Lecții suport pentru început de an școlar – ghid pentru elevi –

9

Informatică

curriculum de specialitate (CS), filiera teoretică, profilul real, specializarea științe ale naturii

clasa a IX-a

SIGMA

Capitolul 1 – Fundamentele programării

(1.1.; 1.3.; 1.5.; 2.1.; 2.3.; 2.5.; 3.1.; 3.3.; 3.5.; 4.1.; 4.3.; 4.5.; 5.1.; 5.3.; 5.5.; 6.1.; 6.5.)

Fișă de lucru pentru elevi

Capitolul 1 – Fundamentele programării // Lecția 1 – Etapele elaborării unui program

Ce vei învăța azi

La finalul lecției, voi ști:

- să definesc noțiunile de informatică, dată și algoritm;
- să clasific datele după tip, moment de utilizare și posibilitatea de modificare;
- să enumăr și să explic cele 5 etape de elaborare a unui program;
- să diferențiez eroarea de sintaxă, de logică și de execuție;
- să aplic etapele de elaborare pentru o problemă simplă, cu teste corespunzătoare.

Să pornim de la o întrebare

Întrebare: De ce un program care rulează fără să se oprească nu este întotdeauna un program corect?

Un magazin online folosește un program care ar trebui să calculeze automat totalul unei comenzi. Pentru o comandă cu trei produse, în valoare de 20, 35 și respectiv 45 de lei, programul afișează 80 de lei în loc de 100. Programul nu se oprește cu eroare, dar rezultatul este vizibil greșit.

Gândește-te: notează, în două-trei rânduri, o ipoteză despre cauza acestei situații și fii pregătit s-o discuți cu profesorul.

Informatica, datele și algoritmul

Informatica studiază modul în care informația poate fi reprezentată, prelucrată și transmisă cu ajutorul sistemelor de calcul. Rezolvarea informatică a unei probleme pornește întotdeauna de la niște date, pe care le prelucrează un algoritm ce poate fi ulterior implementat printr-un program.

Reține! Un ***algoritm*** este o succesiune finită și ordonată de pași, executați într-o ordine bine stabilită, care transformă datele de intrare în rezultatul dorit.

Criteriu de clasificare	Categorii
după tip	numerice, logice, șiruri de caractere, structurate
după momentul utilizării	de intrare, de lucru/intermediare, de ieșire
după posibilitatea modificării	variabile și constante

⚠ **Important!** Pentru a fi corect, un algoritm trebuie să fie finit, clar, executabil și să producă un rezultat pentru orice date de intrare acceptate.

Etapele elaborării unui program

Pentru a realiza un program, care implementează un algoritm este necesară parcurgerea etapelor:

Etapă	Ce presupune
Analiza problemei	identificarea datelor de intrare, de ieșire și a relațiilor dintre ele
Proiectarea soluției	alegerea metodei de rezolvare și schițarea pașilor în pseudocod
Implementarea	transpunerea algoritmului într-un limbaj de programare, cu respectarea sintaxei
Testarea și depanarea	verificarea programului cu date variate și corectarea erorilor găsite
Analiza și optimizarea	evaluarea eficienței, din punctul de vedere al timpului și al memoriei folosite

Testarea și tipurile de erori

Un test are trei componente: datele de intrare, rezultatul așteptat (stabilit înainte de rulare) și rezultatul obținut efectiv. Un set de teste bine construit include date obișnuite, cazuri-limită ale domeniului valid și date invalide, pe care programul ar trebui să le respingă.

Tip de eroare	Cum se manifestă
eroare de sintaxă	codul nu respectă regulile limbajului și nu poate fi interpretat
eroare de logică	programul rulează, dar produce rezultate greșite
eroare de execuție	programul se oprește brusc în timpul rulării (de exemplu, la o împărțire la zero)

⚠ **Important!** Testarea și depanarea nu sunt etape opționale — un program trebuie verificat cu date variate, nu doar cu exemplul cu care a fost gândit inițial.

Exemplu rezolvat

Cerință: Se citesc prețurile a ***n*** produse cumpărate dintr-un magazin. Se cere să se calculeze suma totală cheltuită și să se afișeze dacă suma se încadrează într-un buget de 100 de lei sau îl depășește.

Cum gândim problema

Categorie	Conținut
Date de intrare	***n***, număr natural nenul; n prețuri reale pozitive
Date de ieșire	suma totală și mesajul „ÎN BUGET” sau „PESTE BUGET”
Date de lucru	prețul curent citit la fiecare pas
Relații	suma = suma prețurilor citite; în buget dacă suma ≤ 100
Restricții	n ≥ 1 și fiecare preț este un număr pozitiv

Pașii algoritmului

1. Citește numărul ***n*** de produse.
2. Inițializează suma cu 0.
3. Repetă de ***n*** ori: citește un preț și adaugă-l la sumă.
4. Compară suma cu bugetul de 100 de lei.
5. Afișează suma și mesajul corespunzător.

Pseudocod

```
citește n
suma ← 0
pentru i ← 1, n execută
    citește preț
    suma ← suma + preț
scrie suma
dacă suma ≤ 100 atunci
    scrie „ÎN BUGET”
altfel
    scrie „PESTE BUGET”
```

Verificăm cu câteva exemple

Nr.	Date de intrare	Tip	Rezultat așteptat
1	n = 3; 20, 30, 40	general	suma 90; ÎN BUGET
2	n = 2; 50, 50	limită	suma 100; ÎN BUGET
3	n = 1; 150	special	suma 150; PESTE BUGET
4	n = 4; 10, 10, 10, 10	special	suma 40; ÎN BUGET
5	n = 0	invalid	Număr invalid de produse
6	n = 2; -5, 20	invalid	Mesaj privind un preț negativ

Activitate de rezolvat împreună cu profesorul

Pentru prețurile 15, 60 și 10 lei, completați un tabel cu valorile variabilelor ***i***, ***preț*** și ***sumă*** după fiecare repetare. Calculați suma finală și stabiliți ce mesaj va fi afișat.

Exerciții propuse

Rezolvă exercițiile de mai jos în caiet sau într-un fișier separat.

1. Ordonați etapele: implementare, analiză, testare, proiectare, depanare, optimizare. Explicați de ce proiectarea trebuie să preceadă implementarea.

2. Propuneți câte un caz general, un caz limită și un caz invalid pentru o problemă care citește vârsta unui elev, dintr-un interval permis de la 6 la 19 ani.

3. Modificați algoritmul bugetului de cumpărături astfel încât să afișeze și numărul de produse mai scumpe de 30 de lei.

Temă pentru acasă

Alegeți o problemă simplă din viața cotidiană (de exemplu, planificarea unei excursii sau calculul consumului lunar de energie), parcurgeți cele 5 etape de elaborare și propuneți trei teste: general, limită și invalid.

Autoevaluare!

Afirmație	Da	Nu	Nu sunt sigur(ă)
Știu să definesc noțiunile de informatică, dată și algoritm.	☐	☐	☐
Știu să clasific datele după tip, moment de utilizare și posibilitatea de modificare.	☐	☐	☐
Știu să enumăr și să explic cele 5 etape de elaborare a unui program.	☐	☐	☐
Știu să diferențiez eroarea de sintaxă, de logică și de execuție.	☐	☐	☐
Știu să aplic etapele de elaborare pentru o problemă simplă, cu teste corespunzătoare.	☐	☐	☐

Dacă ai bifat „Nu” sau „Nu sunt sigur(ă)” la vreo afirmație, recitește secțiunea corespunzătoare sau întreabă profesorul.

Ce reținem

- Pentru a fi corect, un algoritm trebuie să fie finit, clar, executabil și să producă un rezultat pentru orice date de intrare acceptate.
- Testarea și depanarea nu sunt etape opționale — un program trebuie verificat cu date variate, nu doar cu exemplul cu care a fost gândit inițial.

Fișă de lucru pentru elevi

Capitolul 1 – Fundamentele programării // Lecția 2 – Pseudocod și limbaje de programare

Ce vei învăța azi

La finalul lecției, voi ști:

- să enumăr cele trei moduri de reprezentare a unui algoritm;
- să descriu structurile liniară, alternativă și repetitivă în pseudocod;
- să transcriu corect un fragment de pseudocod în Python;
- să diferențiez un limbaj interpretat de unul compilat.

Să pornim de la o întrebare

Întrebare: De ce este util să scriem întâi algoritmul în pseudocod, înainte de a-l implementa într-un limbaj de programare?

Un elev scrie direct codul Python pentru o problemă, fără să se gândească întâi la pași. Programul rulează, dar produce un rezultat greșit, iar elevul are dificultăți în a găsi cauza, pentru că a amestecat gândirea despre problemă cu sintaxa limbajului.

Gândește-te: notează, în două-trei rânduri, o ipoteză despre cauza acestei situații și fii pregătit s-o discuți cu profesorul.

Moduri de reprezentare a unui algoritm

Există mai multe moduri de a descrie un algoritm, fiecare având avantaje și limite specifice.

Mod de reprezentare	Caracteristici	Utilizare
schema logică	descriere grafică, prin blocuri și săgeți	vizualizarea rapidă a pașilor algoritmului
pseudocod	descriere textuală, apropiată de limbajul natural	proiectarea algoritmului înainte de scrierea codului
limbaj de programare	instrucțiuni cu sintaxă strictă, executate de calculator	implementarea efectivă a algoritmului

Structuri de control în pseudocod

Indiferent de limbajul în care va fi ulterior implementat, un algoritm este construit din trei tipuri de structuri de control.

Structură	Formă în pseudocod
liniară	citește date instrucțiuni afișare date - executate în ordine

alternativă	dacă <condiție> atunci
	  instrucțiuni 1
	altfel
	  instrucțiuni 2
	sfârșit dacă

repetitivă	cât timp <condiție> execută
	  instrucțiuni
	sfârșit cât timp
	sau
	pentru i ← val_inițială, val_finală, pas_execuție execută
	  instrucțiuni
	sfârșit pentru

⚠ **Important!** Pseudocodul nu este executat direct de calculator, dar permite descrierea clară a pașilor unui algoritm, independent de limbajul de programare ales ulterior.

Limbaje interpretate și limbaje compilate

Limbajul de programare reprezintă mijlocul prin care un algoritm este comunicat calculatorului, sub o formă pe care acesta o poate executa.

Tip	Caracteristici
limbaj cu interpretor	instrucțiunile sunt executate pe rând; Python este un limbaj interpretat
limbaj cu compilator	programul este transformat, înainte de execuție, într-o formă executabilă

Exemplu rezolvat

Cerință: Un produs are atașat un cod format din mai multe cifre. Se cere să se determine produsul cifrelor codului.

Cum gândim problema

Categorie	Conținut
Date de intrare	***cod,*** număr natural
Date de ieșire	produsul cifrelor codului
Date de lucru	cifra extrasă la fiecare pas
Relații	produs = produsul tuturor cifrelor lui cod
Restricții	cod este un număr natural nenul

Pașii algoritmului

1. Citește codul.
2. Inițializează produsul cu 1.
3. Cât timp codul mai are cifre: extrage ultima cifră și înmulțește produsul cu ea, apoi elimină cifra din cod.
4. Afișează produsul obținut.

Pseudocod

```
citește cod
produs ← 1
cât timp cod > 0 execută
   cifra ← cod % 10
   produs ← produs * cifra
   cod ← cod // 10
sfârșit cât timp
scrie produs
```

Codul în Python

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

Verificăm cu câteva exemple

Nr.	Date de intrare	Tip	Rezultat așteptat
1	cod = 1234	general	produs 24
2	cod = 5	limită (o cifră)	produs 5
3	cod = 1000	special (cifre zero)	produs 0
4	cod = 999	special (cifre egale)	produs 729
5	cod = 0	special	bucla nu se execută; produsul rămâne 1 — caz de atenție
6	cod = -25	invalid	codul este negativ; algoritmul nu funcționează corect

Activitate de rezolvat împreună cu profesorul

Pentru codul 306, completați un tabel cu valorile variabilelor ***cod***, ***cifră*** și ***produs*** după fiecare repetare. Observați ce se întâmplă când una dintre cifre este 0.

Exerciții propuse

Rezolvă exercițiile de mai jos în caiet sau într-un fișier separat.

1. Scrieți în pseudocod un algoritm care citește două numere și afișează minimul lor.

2. Descrieți, în cuvinte, pașii unei scheme logice pentru: dacă un număr ***y*** este par, se afișează „par”, altfel se afișează „impar”.

3. Transcrieți în Python algoritmul de calcul al produsului primelor ***n*** numere naturale, completând spațiile libere: n = int(input()); produs = ___ ; for i in range(1, ___): produs *= ___ ; print(___).

Temă pentru acasă

Alegeți o problemă simplă (de exemplu, verificarea dacă un număr este pozitiv, negativ sau nul) și scrieți-o atât în pseudocod, cât și în Python, precizând tipul fiecărei structuri de control folosite.

Autoevaluare!

Afirmație	Da	Nu	Nu sunt sigur(ă)
Știu să enumăr cele trei moduri de reprezentare a unui algoritm.	☐	☐	☐
Știu să descriu structurile liniară, alternativă și repetitivă în pseudocod.	☐	☐	☐
Știu să transcriu corect un fragment de pseudocod în Python.	☐	☐	☐
Știu să diferențiez un limbaj interpretat de unul compilat.	☐	☐	☐

Dacă ai bifat „Nu” sau „Nu sunt sigur(ă)” la vreo afirmație, recitește secțiunea corespunzătoare sau întreabă profesorul.

Ce reținem

- Pseudocodul nu este executat direct de calculator, dar permite descrierea clară a pașilor unui algoritm, independent de limbajul de programare ales ulterior.

Fișă de lucru pentru elevi

Capitolul 1 – Fundamentele programării // Lecția 3 – Variabile, tipuri de date și structuri de control

Ce vei învăța azi

La finalul lecției, voi ști:

- să declar și să utilizez variabile în Python;
- să identific tipurile de date int, float, str, list și bool;
- să folosesc operatori aritmetici, relaționali și logici în expresii;
- să scriu structuri alternative și repetitive corect indentate.

Să pornim de la o întrebare

Întrebare: Cum știe Python ce tip de date are o variabilă, fără ca noi să declarăm acest lucru explicit?

Un elev scrie x = input("Vârsta: ") și apoi încearcă x + 5, dar primește o eroare de tip. Colegul lui, care a folosit int(input(...)), nu are aceeași problemă.

Gândește-te: notează, în două-trei rânduri, o ipoteză despre cauza acestei situații și fii pregătit s-o discuți cu profesorul.

Structura unui program Python

Python este un limbaj interpretat, ale cărui instrucțiuni sunt executate pe rând. Cel mai simplu program afișează un mesaj cu ajutorul funcției print().

```
print("Bun venit la ora de informatică!")
```

Vocabularul limbajului cuprinde: identificatori (pentru variabile, funcții), cuvinte cheie cu înțeles special (if, else, for, while, return), separatori (spațiul, indentarea, parantezele) și comentarii, introduse cu #.

⚠ **Important!** În Python, indentarea nu este doar o convenție estetică — ea delimitează blocurile de instrucțiuni și face parte din sintaxa limbajului.

Variabile: citire și afișare

O variabilă se creează atunci când i se atribuie o valoare printr-un identificator ales de programator. Tipul variabilei este stabilit automat, în funcție de valoarea atribuită.

```
nume = input("Nume: ")
oras = input("Localitate: ")
print(f"Salut, {nume}, din {oras}!")
```

Tipuri de date în Python

Tip de date	Exemplu	Observații
int	scor = 95	număr întreg
float	pret = 12.5	număr real
str	oras = "Cluj"	șir de caractere
list	produse = ["pâine", "lapte"]	colecție ordonată, indexată de la 0
bool	disponibil = True	valoare logică: True sau False

⚠ **Important!** Datele citite cu input() sunt salvate implicit ca șir de caractere; pentru calcule folosim conversii precum int() și float().

Operatori

Categorie	Operatori
aritmetici	+, -, *, **, /, //, %
relaționali	>, >=, <, <=, ==, !=
logici	not, and, or
de atribuire	=, +=, -=, *=, /=, //=, %=

⚠ **Important!** Operatorii % și // sunt utili la prelucrarea cifrelor unui număr: n % 10 extrage ultima cifră, iar n // 10 elimină ultima cifră.

Expresia este o succesiune alcătuită din operanzi, legați prin operatori. Exemplu: 18 + 5 * 3.

Structuri de control în Python

Structura ***alternativă***:

```
temp1 = float(input("Temperatura dimineața: "))
temp2 = float(input("Temperatura seara: "))
if temp1 > temp2:
    print("Dimineața a fost mai cald")
else:
    print("Seara a fost mai cald sau egal")
```

Structura ***repetitivă cu număr necunoscut de pași***:

```
cod = int(input("cod = "))
cnt = 0
while cod > 0:
    cnt += 1
    cod //= 10
print(cnt)
```

Structura ***repetitivă cu număr cunoscut de pași***:

```
n = int(input("n = "))
produs = 1
for i in range(1, n + 1):
    produs *= i
print(produs)
```

⚠ **Important!** Ca și în cazul structurii alternative, indentarea corectă a instrucțiunilor din interiorul unei structuri repetitive este esențială.

Exemplu rezolvat

Cerință: Se citesc două temperaturi înregistrate în aceeași zi, dimineața și seara. Se cere să se afișeze care valoare este mai mare și care este diferența dintre ele.

Cum gândim problema

Categorie	Conținut
Date de intrare	***temp1, temp2 — două numere reale***
Date de ieșire	mesaj despre valoarea mai mare și diferența dintre ele
Date de lucru	diferența calculată
Relații	diferenta = |temp1 - temp2|
Restricții	temp1 și temp2 sunt numere reale, fără altă limitare

Pașii algoritmului

1. Citește cele două temperaturi.
2. Calculează diferența, în valoare absolută.
3. Compară temp1 cu temp2.
4. Afișează mesajul corespunzător și diferența.

Pseudocod

```
citește temp1, temp2
diferenta ← |temp1 - temp2|
dacă temp1 > temp2 atunci
   scrie „Dimineața a fost mai cald”, diferenta
altfel
   dacă temp1 < temp2 atunci
      scrie „Seara a fost mai cald”, diferenta
   altfel
      scrie „Temperaturi egale”
  sfârșit dacă
sfârșit dacă
```

Codul în Python

```
temp1 = float(input("Temperatura dimineața: "))
temp2 = float(input("Temperatura seara: "))
diferenta = abs(temp1 - temp2)
if temp1 > temp2:
    print("Dimineața a fost mai cald, diferență:",
diferenta)
elif temp1 < temp2:
    print("Seara a fost mai cald, diferență:", dife-
renta)
else:
   print("Temperaturi egale")
```

Verificăm cu câteva exemple

Nr.	Date de intrare	Tip	Rezultat așteptat
1	temp1 = 12, temp2 = 18	general	„Seara a fost mai cald”, diferență 6.0
2	temp1 = 0, temp2 = 0	limită	„Temperaturi egale”
3	temp1 = -5, temp2 = 3	special	„Seara a fost mai cald”, diferență 8.0
4	temp1 = 25.5, temp2 = 25.5	special	„Temperaturi egale”
5	temp1 = "abc", temp2 = 20	invalid	eroare la conversia cu float()

Activitate de rezolvat împreună cu profesorul

Activitate: Urmăriți, pas cu pas, o structură for care calculează produsul primelor 4 numere naturale. Completați un tabel cu valorile lui i și produs după fiecare pas.

Exerciții propuse

Rezolvă exercițiile de mai jos în caiet sau într-un fișier separat.

1. Ce va afișa codul: x = "7"; y = 3; print(int(x) * y)?
2. Citiți de la tastatură cele două catete ale unui triunghi dreptunghic și afișați aria acestuia.
3. Scrieți un program care citește un număr întreg și afișează dacă este par sau impar.

Temă pentru acasă

Scrieți un program „Calculator de reducere” care citește prețul unui produs și procentul de reducere, apoi afișează prețul final.

Autoevaluare!

Afirmație	Da	Nu	Nu sunt sigur(ă)
Știu să declar și să utilizez variabile în Python.	☐	☐	☐
Știu să identific tipurile de date int, float, str, list și bool.	☐	☐	☐
Știu să folosesc operatori aritmetici, relaționali și logici în expresii.	☐	☐	☐
Știu să scriu structuri alternative și repetitive corect indentate.	☐	☐	☐

Dacă ai bifat „Nu” sau „Nu sunt sigur(ă)” la vreo afirmație, recitește secțiunea corespunzătoare sau întreabă profesorul.

Ce reținem

- În Python, indentarea nu este doar o convenție estetică — ea delimitează blocurile de instrucțiuni și face parte din sintaxa limbajului.
- Datele citite cu input() sunt salvate implicit ca șir de caractere; pentru calcule folosim conversii precum int() și float().
- Operatorii % și // sunt utili la prelucrarea cifrelor unui număr: n % 10 extrage ultima cifră, iar n // 10 elimină ultima cifră.
- Ca și în cazul structurii alternative, indentarea corectă a instrucțiunilor din interiorul unei structuri repetitive este esențială.

Fișă de lucru pentru elevi

Capitolul 1 – Fundamentele programării //
Lecția 4 – Subprograme: definiție, parametri, variabile locale/globale

Ce vei învăța azi

La finalul lecției, voi ști:

- să definesc un subprogram (funcție) cu parametri și valoare returnată;
- să diferențiez parametrul de argument;
- să folosesc valori implicite ale parametrilor;
- să identific variabilele locale și globale dintr-un program.

Să pornim de la o întrebare

Întrebare: De ce este mai avantajos să scriem un calcul o singură dată, într-un subprogram, decât să îl repetăm de fiecare dată când avem nevoie de el?

Un elev scrie un program în care calculul prețului final, cu adaos și TVA, apare copiat de trei ori, în trei locuri diferite. Când profesorul cere modificarea cotei de TVA, elevul va trebui să găsească și să corecteze fiecare copie.

Gândește-te: notează, în două-trei rânduri, o ipoteză despre cauza acestei situații și fii pregătit s-o discuți cu profesorul.

Ce este un subprogram?

Rolul subprogramelor este de a permite organizarea modulară a codului: o problemă complexă este descompusă în subprograme mai mici, fiecare responsabil de o parte a rezolvării.

Reține! Un ***subprogram*** (numit funcție în Python) este un bloc de instrucțiuni identificat printr-un nume, care rezolvă o sarcină specifică și poate fi reutilizat de câte ori este nevoie.

Avantaj	Explicație
Reutilizarea codului	un subprogram se scrie o singură dată și se apelează de oricâte ori este nevoie
Lizibilitate	programul devine mai ușor de citit și de înțeles
Testare și depanare	fiecare subprogram poate fi testat independent de restul programului
Colaborare	membrii unei echipe pot lucra independent la subprograme diferite

Structura unui subprogram în Python

Componentă	Descriere
Antetul (header)	prima linie; conține cuvântul cheie def, numele funcției și lista parametrilor
Corpul (body)	liniile indentate sub antet; conțin instrucțiunile executate la apel
return	instrucțiune opțională care trimite un rezultat înapoi; fără return, funcția returnează None

Parametri, argumente și valori implicite

Parametrii sunt variabilele prin care o funcție primește date din exterior; iar la apel, valorile ce sunt transmise se numesc argumente.

```
def maxim(a, b):
    if a > b:
        return a
    else:
        return b
x = maxim(7, 4)
y = maxim(2, 9)
print(x, y)
```

O ***funcție*** poate avea parametri cu valoare implicită, folosiți atunci când nu se transmite un argument pentru ei la apel.

```
def calculeaza_pret_final(pret, reducere=0, tva=19):
    pret_redus = pret * (1 - reducere / 100)
    return round(pret_redus * (1 + tva / 100), 2)
print(calculeaza_pret_final(100))
print(calculeaza_pret_final(100, 10))
print(calculeaza_pret_final(100, 10, 9))
```

⚠ **Important!** Instrucțiunea return încheie imediat execuția funcției și trimite rezultatul înapoi, în locul din program de unde a fost apelat subprogramul.

Mecanismul de execuție și variabilele locale/globale

Pas	Ce se întâmplă
1	se oprește execuția din programul principal la întâlnirea apelului
2	se creează un spațiu de memorie nou pentru subprogram
3	valorile argumentelor se transmit parametrilor funcției
4	se execută corpul funcției și se returnează rezultatul
5	execuția programului principal continuă de unde s-a oprit

Variabile locale și variabile globale

```
rata_schimb = 4.97  # variabilă globală
def converteste(suma):
    rezultat = suma / rata_schimb  # rezultat este variabilă locală
    return round(rezultat, 2)
print(converteste(250))
```

Tip	Caracteristici
locale	definite în interiorul unei funcții; există doar pe durata execuției acesteia; parametrii sunt variabile locale
globale	definite în afara oricărei funcții; pot fi citite din interiorul funcțiilor, dar nu modificate fără cuvântul cheie global

⚠ **Important!** Folosirea excesivă a variabilelor globale poate îngreuna depanarea programului; de aceea este preferabil ca funcțiile să primească date prin parametri și să returneze rezultate.

Exemplu rezolvat

Cerință: Se cere o funcție care calculează prețul final al unui produs, pornind de la prețul de bază, un procent de reducere (opțional) și cota de TVA (opțională, implicit 19%).

Cum gândim problema

Categorie	Conținut
Date de intrare	pret — număr real; reducere, tva — numere reale, cu valori implicite
Date de ieșire	prețul final, rotunjit la 2 zecimale
Date de lucru	prețul după aplicarea reducerii
Relații	pret_final = pret × (1 − reducere/100) × (1 + tva/100)
Restricții	pret > 0; reducere și tva sunt procente între 0 și 100

Proiectarea algoritmului

1. Definește funcția cu parametrii pret, reducere (implicit 0) și tva (implicit 19).
2. Calculează prețul după reducere.
3. Aplică TVA asupra prețului redus.
4. Rotunjește rezultatul la 2 zecimale și returnează-l.

Pseudocod

```
subprogram calculeaza_pret_final(pret, reducere=0, tva=19)
    pret_redus ← pret * (1 - reducere/100)
    rezultat ← pret_redus * (1 + tva/100)
    returnează rotunjire(rezultat, 2)
sfârșit subprogram
```

Codul în Python

```
def calculeaza_pret_final(pret, reducere=0, tva=19):
    pret_redus = pret * (1 - reducere / 100)
    rezultat = pret_redus * (1 + tva / 100)
    return round(rezultat, 2)

print(calculeaza_pret_final(100))
print(calculeaza_pret_final(100, 10))
print(calculeaza_pret_final(200, 15, 9))
```

Verificăm cu câteva exemple

Nr.	Date de intrare	Tip	Rezultat așteptat
1	pret=100, reducere=0 (implicit), tva=19 (implicit)	general	119.0
2	pret=100, reducere=10	general	106.71
3	pret=200, reducere=15, tva=9	special	185.3
4	pret=100, reducere=100	limită	0.0
5	pret=0	invalid/special	0.0 — de discutat dacă este o valoare acceptabilă

Activitate de rezolvat împreună cu profesorul

Precizați, pentru secvența de la exemplul rezolvat, care variabile sunt parametri, care sunt locale funcției și dacă există variabile globale folosite.

Exerciții propuse

Rezolvă exercițiile de mai jos în caiet sau într-un fișier separat.

1. Scrieți o funcție cel_mai_mic(a, b, c) care returnează cel mai mic dintre trei numere.
2. La apelul functie(2, 9), valorile 2 și 9 reprezintă: a) parametri formali; b) variabile globale; c) argumente; d) constante.
3. Precizați care variabile sunt locale și care sunt globale în secvența: rata_tva = 19 \n def pret_cu_tva(pret): \n total = pret * (1 + rata_tva/100) \n return total.

Temă pentru acasă

Scrieți o funcție converteste_valuta(suma, curs=4.97) care returnează echivalentul în euro al unei sume în lei. Apelați-o o dată fără al doilea argument și o dată cu un curs diferit.

Autoevaluare!

Afirmație	Da	Nu	Nu sunt sigur(ă)
Știu să definesc un subprogram (funcție) cu parametri și valoare returnată.	☐	☐	☐
Știu să diferențiez parametrul de argument.	☐	☐	☐
Știu să folosesc valori implicite ale parametrilor.	☐	☐	☐
Știu să identific variabilele locale și globale dintr-un program.	☐	☐	☐

Dacă ai bifat „Nu” sau „Nu sunt sigur(ă)” la vreo afirmație, recitește secțiunea corespunzătoare sau întreabă profesorul.

Ce reținem

- Instrucțiunea return încheie imediat execuția funcției și trimite rezultatul înapoi, în locul din program de unde a fost apelat subprogramul.
- Folosirea excesivă a variabilelor globale poate îngreuna depanarea programului; de aceea este preferabil ca funcțiile să primească date prin parametri și să returneze rezultate.

Fișă de lucru pentru elevi

Capitolul 1 – Fundamentele programării //
Lecția 5 – Subprograme predefinite pentru calcule și colecții

Ce vei învăța azi

La finalul lecției, voi ști:

- să folosesc funcțiile predefinite abs(), round(), int(), float(), str();
- să import și să utilizez modulul math;
- să aplic len(), min(), max(), sum() pe o listă de valori;
- să interpretez corect rezultatul funcției round() pentru valori de tip .5.

Să pornim de la o întrebare

Întrebare: De ce round(2.5) și round(3.5) nu se rotunjesc după aceeași regulă la care ne-am aștepta din matematica de gimnaziu?

Un elev calculează media a trei note întregi cu (a + b + c) / 3 și observă că, în anumite cazuri, rezultatul rotunjit cu round() nu corespunde regulii „de la 5 în sus se rotunjește în sus", pe care o știa din clasele anterioare.

Gândește-te: notează, în două-trei rânduri, o ipoteză despre cauza acestei situații și fii pregătit s-o discuți cu profesorul.

Funcții predefinite pentru calcule matematice

Python include o serie de funcții încorporate, care pot fi apelate direct pentru a efectua calcule matematice rapide sau pentru a converti tipurile de date între ele.

Funcție	Rol	Exemplu
abs(x)	returnează valoarea absolută	abs(-18) → 18
math.sqrt(x)	returnează radicalul, întotdeauna de tip float	math.sqrt(9) → 3.0
round(x, n)	rotunjește la n zecimale; fără n, rezultatul este întreg	round(12.7) → 13
int(x)	convertește la număr întreg	int("15") → 15
float(x)	convertește la număr real	float("3") → 3.0
str(x)	convertește la șir de caractere	str(30) → "30"

⚠ **Important!** Funcția ***math.sqrt()*** face parte din modulul math, care trebuie importat la începutul programului cu import math; argumentul trebuie să fie pozitiv sau zero.

Funcții predefinite pentru calcule matematice

Pentru a extrage rapid informații dintr-un set de date, cum ar fi o listă, e nevoie de instrumente potrivite. Python oferă subprograme predefinite pentru acest scop, scutind programatorul de scrierea unor bucle pentru calcule simple.

Funcție	Ce returnează	Exemplu
len(colecție)	numărul de elemente	len([45, 120, 30]) → 3
min(colecție)	cea mai mică valoare	min([45, 120, 30]) → 30
max(colecție)	cea mai mare valoare	max([45, 120, 30]) → 120
sum(colecție)	suma tuturor elementelor	sum([45, 120, 30]) → 195

⚠ **Important!** min() și max() nu pot fi aplicate pe o colecție vidă; sum() funcționează pe colecții de numere; int("3,5") generează eroare, deoarece șirul nu reprezintă un număr întreg valid.

Exemplu rezolvat

Cerință: Se citește o listă cu prețurile a ***n*** produse dintr-un raft. Se cere să se afișeze numărul de produse, prețul minim, prețul maxim, suma totală și prețul mediu, rotunjit la 2 zecimale.

Cum gândim problema

Categorie	Conținut
Date de intrare	preturi — listă de n numere reale
Date de ieșire	numărul de produse, prețul minim, maxim, suma și media
Relații	media = suma(preturi) / len(preturi)
Restricții	lista preturi conține cel puțin un element

Pașii algoritmului

1. Citește lista de prețuri.
2. Determină numărul de produse cu len().
3. Determină prețul minim și maxim cu min() și max().
4. Calculează suma cu sum() și media, rotunjită cu round().
5. Afișează toate rezultatele.

Pseudocod

```
citește lista preturi
n ← lungime(preturi)
pret_min ← minim(preturi)
pret_max ← maxim(preturi)
suma ← suma_elementelor(preturi)
media ← rotunjire(suma / n, 2)
scrie n, pret_min, pret_max, suma, media
```

Codul în Python

```
preturi = [45, 120, 30, 75]
print("Număr produse:", len(preturi))
print("Preț minim:", min(preturi))
print("Preț maxim:", max(preturi))
print("Sumă totală:", sum(preturi))
media = sum(preturi) / len(preturi)
print("Preț mediu:", round(media, 2))
```

Verificăm cu câteva exemple

Nr.	Date de intrare	Tip	Rezultat așteptat
1	preturi = [45, 120, 30, 75]	general	n=4; min=30; max=120; sumă=270; medie=67.5
2	preturi = [50]	limită (un singur element)	n=1; min=max=50; sumă=50; medie=50.0
3	preturi = [20, 20, 20]	special (valori egale)	n=3; min=max=20; sumă=60; medie=20.0
4	preturi = [15.5, 200]	special (valori diferite mult)	medie = 107.75
5	preturi = []	invalid	min() și max() generează eroare pe listă vidă

Activitate de rezolvat împreună cu profesorul

Pentru lista de note [6, 10, 8, 10, 7], calculați manual, apoi verificați cu funcțiile predefinite: numărul de valori, minimul, maximul, suma și media rotunjită la 2 zecimale.

Exerciții propuse

Rezolvă exercițiile de mai jos în caiet sau într-un fișier separat.

1. Ce returnează round(3.5), respectiv round(2.5), în Python 3? Explicați regula folosită.
2. Scrieți un program care citește o listă de n temperaturi și afișează temperatura minimă, maximă, suma și media lor.
3. Scrieți o funcție distanta_orizontala(x1, x2) care calculează distanța dintre două puncte aflate pe aceeași axă.

Temă pentru acasă

Alegeți o listă de cel puțin 5 valori dintr-un context la alegere (note, distanțe, temperaturi) și scrieți un program care afișează toate cele patru rezultate (număr de valori, minim, maxim, sumă, medie).

Autoevaluare!

Afirmație	Da	Nu	Nu sunt sigur(ă)
Știu să folosesc funcțiile predefinite abs(), round(), int(), float(), str().	☐	☐	☐
Știu să import și să utilizez modulul math.	☐	☐	☐
Știu să aplic len(), min(), max(), sum() pe o listă de valori.	☐	☐	☐
Știu să interpretez corect rezultatul funcției round() pentru valori de tip .5.	☐	☐	☐

Dacă ai bifat „Nu” sau „Nu sunt sigur(ă)” la vreo afirmație, recitește secțiunea corespunzătoare sau întreabă profesorul.

Ce reținem

- Funcția math.sqrt() face parte din modulul math, care trebuie importat la începutul programului cu import math; argumentul trebuie să fie pozitiv sau zero.
- min() și max() nu pot fi aplicate pe o colecție vidă; sum() funcționează pe colecții de numere; int("3,5") generează eroare, deoarece șirul nu reprezintă un număr întreg valid.

Glosar de termeni

Termenii sunt ordonați în ordinea în care apar în lecțiile capitolului.

Termen	Explicație
Informatică	Domeniul care se ocupă cu reprezentarea, prelucrarea și transmiterea informației cu ajutorul sistemelor de calcul.
Gândire computațională	Modalitate de abordare a problemelor prin descompunere, recunoașterea modelelor, abstractizare, algoritmizare și evaluare.
Dată	Reprezentare asupra căreia un algoritm poate opera; caracterizată prin nume, tip și valoare.
Algoritm	Succesiune finită și ordonată de pași, executați într-o ordine bine stabilită, care transformă datele de intrare în rezultat.
Etapele elaborării unui program	Analiza problemei, proiectarea soluției, implementarea, testarea și depanarea, analiza și optimizarea.
Schemă logică	Reprezentare grafică a unui algoritm, prin blocuri și săgeți.
Pseudocod	Descriere textuală a unui algoritm, apropiată de limbajul natural, independentă de limbajul de programare folosit ulterior.
Limbaj de programare	Mijloc de comunicare prin care un algoritm este transmis calculatorului, sub o formă pe care acesta o poate executa.
Limbaj interpretat	Limbaj în care instrucțiunile sunt executate pe rând, fără a fi transformate în prealabil (Python este un exemplu).
Limbaj compilat	Limbaj în care programul este transformat, înainte de execuție, într-o formă executabilă.
Variabilă	Identificator căruia i se atribuie o valoare; tipul ei este stabilit automat, în funcție de valoarea atribuită.
Tip de date	Categoria valorii unei variabile: int (întreg), float (real), str (șir de caractere), list (listă), bool (logic).
Operator	Simbol care indică o operație asupra unor date: aritmetic, relațional, logic sau de atribuire.
Structură de control	Mod de organizare a execuției instrucțiunilor unui program: liniară, alternativă sau repetitivă.
Subprogram (funcție)	Bloc de instrucțiuni identificat printr-un nume, care rezolvă o sarcină specifică și poate fi reutilizat.
Parametru	Variabilă prin care un subprogram primește date din exterior.
Argument	Valoarea transmisă efectiv unui subprogram, la apel, pentru un parametru.
Valoare implicită	Valoare pe care o primește automat un parametru atunci când nu i se transmite un argument la apel.
Variabilă locală	Variabilă definită în interiorul unei funcții; există doar pe durata execuției acesteia.
Variabilă globală	Variabilă definită în afara oricărei funcții; poate fi citită din interiorul funcțiilor.
Funcție predefinită	Funcție inclusă în limbaj sau într-un modul, gata de utilizare (de exemplu: abs, round, len, min, max, sum).
Eroare de sintaxă	Eroare provocată de nerespectarea regulilor limbajului de programare; codul nu poate fi interpretat.
Eroare de logică	Eroare care nu oprește execuția programului, dar produce rezultate greșite.
Eroare de execuție	Eroare care oprește brusc rularea programului (de exemplu, o împărțire la zero).
Testare	Verificarea unui program cu date variate — normale, limită și eronate — pentru a-i confirma corectitudinea.
Depanare	Localizarea și corectarea cauzei unui rezultat greșit, identificată prin testare.

Hartă conceptuală

Schema de mai jos arată cum se leagă între ele noțiunile parcurse în cele 5 lecții ale capitolului.