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

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.)

Capitolul 1 – Fundamentele programării

Lecția 1 – Etapele elaborării unui program

Fișa de identificare a lecției

Element	Conținut
Clasa	a IX-a
Durata	50 de minute
Tipul lecției	Dobândire de cunoștințe noi
Limbaj	Pseudocod și Python
Prerechizite	Operații aritmetice de bază, noțiuni elementare despre calculator
Resurse	Calculator, mediu Python, videoproiector, fișă de lucru, tablă
Conținuturi	Informatică, date, algoritm, etapele elaborării unui program, tipuri de erori

Corelare cu competențele specifice (conform programei)

Cod	Competența specifică
CS 1.1	Identifică principalele caracteristici ale organizării datelor în cadrul unor modele conceptuale fundamentale — date simple sau liste, pentru a structura și accesa date în vederea prelucrării acestora
CS 1.3	Identifică specificul, caracteristicile și etapele unor strategii de rezolvare a problemelor prin proiectarea modulară a algoritmilor
CS 2.3	Explică etapele unor strategii de rezolvare a problemelor prin proiectarea modulară a algoritmilor
CS 5.3	Evaluează corectitudinea și eficiența algoritmilor care utilizează strategii de rezolvare a problemelor prin proiectarea modulară a algoritmilor

Competențe urmărite

- Analizarea unei probleme și identificarea datelor necesare rezolvării ei.
- Descrierea etapelor de elaborare a unui program.
- Recunoașterea tipurilor de erori care pot apărea într-un program.
- Aplicarea unei metode sistematice pentru rezolvarea unei probleme simple.

Obiective de învățare

La finalul lecției, elevul va fi capabil:

- să definească noțiunile de informatică, dată și algoritm;
- să clasifice datele după tip, moment de utilizare și posibilitatea de modificare;
- să enumere și să explice cele 5 etape de elaborare a unui program;
- să diferențieze eroarea de sintaxă, de logică și de execuție;
- să aplice etapele de elaborare pentru o problemă simplă, cu teste corespunzătoare.

Scenariul didactic

Etapă	Timp	Activitatea profesorului	Activitatea elevilor	Dovada învățării
Captarea atenției	5 min	Prezintă un program care rulează fără să se oprească, dar afișează un total greșit pentru o comandă online.	Formulează ipoteze despre cauza posibilă a erorii.	Ipoteze notate pe tablă
Reactualizare	5 min	Recapitulează prin întrebări ce știu elevii despre calculator și operații aritmetice.	Răspund și dau exemple din activitatea proprie.	Răspunsuri orale
Predarea conținutului nou	20 min	Explică noțiunile de dată și algoritm și prezintă cele 5 etape de elaborare a unui program.	Notează, pun întrebări și oferă exemple proprii.	Fișă de notițe completată
Fixarea cunoștințelor	15 min	Ghidează rezolvarea problemei bugetului de cumpărături parcurgând toate etapele.	Analizează problema, proiectează algoritmul și propun teste.	Algoritm în pseudocod și tabel de teste
Feedback și încheiere	5 min	Aplică un bilet de ieșire cu 2-3 întrebări scurte.	Răspund și își autoevaluează gradul de înțelegere.	Bilet de ieșire completat

Întrebarea esențială

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

Situația de pornire

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.

Predicție: Elevii notează ipoteze despre locul unde ar putea apărea problema: citirea prețurilor, calculul sumei sau modul de afișare a rezultatului, și justifică alegerea.

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.

Definiție: 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

Reține! 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)

Reține! 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.

Analiza problemei

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

Proiectarea 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”
```

Tabel de teste

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

Complexitate: algoritmul citește fiecare preț o singură dată; timpul de execuție este O(n), iar memoria suplimentară folosită este constantă, O(1).

Activități și exerciții pentru elevi

Activitate ghidată

Sarcină: 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.

Criterii de reușită: suma este actualizată corect la fiecare pas; comparația cu bugetul se face după citirea tuturor prețurilor; mesajul afișat corespunde rezultatului calculat.

Exerciții

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.

Diferențiere

- Sprijin: elevul primește algoritmul cu doi pași lipsă și un tabel de urmărire parțial completat.
- Nivel de bază: elevul identifică datele de intrare, de ieșire și tipurile de erori posibile.
- Aprofundare: elevul compară două variante ale algoritmului și justifică diferențele de eficiență.

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.

Evaluare și autoevaluare

1. Ce diferență există între o dată de intrare și o dată de ieșire?
2. De ce trebuie ca testarea să preceadă depanarea?
3. Scrieți un test invalid pentru problema bugetului de cumpărături și precizați ce ar trebui să afișeze programul.

Repere pentru profesor

Sugestii metodice

- Porniți de la un exemplu concret din viața reală (o comandă online) — situația arată rapid că „rulează” nu înseamnă „este corect”.
- Cereți elevilor să precizeze rezultatul așteptat înainte de a discuta soluția, altfel testarea devine simplă observare.
- Insistați că validitatea unei valori depinde de restricțiile problemei, nu de o regulă generală.

Întrebări de ghidare

- Ce informații din enunț devin date de intrare?
- Cum stabilim ce rezultat ar trebui obținut, înainte de a rula programul?
- Ce se întâmplă dacă una dintre valorile citite nu respectă restricțiile problemei?

Soluții orientative

Exercițiu	Răspuns orientativ
1	Analiză, proiectare, implementare, testare, depanare, optimizare; proiectarea clarifică pașii înainte ca aceștia să fie scrișI în cod, reducând erorile de logică.
2	Exemple posibile: 12 ani (general); 6 și 19 ani (limită); 5 sau 20 de ani (invalid).
3	Se adaugă un contor, incrementat de fiecare dată când prețul citit este mai mare decât 30.

Lecția 2 – Pseudocod și limbaje de programare

Fișa de identificare a lecției

Element	Conținut
Clasa	a IX-a
Durata	50 de minute
Tipul lecției	Dobândire și aplicare de cunoștințe
Limbaj	Pseudocod și Python
Prerechizite	Noțiunile de dată și algoritm, etapele elaborării unui program
Resurse	Calculator, mediu Python, videoproiector, fișă de lucru, tablă
Conținuturi	Scheme logice, pseudocod, limbaje interpretate/compilate, transcrierea pseudocod–Python

Corelare cu competențele specifice (conform programei)

Cod	Competența specifică
CS 3.3	Aplică strategii de rezolvare a problemelor prin proiectarea modulară a algoritmilor
CS 4.3	Analizează aplicabilitatea, avantajele și dezavantajele utilizării unor strategii de rezolvare a problemelor prin proiectarea modulară a algoritmilor

Competențe urmărite

- Recunoașterea modurilor de reprezentare a unui algoritm.
- Descrierea structurilor de control folosind pseudocodul.
- Transcrierea unui algoritm din pseudocod într-un limbaj de programare.
- Diferențierea unui limbaj interpretat de unul compilat.

Obiective de învățare

La finalul lecției, elevul va fi capabil:

- să enumere cele trei moduri de reprezentare a unui algoritm;
- să descrie structurile liniară, alternativă și repetitivă în pseudocod;
- să transcrie corect un fragment de pseudocod în Python;
- să diferențieze un limbaj interpretat de unul compilat.

Scenariul didactic

Etapă	Timp	Activitatea profesorului	Activitatea elevilor	Dovada învățării
Captarea atenției	5 min	Prezintă un elev (ipotetic) care scrie direct cod Python, fără plan, și obține un program greu de corectat.	Discută ce ar fi putut face diferit.	Opinii formulate oral
Reactualizare	5 min	Recapitulează, prin întrebări, etapele de elaborare a unui program, discutate anterior.	Reamintesc etapele și rolul fiecăreia.	Răspunsuri orale
Predarea conținutului nou	20 min	Prezintă modurile de reprezentare a unui algoritm și structurile de control în pseudocod.	Notează, compară exemple, pun întrebări.	Fișă de notițe
Fixarea cunoștințelor	15 min	Ghidează transcrierea unui algoritm din pseudocod în Python, pas cu pas.	Transcriu algoritmul și îl testează la calculator.	Cod Python funcțional
Feedback și încheiere	5 min	Adresează 2-3 întrebări de verificare rapidă.	Răspund și își evaluează înțelegerea.	Răspunsuri scurte

Întrebarea esențială

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

Situația de pornire

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.

Predicție: Elevii formulează ipoteze despre ce ar fi putut evita această situație și dacă scrierea prealabilă a pașilor ar fi ajutat la localizarea erorii.

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
	  instrucțiune 1
	altfel
	  instrucțiune 2
	sfârșit dacă

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

Reține! 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 (nu suma lor).

Analiza problemei

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

Proiectarea 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
scrie produs
```

Transcriere în Python

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

Tabel de teste

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

Complexitate: algoritmul parcurge fiecare cifră a codului o singură dată; numărul de pași este proporțional cu numărul de cifre.

Activități și exerciții pentru elevi

Activitate ghidată

Sarcină: 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.

Criterii de reușită: fiecare cifră este extrasă corect cu operatorii % și //; produsul este actualizat la fiecare pas; se identifică motivul pentru care rezultatul final este 0.

Exerciții

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(___).

Diferențiere

- Sprijin: elevul primește pseudocodul cu o linie lipsă și trebuie doar să identifice ce lipsește.
- Nivel de bază: elevul identifică tipul reprezentării (schemă logică, pseudocod, cod) pentru exemple date.
- Aprofundare: elevul compară eficiența pseudocodului față de schema logică pentru un algoritm cu mai multe ramificații.

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.

Evaluare și autoevaluare

1. Care este diferența dintre o schemă logică și pseudocod?
2. De ce este util să scriem algoritmul în pseudocod înainte de a scrie codul Python?
3. Transcrieți în Python următoarea structură din pseudocod: pentru i ← 1, 5 execută scrie i*i.

Repere pentru profesor

Sugestii metodice

- Porniți de la un exemplu în care amestecarea gândirii asupra problemei cu sintaxa limbajului a dus la o eroare greu de găsit.
- Insistați asupra faptului că pseudocodul este independent de limbajul de programare ales ulterior.
- Folosiți exemplul cu cifra 0 pentru a discuta un caz special care merită atenție la proiectare, nu doar la testare.

Întrebări de ghidare

- Ce tip de structură de control folosim atunci când repetăm un pas de un număr necunoscut de ori?
- Ce se schimbă în pseudocod atunci când trecem de la o structură alternativă la una repetitivă?
- Ce diferență există între rularea unui program interpretat și rularea unuia compilat?

Soluții orientative

Exercițiu	Răspuns orientativ
1	citește a, b; dacă a < b atunci scrie a; altfel scrie b.
2	Se verifică restul împărțirii lui y la 2; dacă restul este 0, se afișează „par”, altfel „impar”.
3	produs = 1; for i in range(1, n + 1): produs *= i; print(produs).

Lecția 3 – Variabile, tipuri de date și structuri de control

Fișa de identificare a lecției

Element	Conținut
Clasa	a IX-a
Durata	50 de minute
Tipul lecției	Dobândire și aplicare de cunoștințe
Limbaj	Python
Prerechizite	Pseudocod, noțiunile de dată și algoritm
Resurse	Calculator, mediu Python, videoproiector, fișă de lucru, tablă
Conținuturi	Variabile, tipuri de date, operatori, structuri de control în Python

Corelare cu competențele specifice (conform programei)

Cod	Competența specifică
CS 2.1	Explică principiile de organizare a datelor în cadrul unor modele conceptuale fundamentale — date simple sau liste, pentru a le gestiona eficient
CS 3.1	Utilizează modul de organizare și prelucrare a datelor în cadrul unor modele conceptuale fundamentale — date simple sau liste, pentru a rezolva probleme de gestionare a datelor
CS 4.1	Analizează caracteristicile, modul de prelucrare, avantajele și dezavantajele utilizării unor modele conceptuale fundamentale — date simple sau liste, pentru a alege structura adecvată în rezolvarea unor probleme concrete
CS 5.1	Argumentează alegerea unor modele conceptuale fundamentale — date simple sau liste — și a modurilor de prelucrare a acestora pentru rezolvarea unor probleme informatice concrete
CS 6.1	Proiectează algoritmi personalizați care utilizează modele conceptuale fundamentale — date simple sau liste — pentru organizarea și prelucrarea eficientă a datelor, utilizând algoritmi de bază, în vederea rezolvării unor probleme

Competențe urmărite

- Utilizarea variabilelor și a tipurilor de date de bază în Python.
- Aplicarea operatorilor aritmetici, relaționali și logici.
- Scrierea corectă a structurilor alternative și repetitive.

Obiective de învățare

La finalul lecției, elevul va fi capabil:

- să declare și să utilizeze variabile în Python;
- să identifice tipurile de date int, float, str, list și bool;
- să folosească operatori aritmetici, relaționali și logici în expresii;
- să scrie structuri alternative și repetitive corect indentate.

Scenariul didactic

Etapă	Timp	Activitatea profesorului	Activitatea elevilor	Dovada învățării
Captarea atenției	5 min	Prezintă secvența x = input("Vârsta: "); print(x + 5), care generează eroare la rulare.	Formulează ipoteze despre cauza erorii.	Ipoteze notate
Reactualizare	5 min	Recapitulează pseudocodul și structurile de control învățate anterior.	Reamintesc structurile liniară, alternativă, repetitivă.	Răspunsuri orale
Predarea conținutului nou	20 min	Prezintă variabilele, tipurile de date, operatorii și structurile de control în Python.	Notează, testează exemple scurte la calculator.	Cod scurt executat corect
Fixarea cunoștințelor	15 min	Ghidează rezolvarea unei probleme cu temperaturi folosind o structură alternativă.	Scriu, testează și corectează codul propriu.	Program Python funcțional
Feedback și încheiere	5 min	Aplică un bilet de ieșire cu întrebări scurte.	Răspund și se autoevaluează.	Bilet de ieșire completat

Întrebarea esențială

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

Situația de pornire

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.

Predicție: Elevii formulează ipoteze despre tipul de date returnat de input() și despre motivul pentru care operația x + 5 eșuează.

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 #.

Reține! 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

Reține! 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	=, +=, -=, *=, /=, //=, %=

Reține! 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 paşi***:

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

Reține! 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.

Analiza problemei

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

Proiectarea 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
```

Transcriere 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:", diferenta)
else:
    print("Temperaturi egale")
```

Tabel de teste

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()

Complexitate: algoritmul execută un număr constant de operații, indiferent de valorile citite — O(1).

Activități și exerciții pentru elevi

Activitate ghidată

Sarcină: 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.

Criterii de reușită: valoarea lui produs este actualizată corect la fiecare pas al buclei; produsul final corespunde calculului 1×2×3×4.

Exerciții

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.

Diferențiere

- Sprijin: elevul primește codul cu o linie de indentare greșită și trebuie doar să o corecteze.
- Nivel de bază: elevul asociază fiecare valoare din exemple cu tipul de date corespunzător (int, float, str, list, bool).
- Aprofundare: elevul rescrie exemplul cu temperaturi folosind o listă de valori, în loc de doar două variabile separate.

Temă pentru acasă

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

Evaluare și autoevaluare

1. Care este diferența dintre operatorii / și // în Python?
2. De ce este importantă indentarea în interiorul unei structuri alternative sau repetitive?
3. Scrieți o structură alternativă care afișează „major” dacă vârsta citită este cel puțin 18, altfel „minor”.

Repere pentru profesor

Sugestii metodice

- Porniți de la eroarea de tip generată de x + 5 pentru a introduce noțiunea de conversie explicită.
- Cereți elevilor să testeze fiecare exemplu la calculator imediat după prezentare, nu doar să îl copieze.
- Insistați asupra diferenței dintre / (împărțire reală) și // (împărțire întreagă), o sursă frecventă de confuzie.

Întrebări de ghidare

- Ce tip de date returnează input() înainte de orice conversie?
- Ce se întâmplă dacă indentarea unei linii din interiorul unui if nu este corectă?
- Care este diferența dintre o structură repetitivă cu număr cunoscut și una cu număr necunoscut de pași?

Soluții orientative

Exercițiu	Răspuns orientativ
1	21 — x este convertit la întreg (7), apoi înmulțit cu y (3).
2	aria = (cateta1 * cateta2) / 2, cu ambele catete citite ca numere reale.
3	if n % 2 == 0: print("par") else: print("impar").

Lecția 4 – Subprograme: definiție, parametri, variabile locale/globale

Fișa de identificare a lecției

Element	Conținut
Clasa	a IX-a
Durata	50 de minute
Tipul lecției	Dobândire și aplicare de cunoștințe
Limbaj	Python
Prerechizite	Variabile, tipuri de date, structuri de control în Python
Resurse	Calculator, mediu Python, videoproiector, fișă de lucru, tablă
Conținuturi	Subprograme (funcții), parametri, argumente, valoare returnată, variabile locale/globale

Corelare cu competențele specifice (conform programei)

Cod	Competența specifică
CS 1.5	Identifică elementele de sintaxă și componentele din definiția și apelul subprogramelor și subprogramelor predefinite pentru operații uzuale, pentru a le utiliza în implementarea algoritmilor în limbaj de programare
CS 2.5	Explică mecanismul de executare a subprogramelor și subprogramelor predefinite pentru operații uzuale, pentru a le utiliza în implementarea algoritmilor în limbaj de programare
CS 3.5	Utilizează elementele de sintaxă și componentele din definiția și apelul subprogramelor și subprogramelor predefinite pentru operații uzuale, în implementarea algoritmilor în limbaj de programare

Competențe urmărite

- Definirea și apelarea unui subprogram (funcție) în Python.
- Utilizarea parametrilor, inclusiv a celor cu valoare implicită.
- Diferențierea variabilelor locale de variabilele globale.

Obiective de învățare

La finalul lecției, elevul va fi capabil:

- să definească un subprogram (funcție) cu parametri și valoare returnată;
- să diferențieze parametrul de argument;
- să folosească valori implicite ale parametrilor;
- să identifice variabilele locale și globale dintr-un program.

Scenariul didactic

Etapă	Timp	Activitatea profesorului	Activitatea elevilor	Dovada învățării
Captarea atenției	5 min	Prezintă un program lung, cu același calcul repetat de trei ori, în locuri diferite.	Observă repetarea și propun soluții de simplificare.	Idei notate pe tablă
Reactualizare	5 min	Recapitulează structurile de control și tipurile de date folosite anterior.	Reamintesc exemple din lecția precedentă.	Răspunsuri orale
Predarea conținutului nou	20 min	Explică structura unui subprogram, parametrii, valorile implicite și variabilele locale/globale.	Notează, testează exemple scurte de funcții.	Funcții scurte executate corect
Fixarea cunoștințelor	15 min	Ghidează scrierea unei funcții pentru calculul prețului final al unui produs.	Scriu, apelează și testează funcția proprie.	Funcție Python funcțională
Feedback și încheiere	5 min	Adresează întrebări scurte despre parametri și variabile.	Răspund și se autoevaluează.	Răspunsuri scurte

Întrebarea esențială

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?

Situația de pornire

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.

Predicție: Elevii formulează ipoteze despre ce s-ar întâmpla dacă una dintre cele trei copii ar fi uitată necorectată și propun o soluție mai sigură.

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.

Definiție: 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 se transmit 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))
```

Reține! 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

Reține! 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%).

Analiza problemei

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
```

Transcriere 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))
```

Tabel de teste

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

Complexitate: funcția execută un număr constant de operații aritmetice, indiferent de valorile primite — O(1).

Activități și exerciții pentru elevi

Activitate ghidată

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

Criterii de reușită: identificarea corectă a parametrilor cu valoare implicită; recunoașterea variabilei pret_redus ca variabilă locală; observația că exemplul nu folosește nicio variabilă globală.

Exerciții

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.

Diferențiere

- Sprijin: elevul primește funcția calculeaza_pret_final cu un parametru implicit lipsă și trebuie doar să îl completeze.
- Nivel de bază: elevul identifică antetul, corpul și instrucțiunea return într-o funcție dată.
- Aprofundare: elevul proiectează o funcție cu trei parametri, dintre care doi au valori implicite, pentru o problemă la alegere.

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.

Evaluare și autoevaluare

1. Care este diferența dintre un parametru și un argument?
2. De ce sunt utile valorile implicite ale parametrilor unei funcții?
3. Ce se întâmplă dacă o funcție nu conține instrucțiunea return?

Repere pentru profesor

Sugestii metodice

- Porniți de la exemplul cu codul repetat de trei ori, pentru a motiva nevoia de subprograme.
- Insistați asupra faptului că return încheie imediat execuția funcției, spre deosebire de print, care doar afișează.
- Discutați explicit riscurile folosirii variabilelor globale, folosind exemplul cu rata_schimb.

Întrebări de ghidare

- Ce se întâmplă cu o variabilă creată în interiorul unei funcții, după ce funcția își încheie execuția?
- Cum decidem dacă un parametru ar trebui să aibă o valoare implicită?
- Ce diferență există între o funcție care returnează un rezultat și una care doar îl afișează cu print?

Soluții orientative

Exercițiu	Răspuns orientativ
1	def cel_mai_mic(a, b, c): return min(a, min(b, c)) — sau o secvență echivalentă cu if-uri.
2	c) argumente.
3	rata_tva este variabilă globală; pret este parametru (deci variabilă locală); total este variabilă locală.

Lecția 5 – Subprograme predefinite pentru calcule și colecții

Fișa de identificare a lecției

Element	Conținut
Clasa	a IX-a
Durata	50 de minute
Tipul lecției	Dobândire și aplicare de cunoștințe
Limbaj	Python
Prerechizite	Subprograme, parametri, liste — noțiuni introductive
Resurse	Calculator, mediu Python, videoproiector, fișă de lucru, tablă
Conținuturi	Funcții predefinite pentru calcule (abs, round, math.sqrt) și pentru colecții (len, min, max, sum)

Corelare cu competențele specifice (conform programei)

Cod	Competența specifică
CS 4.5	Analizează aplicabilitatea, avantajele și dezavantajele utilizării subprogramelor, în implementarea algoritmilor în limbaj de programare
CS 5.5	Evaluează corectitudinea și eficiența programelor care utilizează subprograme, pentru a rezolva probleme informatice
CS 6.5	Implementează aplicații în limbaj de programare care integrează subprograme personalizate, pentru a modulariza și organiza eficient codul în cadrul soluțiilor informatice

Competențe urmărite

- Utilizarea funcțiilor predefinite pentru calcule matematice.
- Utilizarea funcțiilor predefinite pentru prelucrarea listelor.
- Interpretarea corectă a rezultatelor rotunjirii și conversiilor de tip.

Obiective de învățare

La finalul lecției, elevul va fi capabil:

- să folosească funcțiile predefinite abs(), round(), int(), float(), str();
- să importe și să utilizeze modulul math;
- să aplice len(), min(), max(), sum() pe o listă de valori;
- să interpreteze corect rezultatul funcției round() pentru valori de tip .5.

Scenariul didactic

Etapă	Timp	Activitatea profesorului	Activitatea elevilor	Dovada învățării
Captarea atenției	5 min	Prezintă rezultatele round(2.5) și round(3.5), ambele diferite de ceea ce elevii ar anticipa intuitiv.	Formulează ipoteze despre regula de rotunjire folosită.	Ipoteze notate
Reactualizare	5 min	Recapitulează noțiunile de funcție, parametru și listă.	Reamintesc exemple din lecțiile anterioare.	Răspunsuri orale
Predarea conținutului nou	20 min	Prezintă funcțiile predefinite pentru calcule și pentru colecții, cu exemple.	Notează, testează exemplele la calculator.	Cod testat corect
Fixarea cunoștințelor	15 min	Ghidează analiza unei liste de prețuri folosind funcțiile predefinite.	Scriu și testează programul propriu.	Program Python funcțional
Feedback și încheiere	5 min	Adresează întrebări scurte de verificare.	Răspund și se autoevaluează.	Răspunsuri scurte

Întrebarea esențială

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?

Situația de pornire

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.

Predicție: Elevii formulează ipoteze despre motivul acestei diferențe și despre ce alt mecanism ar putea folosi Python pentru rotunjire.

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"

Reține! 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 colecții de valori

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

Reține! 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.

Analiza problemei

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

Proiectarea 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
```

Transcriere 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))
```

Tabel de teste

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

Complexitate: fiecare dintre funcțiile len, min, max, sum parcurge lista o singură dată; complexitatea globală este O(n).

Activități și exerciții pentru elevi

Activitate ghidată

Sarcină: 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.

Criterii de reușită: valorile calculate manual coincid cu cele obținute din len/min/max/sum; media este rotunjită corect la 2 zecimale.

Exerciții

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.

Diferențiere

- Sprijin: elevul primește lista de prețuri deja introdusă în cod și trebuie doar să completeze apelurile funcțiilor predefinite lipsă.
- Nivel de bază: elevul asociază fiecare funcție predefinită (len, min, max, sum) cu rolul ei, fără a scrie cod.
- Aprofundare: elevul tratează, în plus, cazul unei liste vide, afișând un mesaj corespunzător înainte de a apela min/max.

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).

Evaluare și autoevaluare

1. Ce returnează sum([]) — o listă vidă?
2. Care este diferența dintre int(-4.8) și round(-4.8)?
3. Scrieți un test cu date invalide pentru problema din exemplul rezolvat și precizați ce s-ar întâmpla.

Repere pentru profesor

Sugestii metodice

- Porniți de la round(2.5) și round(3.5) pentru a arăta că regula „rotunjire bancară” diferă de cea învățată la matematică.
- Insistați asupra faptului că min() și max() nu funcționează pe o listă vidă — este o sursă frecventă de eroare la testare.
- Cereți elevilor să scrie manual, pe hârtie, un exemplu mic înainte de a rula funcțiile predefinite, pentru a compara rezultatele.

Întrebări de ghidare

- Ce se întâmplă dacă aplicăm min() pe o listă cu un singur element?
- De ce sum(preturi) / len(preturi) poate produce o eroare dacă lista este vidă?
- Ce diferență există între trunchierea unui număr cu int() și rotunjirea cu round()?

Soluții orientative

Exercițiu	Răspuns orientativ
1	round(3.5) → 4; round(2.5) → 2 — Python 3 rotunjește la cel mai apropiat număr par (rotunjire bancară).
2	int(-4.8) trunchiază spre zero → -4; round(-4.8) rotunjește la cel mai apropiat întreg → -5.
3	preturi = [] — apelul min(preturi) sau max(preturi) ar genera o eroare, deoarece lista este vidă.

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.