test-e.ro

CAIET ELEV

Lecția 5

Recapitulare L1–L4: O zi din viața ta, scrisă în cod

O zi întreagă, de la alarmă până seara: fiecare moment al ei recapitulează o idee din lecțiile 1–4.

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

NUMELE ELEVULUI .....

CUM FOLOSEȘTI CAIETUL

Rezolvă exercițiile pe hârtie: ordonezi, încercuiești, completezi și scrii singur soluția, apoi le verifici pe platformă. Aceasta este o recapitulare a lecțiilor 1–4: algoritmi, decizii (if/elif/else), bucle (while, for cu range), operatorii % și //. Ce ai voie să folosești: exact aceste construcții, în Python. Fără funcții (def/return), fără liste sau alte structuri neintroduse încă.

CUPRINS

1 Alarma sună · algoritm și schemă logică .....
2 Cu ce mă îmbrac? · if / if-else .....
3 Punctele din joc · if-elif-else .....
4 Autobuzul · bucla while .....
5 Orele de la școală · for cu range() .....
6 Recreația · operatorii % și // .....
7 Cod ghidat · 1 problemă .....

SECȚIUNEA 1 · 06:45 · L1 · ALGORITM · PSEUDOCOD · SCHEME LOGICE

1 Alarma *sună*

Te trezești. Înainte să faci orice, mintea ta rulează un **algoritm**: pașii rutinei de dimineață, în ordine.

> **SCENA**
>
> Alarma sună la 6:45. Ce faci, pas cu pas, ca să ajungi gata de plecare? Ordinea contează — la fel ca într-un algoritm.

Un **algoritm** e o secvență de pași clari, în ordine, care duce sigur la un rezultat. Pașii trebuie să fie clari, fără ambiguitate, iar algoritmul trebuie să se termine.

O **structură de decizie (dacă)** se desenează în schema logică printr-un romb: rombul pune o întrebare, iar răspunsul desparte drumul în **două ramuri** — una pentru **DA**, una pentru **NU**. Fiecare ramură duce la altă acțiune, apoi drumurile se reunesc.

> **FORMELE DIN SCHEMA LOGICĂ**
>
> **Oval** = start/stop, **romb** = decizie, **dreptunghi** = acțiune.

A

EXERCIȚIUL A · ORDONARE

Pune-ți dimineața în ordine

ORDONARE

Numerotează de la 1 la 5 pașii algoritmului „de la alarmă la ușă”, în ordinea corectă a dimineții tale. Atenție: doi pași sunt capcane — încalcă o proprietate a algoritmului (un pas neclar și unul care n-are sfârșit). În căsuța lor scrie X.

- [ ] Ia micul dejun
- [ ] Apasă snooze la nesfârșit
- [ ] Ieși pe ușă
- [ ] Oprește alarma
- [ ] Fă „ceva” prin cameră
- [ ] Spală-te și îmbracă-te
- [ ] Ridică-te din pat

HINT *Gândește cronologic: ce faci prima dată după ce sună alarma? Algoritmul cere pași clari și care se termină.*

B

EXERCIȚIUL B · SCHEMĂ LOGICĂ

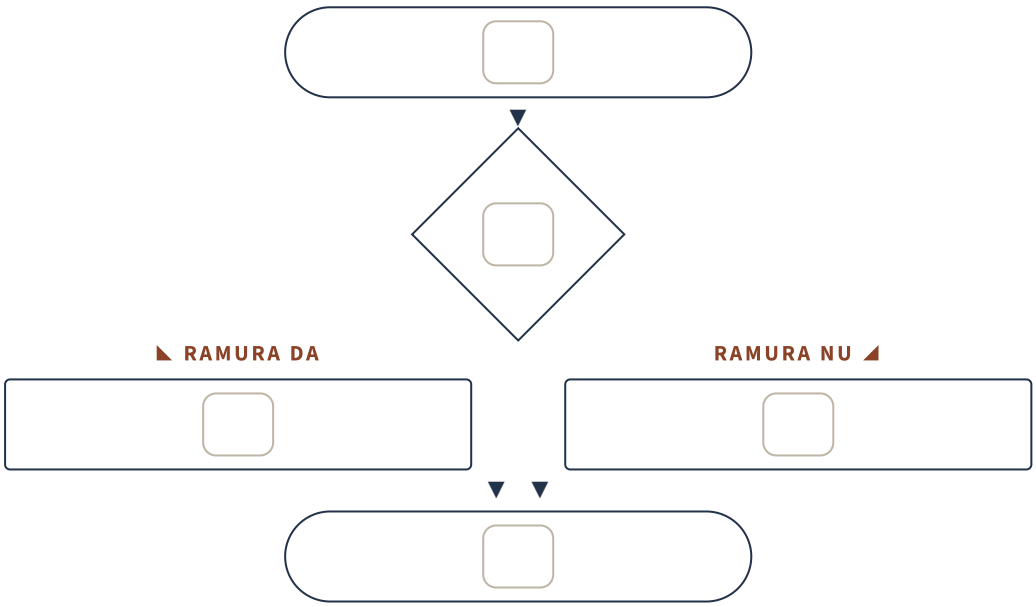
Schema logică a deciziei „mai dorm 5 minute?”

SCHEMĂ LOGICĂ

Scrie în fiecare formă a schemei litera etichetei potrivite. Ține cont de formă: oval = start/stop, romb = decizie, dreptunghi = acțiune. O etichetă rămâne nefolosită.

A Te ridici din pat | B STOP | C Citește un fișier | D Mai e devreme? (ora < 7)

E START | F Mai dormi 5 minute



HINT *Ramura DA și ramura NU duc la acțiuni diferite. „Citește un fișier” n-are ce căuta aici.*

SECȚIUNEA 2 · 07:10 · L2 · STRUCTURA DE DECIZIE (IF / IF-ELSE)

2 Cu ce mă *îmbrac?*

Te uiți pe telefon la vreme. O singură **decizie** îți schimbă ținuta.

> SCENA
>
> Afară plouă? Atunci iei umbrela. Altfel, nu. O condiție, două ramuri — exact ce face if-else.

La fiecare cod, o singură **condiție** decide ce ramură se execută.

A

EXERCIȚIUL A · GRILĂ

Ghicește ramura

GRILĂ

Citește fiecare program și încercuiește ce se afișează.

```
1  temperatura = 8
2  if temperatura < 10:
3      print("Ia geaca groasă")
4  else:
5      print("Geaca subțire e ok")
```

1. temperatura este 8. Ce se afișează?

Ambele | Geaca subțire e ok | Nimic | Ia geaca groasă

```
1  ploua = False
2  if ploua:
3      print("Ia umbrela")
```

2. ploua este False. Ce se afișează?

False | Ia umbrela | Eroare | Nimic

```
1  temp = 24
2  if temp >= 25:
3      print("Tricou")
4  else:
5      print("Bluză")
```

3. temp este 24. Ce se afișează?

Nimic | Tricou | Bluză | Ambele

```
1  vant = 30
2  ploua = True
3  if ploua and vant > 20:
4      print("Stai în casă")
5  else:
6      print("Poți ieși")
```

4. ploua = True, vant = 30. Ce se afișează?

Poți ieși | Ambele | Stai în casă | Nimic

```
temp = 12
if temp < 0:
    print("Fular")
else:
    print("Fără fular")
```

5. temp este 12. Ce se afișează?

Fără fular | Eroare | Fular | Ambele

B

EXERCIȚIUL B · GRILĂ

Întrebarea cheie — de ce if-else și nu două if?

GRILĂ

Încercuiește răspunsul corect.

1. Vrei să afișezi „Ia umbrela” când plouă, altfel „Lasă umbrela acasă”. Care variantă e corectă și sigură?

if ploua: … / if not ploua: … (două if-uri) | if ploua: … / else: … (if-else)

3 Punctele din *joc*

SECȚIUNEA 3 · 07:30 · L2 · DECIZII ÎN CASCADĂ (IF-ELIF-ELSE)

Ieși pe ușă și, în drum spre stație, deschizi jocul preferat să vezi câte puncte ai strâns. În funcție de cât ai, deblochezi alt skin — **decizii înlănțuite**.

> SCENA
>
> Dacă ai ≥ 500 puncte → skin legendar. Altfel dacă ai ≥ 200 → skin rar. Altfel dacă ai ≥ 50 → skin comun. Praguri verificate în ordine.

La **if-elif-else**, condițiile se verifică **de sus în jos** și se oprește la **prima adevărată**.

A

EXERCIȚIUL A · ORDONARE

Repară cascada de decizii

ORDONARE

Ai strâns 500 de puncte în joc și aștepți „Skin legendar”. Dar programul îți oferă altceva! Numerotează ramurile de la 1 la 3, în ordinea în care trebuie scrise, ca fiecare punctaj să deblocheze skin-ul corect. Prima ramură se scrie cu if, celelalte cu elif; ramura else rămâne la final.

- [ ] puncte >= 200 → "Skin rar (200 puncte)"
- [ ] puncte >= 50 → "Skin comun (50 puncte)"
- [ ] puncte >= 500 → "Skin legendar (500 puncte)"

```
else:
    print("Nu ai destule puncte")
```

HINT *Dacă pui puncte >= 50 primul, atunci și 520 de puncte „prind” prima ramură (520 >= 50 e adevărat) și programul se oprește acolo.*

SECȚIUNEA 4 · 07:45 · L3 · STRUCTURA REPETITIVĂ CU WHILE

4 Autobuzul

Aștepți în stație. Verifici iar și iar dacă a venit autobuzul tău — **repeți până** când vine.

> **SCENA**
> Nu știi câte autobuze trec până vine al tău. Repeți verificarea cât timp nu a sosit. De aceea while, nu for.

Aștepți în stație autobuzul **linia 5**. La fiecare autobuz verifici: „e al meu?”. **Cât timp** nu e linia 5, mai aștepți. Nu știi dinainte câte trec — de aceea folosești **while**.

```
linia = 0   # inca n-a venit niciun autobuz
verificari = 0
while linia != 5:        # cat timp NU e al meu
    linia = urmatorul_autobuz()
    verificari = verificari + 1
print("A venit! Verificari:", verificari)
```

> **TEST INIȚIAL**
> Condiția se verifică **înainte** de fiecare pas, de aceea `while` poate rula și de zero ori dacă autobuzul tău e deja în stație.

A

EXERCIȚIUL A · GRILĂ

Oprește bucla

GRILĂ

Încercuiește răspunsul corect.

1. Autobuzele care trec, în ordine: 3 · 7 · 2 · 5 · 9. Câte verificări faci până vine linia 5?

   2 3 4 5 6

B

EXERCIȚIUL B · GRILĂ

De ce while și nu for?

GRILĂ

Încercuiește răspunsul corect.

1. La autobuz, de ce am folosit while și nu for cu range()?

   Pentru că o buclă for nu poate folosi deloc condiții, iar aici avem nevoie de o condiție clară.

   Pentru că while rulează mereu mai rapid decât for și ajunge mai repede la autobuzul potrivit.

   Pentru că nu știm dinainte câte autobuze trec până vine al nostru — repetăm cât timp nu a sosit.

SECȚIUNEA 5 · 09:00 · L4 · STRUCTURA REPETITIVĂ CU FOR + RANGE()

5 Orele de la *școală*

Ai 6 ore în orar, una după alta. Știi exact câte — deci le **parcurgi cu for**.

> SCENA
>
> Ora 1, ora 2, … ora 6. Un număr cunoscut de pași. Aici for cu range() e alegerea naturală.

Ai un **număr cunoscut** de ore în orar, deci le parcurgi cu **for** și **range()**.

A

EXERCIȚIUL A · GRILĂ

Vânătoarea de pași

GRILĂ

Pentru fiecare buclă, încercuiește toate valorile pe care le generează range-ul — nici una în plus, nici una în minus.

1. Ce valori parcurge bucla?

```
1  for ora in range(1, 7):
```

0 1 2 3 4 5 6 7

2. Ce valori parcurge bucla?

```
1  for i in range(0, 10, 2):
```

0 1 2 3 4 5 6 7 8 9 10 11

3. Ce valori parcurge bucla?

```
1  for i in range(1, 6):
```

0 1 2 3 4 5 6

B

EXERCIȚIUL B · GRILĂ

for sau while?

GRILĂ

Încercuiește răspunsul corect.

1. La autobuz (secțiunea 4) ai folosit while. La orele de școală folosești for. Care e diferența esențială?

- for e o structură mai nouă, deci e mereu mai bună și o folosim oriunde în locul lui while.
- La autobuz nu știam câte trec (while); la școală știu exact că am 6 ore (for cu range).
- Nu există nicio diferență între ele: for și while sunt mereu interschimbabile, indiferent de situație.

6

SECȚIUNEA 6 · 13:30 · TRANSVERSAL · OPERATORII % ȘI //

Recreația

Împarți gustarea la prieteni și calculezi cât mai e până acasă — **cât întreg** și **rest**.

> **SCENA**
>
> Ai 20 de biscuiți și sunteți 6 prieteni. Câți primește fiecare în mod egal și câți rămân? Câtul și restul, în viața reală.

În recreație tot calculezi: cât îi dai fiecăruia (`//` = câtul întreg) și cât rămâne (`%` = restul).

A

EXERCIȚIUL A · COMPLETARE

Cât și rest, în viața reală

COMPLETARE

Calculează fiecare expresie și scrie rezultatul pe linie.

1. ***Împarte biscuiții***

   Ai 20 de biscuiți și sunteți 6 prieteni. Vrei să dai la fiecare câți poți, egal. Câți biscuiți primește fiecare?

   `20 // 6 =` _____

2. ***Cât rămâne?***

   Aceiași 20 de biscuiți la 6 prieteni, după ce fiecare a luat partea egală. Câți biscuiți rămân nedați?

   `20 % 6 =` _____

3. ***Câte pagini?***

   O listă are 50 de melodii și afișezi câte 10 pe pagină. Toate trebuie să încapă. Câte pagini complete ies?

   `50 // 10 =` _____

4. ***A câta oră?***

   Sunt 13:30. În format de 24h, vrei ora în format de 12h (ceasul de perete). Ce oră arată ceasul de 12h?

   `13 % 12 =` _____

HINT *La // împarți și păstrezi doar partea întreagă (cât de multe încap). La % cauți ce rămâne după împărțirea egală.*

7 SECȚIUNEA 7 · COD GHIDAT

Scrie cod care *trece testele*

O singură problemă — boss-ul zilei, la 17:00. Aici se întâlnesc tot ce-ai recapitulat: o **buclă for** care citește rundele și un **if** care decide dacă streak-ul continuă. Scrie soluția pe hârtie, urmărește-o mental pe exemple și abia apoi tastează-o pe platformă.

1

PROBLEMA 7.1 · COD GHIDAT

Streak-ul din joc

SCRIERE COD

Se citește n (numărul de runde), apoi n scoruri. Afișează câte runde consecutive de la început ai avut scor >= 100. La prima rundă sub 100, streak-ul se oprește, dar codul citește toate cele n scoruri — fără break. Completează scheletul:

```
n = int(input())
streak = 0
activ = True
for i in range(n):
    scor = int(input())
    # TODO: completeaza aici logica.
    # atunci cand un scor este >= 100, streak creste cu 1,
    # dar la primul scor sub 100, activ devine fals.
    # bucla trebuie insa sa citeasca
    # toate cele n scoruri.
print("Streak:", streak)
```

Ex.: pentru intrarea 3 / 120 / 150 / 90 se afișează Streak: 2; pentru intrarea 4 / 200 / 200 / 200 / 200 se afișează Streak: 4.

HINT **Atenție.** *Streak crește doar cât timp activ e adevărat ȘI scor >= 100. La primul scor sub 100, activ devine False — dar bucla citește toate cele n scoruri.*

HINT *Fără break: indicatorul activ, odată devenit False, nu mai lasă streak-ul să crească.*

SOLUȚIA TA · PYTHON