

PYTHON

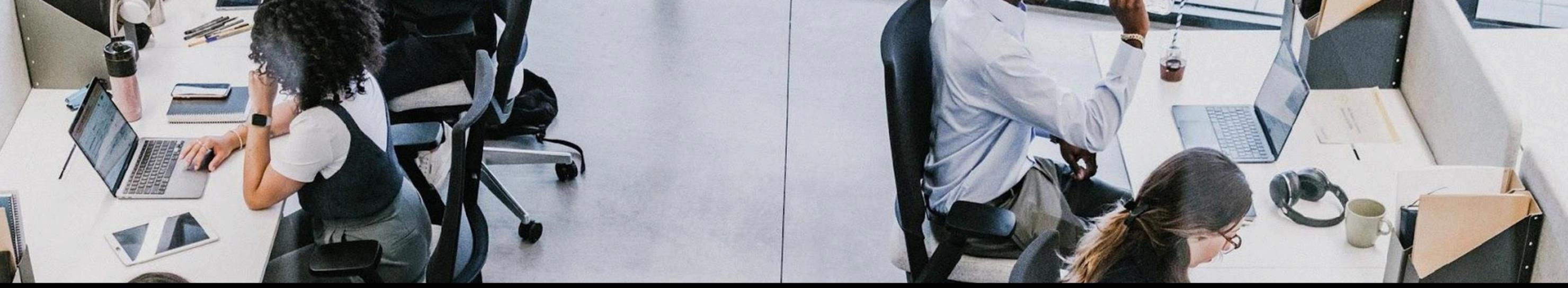
Introducere

{iT} Factory



Structură Sesiune

➤ Sfaturi Generale	03	➤ Variabile	13
➤ Reguli Curs	04	➤ Tipuri de Date	15
➤ Obiective	05	➤ Funcția Type + Type Casting	17
➤ Principii de bază	08	➤ Funcția Input	20
➤ Primul Program	11	➤ Prelucrarea Stringurilor	22
➤ Comentarii	12		



Sfaturi Generale

- Tratați cursul cu **seriozitate și profesionalism**. Cei care își ating obiectivele nu sunt întotdeauna cei mai smart, ci aceia care au consistență în ceea ce fac. Consistența duce la excelență.
- **Lăsați comentarii** explicative în notițele voastre, informații pentru voi cei din viitor.
- Recomandăm să **vizualizați înregistrarea** și să vă notați aspectele importante + întrebări pentru trainer pentru sesiunea următoare.
- Să faceți exercițiile din **studiul de echipă**
- În timpul orelor, să aveți curajul să **puneți întrebări** când ceva nu e clar. Acest lucru va fi cel care va face cu adevărat diferență

Reguli Curs



Va exista un sheet de prezență unde va trebui să vă asigurați constant că vă notați atunci când participați la curs



Vă rugăm să îñtrerupeți trainerul oricând aveți întrebări. Doar aşa își poate da seama unde trebuie să mai insiste cu explicații / exemple.



E important să intrați cu 3 minute mai devreme la curs în caz că apar probleme tehnice. Astfel puteți profita la maxim de cele 2 ore alocate.

Obiective Principale Curs

Până la final TOȚI* veți avea:

- Cunoștințe de bază despre Programare Python
- Înțelegerea conceptului de Programare Orientată pe Obiecte
- Capabilitatea de a explica diferența între clase și obiecte
- Posibilitatea să învățați ce înseamnă testarea automată
- Să puteți să creați de la zero trei framework-uri de testare automată: BDD, Unit Testing Library, API

* *toti cei care sunt activi, implicați, își fac temele, dedică timp pentru studiu individual / de echipă și pun întrebări trainerului vor atinge aceste obiective.*

Obiective Secundare

Nu fac parte din curricula cursului LIVE, dar vă punem la dispoziție materiale extra ca să aveți un avantaj la interviuri. Sfatul nostru e să vă concentrați pe ele doar după cursul live, astfel încât să nu fiți copleșiți de noile informații.

Așadar, următoarele elemente vor fi disponibile pentru voi pe platforma [hello](#):

- Cunoștințe teoretice despre **bazele programării** - Primii Pași în Programare.
- Capacitatea de a construi un mic **brand personal** (Curs Portofoliu Wordpress) care va consta în:
 - Website propriu prin care angajatorul să vă cunoască pe voi și pe munca voastră;
 - CV în limba Engleză;
 - Profil LinkedIn;
 - Github public (un loc în cloud unde se pune codul scris de voi);
 - Veți primi feedback dacă ne trimiteți un email cu ele la adresa hello@itfactory.ro.



Setup Funcțional



Principii de bază în programare - primul program



Print statement, comentarii, variabile, tipuri de date



Type Casting, Input statement, Complexitate String (partea 1 - string index, string length)

Obiective Sesiune 1

Principii de bază în programare...

• **terminal** = zona în care trimitem instrucțiuni către program (altele decât cod Python). Tot de aici putem instala librării externe

Exemple: python --version, pip install selenium

• **Consolă** = *zona în care primim output (răspuns vizual) de la programul rulat*

• **IDE** = *Integrated Development Environment - Pycharm. Este un editor de cod*
• **venv** = *Virtual environment - zona care folosește în mod izolat și securizat toate librăriile externe*

...Principii de bază în programare

a compila = a traduce din 'human reading syntax' în 'machine language'. Codul se interpretează secvențial, linie cu linie, de sus în jos

Machine language = *binary code (cod binar)* - combinații diferite de 0 și 1. Principiul seamănă cu cel din codul Morse. Pentru 1 se transmite un impuls electric, pentru 0 o pauză. Impulsurile sunt reprezentate pe adrese de memorie în felul următor:

- 1 bit = memorie în care începe doar o singura valoare. 1 (true), 0 (false)
- 1 Byte = 8 biti. Numere între 0 (00000000) și 255 (11111111)
- 1 Kilobyte = 1.024 bytes
- 1 Megabyte = 1.024 kilobytes (1.048.575 bytes)

Limbaj compilat = Limbaj care asigură transformarea codului din sintaxa în cod citibil de processor.

Codul e transformat (compilat) în limbaj mașină iar ulterior rulat

Limbaj compilat: **Java**

Limbaj interpretat = Limbaj care asigură citirea codului la momentul execuției linie cu linie și transformat în cod mașină 'on the go'. Interpretorul citește codul linie cu linie și îl transformă în cod mașină la momentul execuției

Limbaj interpretat: **Python**

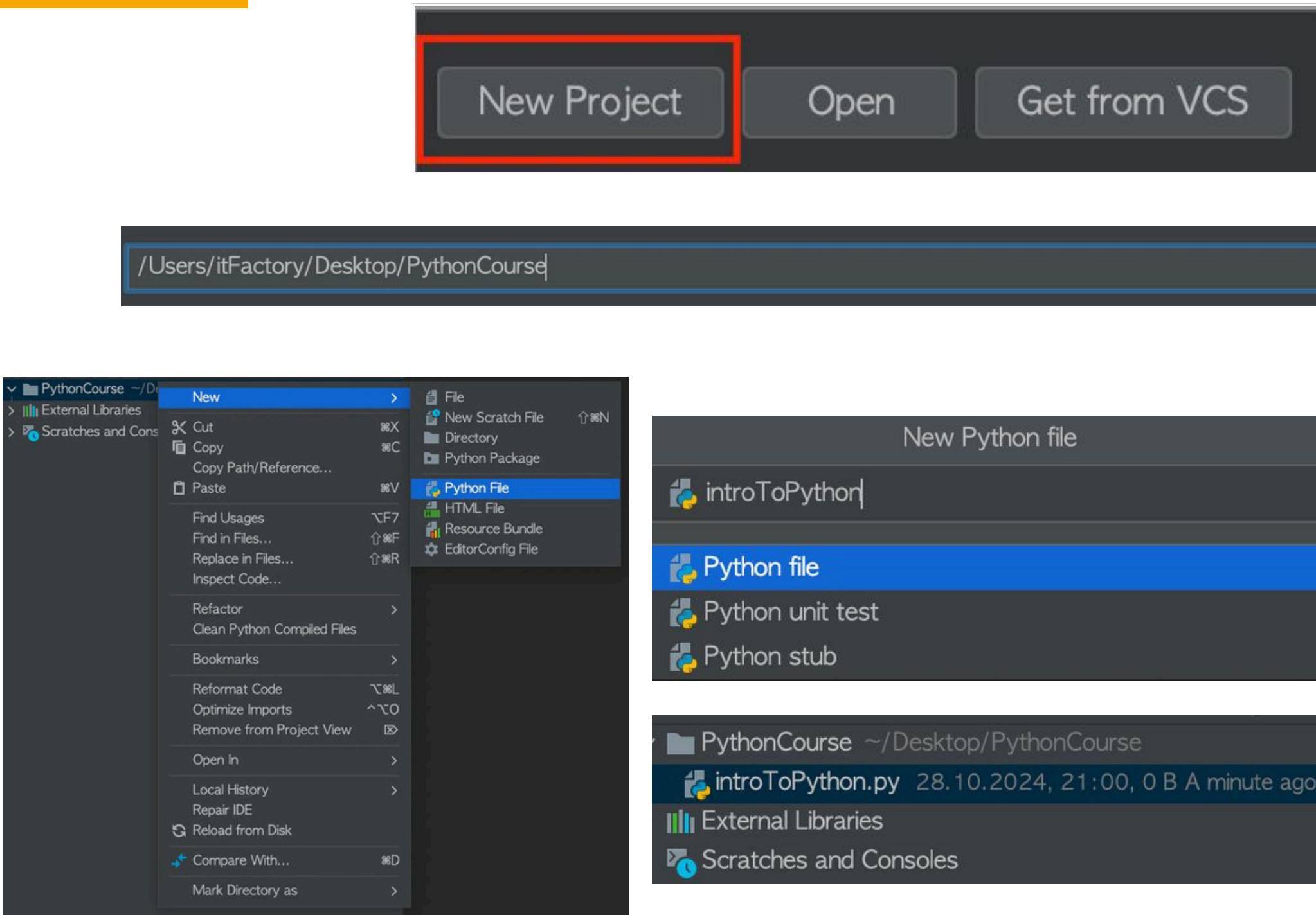


PROJECT - CREARE

Pentru crearea unui Proiect nou **deschideți** mai întâi programul **Pycharm** și apoi dați click pe **New Project**.

Alegeți locația în care vreți să creați proiectul, dați un nume acestuia și apoi apăsați **Create**. Proiectul se va deschide automat.

Ca să puteți începe să lucrați, e nevoie să creați un fișier python. Pentru asta, dați click pe numele proiectului din meniul din stânga (numit root) și selectați **New Python File**. Dați un nume fișierului și apoi apăsați Enter. Fișierul va apărea în meniul din stânga.



PRIMUL PROGRAM PYTHON

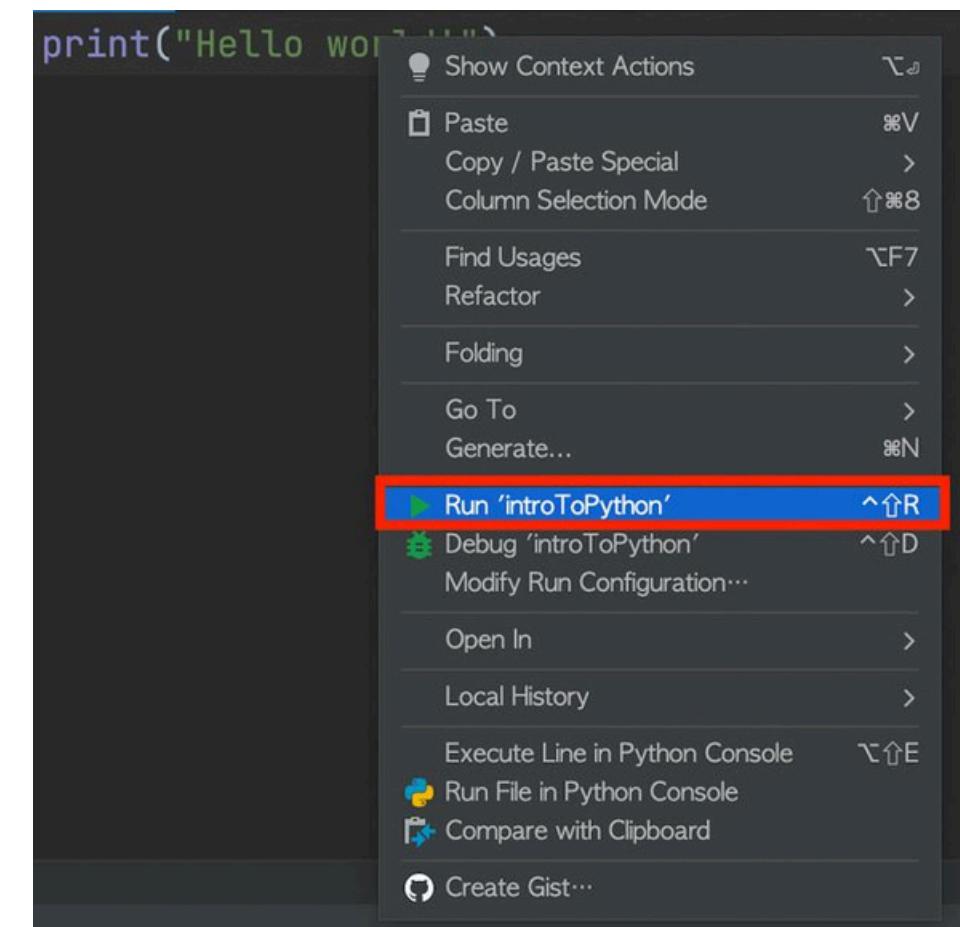
```
print("Hello world!")
```

Funcția `print` afișează în consolă informațiile care sunt scrise între paranteze.

```
Hello world!  
  
Process finished with exit code 0
```

scrieți instrucțiunea de mai sus în fișierul nou creat și rulați codul. Rularea se face prin apăsarea butonului verde în formă de triunghi din partea de sus a programului, sau prin **click dreapta oriunde** în cod și selectarea opțiunii **Run**.

Rularea cu succes este marcată de mesajul “**Process finished with exit code 0**”



COMENTARII

- Comentariile sunt linii de cod care sunt ignoreate în timpul rulării unui program.

- De ce avem nevoie de comentarii?

- documentarea codului
 - creșterea înțelegerii codului
 - prevenirea executării uneia sau mai multor linii de cod

Tipuri de comentarii:

- single-line comments – definite folosind semnul "#" la începutul liniei de cod
- multi-line comments – definite folosind semnul "#" la începutul fiecărei linii de cod sau folosind trei seturi de ghilimele simple sau duble

1 # Acesta este un comentariu pe o singura linie

8 """
9 Acesta este alt comentariu
10 pe mai multe linii
11 """

VARIABILE

- O variabilă este un container din memorie care stochează valori ce pot fi modificate de mai multe ori pe parcursul execuției programului.
- Adresa de memorie este identificată de numele variabilei
- Procesul de **alocare de memorie** pentru o variabilă se numește **declarare**
- Procesul de **salvare a unei valori inițiale** într-o variabilă se numește **initializare**
- În Python declararea și initializarea se fac concomitent.
- Variabilele **au un nume unic**, pentru a putea fi identificate și folosite ulterior
- Variabila este **creată în momentul în care îi atribuim o valoare**
- **NU putem pune spațiu** în numele unui variabilă
- Numele variabilelor **se scrie cu litere mici**
- Numele variabilelor **trebuie să înceapă cu literă mică sau simbolul “_”**, dar poate conține și cifre (user1)
- Variabilele **sunt CASE-SENSITIVE** (my_var = 5 e diferită de my_Var = 5)
- Variabilele **pot sa isi schimbe valoarea** pe parcursul execuției programului (suprascriere)
- Variabilele **pot să își schimbe tipul de date** pe parcursul execuției programului prin suprascriere

CREARE VARIABILE

Exemplu creare variabile:

```
marca_masina = "Volvo"  
model_masina = "XC 60"
```

Putem crea mai multe variabile într-o singură linie de cod, cu valori diferite sau aceeași valoare.

```
1 # cream trei variabile cu valori diferite  
2 x, y, z = "Orange", "Banana", "Cherry"  
3  
4 print(x) # Orange  
5 print(y) # Banana  
6 print(z) # Cherry  
7  
8  
9 # cream trei variabile cu aceeași valoare  
10 a = b = c = "Apple"  
11  
12 print(a) # Apple  
13 print(b) # Apple  
14 print(c) # Apple
```

TIPURI DE DATE

Tipurile de date sunt proprietăți ale adreselor de memorie care descriu ce fel de informații pot fi salvate în acea adresă de memorie și respectiv ce fel de operații se pot efectua cu informația salvată într-o adresă de memorie.

În Python, **tipul de date este detectat automat de sistem** pe baza valorii salvate într-o adresă de memorie, motiv pentru care este permisă schimbarea tipului de date.

Atenție! În programare există și conceptul de **CONSTANTĂ**, adică **adresă de memorie care nu variază pe parcursul execuției**. În **Python constanta există doar ca și convenție** de scriere cu capitals, dar tehnic nu există adrese de memorie constante. Putem doar să le marcăm ca atare pentru ca persoanele care lucrează pe proiect să știe că acea adresă de memorie nu trebuie modificată.

Există mai multe tipuri de date în Python, dar cele mai importante/folosite sunt:

- **int** = număr întreg
- **float** = număr zecimal
- **bool** = valoare de adevărat sau fals (True sau False)
- **string** = sir de caractere de la tastatură delimitat de ghilimele simple sau ghilimele duble

```
1 # initializam o variabila de tip int/numar intreg
2 cantitate = 10
3
4 # initializam o variabila de tip float/numar zecimal
5 pret = 10.45
6
7 # initializam variabile de tip boolean
8 este_impar = False
9 este_par = True
10
11 # definim o variabila string folosind ghilimele simple
12 nume = 'Popa'
13
14 # definim o variabila string folosind ghilimele duble
15 prenume = "Maria"
```

TIPURI DE DATE

- EXERCITII -

Exercițiu 1:

- Definește o variabilă de tip int, numita ‘latime’.
- Definește o variabilă de tip string, numită ‘descriere’.
- Definește două variabile de tip float, numite ‘pret’ și ‘discount’.
- Definește o variabilă de tip bool, numită ‘initializat’, care are valoarea True.
- Printează variabilele definite la punctele anterioare.

Exercițiu 2:

Folosind o singură linie de cod, definește 2 variabile, de tip int, cu valoarea 10.

Exercitiul 3:

Folosind o singură linie de cod, initializează 2 variabile de tip string cu valori diferite.

FUNCȚIA TYPE

Functia `type()` e o funcție predefinită (există deja creată în sintaxa Python) care ne expune tipul de date al variabilei dată ca input.

```
culoare = "rosu"
print(type(culoare))

varsta = 35
print(type(varsta))

pret = 15.66
print(type(pret))

absolvit = False
print(type(absolvit))
```

```
<class 'str'>
<class 'int'>
<class 'float'>
<class 'bool'>
```

```
Process finished with exit code 0
```

TYPE CASTING

type casting = convertirea tipurilor de date

Type casting este util atunci când datele ne vin într-un format în care nu le putem procesa. De exemplu, dacă ne vine un număr în format string nu putem opera adunări sau scăderi cu acel număr. De aceea trebuie să îl transformăm în int ca să putem să îl folosim.

Functiile int(), str(), bool(), float() schimba tipul de date.

```
1 numar1 = 10
2 numar2 = '10'
3
4 # Sunt numar1 si numar2 de acelasi tip?
```

TYPE CASTING

- EXERCITII -

Exercitiul 6:

- Definește o variabilă de tip int, afișează-o în consolă. Afișează de asemenea și tipul acesteia.
- Definește o variabilă de tip float, afișează-o în consolă. Afișează de asemenea și tipul acesteia.
- Definește o variabilă de tip string, afișează-o în consolă. Afișează de asemenea și tipul acesteia.
- Definește o variabilă de tip bool, afișează-o în consolă. Afișează de asemenea și tipul acesteia.

Exercitiul 7:

- Definește o variabilă de tip int. Afișează-o în consolă.
- Afișează în consolă tipul variabilei definite la punctul anterior, folosind funcția type().
- Convertește variabila de tip int de la punctul 1, la tipul float și salvează rezultatul într-o altă variabilă.
- Afișează în consolă tipul variabilei generate la punctul anterior.

Exercitiul 8:

- a. Definește o variabilă de tip string. Afișează-o în consolă.

FUNCȚIA INPUT

- Funcția `input()` ne ajută să citim date de la tastatură și să le salvăm într-o variabilă
- Dacă nu facem type casting, valorile implicite ale datelor furnizate de utilizator vor fi de tip `string`
- Ulterior, putem accesa valorile salvate în variabile după necesitate

```
1 nume = input('Cum te numesti? ') # default - string
2 varsta = int(input('Cati ani ai? ')) # fortam varsta sa fie un int
```

FUNCȚIA INPUT

- EXERCIȚII -

Exercițiu 9:

- Citește de la tastatură un nume de produs. Salvează rezultatul într-o variabilă
- Afisează un mesaj care să conțină variabila salvată.

Exercițiu 10:

- Citește de la tastatură un preț. Obligă utilizatorul să introducă prețul ca și număr zecimal. Salvează rezultatul într-o variabilă
- Afisează un mesaj care să conțină variabila salvată.

PRELUCRAREA STRINGURIILOR

- Concatenare -

Concatenarea reprezintă unirea mai multor stringuri într-unul singur și este utilă în special atunci când facem printare.

```
nume = "Popescu"
prenume = "Ion"
# mai jos am concatenat variabila nume
# cu variabila prenume folosind operatorul "+"
nume_complet = nume + prenume
# mai jos am concatenat un string dat de la tastatura
# cu două stringuri salvate în variabile
print("Numele meu este" + nume + prenume)
```

Concatenarea se poate face în două feluri:

- cu semnul “+”
- folosind *f-strings (printare cu formatare)*

```
varsta = 20
# instructiunea de mai jos va returna eroare
# deoarece operatorul + inseamna concatenare pentru stringuri si adunare pentru numere
# si sistemul avand si numere si stringuri nu stie ce sa faca
print("Varsta mea este de: " + varsta, "ani")

# folosind printare cu formatare, inglobam variabila numerica direct în string,
# unde este tratată și ea ca un string, rezolvând problema
print(f"Varsta mea este de: {varsta} ani")
```

PRELUCRAREA STRINGURIILOR

- Lungime -

- Lungimea unui string este dată de numărul de caractere dintr-un string.
- Lungimea poate să varieze atunci când numărul de caractere din string crește.
- Putem afla numărul de caractere care se află într-un string folosind funcția len()

```
1 info = 'Afara sunt 2 grade'  
2  
3  
4 # afisam lungimea string-ului info  
5 print(len(info)) # => 18
```

PRELUCRAREA STRINGURIILOR

- Index -

- Stringul este format din unul sau mai multe caractere.
- Fiecare caracter dintr-un string are un număr asociat/**poziție asociată**, numit **index**.
- Indexul începe de la 0.

```
info = 'Afara sunt 2 grade'
print(info[0]) # => 'A' (primul caracter din string se afla la indexul 0)
print(info[1]) # => 'f'
print(info[5]) # => ' ' (la indexul 5 avem un spatiu gol)
```

Putem să **accesăm ultimul caracter** dintr-un string în mod dinamic, folosindu-ne de lungimea acestuia. **NU e recomandat** să hardcodăm indexul ultimului element.

```
1 info = 'Afara sunt 2 grade'
2 # prima varianta - e o modalitate dar NU e indicata - DE CE?
3 print(info[17]) # => e
4
5
6 # a doua varianta (PREFERATA)
7 print(info[-1]) # => e
```

PRELUCRARE STRINGURI

- EXERCITII.

Exercitiul 11 :

- Definește două variabile de tip string, 'nume', respectiv 'oras'.
- Afisează în consolă un mesaj care să conțină cele două variabile.

Exercitiul 12:

- Definește două variabile: *nume* (*string*) și *varsta* (*int*).
- Folosind f-string, afisează în consolă, o propoziție care să conțină cele două

Exercitiul 13:

Se dă variabila *prop1* = 'Andy este prescurtarea de la Andrei"

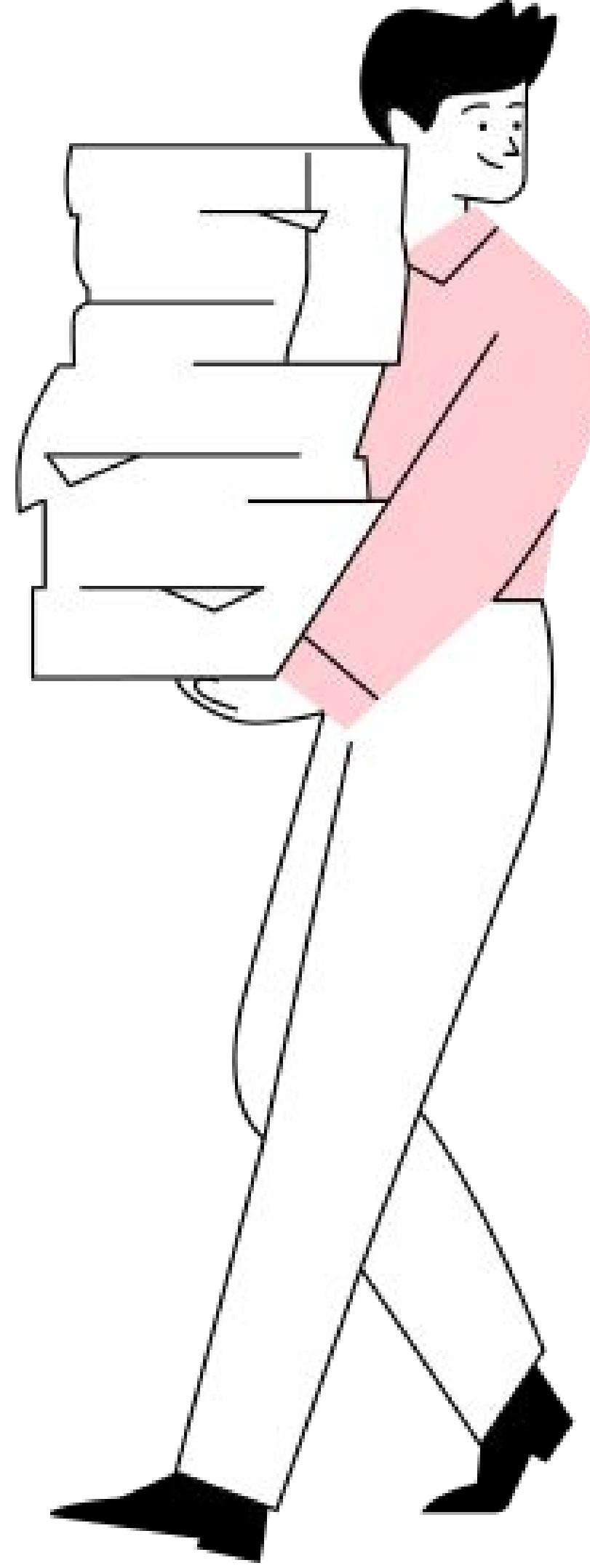
- Afisează primul caracter.
- Afisează al 4-lea caracter.
- Afisează ultimul caracter.

Exercitiul 14:

Se dă string-ul *prop2* = 'Masina e rosie."

- Afisează lungimea stringului *prop2*.

Elemente Esențiale Învățate



Variabile, comentarii

Tipuri de date, type casting

Inputuri și prelucrarea stringurilor

FORMULAR DE FEEDBACK



Dorim să oferim servicii de cea mai înaltă calitate și să ne îmbunătățim în mod continuu pentru a vă satisface nevoile.

Dorim să vă ascultăm părerea și să aflăm cum putem oferi o experiență și mai bună.

Am creat un **formular de feedback activ** permanent, care vă oferă posibilitatea de a ne împărtăși **gândurile, sugestiile și observațiile** voastre. Acesta se găsește în link-ul de mai jos sau pe platforma voastră de grupă, pentru a ajunge la el cu ușurință.

Nu durează mai mult de câteva minute să completați formularul, dar contribuția dvs. este extrem de valoroasă pentru noi. Feedback-ul pe care îl primim ne ajută să identificăm punctele noastre forte și să corectăm eventualele deficiențe.