test-e.ro

CAIET ELEV

Lecția 14

Recapitulare integrată (lecțiile 6–13)

Lecția adună la un loc două blocuri învățate separat.

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

NUMELE ELEVULUI .....

CUM FOLOSEȘTI CAIETUL

Rezolvă exercițiile pe hârtie: urmărești execuția pas cu pas, completezi, asociezi și scrii singur soluțiile, apoi le verifici pe platformă. Ce ai voie să folosești: secvențial, if/elif/else, while, for cu range, %, //, **, ord() și chr(). Fără liste, fără def/return, fără atribuiri simultane, fără break, fără diacritice în șirurile Python. Cifrele se parcurg doar cu while, prin % și //, niciodată prin conversie la șir. Paritatea se scrie n % 2 != 0.

CUPRINS

1 Registratura .....
2 Masa de analiză .....
3 Panoul de schițe .....
4 Camera de teste .....
5 Atelierul de mecanică .....
6 Camera serverelor .....
7 Cod ghidat · 6 probleme .....

SECȚIUNEA 1 · L6–L8 · VOCABULARUL ELABORĂRII UNUI PROGRAM

1 Registratura

Ai intrat în tura de noapte la **Laboratorul de Coduri**. Primul lucru pe care îl face orice verificator: își ia fișa de tură și își verifică uneltele.

> **SCENA**
>
> Ecranul de la intrare cere două lucruri înainte să te lase mai departe: să dovedești că știi vocabularul meseriei și să recunoști, dintr-o privire, ce fel de eroare are un program.

A EXERCIȚIUL A · GRILĂ

Fișa de tură

GRILĂ

Șase întrebări despre meseria de programator, exact cum le-am învățat la lecțiile 6–8. Încercuiește răspunsul corect la fiecare întrebare.

1. Într-un program care validează un CNP, numărul citit de la tastatură este o dată de:

   `ieșire` `niciuna dintre acestea` `de lucru` `intrare`

2. Un program rulează până la capăt, fără niciun mesaj de eroare, dar afișează un rezultat greșit. Ce fel de eroare are?

   `eroare logică` `nu are nicio eroare, doar datele sunt greșite` `eroare de sintaxă` `eroare de execuție`

3. Care dintre următoarele NU este o modalitate de reprezentare a unui algoritm?

   `schema logică` `limbajul natural` `fișierul de date de test` `pseudocodul`

4. Un set bun de date de test pentru o problemă cu numere naturale conține:

   `numai numere foarte mari, ca să fie sigur`

   `doar cazuri obișnuite, fiindcă restul nu apar în practică`

   `cazuri obișnuite, cazuri-limită și cazuri invalide` `un singur caz, dacă acela merge`

5. Instrucțiunile `input()` și `print()` reprezintă comunicarea cu programul prin:

   `interfață grafică` `fișiere` `consolă` `rețea`

6. În etapele elaborării unui program, depanarea vine:

   `înainte de analiza problemei` `în locul testării, ca să nu pierdem timp`

   `după testare, ca să corectăm erorile descoperite`

doar dacă programul nu pornește deloc

B

EXERCIȚIUL B · CLASIFICARE

Triajul erorilor

CLASIFICARE

Șase probleme raportate azi de colegi. Scrie în căsuța din fața fiecărei probleme litera sertarului potrivit: **eroare de sintaxă** (programul nu pornește), **eroare de execuție** (pornește și se oprește brusc) sau **eroare logică** (merge până la capăt, dar rezultatul e greșit). O literă se poate folosi de mai multe ori.

- ☐ S-a scris `prin(n)` în loc de `print(n)`
- ☐ Pentru „număr impar” s-a scris `n % 2 == 0`
- ☐ Lipsesc cele două puncte: `while n > 0`
- ☐ La un moment dat se execută `n // 0`
- ☐ Utilizatorul scrie `abc`, iar programul face `int("abc")`
- ☐ Contorul cifrelor pornește de la 1 în loc de 0

A Eroare de sintaxă | B Eroare de execuție | C Eroare logică

SECȚIUNEA 2 · L6 · DATE DE INTRARE / IEȘIRE / DE LUCRU · ETAPELE ELABORĂRII UNUI ALGORITM

2 Masa de *analiză*

Pe masă te așteaptă primul dosar: un **CNP** care trebuie validat. Înainte să scrii o singură linie de cod, un verificator bun face analiza.

> **SCENA**
>
> Un CNP are 13 cifre. Ultima este cifra de control: se calculează din primele 12, fiecare înmulțită cu o pondere, iar suma se împarte la 11 — restul obținut este chiar cifra de control, cu o singură excepție: dacă restul este 10, cifra de control este 1. Ca să ajungi acolo, trebuie mai întâi să știi ce date intră, ce date ies și ce date țin socoteala pe drum.

A

EXERCIȚIUL A · CLASIFICARE

Cele trei feluri de date

CLASIFICARE

Iată variabilele din programul de validare a CNP-ului. Scrie în căsuța din fața fiecărei variabile litera categoriei ei. **Nu te mira** că sunt mai multe date de lucru decât de intrare: așa arată majoritatea programelor — citim puțin, calculăm mult, afișăm puțin.

- ☐ `c` — cifra extrasă la pasul curent
- ☐ `mesaj` — textul DA sau NU, afișat la final
- ☐ `cnp` — numărul de 13 cifre citit de la tastatură
- ☐ `rest` — restul împărțirii sumei la 11
- ☐ `suma` — totalul produselor cifră × pondere

A Date de intrare | B Date de lucru | C Date de ieșire

ÎNAINTE DE A MERGE MAI DEPARTE

Toate codurile folosite în această lecție sunt **fictive**. Nu introduce niciodată un CNP real — al tău sau al altcuiva — într-un program pe care îl testezi. Datele personale se protejează exact ca parolele.

B

EXERCIȚIUL B · ORDONARE

Ordinea de pe fișa de lucru

ORDONARE

Fișa de lucru a laboratorului s-a dezlipit și etapele s-au amestecat. Numerotează etapele de la 1 la 6, în ordinea în care se elaborează de fapt un program.

- ☐ Depanează: găsește și corectează erorile descoperite la testare
- ☐ Identifică datele de intrare, de ieșire și de lucru
- ☐ Analizează problema: ce se dă și ce se cere
- ☐ Concepe algoritmul și reprezintă-l (pseudocod sau schemă logică)
- ☐ Testează programul pe cazuri obișnuite, limită și invalide
- ☐

SECȚIUNEA 3 · L6 · SCHEMA LOGICĂ · L9 · CIFRELE UNUI NUMĂR

3 Panoul de *schițe*

Pe perete e un panou magnetic. Aici algoritmul se desenează înainte să fie scris — fiindcă e mult mai ieftin să muți un magnet decât să rescrii un program.

> **SCENA**
>
> Ca să verifici cifra de control a unui CNP ai nevoie de suma cifrelor. Șeful de tură îți cere două lucruri: mai întâi desenează schema logică — cu forma potrivită pentru fiecare rol — și abia apoi urmărește algoritmul pas cu pas pe o valoare concretă.

A

EXERCIȚIUL A · SCHEMĂ LOGICĂ

Desenează algoritmul

SCHEMĂ LOGICĂ

Mai jos ai pseudocodul care calculează suma cifrelor unui număr. Desenează în chenar schema logică, cu blocurile în ordinea execuției. **Atenție la formă:** ovalul e pentru început și sfârșit, paralelogramul pentru citire și afișare, dreptunghiul pentru calcule, rombul pentru decizii. Rombul are două ieșiri, marcate DA și NU.

```
1 START
2 citește n
3 s ← 0
4 cât timp n > 0 execută
5     s ← s + n % 10
6     n ← n // 10
7 afișează s
8 STOP
```

SCHEMA TA LOGICĂ

B

EXERCIȚIUL B · DRY-RUN

Urmărește algoritmul pas cu pas

DRY-RUN

Rulează algoritmul cu creionul, pe n = 4721. Completează tabelul: la fiecare rotație a buclei, care e cifra extrasă, cât devine suma și ce rămâne din număr. Primul rând e completat ca model, iar coloana n (la intrare) este dată.

```
n = 4721
s = 0
while n > 0:
    c = n % 10
    s = s + c
    n = n // 10
print(s)
```

n (la intrare)	c = n % 10	s (după)	n (după)
4721	1	1	472
472			
47			
4			

HINT *Recalculează: la fiecare pas c este ultima cifră (n % 10), s crește cu c, iar n pierde ultima cifră (n // 10).*

4 Camera de *teste*

Aici programele sunt puse la încercare. Un program care merge pe un singur exemplu nu e un program verificat — e un program norocos.

> **SCENA**
>
> Pe ecran stă un program care spune dacă un număr e perfect. A trecut toate probele colegilor și pare în regulă. Sarcina ta: alege datele de test relevante, apoi urmărește ce se întâmplă la o valoare la care nimeni nu s-a gândit.

A

EXERCIȚIUL A · CLASIFICARE

Alege datele de test

CLASIFICARE

Programul citește un număr natural n și spune dacă este **număr perfect** (egal cu suma divizorilor săi proprii, ca 6 = 1 + 2 + 3). Scrie în căsuța din fața fiecărei valori litera categoriei potrivite: **caz obișnuit**, **caz-limită** (la marginea domeniului) sau **caz invalid** (nu respectă enunțul).

- ☐ `n = 6` — cel mai mic număr perfect
- ☐ `n = -8` — număr negativ
- ☐ `n = 0` — cea mai mică valoare posibilă
- ☐ `n = 2.5` — număr cu virgulă
- ☐ `n = 1` — numărul fără divizori proprii
- ☐ `n = 12` — un număr oarecare, care nu e perfect

A Caz obișnuit | B Caz-limită | C Caz invalid

B

EXERCIȚIUL B · GRILĂ

Capcana lui zero

GRILĂ

Valoarea n = 0 ascunde o capcană. Urmărește programul de mai jos pentru n = 0 și încercuiește răspunsul corect la fiecare dintre cele patru întrebări.

```
1  n = int(input())
2  s = 0
3  d = 1
4  while d < n:
5      if n % d == 0:
6          s = s + d
7      d = d + 1
8  if s == n:
9      print("DA")
10 else:
11     print("NU")
```

1. Pentru `n = 0`, de câte ori se execută bucla `while d < n`?

`la nesfârșit` | `niciodată, fiindcă 1 < 0 este fals de la început` | `o singură dată` | `de zero ori, dar d tot crește la 1`

2. Așadar, ce valoare are `s` după buclă, pentru `n = 0`?

`niciuna, s nu există` | `depinde de d` | `0, valoarea cu care a fost inițializată` | `1`

3. Programul compară apoi `s == n`. Ce afișează pentru `n = 0`?

`nimic` | `NU — răspunsul corect` | `o eroare de execuție` | `DA — deși 0 nu este număr perfect`

4. Ce fel de eroare este aceasta și cum o reparăm?

`eroare logică; adăugăm condiția n > 0 înainte de comparare`

`eroare de sintaxă; punem două puncte după while`

`nu e nicio eroare, 0 chiar este număr perfect`

`eroare de execuție; verificăm ca utilizatorul să introducă un număr`

SECȚIUNEA 5 · L10–L11 · ALGORITMUL LUI EUCLID · CMMDC ȘI CMMMC · DESCOMPUNEREA ÎN FACTORI PRIMI

5 Atelierul de *mecanică*

Două utilaje, două calendare de întreținere, o singură întrebare: când cad în aceeași zi?

> **SCENA**
>
> Pe perete atârnă două fișe. Utilajul A se revizuiește la fiecare 12 zile, utilajul B la fiecare 18. Alături, o rețetă industrială cu 24 de kilograme de făină la 36 de litri de apă, care trebuie adusă la cea mai simplă formă. Ambele probleme se rezolvă cu același algoritm, vechi de peste două mii de ani. Iar ca să simplifici cu adevărat, trebuie să știi din ce factori e făcut fiecare număr.

A EXERCIȚIUL A · COMPLETARE

Fișa algoritmului lui Euclid

COMPLETARE

Fișa de pe perete s-a decolorat și câteva locuri au rămas goale. Completează golurile cu fragmentele potrivite din rezervă (cinci fragmente rămân nefolosite). Ai grijă: în rezervă sunt și variante care *par* corecte.

`x * y // cmmdc` `x * y * cmmdc` `r` `cmmdc // x * y` `a // b` `a % b` `a` `b % a` `a - b` `b`

Algoritmul lui Euclid cu împărțiri, pentru numerele naturale a și b. Cât timp b este diferit de 0: restul r ia valoarea _____, apoi a ia valoarea _____, iar b ia valoarea _____. Când bucla se oprește, CMMDC este valoarea din _____. Dacă am păstrat numerele inițiale în x și y, atunci cel mai mic multiplu comun se calculează cu _____.

B EXERCIȚIUL B · DRY-RUN

Rulează Euclid cu creionul

DRY-RUN

Rețeta cere raportul dintre 1071 și 462 adus la cea mai simplă formă. Urmărește algoritmul pas cu pas și completează tabelul. Primul rând e dat ca model, iar coloanele a și b (la intrare) sunt date.

```
a = 1071
b = 462
while b != 0:
    r = a % b
    a = b
    b = r
print(a)
```

a (la intrare)	b (la intrare)	r = a % b	a (după)	b (după)
1071	462	147	462	147
462	147			
147	21			

HINT *Recalculează: r este restul împărțirii lui a la b, apoi a preia vechiul b, iar b preia restul.*

SPRIJIN · SCRIE CODUL

Cea mai mică piesă

SCRIERE COD

Ca să aduci o rețetă la forma cea mai simplă, trebuie să știi din ce factori sunt făcute numerele. Înainte să descompui un număr în *toți* factorii lui primi, e destul să știi să-l găsești pe **primul**. Se citește un număr natural n (n ≥ 2). Afișează cel mai mic divizor al lui n mai mare decât 1. Ex.: pentru 12 se afișează 2; pentru 35 se afișează 5; pentru 13 se afișează 13, fiindcă 13 este prim, iar singurul lui divizor mai mare decât 1 este el însuși.

HINT **Observă.** *Acest divizor este* **întotdeauna un număr prim** *. Dacă ar avea el însuși un divizor mai mare decât 1 și mai mic decât el, acela ar fi și divizor al lui n — deci l-ai fi găsit tu mai devreme.*

HINT *Nu ai nevoie să treci prin toți divizorii: e destul să cauți cât timp d * d nu depășește n. Dacă bucla se termină fără să fi găsit ceva, numărul e prim.*

SOLUȚIA TA · PYTHON

SECȚIUNEA 6 · L12 · BAZE DE NUMERAȚIE · CODUL ASCII

6 Camera *serverelor*

Ventilatoarele huruie. Aici numerele își schimbă haina: din baza 10 în baza 2 și invers, iar literele redevin numere.

> **SCENA**
>
> Pe monitorul din mijloc clipește o adresă IP: 192.168.0.1. Ruterul n-o vede așa — el vede numai biți. Iar pe al doilea ecran curge un jurnal în care numele fișierelor apar ca șiruri de numere. Ca să ieși din camera asta trebuie să faci ambele traduceri.

A

EXERCIȚIUL A · ORDONARE

Traducătorul de adrese

ORDONARE

Ruterul cere octetul 172 scris în baza 2. Citește etapele algoritmului, apoi numerotează instrucțiunile Python de la 1 la 9, în ordinea corectă. Reține: fiecare rest se adaugă **în stânga** celor dinainte, de aceea puterea lui 10 crește la fiecare pas.

Etapele algoritmului, în cuvinte:

1. Citește numărul, scris în baza 10.
2. Pornește rezultatul de la 0, iar poziția de scriere de la unități.
3. Cât timp numărul este mai mare decât 0, repetă pașii următori.
4. Află restul împărțirii numărului la 2 — acesta este următoarea cifră binară.
5. Adaugă cifra la rezultat, pe poziția de scriere curentă.
6. Mută poziția de scriere cu un rang mai la stânga.
7. Împarte numărul la 2, cu împărțire întreagă, și reia verificarea condiției.
8. Când numărul a ajuns la 0, afișează rezultatul.

```
p = 1
```

```
while n > 0:
```

```
    b = b + c * p
```

```
print(b)
```

```
b = 0
```

```
    n = n // 2
```

```
    c = n % 2
```

```
    p = p * 10
```

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

B

EXERCIȚIUL B · GRILĂ

Jurnalul în numere

GRILĂ

Cinci întrebări despre traducerea dintre litere și numere — și una despre adresa IP de pe primul monitor. Încercuiește răspunsul corect la fiecare întrebare.

1. Ce valoare are `ord("A")`? 97 · 65 · 48 · 1

2. Ce afișează `print(chr(74))`? J · G · 74 · j

3. Litera `j` are codul 106, iar `J` are 74. Ce diferență este mereu între o literă mică și corespondenta ei mare?

   depinde de literă · 65 · 32 · 26

4. Cum aflăm poziția literei `J` în alfabet, folosind coduri?

   `ord("J") // 10` · `ord("J") - 1` · `ord("J") - ord("A") + 1` · `chr("J") - 65`

5. La conversia din baza 10 în baza 2 prin împărțiri repetate, de ce se citesc resturile de jos în sus?

   - fiindcă baza 2 se scrie invers față de baza 10
   - fiindcă împărțirea întreagă inversează ordinea cifrelor
   - fiindcă primul rest obținut este cifra cea mai puțin semnificativă, deci ultima din scriere
   - e doar o convenție, ordinea nu contează

6. Octetul `172` scris în baza 2 este: `10110100` `11010100` `11001010` `10101100`

SECȚIUNEA 7 · COD GHIDAT · INTEGRARE

7 Seiful

Ultima ușă dinaintea ieșirii. Aici **scrii cod**: trei probleme obligatorii, una de remediere și două provocări. Scrie soluția pe hârtie și urmărește-o mental pe exemplul dat.

PROBLEMA 7.1 · COD GHIDAT

1 *Cifra de control a codului de bare*

SCRIERE COD

Fiecare cod de bare se termină cu o **cifră de control**, calculată din celelalte. Se citește un cod de bare de 12 cifre, fără cifra de control. Parcurge cifrele **de la dreapta la stânga**: prima (cea mai din dreapta) se înmulțește cu 3, următoarea cu 1, apoi iar cu 3, alternativ. Aduni toate produsele și obții suma s. Cifra de control este (10 - s % 10) % 10. Afișeaz-o. Ex.: pentru 590123412345 se afișează 7 — chiar cifra cu care se termină codul real 5901234123457.

SOLUȚIA TA · PYTHON

2

PROBLEMA 7.2 · COD GHIDAT

Când cad reviziile în aceeași zi

SCRIERE COD

Cele două utilaje din atelier se revizuiesc la intervale diferite. Se citesc două numere naturale x și y (x, y ≥ 1), pe câte o linie: intervalele de revizie, în zile. Afișează peste câte zile de la pornire cele două revizii cad prima dată în aceeași zi. Ex.: pentru x = 12 și y = 18 se afișează 36.

HINT *Folosește algoritmul lui Euclid pentru CMMDC, apoi formula cu care ai completat fișa din atelier.*

SOLUȚIA TA · PYTHON

3

PROBLEMA 7.3 · COD GHIDAT

Octet valid pentru rutere

SCRIERE COD

Ruterul acceptă numai octeți: numere întregi între 0 și 255. Se citește un număr întreg n. Dacă n este un octet valid, afișează reprezentarea lui în baza 2; altfel afișează INVALID. Ex.: pentru 172 se afișează 10101100; pentru 256 se afișează INVALID; pentru 0 se afișează 0.

SOLUȚIA TA · PYTHON

PROBLEMA 7.4 · REPARĂ CODUL

4 Numărul perfect care mințea

SCRIERE COD

Iată programul pe care l-ai demascat în camera de teste. Pentru n = 0 afișează DA, deși 0 nu este număr perfect. Repară-l, păstrând restul comportamentului: pentru 6 și 28 trebuie să afișeze DA, pentru 0, 1 și 12 trebuie să afișeze NU.

Programul stricat:

```
n = int(input())
s = 0
d = 1
while d < n:
    if n % d == 0:
        s = s + d
    d = d + 1
if s == n:
    print("DA")
else:
    print("NU")
```

PROGRAMUL CORECTAT · PYTHON

PROBLEMA 7.5 · PROVOCARE

Rădăcina digitală

SCRIERE COD

Se citește un număr natural n (n ≥ 0). Adună cifrele lui; dacă rezultatul are mai mult de o cifră, adună din nou cifrele rezultatului, și tot așa până rămâne o singură cifră. Afișeaz-o. Ex.: pentru 9875 → 9 + 8 + 7 + 5 = 29 → 2 + 9 = 11 → 1 + 1 = 2, deci se afișează 2.

HINT *Ai nevoie de o buclă în interiorul altei bucle.*

SOLUȚIA TA · PYTHON

PROBLEMA 7.6 · PROVOCARE

Din ce e făcut numărul

SCRIERE COD

În secțiunea 5 ai găsit prima piesă. Acum mergi până la capăt. Se citește un număr natural n (n ≥ 2). Afișează descompunerea lui în factori primi: **câte un factor pe linie**, în forma „factorul, apoi exponentul”, în ordinea crescătoare a factorilor. Ex.: pentru 360 se afișează trei linii — 2 3, apoi 3 2, apoi 5 1 — fiindcă $360 = 2^3 \cdot 3^2 \cdot 5$.

HINT *Ai nevoie de o buclă în interiorul altei bucle: cea din afară caută următorul factor, cea din interior numără de câte ori se împarte exact. Nu uita de factorul care poate rămâne la final, mai mare decât rădăcina lui n.*

SOLUȚIA TA · PYTHON