

## Lucrarea 1. Introducere în Python

### A. Obiectivele lucrării

1. Prezentarea generală a mediului de dezvoltare *PyCharm*
2. Prezentarea modului de instalare a librăriilor *Python*
3. Prezentarea bazelor limbajului *Python*
4. Prezentarea modului de definire și afișare a variabilelor în *Python*
5. Prezentarea operatorilor de bază în *Python*

### B. Ce este Python?

**Python** este un limbaj de programare de nivel înalt, de utilitate generală, putând fi folosit în aproape orice domeniu (dezvoltarea de aplicații mobile/PC/web, dezvoltarea de jocuri video, dezvoltarea de aplicații industriale sau de laborator: inteligență artificială, procesare de date, controlul sistemelor, calcul ingineresc, etc) [1].

Printre avantajele acestui limbaj de programare se numără [2]:

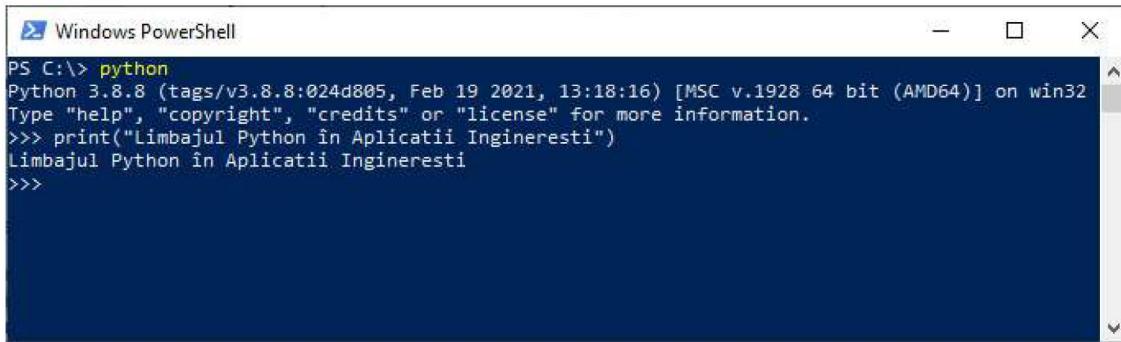
1. Este considerat un limbaj de nivel înalt, adică instrucțiunile acestuia sunt apropriate de limbajul uman.
2. Codul nu este compilat, ci interpretat, economisindu-se astfel timp atât la dezvoltarea, cât și la depanarea aplicațiilor.
3. Există un număr foarte mare de librării ce sunt disponibile în mod gratuit.
4. Oferă posibilitatea mai multor tipuri de programare: programare structurată, programare funcțională și programare orientată pe obiect.

### C. Instalare

Descărcarea kit-ului de instalare se poate face din secțiunea „*Downloads*” a site-ului oficial Python: <https://www.python.org/>. Se recomandă instalarea versiunii *Python 3.9* pentru buna funcționare a exemplelor de aplicații din acest îndrumar.

Există mai multe metode de a utiliza Python:

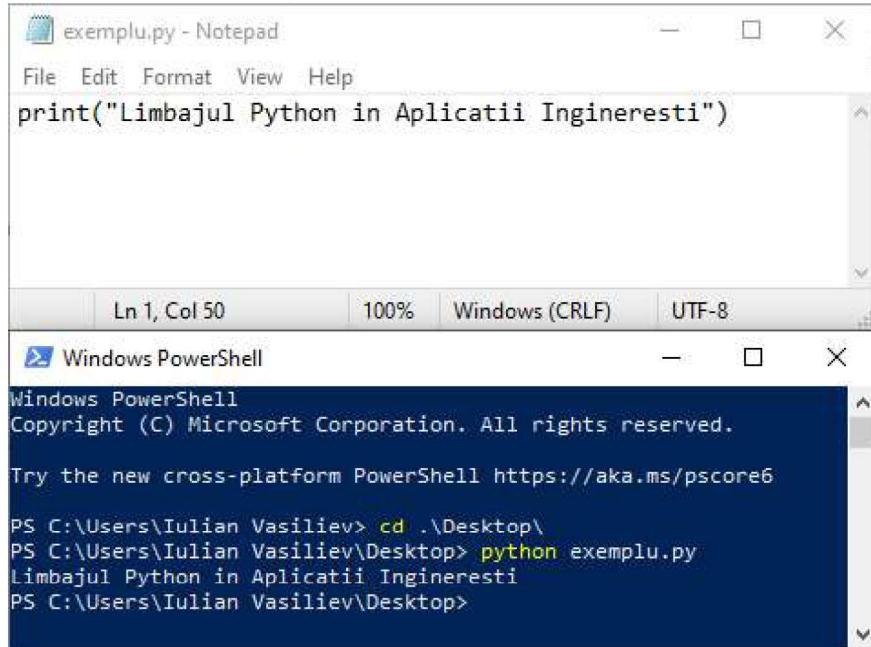
1. Comenzile pot fi introduse în mod interactiv. Pentru această metodă se va deschide o fereastră de „*Command Prompt*” sau „*PowerShell*” în care se va introduce comanda **python**. După apariția prompt-ului **>>>**, se pot introduce instrucțiuni *Python* ce se vor executa în momentul apăsării tastei **Enter**. Figura 1 prezintă un exemplu de utilizare a limbajului *Python* în mod interactiv.



```
PS C:\> python
Python 3.8.8 (tags/v3.8.8:024d805, Feb 19 2021, 13:18:16) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Limbajul Python în Aplicatii Ingineresti")
Limbajul Python în Aplicatii Ingineresti
>>>
```

*Figura 1 – Exemplu de utilizare a limbajului Python în mod interactiv*

2. Instrucțiunile **Python** pot fi scrise într-un fișier script cu extensia **.py** folosind orice editor de text și apoi rulate din „**Command Prompt**” sau „**PowerShell**” introducând comanda **python** urmată de numele fișierului script, incluzând extensia. *Figura 2* prezintă un astfel de exemplu.



The figure shows two windows. The top window is a Notepad titled "exemplu.py - Notepad" containing the Python code: `print("Limbajul Python in Aplicatii Ingineresti")`. The bottom window is a Windows PowerShell window showing the command and its output:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Iulian Vasiliev> cd ..\Desktop\
PS C:\Users\Iulian Vasiliev\Desktop> python exemplu.py
Limbajul Python in Aplicatii Ingineresti
PS C:\Users\Iulian Vasiliev\Desktop>
```

*Figura 2 – Exemplu de rulare a scripturilor Python din linie comandă*

3. Utilizarea unui mediu de dezvoltare (IDE – Integrated Development Environment) care facilitează dezvoltarea aplicațiilor prin îmbinarea, într-o singură interfață, a unei serii de instrumente pentru dezvoltare precum: managementul aplicației, editor de cod sursă, managementul librăriilor, depanare, etc. În cadrul acestui laborator, se va folosi versiunea community a mediului de dezvoltare **PyCharm**. Descărcarea kit-ului de instalare al acestui IDE se poate face din secțiunea „**Downloads**” a site-ului oficial PyCharm: <https://www.jetbrains.com/pycharm/>.

#### D. Prezentarea mediului de dezvoltare PyCharm

La deschiderea mediului de dezvoltare PyCharm veți fi întâmpinați de fereastra de bun venit (*Figura 3*) care va permite crearea unui nou proiect sau deschiderea unui proiect deja existent. În cazul în care la rularea anterioară proiectul în care s-a lucrat nu a fost închis, PyCharm va deschide direct acel proiect. În acest caz, pentru a ajunge la fereastra de bun venit se va închide proiectul deschis din meniul „*File*” → „*Close project*”.

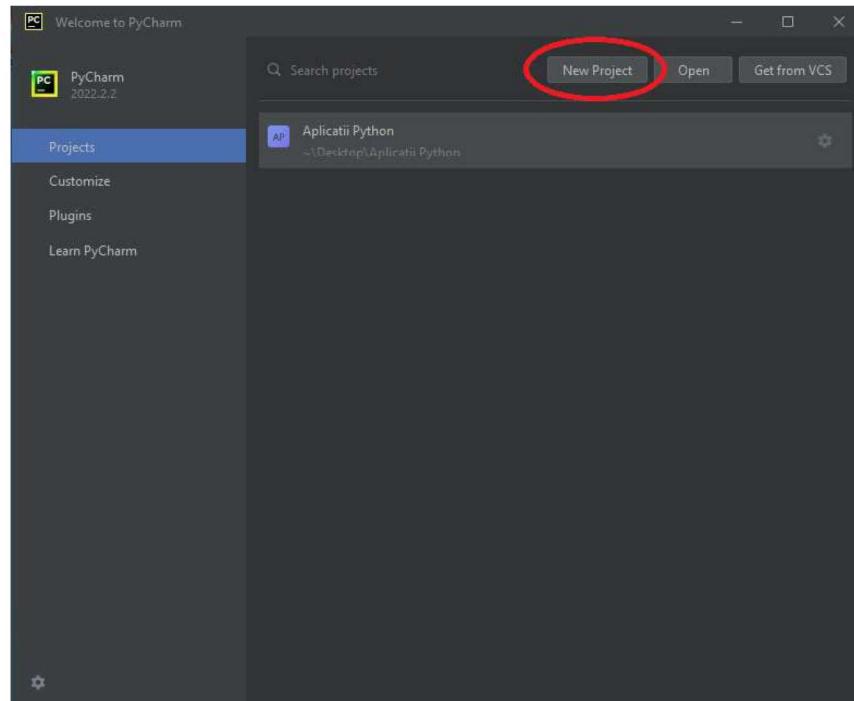


Figura 3 – Fereastra de bun venit a mediului de dezvoltare PyCharm

În fereastra de bun venit, prin apăsarea butonului „*New Project*” (încercuit cu roșu în *Figura 3*), va apărea fereastra ce permite crearea unui nou proiect (*Figura 4*). În această fereastră se poate seta, la câmpul „*Location*” locația proiectului, după care, prin apăsarea butonului „*Create*”, noul proiect va fi creat și va apărea fereastra ce va permite editarea codului proiectului creat (*Figura 5*).

Această fereastră conține mai multe sub-fereestre („*views*”) de bază. În partea stânga se află sub-fereastra „*Project*” ce permite navigarea prin fișierele și directoarele proiectului, iar în partea dreaptă avem editorul de cod propriu-zis. Din partea de jos se pot deschide o serie de alte sub-ferestre cu diferite utilități.

## Lucrarea 1. Introducere în Python

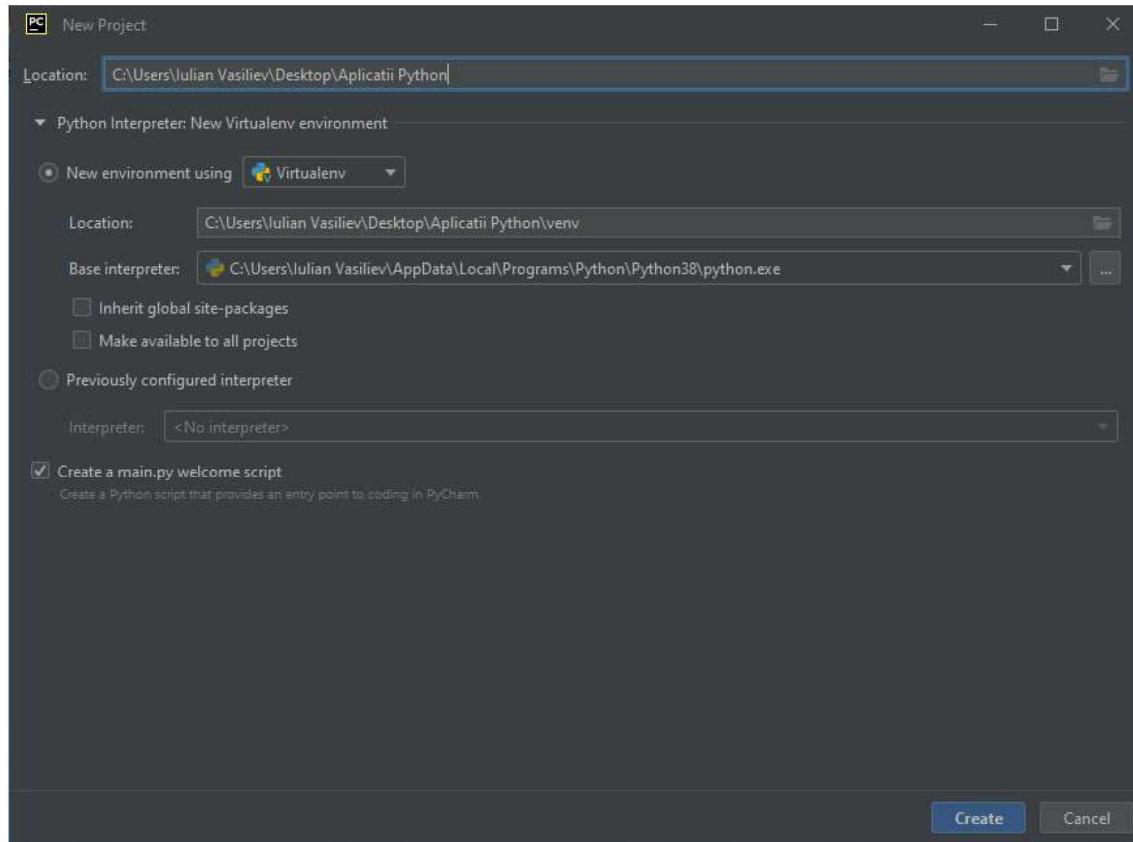


Figura 4 – Fereastră pentru crearea unui nou proiect în mediul de dezvoltare PyCharm

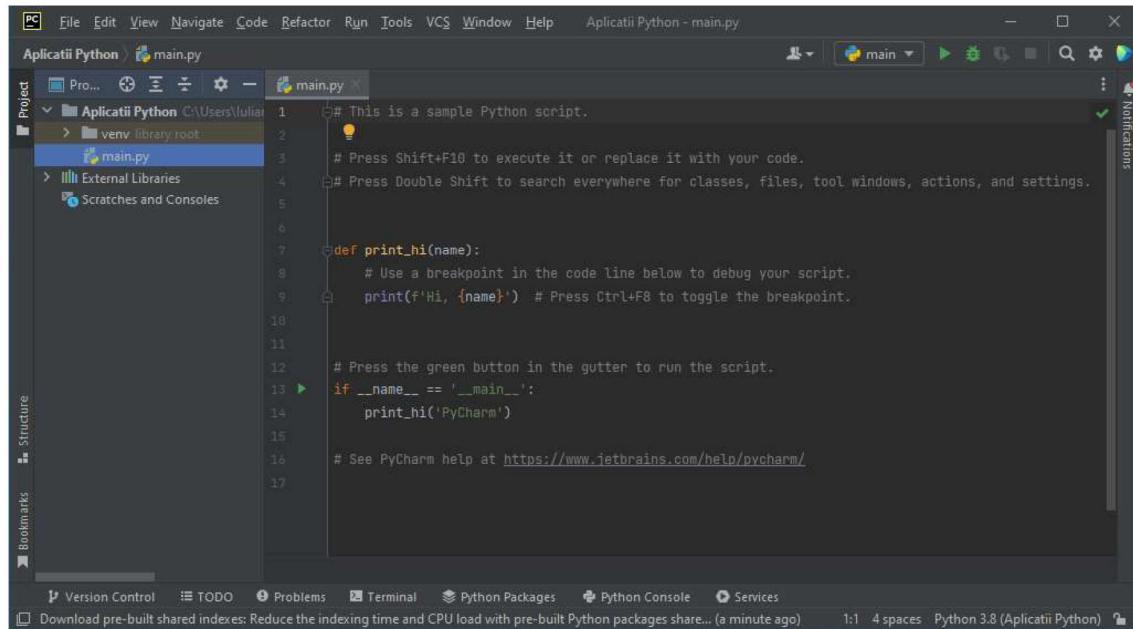


Figura 5 – Fereastra de bază a mediului de dezvoltare PyCharm

Când se creează un nou proiect, se creează implicit și un script „main.py” care este fișierul principal al proiectului. În mod implicit, acest fișier conține o funcție numită `print_hi` ce primește ca parametru o variabilă `name` și care apelată cu parametrul „PyCharm” va afișa în sub-fereastra „Run” mesajul „Hi, PyCharm”. Rularea proiectului se poate face accesând din meniu „Run” → „Run ‘main’”. La rularea proiectului, va apărea în partea de jos sub-fereastra „Run” în care se vor afișa rezultatele rulării (*Figura 6*).

The screenshot shows the PyCharm IDE interface. The top menu bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help, and the current project name "Aplicatii Python - main.py". The left sidebar has sections for Project, Structure, Bookmarks, and Run. The main editor window displays the code for main.py:

```
def print_hi(name):
    # Use a breakpoint in the code line below to debug your script.
    print(f'Hi, {name}!') # Press Ctrl+F8 to toggle the breakpoint.

# Press the green button in the gutter to run the script.
if __name__ == '__main__':
    print_hi('PyCharm')

# See PyCharm help at https://www.jetbrains.com/help/pycharm/
```

The Run tab at the bottom shows the command "C:\Users\Iulian Vasiliev\Desktop\Aplicatii Python\venv\Scripts\python.exe" "C:\Users\Iulian Vasiliev\Desktop\Aplicatii Python\main.py" and the output "Hi, PyCharm". Below that, it says "Process finished with exit code 0".

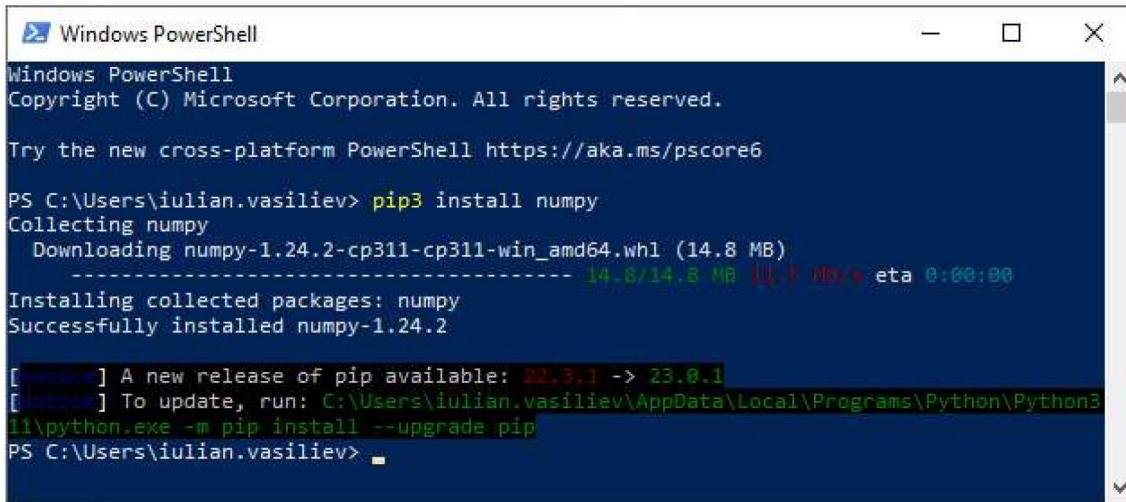
*Figura 6 – Rezultatul rulării proiectului implicit în mediul de dezvoltare PyCharm*

### E. Instalarea librăriilor

Extinderea capacitaților de bază ale limbajului **Python** se face cu ajutorul librăriilor. Librăriile sunt colecții de module specializate pe anumite funcționalități (ex: **numpy** – lucru cu matrice multi-dimensionale, **matplotlib** – afișarea grafică a datelor, etc.). Aceste librării sunt disponibile în mod gratuit și se instalează diferit în funcție de modul de rulare al aplicațiilor **Python**.

Astfel, în cazul în care **Python** se utilizează în mod interactiv sau scripturile sunt rulate din linie comandă, instalarea unei librării se face din „**Command Prompt**” sau „**PowerShell**” folosind comanda `pip3 install` urmată de numele librăriei ce se dorește instalată. Un exemplu se poate observa în figura următoare.

## Lucrarea 1. Introducere în Python



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\iulian.vasiliev> pip3 install numpy
Collecting numpy
  Downloading numpy-1.24.2-cp311-cp311-win_amd64.whl (14.8 MB)
    14.8/14.8 MB 14.8 MB/s eta 0:00:00
Installing collected packages: numpy
Successfully installed numpy-1.24.2

[...]
  A new release of pip available: 22.3.1 -> 23.0.1
[...] To update, run: C:\Users\iulian.vasiliev\AppData\Local\Programs\Python\Python311\python.exe -m pip install --upgrade pip
PS C:\Users\iulian.vasiliev>
```

Figura 7 – Exemplu de instalare a unei librării Python (NumPy) folosind utilitarul pip

În cazul în care se utilizează un mediu de dezvoltare pentru crearea aplicațiilor **Python**, instalarea unei librării poate dифeаt de la IDE la IDE. În cazul PyCharm, instalarea unei librării implică o serie de pași (notați cu **roșu** în Figura 8):

1. se accesează sub-fereastra „**Python Packages**” din partea de jos a ferestrei principale.
2. vom scrie numele librăriei în câmpul de căutare al sub-ferestrei.
3. se selectează librăria ce se dorește instalată.
4. se apasă butonul „**Install package**” din partea dreaptă a sub-ferestrei.

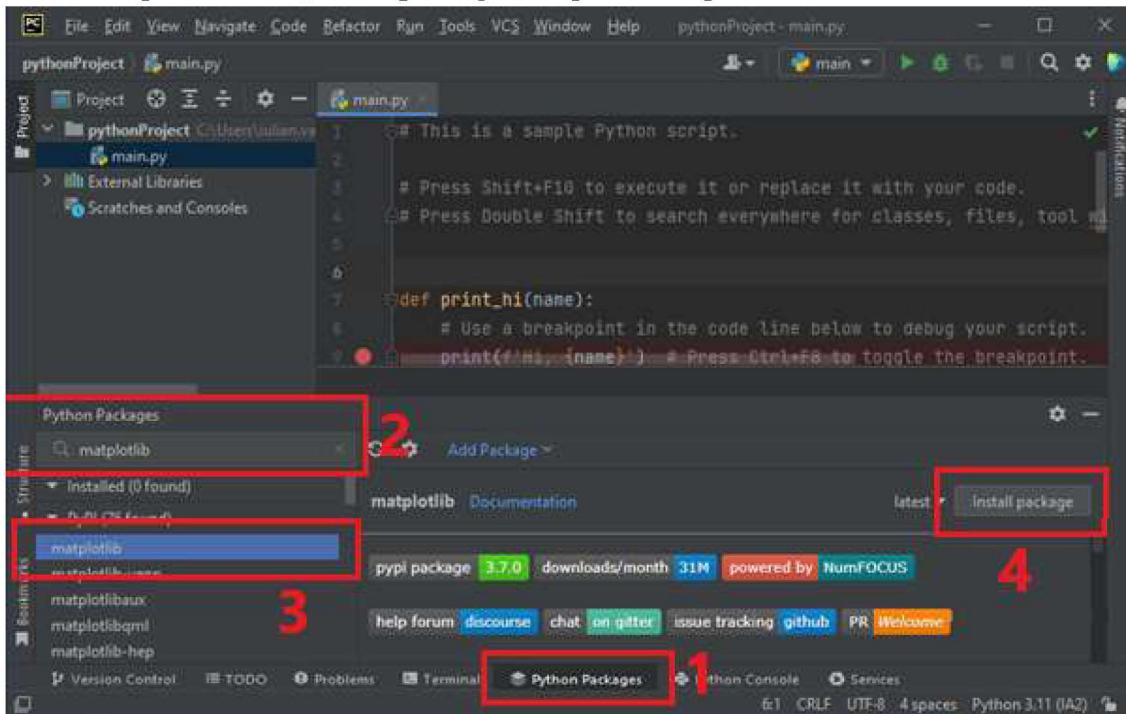


Figura 8 – Procedura de instalare a unei librării Python în IDE-ul PyCharm

## F. Bazele programării în limbajul Python

Una dintre cele mai simple instrucțiuni de bază ale limbajului **Python** este funcția `print` ce poate fi folosită la afișarea mesajelor în consolă. De asemenea, în limbajul **Python** există două tipuri de comentarii în cod: comentarii pe o singură linie, acestea fiind precedate de simbolul `#` și comentarii pe mai multe linii, acestea fiind cuprinse între seturi de ghilimelele `"""` [1]. Exemplu:

<b>main.py</b>
<pre>print("Informatica Aplicata II") # Exemplu de comentariu pe o singură linie """  Exemplu de comentariu pe mai multe linii """  <b>rezultat</b></pre>
Informatica Aplicata II

Așa cum se observă în exemplul anterior, spre deosebire de alte limbaje, în acesta nu este necesar să punem simbolul ; (punct și virgulă) la sfârșitul liniilor de cod. Un lucru important de menționat este că în limbajul **Python**, identarea este foarte importantă. Astfel, o linie de cod este identată doar dacă ea aparține interiorului unei structuri de control (condițională, buclă, funcție, etc.). **Python** nu utilizează accolade pentru a defini blocurile interioare structurilor de control, ci identarea [1]. Aceasta se poate face atât cu spații, cât și folosind caracterul Tab. Implicit, **PyCharm** se va ocupa singur de identare, folosind 4 spații. O identare greșită va duce la returnarea unei erori la rularea aplicației implementate. Exemplu de utilizare a identării pentru definirea codului unei structuri condiționale:

<b>main.py</b>
<pre>a = 1 if a == 2:     # cod interior structurii if     print("Sunt in structura if")     # alte instructiuni in interiorul structurii if # cod in afara structurii if print("OK")</pre>
<b>rezultat</b>
OK

Exemplu de utilizare a unei identări greșite:

<b>main.py</b>
<pre>def test():     x = 7     y = 2    # Aceasta este o identare greșită      if x &lt; y:</pre>

```
        print("x este mai mic decât y")
    else:
print("x nu este mai mic decât y") # Aceasta este o identare greșită
test()

rezultat
File "C:\User\Desktop\test.py", line 3
    y = 10 # Aceasta este o identare greșită
IndentationError: unexpected indent
```

O identare corectă ar arata altfel:

```
main.py
def test():
    x = 7
    y = 2

    if x < y:
        print("x este mai mic decât y")
    else:
        print("x nu este mai mic decât y")
test()

rezultat
x nu este mai mic decât y
```

### a) Tipuri de date și definirea variabilelor în Python

**Python** oferă mai multe tipuri de date, dintre care cele mai simple sunt cele numerice și sirurile de caractere. Datorită orientării spre obiecte a acestui limbaj, orice variabilă **Python** poate fi văzută ca un obiect. De asemenea, este important de menționat că în **Python** nu trebuie declarat tipul de variabilă înainte. Astfel, pentru a defini și atribui o valoare unei variabile se va scrie numele variabilei, urmat de simbolul = și de valoarea pe care o atribuim.

Important [1]:

- Fiecare variabilă trebuie să aibă nume unic;
- Nu sunt permise spații în numele variabilelor;
- Numele variabilelor ar trebui să înceapă cu literă sau cu „\_”;
- Cifrele sunt permise în numele variabilelor (dar nu ca prim caracter);
- Variabilele sunt CASE-SENSITIVE;
- Pentru a respecta convențiile de denumire în Python, variabilele ar trebui să fie denumite folosind formatele: Snake Case, Camel Case;
- Variabilele își pot schimba valorile și tipul de date în timpul execuției programului (suprascriere).

Exemple:

**main.py**

```
# Variabilă care începe cu literă sau cu "_"
varsta_tatalui = 45
_varsta_tatalui = 54

# Numere în numele variabilelor
camera_10 = "Recepție"

# Snake Case
prima_variabla = 10
nr_total = 100
nume_utilizator = "PopescuIon"

# Camel Case
primaVariabla = 20
nrTotal = 200
numeUtilizator = "PopescuIon"
```

Limbajul **Python** oferă suport pentru trei tipuri de date numerice: numere întregi, numere în virgulă mobilă și numere complexe [1]. Exemplu:

**main.py**

```
# numere intregi
nr_intreg = 42
print(nr_intreg)

# numere în virgula mobilă
nr_real1 = 7.0
print(nr_real1)
nr_real2 = float(7)
print(nr_real2)

# numere complexe
nr_complex = 7 + 0j
print(nr_complex)
```

**rezultat**

```
42
7.0
7.0
(7+0j)
```

Un alt tip de date de bază este sirul de caractere, numit și string. Aceasta se poate defini folosind atât ghilimele simple, cât și ghilimele duble. Exemplu:

**main.py**

```
sir_de_caractere = "Informatica Aplicata II"  
print(sir_de_caractere)
```

**rezultat**

```
Informatica Aplicata II
```

Un alt tip de date disponibil în **Python** este lista, reprezentând un vector unidimensional ce poate conține orice număr de elemente de orice tip. Valoarea acesteia se poate inițializa enumerând valorile elementelor acesteia, separate de virgulă între un set de paranteze drepte. Ulterior, un element al listei se poate accesa scriind numele variabilei urmat de indexul elementului între paranteze pătrate. În **Python** numărarea index-urilor începe de la 0. Exemplu:

**main.py**

```
lista = [3, 2, 1]  
print(lista)  
print(lista[1])
```

**rezultat**

```
[3, 2, 1]  
2
```

**b) Operatori matematici de bază în Python**

Toți operatorii matematici de bază (adunare, scădere, înmulțire, împărțire, dar și ridicare la putere) pot fi folosiți în **Python** între oricare tipuri de date numerice, fiind inclusi în limbajul de bază. Pentru adunare, scădere, înmulțire și împărțire operatorii sunt aceiași ca în celelalte limbaje de programare (+, -, \* și /). Pentru ridicare la putere, în **Python** operatorul este „\*\*”. Ordinea operațiilor este cea matematică, având, de asemenea, posibilitatea grupării cu ajutorul parantezelor rotunde. Exemplu:

**main.py**

```
rezultat = 5 + 2 * 4 ** (2 - 1) - (3 + 2j) / 1  
print(rezultat)
```

**rezultat**

```
(10-2j)
```

Un alt operator matematic ce poate fi folosit în **Python** este operatorul modulo (%) ce va calcula restul unei împărțiri.

Din operatorii aritmetici de bază, doi pot fi folosiți și cu siruri de caractere. Adunarea se poate face între două sau mai multe siruri de caractere, rezultatul fiind concatenarea acestor siruri, iar înmulțirea se poate face între un sir de caractere și un număr întreg, rezultatul fiind

multiplicarea sirului de caractere de un număr de ori egal cu operatorul de tip număr întreg. Exemplu:

**main.py**

```
print("Informatica" + " " + "Aplicata" + " " + "II")
print("Laborator" * 3)
```

**rezultat**

Informatica Aplicata II
LaboratorLaboratorLaborator

Operatorul de adunare nu poate fi pus între un sir de caractere și un număr. Dacă dorim să inserăm valoarea unei variabile în interiorul unui sir de caractere se va folosi una din funcționalitățile de bază ale limbajului, și anume „**Literal String Interpolation**” [1]. Pentru aceasta se va pune litera **f** înainte de ghilimelele ce definesc sirul de caractere. Astfel, atunci când dorim să inserăm valoarea unei variabile în acel sir de caractere se va scrie numele variabilei între accolade. Exemplu de utilizarea „**Literal String Interpolation**”:

**main.py**

```
nume = "Informatica Aplicata II"
nr_laborator = 1

print(f"Acesta este laboratorul numarul {nr_laborator} de {nume}")
```

**rezultat**

Acesta este laboratorul numarul 1 de Informatica Aplicata II
--

### c) Funcții matematice de bază

Marea majoritate a funcțiilor matematice de bază nu este disponibilă direct în limbajul de bază, fiind necesară utilizarea librăriei **math** (librăria este pre-instalată). Pentru a importa o librărie în codul aplicației se va folosi instrucțiunea `import` urmată de numele librăriei. Astfel, funcțiile și constantele definite de acea librărie vor putea fi accesate scriind numele librăriei, urmat de simbolul punct și de numele constantei / funcției dorite.

Librăria **math** pune la dispoziție mai multe constante matematice și funcții matematice. Cele mai uzuale se regăsesc în tabelul următor [2]:

*Tabelul 1 - Constante și funcții matematice de bază din librăria math*

<b>Constante Matematice</b>	
<code>math.pi</code>	Constanta matematică $\pi$
<code>math.e</code>	Constanta matematică $e$ (numărul lui Euler)
<b>Funcții pentru reprezentarea numerelor</b>	
<code>math.ceil(x)</code>	Cel mai mic număr întreg mai mare sau egal cu $x$

<code>math.floor(x)</code>	Cel mai mare număr întreg mai mic sau egal cu <b>x</b>
<code>round(x)</code>	Cel mai apropiat întreg de numărul <b>x</b> – funcția este în limbajul de bază (nu trebuie importată librăria <b>math</b> )
<code>abs(x)</code>	Modulul numărului <b>x</b> – funcția este în limbajul de bază
<b>Functii exponentiale și logaritmice</b>	
<code>math.exp(x)</code>	<b>e</b> la puterea <b>x</b>
<code>math.log(x, b)</code>	Logaritm în bază <b>b</b> din <b>x</b> (dacă <b>b</b> lipsește va calcula logaritm natural)
<code>math.log2(x)</code>	Logaritm în baza 2 din <b>x</b>
<code>math.log10(x)</code>	Logaritm în baza 10 din <b>x</b>
<code>math.sqrt(x)</code>	Rădăcina pătrată a lui <b>x</b>
<b>Functii trigonometrice</b>	
<code>math.sin(x)</code>	Sinus de <b>x</b> ( <b>x</b> în radiani)
<code>math.cos(x)</code>	Cosinus de <b>x</b> ( <b>x</b> în radiani)
<code>math.tan(x)</code>	Tangent de <b>x</b> ( <b>x</b> în radiani)
<code>math.asin(x)</code>	Arc sinus de <b>x</b> în radiani (rezultatul între $-\pi/2$ și $\pi/2$ )
<code>math.acos(x)</code>	Arc cosinus de <b>x</b> în radiani (rezultatul între 0 și $\pi$ )
<code>math.atan(x)</code>	Arc tangent de <b>x</b> în radiani (rezultatul între $-\pi/2$ și $\pi/2$ )
<code>math.atan2(y, x)</code>	Arc tangent de <b>y / x</b> în radiani (rezultatul între $-\pi$ și $\pi$ )
<b>Functii de conversie unghiulară</b>	
<code>math.degrees(x)</code>	Convertește în grade unghiul <b>x</b> specificat în radiani
<code>math.radians(x)</code>	Convertește în radiani unghiul <b>x</b> specificat în grade

Spre exemplu, dacă se dorește calcularea lui  $\sin(2 \cdot \pi/3)$  se va scrie:

<b>main.py</b>
<code>import math print(math.sin(2 * math.pi / 3))</code>
<b>rezultat</b>
0.8660254037844387

#### d) Operatori logici

**Python** utilizează logica booleană pentru a evalua condiții. Astfel, acest limbaj folosește două valori booleene ce pot fi returnate la evaluarea unei expresii logice (condiții): `True` și `False`. Operatorii ce permit compararea a două valori sunt asemănători cu cei din alte limbaje. Considerând variabila `x` având valoarea 2, operatorii de comparație se pot observa în tabelul următor:

*Tabelul 2 - Operatori logici în Python*

Operator	Descriere	Exemplu	Rezultat
<code>==</code>	egal cu	<code>print(x == 2)</code>	True
<code>!=</code>	diferit de	<code>print(x != 2)</code>	False
<code>&lt;</code>	mai mic ca	<code>print(x &lt; 2)</code>	False
<code>&lt;=</code>	mai mic sau egal cu	<code>print(x &lt;= 2)</code>	True
<code>&gt;</code>	mai mare ca	<code>print(x &gt; 2)</code>	False
<code>&gt;=</code>	mai mare sau egal cu	<code>print(x &gt;= 2)</code>	True

Un alt operator logic definit de **Python** este operatorul `in` ce se poate utiliza pentru a verifica dacă o anumită valoare se regăsește într-un obiect iterabil. Exemplu:

main.py
<pre>lista_forme = ["triunghi", "patrat", "cerc"] print("triunghi" in lista_forme) print("hexagon" in lista_forme)</pre>
rezultat
<pre>True False</pre>

Pentru a construi expresii booleene complexe, limbajul **Python** pune la dispoziție 3 operatori booleeni, conform tabelului următor:

*Tabelul 3 - Operatori booleeni în Python*

Operator	Descriere	Exemplu	Rezultat
<code>not</code>	negarea expresiei precedate de operator	<code>not True</code>	False
<code>or</code>	operația de „sau logic” între două expresii	<code>True or False</code>	True
<code>and</code>	operația de „și logic” între două expresii	<code>True and False</code>	False

### e) Structura condițională

Așa cum a fost specificat mai devreme, în **Python** blocurile interioare structurilor de control nu sunt definite utilizând accolade ci identare. Structura condițională permite executarea unor secvențe de cod doar dacă anumite condiții sunt îndeplinite. Această structură folosește 3 cuvinte cheie (`if`, `elif` și `else`) pentru definirea ramurilor acesteia. După definirea fiecărei ramuri, înainte de blocul de cod aferent acelei ramuri, este necesar să se pună simbolul „două puncte”.

Sintaxă generală a structurii condiționale:

**main.py**

```
if expr_logical:  
    # bloc de cod - ramura if  
    ...  
elif expr_logica2: # else if  
    # bloc de cod - ramura elif  
    ...  
# alte ramuri elif  
else:  
    # bloc de cod - ramura else  
    ...
```

Blocul de cod interior ramurii `if` se execută doar dacă expresia logică `expr_logical` dată acestei ramuri are valoarea `True`. Blocul de cod interior ramurii `elif` se execută doar dacă expresia logică `expr_logica2` dată acestei ramuri are valoarea `True` și dacă expresiile logice date ramurilor precedente au avut valoarea `False`. Blocul de cod aferent ramurii `else` se execută dacă nici una din expresiile logice aferent ramurilor precedente nu au avut valoarea `True`. Este important de menționat faptul că ramurile `elif` și `else` pot lipsi din structurile condiționale și faptul că pot exista mai multe ramuri `elif`.

**main.py**

```
x = 33  
if x % 2 == 0:  
    print("x este divizibil cu 2!")  
elif x % 5 == 0:  
    print("x este divizibil cu 5!")  
else:  
    print("x nu este divizibil nici cu 2, nici cu 5!")
```

**rezultat**

```
x nu este divizibil nici cu 2, nici cu 5!
```

**f) Structuri repetitive**

**Python** oferă două tipuri de structuri repetitive: bucle `while` – se utilizează pentru a repeta un bloc de cod atât timp cât o condiție este adevărată și bucle `for` – se utilizează pentru a repeta o secvență de cod pentru fiecare element al unui obiect iterabil.

Exemplu de utilizare a structurii repetitive `while`:

**main.py**

```
a = 0  
while a < 4:  
    print(f"Iteratia {a}")  
    a = a + 1
```

**rezultat**

```
Iteratia 0  
Iteratia 1  
Iteratia 2  
Iteratia 3
```

Exemplu de utilizare a structurii repetitive `for`:

**main.py**

```
lista_forme = ["triunghi", "patrat", "cerc"]  
for forma in lista_forme:  
    print(forma)
```

**rezultat**

```
triunghi  
patrat  
cerc
```

Funcția `range` poate fi folosită pentru a crea un obiect iterabil ce conține o secvență de numere. Această funcție poate primi 3 parametri:

*Tabelul 4 - Parametrii funcției range*

Parametru	Descriere	Observație
<code>start</code>	valoarea de start a secvenței	poate lipsi, caz în care se consideră valoarea implicită 0
<code>stop</code>	valoarea de stop a secvenței	nu poate lipsi; de asemenea, valoarea de stop a secvenței nu este inclusă în secvență
<code>step</code>	pasul de incrementare	poate lipsi, valoarea implicită 1

Exemplu:

**main.py**

```
for x in range(1, 11, 2):  
    print(x)
```

**rezultat**

```
1  
3  
5  
7  
9
```

În buclele `while` și `for` se pot utiliza două comenzi pentru a opri execuția blocului de cod:

*Tabelul 5 - Comenzi ce se pot utiliza în interiorul buclelor*

Comanda	Descriere
break	oprește execuția blocului de cod și ieșe din buclă
continue	oprește execuția blocului de cod și continuă cu următoarea iterare a buclei

Exemple:

<b>main.py</b>	<pre># Exemplul 1: bucla se va opri cand cnt ajunge la 4 cnt = 0 while True:     print(cnt)     cnt += 1     if cnt &gt;= 4:         break  # Exemplul 2: Va afisa doar numerele pare for cnt in range(10):     if cnt % 2 != 0:         continue     print(cnt)</pre>
<b>rezultat</b>	<pre>1 3 5 7 9</pre>

## G. Cerințe

1. Deschideți mediul de dezvoltare PyCharm și creați un proiect cu numele dumneavoastră pe Desktop. Familiarizați-vă cu mediul de dezvoltare. Creați din PyCharm în interiorul proiectului creat un director numit „Lab1” în care creați 6 fișiere python: ex2.py, ex3.py, ex4.py, ex5.py, ex6.py, ex7.py, ex8.py, ex9.py, ex10.py. Rezolvați următoarele exerciții câte unul în fiecare fișier creat. Observație: pentru a rula un anumit fișier în PyCharm, veți da click-dreapta pe el în sub-fereastra „*Project*” și apoi „**Run nume\_fisier**”.
2. Scrieți un program Python care să afișeze numele vostru.
3. Creați o variabilă de tip număr întreg, numită *vârstă* și afișați-o în consolă.
4. Creați o variabilă de tip sir de caractere, numită *nume* și afișați-o în consolă.

5. Să se calculeze și afișeze rezultatele:

a)  $5 + 4 \cdot 2$

b)  $\frac{2}{15} + 3^2 \cdot 2$

c)  $8 - 9 \cdot (1 - 3)$

d)  $1 - \frac{3}{7} + 2$

e)  $\frac{2+5^3}{3-\frac{1}{9}}$

f)  $\left( \frac{\frac{2+3}{3}+1}{3^2+2^3} \right)^{\frac{1}{2}}$

g)  $15.601 - 12.8$

h)  $3.256^{1.34}$

6. Stocați valoarea 12 într-o variabilă numită „*a*” și valoarea 3 într-o variabilă numită „*b*”.

Calculați suma, diferența, înmulțirea, împărțirea și ridicarea la putere a variabilelor stocate și salvați rezultatul pentru fiecare situație într-o variabilă separată numită „rezultat”. Afişați rezultatele folosind „**Literal String Interpolation**”. Exemplu: în cazul sumei se va afișa „*12 + 3 = 15*”.

7. Pentru  $a = 22$ ,  $b = a - 8.5$  și  $c = 3 \cdot b + 1$ , calculați:

a)  $c \cdot \frac{a}{b} + \frac{c^2}{2+2 \cdot c} - (b^2 - c) \cdot (c + a^{\frac{1}{2}})$

b)  $\frac{c^2+a^2}{b-c} - (a - c + b)^{\frac{1}{3}}$

8. Să se calculeze și afișeze pe ecran rezultatele:

a)  $\frac{1+\sqrt{2}}{4}$

b)  $\left| e^{-1} - 2 \cdot \sin \frac{\pi}{6} \right|$

c)  $\pi^{0.5} + (1 + \tan 60^\circ)^3$

d)  $2 \cdot \cot^2 \frac{\pi}{6} - \cos \frac{\pi}{2}$

e)  $\frac{\lg(86)+\sin(\cos(\sqrt{22.55}))}{\sqrt[3]{\ln(100)-\log_4(\sin \frac{7\pi}{3})}}$

9. Să se calculeze și să se afișeze rezultatul sumelor:

a)  $\sum_{x=1}^{100} \frac{1}{x^2}$

b)  $\sum_{x=1}^5 x^2 - x$

c)  $\sum_{x=4}^{15} \frac{2^{x+1}}{x-1}$

10. Implementați instrucțiunile pentru a afișa toate numerele în intervalul [0 132] care sunt atât pare cât și divizibile cu 11.