cifre **vecine** egale și suma cifrelor este pară. Afișează `VALID` sau `INVALID`. Teste: `1234 → VALID`, `1124 → INVALID`, `1235 → INVALID`.

P18. Tabla înmulțirii. Pentru `n` citit, afișează tabla `n × n`, fiecare număr pe 4 caractere (`f"{x:4}"`). Pentru `n = 3`:

```
   1   2   3
   2   4   6
   3   6   9
```

P19. Ping-Pong. Afișează numerele de la 1 la `n`, dar multiplii de 3 devin `Ping`, multiplii de 5 devin `Pong`, iar multiplii de 15 devin `PingPong`. Test: pentru `n = 15`, ultimele patru valori afișate sunt `Ping 13 14 PingPong`.

P20. Cronometrul de bucătărie. Citește un număr de minute și afișează-l ca `ore:minute`, cu minutele pe două cifre. Folosește `divmod`. Teste: `135 → 2:15`, `59 → 0:59`, `600 → 10:00`.

Nivelul 4 · Pune mintea la contribuție

★★★ pune mintea la contribuție

P21. range invers. Vrem ca `range(20, b, -5)` să producă **exact** `20, 15, 10, 5`. Câte valori întregi poate avea `b`? Care sunt?

P22. De ce −7 // 2 = −4? Verifică pentru perechile `(-7, 2)`, `(7, -2)`, `(-9, 4)`, `(13, 5)` că `a == (a // b) * b + a % b`. Ce regulă respectă Python, pe care C++ n-o respectă la fel?

P23. Numere uriașe. Câte cifre are `2 ** 1000`? Calculează în **două** feluri: o dată cu `len(str(...))` și o dată cu o buclă `while` care taie cifre. Ar fi mers asta în C++ cu `long long`?

P24. Fără do-while. Traduce în Python programul C++ de mai jos (citește numere până la 0, inclusiv, și afișează suma lor). Folosește `while True` și `break`. Ce afișează pentru `4 7 0`?

```
int x, s = 0;
do { cin >> x; s += x; } while (x != 0);
cout << s;
```

P25. Ghicește numărul. Scrie un joc: calculatorul alege un număr între 1 și 100 (`import random`, `random.randint(1, 100)`), iar jucătorul primește „mai mare” / „mai mic” până ghicește. La final afișează numărul de încercări. Întrebare: dacă joci perfect (înjumătățești mereu intervalul), de câte încercări ai nevoie **cel mult**? De ce?

Probleme de lucru în clasă (pbinfo)

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

- [ ] `#939` **sum00** — ușoară
  https://www.pbinfo.ro/probleme/939/sum00
- [ ] `#105` **max2** — ușoară
  https://www.pbinfo.ro/probleme/105/max2
- [ ] `#51` **suma** — ușoară
  https://www.pbinfo.ro/probleme/51/suma

Temă pentru acasă (pbinfo)

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

- [ ] `#109` **paritate** — ușoară
  https://www.pbinfo.ro/probleme/109/paritate
- [ ] `#2` **suma-n** — ușoară
  https://www.pbinfo.ro/probleme/2/suma-n
- [ ] `#10` **suma-cifrelor** — ușoară
  https://www.pbinfo.ro/probleme/10/suma-cifrelor