

FUNDAMENTE DE SISTEME BIOLOGICE ȘI INFORMATICĂ MEDICALĂ - Cursul 6 - **Bazele Python**

Ș.I.dr.inf. Oana Sorina CHIRILA

oana.chirila@aut.upt.ro

PYTHON și Informatica medicală



- ▶ Python este adesea o cerință pentru angajare ca **analist de date din sănătate**.
- ▶ De asemenea, este un element cheie pentru a lucra în **analiza asistenței medicale și a datelor medicale**.
- ▶ Python ar putea deveni o componentă majoră a sistemelor de asistență medicală în viitor. Învățarea acesteia, deși poate nu este necesară, s-ar putea dovedi utilă celor care se uită la o carieră în domeniul informaticii medicale și nu numai.

Generalități despre PYTHON



- ▶ Python este un limbaj de programare de nivel înalt, dezvoltat de programatorul olandez Guido van Rossum la începutul anilor 1990.
- ▶ Python are o sintaxă ușor de învățat, care subliniază lizibilitatea, modulele de suport și pachetele, care promovează reutilizarea codului și modularitatea programului.
- ▶ Programatorii „se îndrăgostesc de Python”, deoarece crește productivitatea, potrivit Python.org. De exemplu, ciclul edit-test-debug este rapid, deoarece nu există compilare.
- ▶ Python fiind un limbaj interpretat, are un avantaj față de limbajele compilate cum ar fi C/C++, deoarece necesită mai puțin cod pentru a realiza anumite instrucțiuni.

Generalități despre PYTHON

- ▶ Python **este mai lent decât C**.
- ▶ Aplicațiile Python sunt foarte ușor de depanat, codul putând fi ușor inspectat în timpul rulării.
- ▶ Este foarte ușor de experimentat cu mici fragmente de cod folosind interpretorul Python.
- ▶ **Sintaxa** este gândită în așa fel încât programele Python să fie **ușor de citit**. Acest lucru este obținut prin folosirea de **cuvinte în locul semnelor** (de exemplu, and în loc de &&) și prin includerea indentării în limbaj.
- ▶ În Python **nu se folosesc acolade** (ca în C/C++, Java), ci blocurile de cod se delimitează prin **indentare**.
- ▶ Programele Python sunt, de multe ori, foarte aproape de o “implementare” echivalentă în **pseudocod**.

Generalități despre PYTHON

- ▶ Limbajul Python **este interpretat, nu compilat**. Asta înseamnă că programele Python sunt transformate într-un limbaj intermediar. Acest lucru permite codului să fie ușor de portat pe **diverse sisteme de operare și arhitecturi hardware**.
- ▶ Codul este executat **linie cu linie**. Astfel, dacă - de exemplu - apelăm o funcție care nu există, vom primi un mesaj de eroare abia când se încearcă executarea liniei respective.
- ▶ Erorile de sintaxă sunt raportate înainte de rularea programului.

Aplicații ale limbajului Python

- 1. **Dezvoltarea web:** Python este folosit în dezvoltarea aplicațiilor web și a site-urilor web. Framework-uri populare precum Django și Flask facilitează crearea de aplicații web scalabile și sigure.
- 2. **Analiza datelor:** Python este larg utilizat în știința datelor și analiza datelor. Bibliotecile precum NumPy, Pandas și SciPy oferă funcții puternice pentru manipularea datelor, efectuarea de calcule matematice complexe și crearea de modele statistice.
- 3. **Inteligența artificială și învățarea automată:** Python este un limbaj preferat în domeniul inteligenței artificiale și învățării automate. Bibliotecile TensorFlow, PyTorch și multe altele sunt utilizate pentru a construi și antrena modele de învățare automată și rețele neuronale.
- 4. **Automatizare și scripting:** Python este excelent pentru automatizarea sarcinilor repetitive și crearea de scripturi. Poți dezvolta scripturi pentru administrarea sistemelor, procesarea fișierelor și multe altele.
- 5. **Aplicații desktop:** Python poate fi folosit pentru dezvoltarea aplicațiilor desktop utilizând biblioteci precum PyQt și Tkinter. Aceste aplicații pot fi utilizate pe diferite platforme, inclusiv Windows, macOS și Linux.

Aplicații ale limbajului Python

- ▶ **6. Dezvoltarea de jocuri:** Chiar și în dezvoltarea de jocuri, Python poate fi folosit cu ajutorul bibliotecilor precum Pygame. Deși nu este la fel de performant ca alte limbaje specializate pentru jocuri, Python este potrivit pentru prototiparea rapidă și dezvoltarea de jocuri mai mici și cu grafică mai puțină.
- ▶ **7. Proiecte IoT** (Internet of Things): Python poate fi utilizat în dezvoltarea de proiecte IoT datorită ușurinței de a lucra cu hardware și de a comunica cu senzori și dispozitive înglobate.
- ▶ **8. Aplicații mobile:** Cu ajutorul framework-urilor precum Kivy sau BeeWare, se pot dezvolta aplicații mobile pentru Android și iOS utilizând Python.
- ▶ **9. Aplicații pentru analiza imaginilor și prelucrarea semnalelor:** Python are biblioteci puternice precum OpenCV și SciPy, care permit dezvoltarea de aplicații pentru prelucrarea imaginilor și analiza semnalelor.
- ▶ **10. Aplicații pentru blockchain și criptomonedă:** Există biblioteci Python care facilitează dezvoltarea de aplicații blockchain și pentru lucrul cu criptomonede.

Aplicații ale limbajului Python

- **11. Automatizare a testelor software:** Python poate fi folosit pentru automatizarea testelor software, ceea ce facilitează testarea și asigurarea calității aplicațiilor.
- **12. Instrumente de rețea:** Python are biblioteci care permit dezvoltarea de instrumente pentru administrarea rețelelor, cum ar fi crearea de scanere de porturi, automatizarea configurării echipamentelor de rețea și multe altele.
- **13. Aplicații pentru calcul numeric și simulare:** Python are biblioteci precum NumPy și SciPy, care permit dezvoltarea de aplicații pentru calcul numeric și simulare în diverse domenii științifice și ingineresti.
- **14. Aplicații pentru roboți și drona:** Python este folosit în dezvoltarea de aplicații pentru controlul roboților și a dronelor, datorită ușurinței de a comunica cu diferite componente hardware și senzori.
- **15. Aplicații pentru analiza textului și procesarea limbajului natural:** Python are biblioteci precum NLTK (Natural Language Toolkit) și spaCy care permit analiza și prelucrarea textului, cum ar fi extragerea de informații, analiza sentimentelor și generarea automată de texte.

















Aplicații ale limbajului Python

- **16. Aplicații pentru interacțiunea cu baze de date:** Python se integrează bine cu diverse sisteme de gestiune a bazelor de date (MySQL, PostgreSQL, MongoDB etc.), facilitând dezvoltarea de aplicații pentru interacțiunea cu baze de date și stocarea datelor.
- **17. Aplicații pentru domeniul medical:** Python poate fi utilizat pentru dezvoltarea de aplicații medicale, cum ar fi analiza datelor medicale, simulări medicale, asistența în diagnostic și gestionarea informațiilor despre pacienți.
- Acestea sunt doar câteva dintre numeroasele aplicații pentru care Python este utilizat.

Top limbaje de programare

➡ Conform:
<https://www.tiobe.com/tiobe-index/>

Oct 2023	Oct 2022	Change	Programming Language		Ratings	Change
1	1			Python	14.82%	-2.25%
2	2			C	12.08%	-3.13%
3	4	⬆️		C++	10.67%	+0.74%
4	3	⬇️		Java	8.92%	-3.92%
5	5			C#	7.71%	+3.29%
6	7	⬆️		JavaScript	2.91%	+0.17%
7	6	⬇️		Visual Basic	2.13%	-1.82%
8	9	⬆️		PHP	1.90%	-0.14%
9	10	⬆️		SQL	1.78%	+0.00%
10	8	⬇️		Assembly language	1.64%	-0.75%
11	11			Go	1.37%	+0.10%
12	23	⬆️⬆️		Scratch	1.37%	+0.69%
13	18	⬆️⬆️		Delphi/Object Pascal	1.30%	+0.46%
14	14			MATLAB	1.27%	+0.09%
15	15			Swift	1.07%	+0.02%
16	19	⬆️		Fortran	1.02%	+0.23%
17	12	⬇️⬇️		R	0.96%	-0.26%
18	28	⬆️⬆️		Kotlin	0.96%	+0.53%
19	16	⬇️		Ruby	0.92%	+0.05%
20	20			Rust	0.91%	+0.22%



Avantaje Python



- 1. **Simplitate și ușurință în învățare:** Sintaxa simplă și clară a limbajului Python face ca acesta să fie ușor de învățat și de înțeles, chiar și pentru cei fără experiență în programare.
- 2. **Versatilitate:** Python este un limbaj de programare general, ceea ce înseamnă că poate fi folosit pentru o varietate largă de aplicații, de la dezvoltarea web și analiza datelor la crearea de aplicații desktop și jocuri.
- 3. **Biblioteci bogate:** Python beneficiază de o vastă colecție de biblioteci și framework-uri care facilitează dezvoltarea rapidă și simplă a diferitelor aplicații.
- 4. **Portabilitate:** Codul scris în Python este portabil și poate fi rulat pe diferite sisteme de operare, cum ar fi Windows, macOS și Linux, fără a necesita modificări semnificative.
- 5. **Comunitate puternică:** Python are o comunitate vastă și activă de dezvoltatori care contribuie cu cod open-source și oferă suport prin forumuri, grupuri de discuții și alte resurse online.

Avantaje Python

- ▶ **6. Integrare ușoară:** Python poate fi integrat cu ușurință cu alte limbaje de programare, permițând dezvoltatorilor să folosească module scrise în alte limbaje, cum ar fi C și C++, C#, Java, etc.
- ▶ **7. Productivitate ridicată:** Datorită sintaxei simple și numeroaselor biblioteci, dezvoltarea în Python este rapidă și eficientă, ceea ce conduce la o creștere a productivității dezvoltatorilor.
- ▶ **8. Orientat către soluții:** Python se concentrează pe oferirea soluțiilor la problemele reale, permițând dezvoltatorilor să scrie cod mai puțin și să rezolve sarcini complexe cu mai puțin efort.
- ▶ **9. Suport pentru programare orientată pe obiecte (POO):** Python permite dezvoltatorilor să utilizeze conceptele de programare orientată pe obiecte, facilitând astfel dezvoltarea de software modular, flexibil și ușor de întreținut.
- ▶ **10. Acesta este gratuit și open-source:** Python este disponibil gratuit și este distribuit sub licența Python Software Foundation License, ceea ce înseamnă că puteți folosi, modifica și distribui Python fără nicio taxă.

Dezavantajele Python

- ▶ 1. **Viteza de execuție:** Python este un limbaj interpretat. Acest lucru poate duce la o performanță mai lentă în comparație cu limbaje de programare compilate, cum ar fi C++ sau Java. Cu toate acestea, pentru multe aplicații, performanța Python este suficientă, iar în cazurile în care se necesită o viteză mai mare, pot fi utilizate extensii scrise în C sau C++ pentru a optimiza anumite părți ale codului.
- ▶ 2. **Gestionarea memoriei:** Python utilizează un sistem automat de gestionare a memoriei (garbage collection), ceea ce înseamnă că dezvoltatorul nu trebuie să se preocupe de alocarea și eliberarea memoriei. Cu toate acestea, acest sistem poate duce la o utilizare inefficientă a memoriei și la o performanță mai slabă în anumite situații.
- ▶ 3. **Compatibilitate versiuni anterioare:** Din cauza schimbărilor aduse în mod regulat în versiunile noi de Python, este posibil ca codul scris într-o versiune mai veche să nu fie complet compatibil cu versiunile mai noi, necesitând ajustări sau actualizări.

Dezavantajele Python

- ▶ 4. **Capacități limitate în aplicații mobile:** Deși există framework-uri care permit dezvoltarea de aplicații mobile în Python, cum ar fi Kivy și BeeWare, Python nu este atât de popular ca limbaj de programare pentru dezvoltarea de aplicații mobile în comparație cu Java (pentru Android) și Swift (pentru iOS).
- ▶ 5. **Consumul de resurse:** Python poate consuma mai multe resurse în comparație cu alte limbaje, cum ar fi memoria și puterea de procesare. Acest lucru poate fi o problemă în cazul aplicațiilor care trebuie să ruleze pe dispozitive cu resurse limitate.
- ▶ 6. **GIL (Global Interpreter Lock):** Python are un GIL, care este un mecanism de sincronizare care permite un singur fir de execuție să ruleze la un moment dat într-un proces Python. Acest lucru poate restricționa performanța paralelismului în aplicațiile de calcul intens.

Medii de dezvoltare Python

- Există mai multe medii de dezvoltare (IDE - Integrated Development Environments) și editoare de text care sunt folosite în mod obișnuit pentru dezvoltarea în Python.
 - **PyCharm:** Dezvoltat de JetBrains, PyCharm este un IDE puternic, complet echipat și dedicat Python. Acesta oferă o serie de caracteristici avansate, cum ar fi completarea automată, depanarea (debugging), analiza statică, gestionarea mediilor virtuale și integrarea cu sistemul de control al versiunii. Versiunea Community este gratuită, iar versiunea Professional oferă mai multe funcționalități avansate.
 - **Visual Studio Code (VS Code):** Este un editor de text gratuit dezvoltat de Microsoft, care are un suport excelent pentru Python prin intermediul unor extensii. Deși nu este un IDE dedicat exclusiv Python, mulți dezvoltatori Python preferă să lucreze cu VS Code datorită flexibilității, performanței și ecosistemului său bogat de extensii.
 - **Jupyter Notebook:** Este un mediu interactiv bazat pe web care permite crearea și partajarea de documente care conțin cod Python, grafice și texte explicative. Jupyter Notebook este adesea utilizat în analiza datelor, machine learning și în domeniul științei datelor.

Medii de dezvoltare Python

- **IDLE:** Acesta este un mediu de dezvoltare inclus în pachetul standard al Python. Este simplu și ușor de folosit, oferind facilități de bază pentru scrierea și rularea de cod Python.
- **Spyder:** Este un IDE specializat în analiza datelor și machine learning, construit folosind Python și numeroase biblioteci populare din ecosistemul științei datelor.
- **Sublime Text:** Este un editor de text rapid, ușor și personalizabil care este adesea folosit pentru dezvoltarea în Python cu ajutorul unor extensii disponibile.
- **Atom:** Este un editor de text open-source, flexibil și personalizabil, care oferă suport pentru Python prin intermediul unor extensii.
- **Google Colab** este o platformă interactivă de programare în Python, ce funcționează direct în browser, fără a necesita nicio configurare pe dispozitivul local. Singura cerință pentru utilizarea Colab este să aveți un cont Google

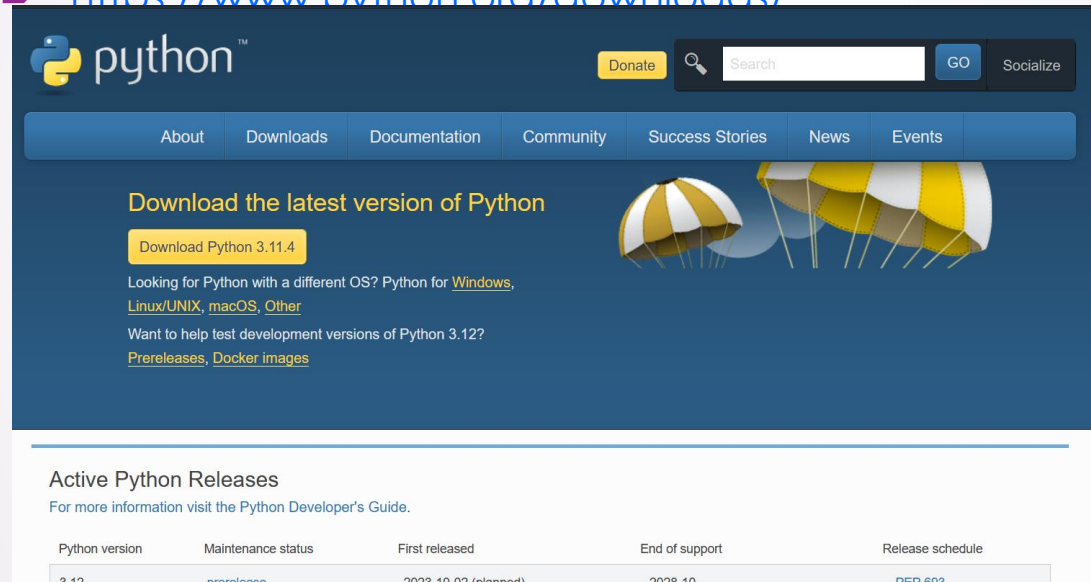
Acestea sunt doar câteva dintre cele mai folosite medii de dezvoltare și editoare de text pentru Python.

Mai puteți vedea și altele aici:
<https://wiki.python.org/moin/IntegratedDevelopmentEnvironments>

Instalare PYTHON

► De la link-ul:




► <https://www.python.org/downloads/>



► Pentru alte sisteme de operare puteți urmări ghidul de la următorul link:

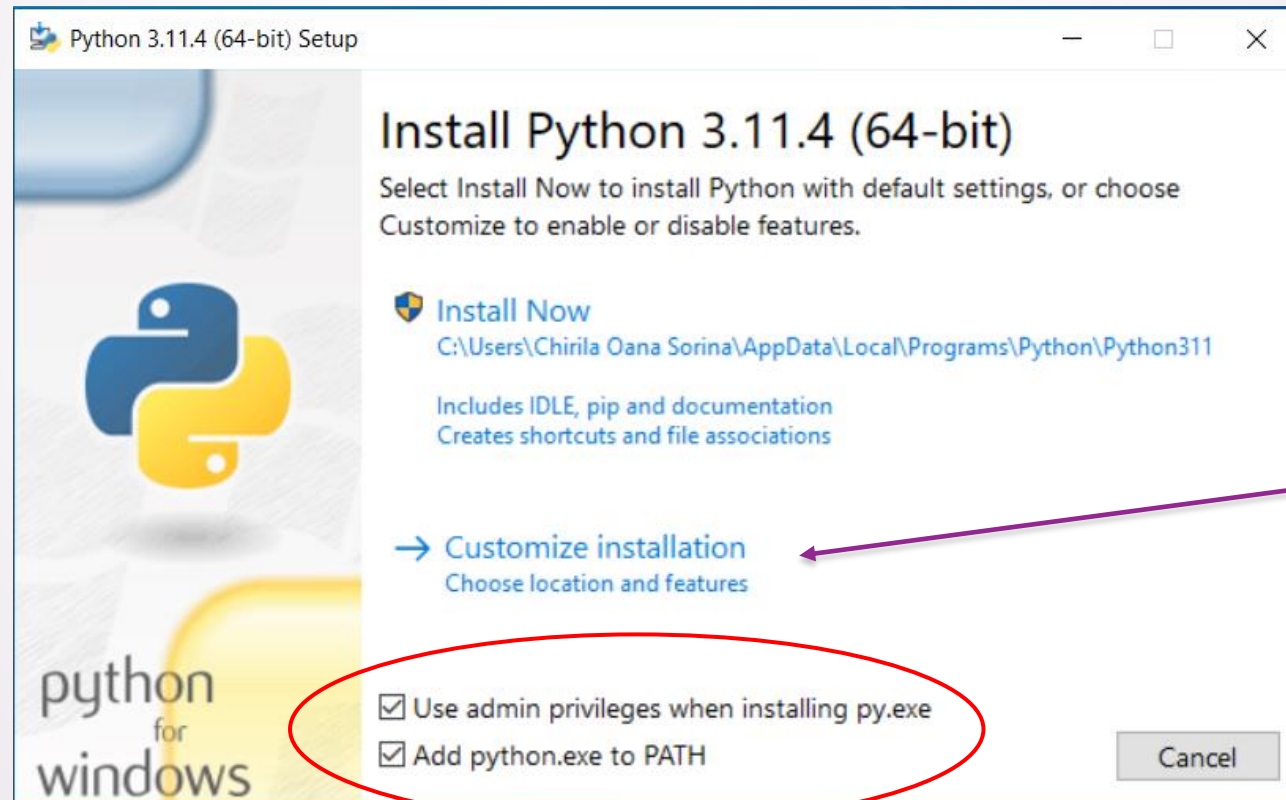
► <https://wiki.python.org/moin/BeginnersGuide/Download>

Instalare PYTHON

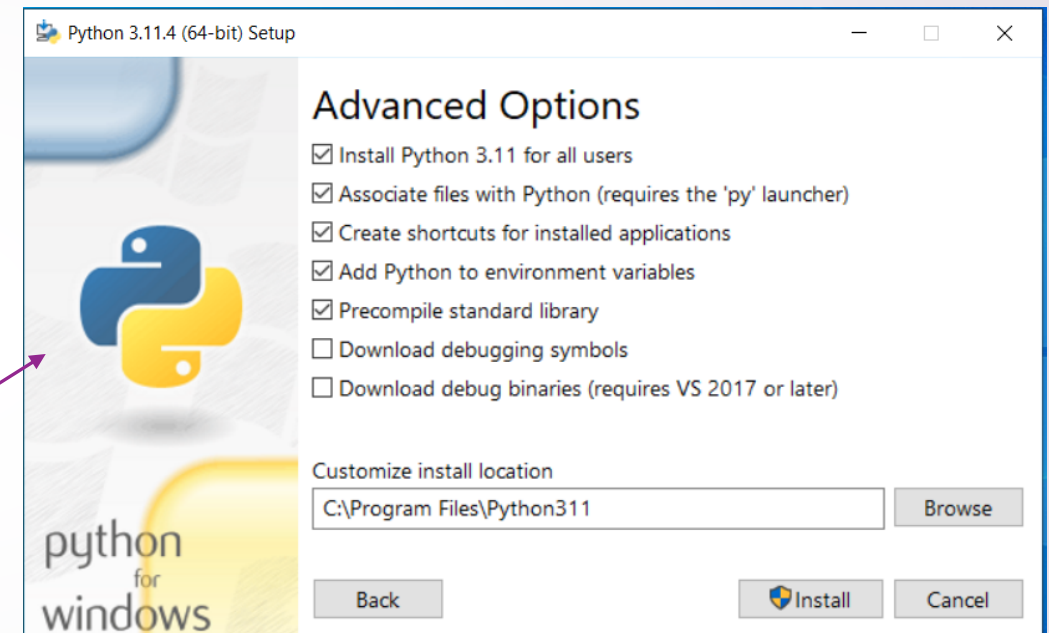
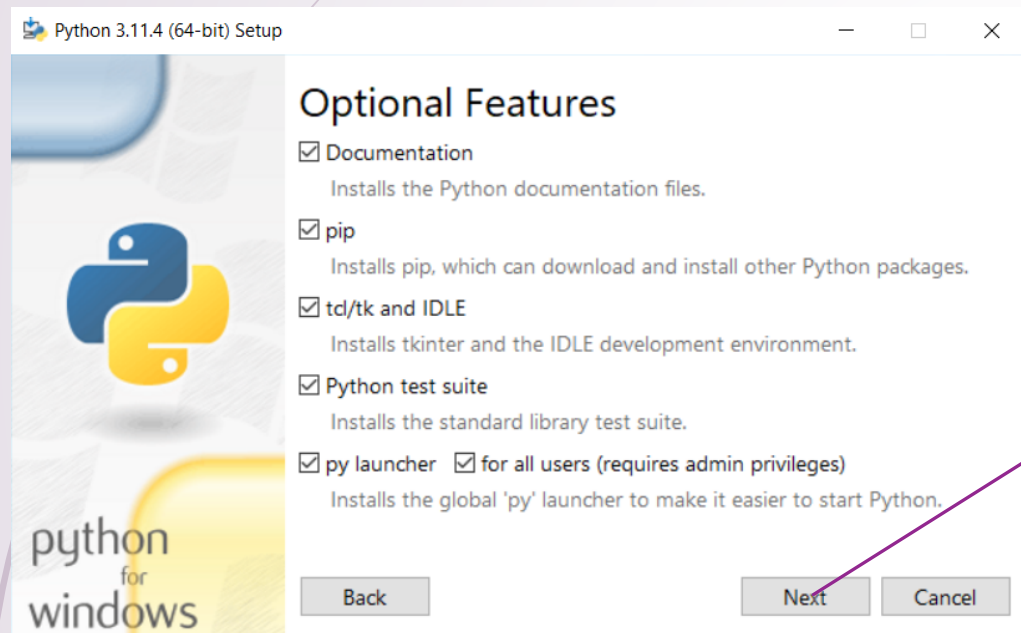
 https://www.python.org/downloads/release/python-3114/  						
<h2>Files</h2>						
Version	Operating System	Description	MD5 Sum	File Size	GPG	Sigstore
Gzipped source tarball	Source release		bf6ec50f2f3bfa6ffbdb385286f2c628	26526163	SIG	.sigstore
XZ compressed source tarball	Source release		fb7f7eae520285788449d569e45b6718	19954828	SIG	.sigstore
macOS 64-bit universal2 installer	macOS	for macOS 10.9 and later	91498b67b9c4b5ef33d1b7327e401b17	43120982	SIG	.sigstore
Windows embeddable package (32-bit)	Windows		81b0acfcdd31a73d1577d6e977acbd6	9596761	SIG	.sigstore
Windows embeddable package (64-bit)	Windows		d0e85bf50d2adea597c40ee28e774081	10591509	SIG	.sigstore
Windows embeddable package (ARM64)	Windows		bdce328de19973012123dc62c1cfa7e9	9965162	SIG	.sigstore
Windows installer (32-bit)	Windows		9ec180db64c074e57bdcca8374e9ded6	24238000	SIG	.sigstore
Windows installer (64-bit)	Windows	Recommended	e4413bb7448cd13b437dffffba294ca0	25426160	SIG	.sigstore
Windows installer (ARM64)	Windows	Experimental	60785673d37c754ddceb5788b5e5baa9	24714240	SIG	.sigstore

Instalare PYTHON

- Se instalează Python, iar la prima fereastră care apare pe ecran se vor bifa **ambele check-box-uri (Figura de mai sus)**.



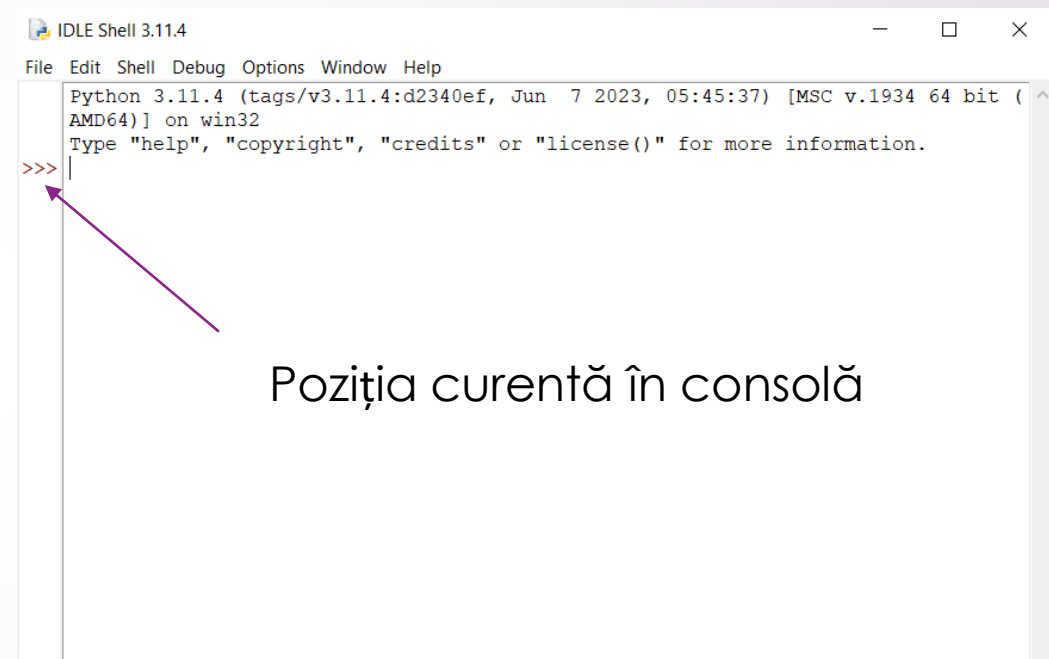
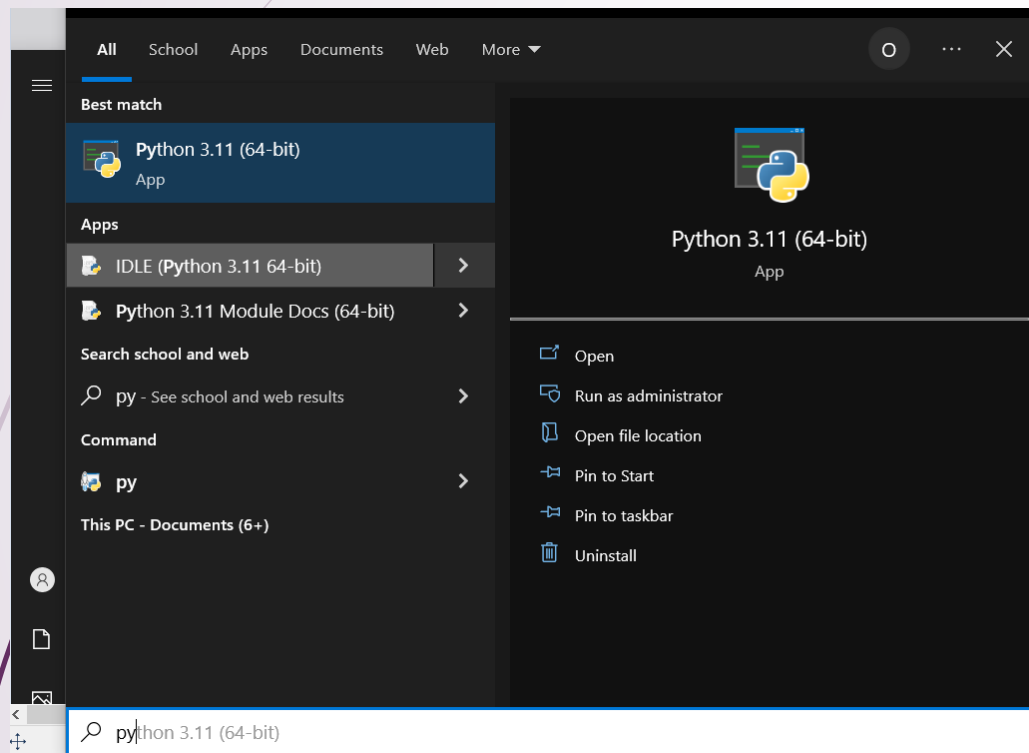
Instalare PYTHON



Lăsați totul bifat în prima fereastră, iar în a doua bifați opțiunea **Install for all users**. Astfel, locația de instalare va fi în „C:\Program Files\Python311”, care va fi accesibilă tuturor utilizatorilor.

Apăsați butonul **Install**.

Python IDLE



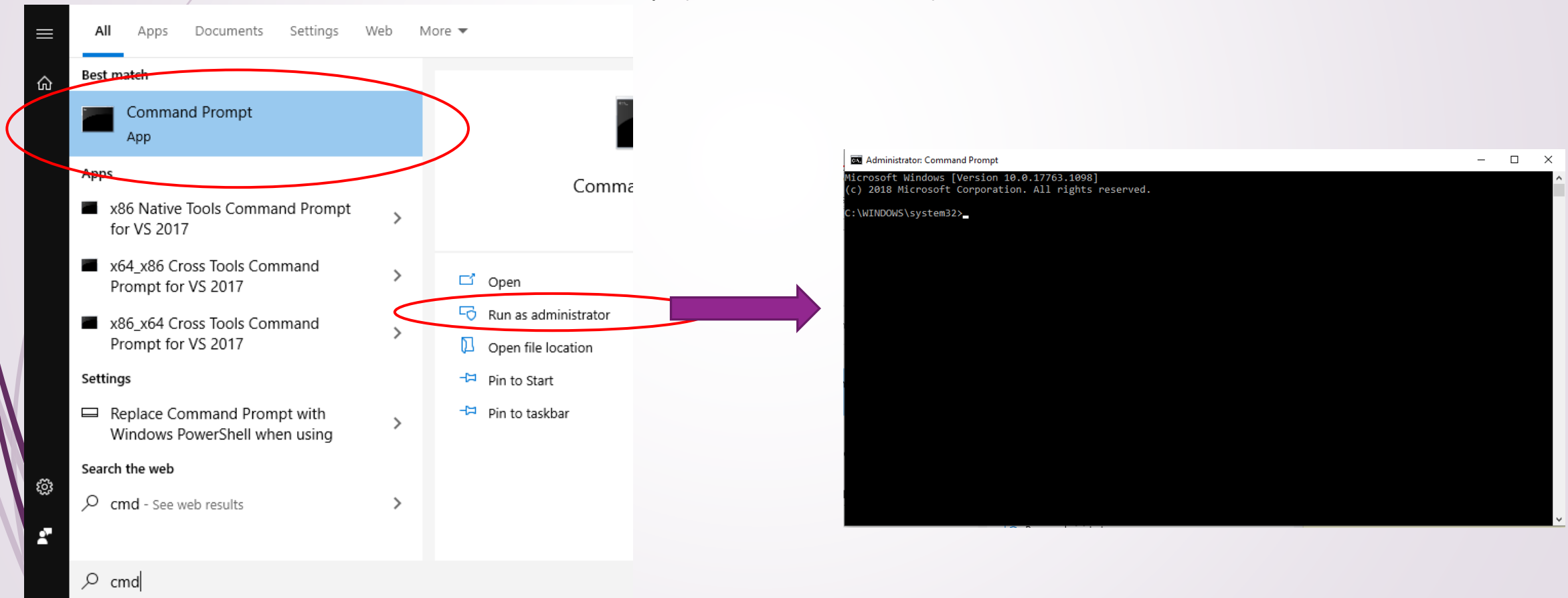
Poziția curentă în consolă

IDLE (Integrated Development and Learning Environment) - un mediu integrat de dezvoltare și învățare.

În fereastra din dreapta avem consola (Interactive Python Shell) în care putem scrie cod python

Instalare biblioteci necesare

- Se deschide command prompt din Windows:
- Start->Command Prompt (Run as administrator)

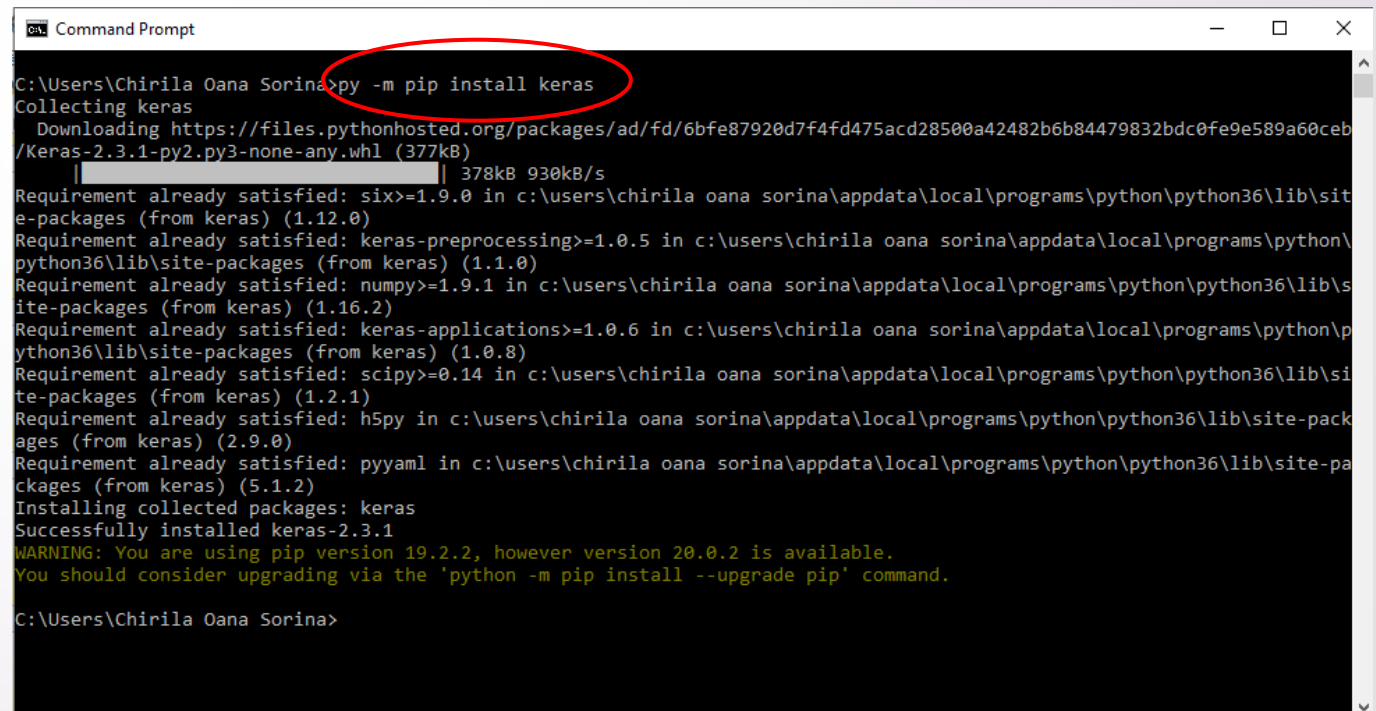


Instalare biblioteci necesare

- În command prompt se vor executa comenzile de instalare pentru diferite biblioteci de care vom avea nevoie sub următoarea formă:
- **py -m pip install nume_bibliotecă // pip install nume_biblioteca**

ex: Pentru instalarea bibliotecii *keras*:

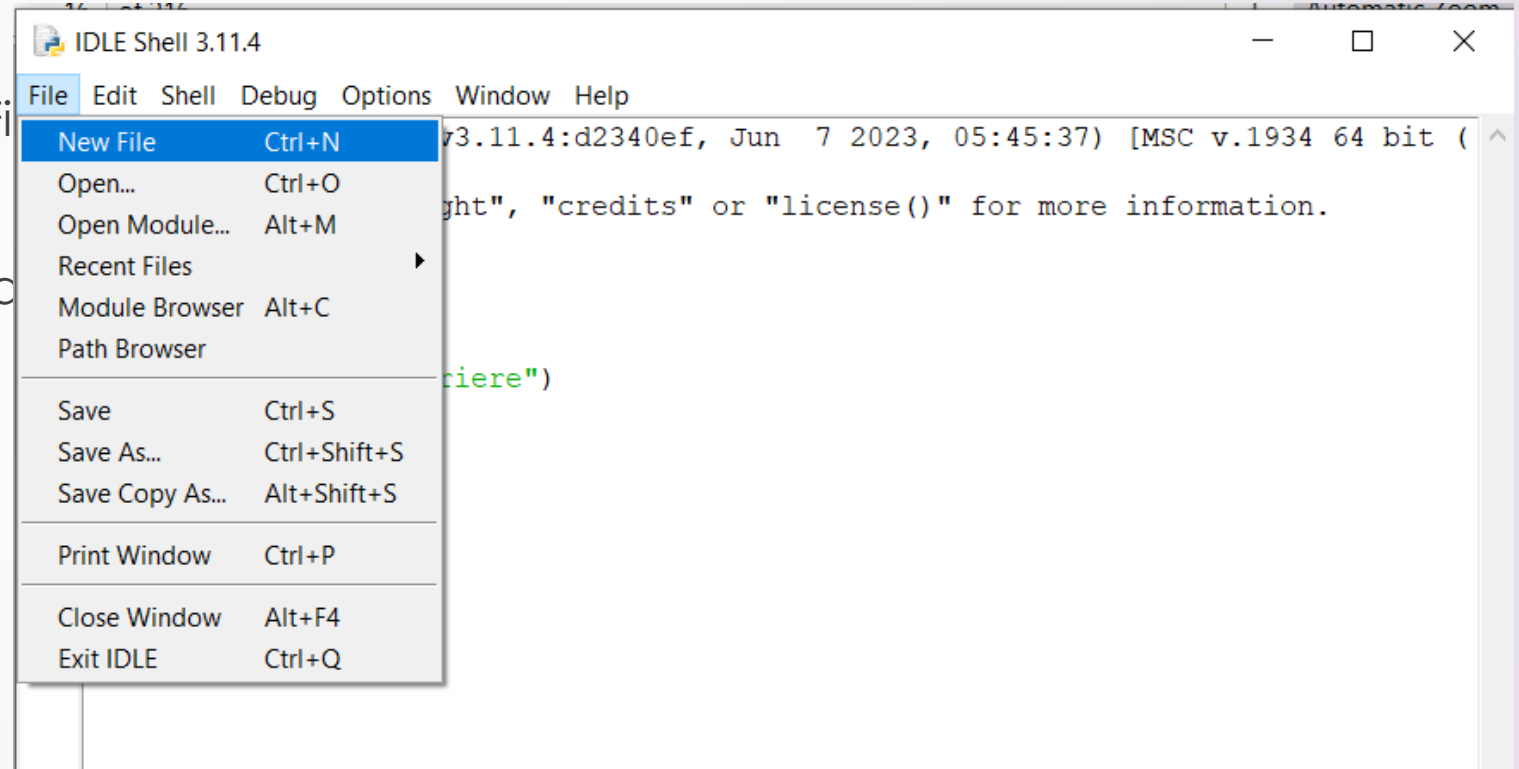
py -m pip install keras



```
Command Prompt
C:\Users\Chirila Oana Sorina>py -m pip install keras
Collecting keras
  Downloading https://files.pythonhosted.org/packages/ad/fd/6bfe87920d7f4fd475acd28500a42482b6b84479832bdc0fe9e589a60ceb/Keras-2.3.1-py2.py3-none-any.whl (377kB)
    | 378kB 930kB/s
Requirement already satisfied: six>=1.9.0 in c:\users\chirila oana sorina\appdata\local\programs\python\python36\lib\site-packages (from keras) (1.12.0)
Requirement already satisfied: keras-preprocessing>=1.0.5 in c:\users\chirila oana sorina\appdata\local\programs\python\python36\lib\site-packages (from keras) (1.1.0)
Requirement already satisfied: numpy>=1.9.1 in c:\users\chirila oana sorina\appdata\local\programs\python\python36\lib\site-packages (from keras) (1.16.2)
Requirement already satisfied: keras-applications>=1.0.6 in c:\users\chirila oana sorina\appdata\local\programs\python\python36\lib\site-packages (from keras) (1.0.8)
Requirement already satisfied: scipy>=0.14 in c:\users\chirila oana sorina\appdata\local\programs\python\python36\lib\site-packages (from keras) (1.2.1)
Requirement already satisfied: h5py in c:\users\chirila oana sorina\appdata\local\programs\python\python36\lib\site-packages (from keras) (2.9.0)
Requirement already satisfied: pyyaml in c:\users\chirila oana sorina\appdata\local\programs\python\python36\lib\site-packages (from keras) (5.1.2)
Installing collected packages: keras
Successfully installed keras-2.3.1
WARNING: You are using pip version 19.2.2, however version 20.0.2 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.
C:\Users\Chirila Oana Sorina>
```

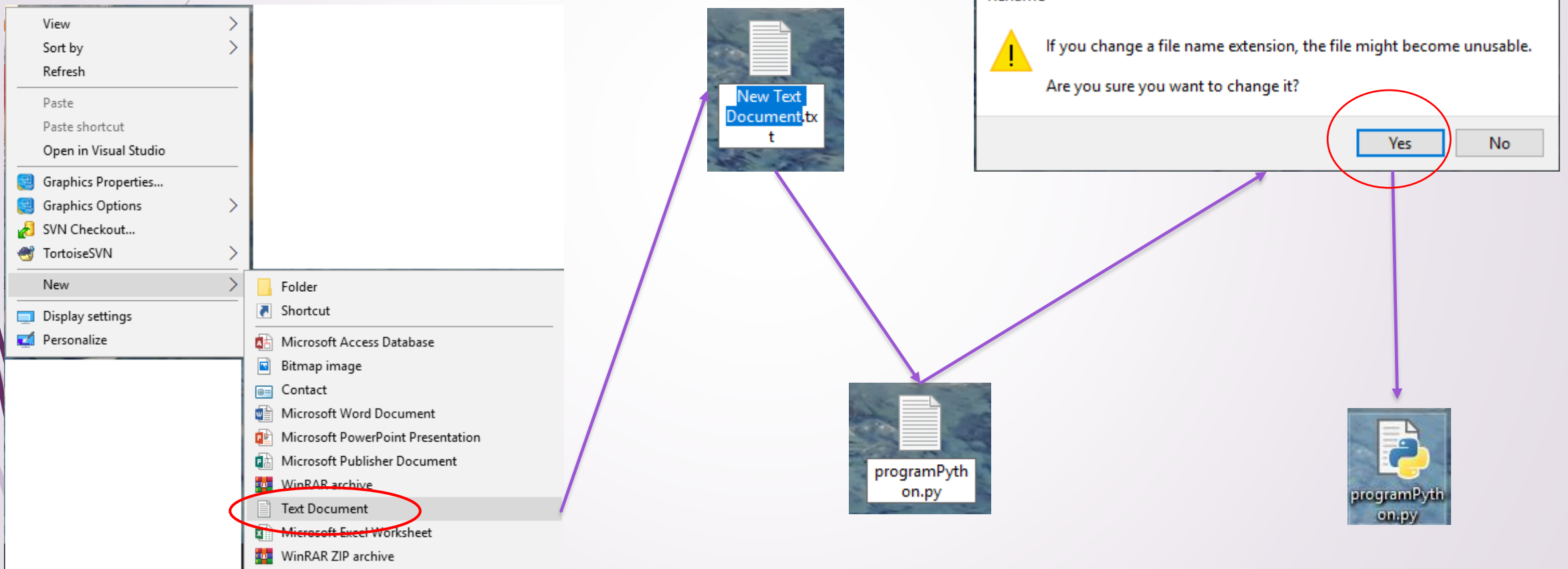
Crearea unui fisier Python (Metoda 1)

- - File -> New File
- - Ctrl+N
- - se va deschide un editor
- în care putem scrie



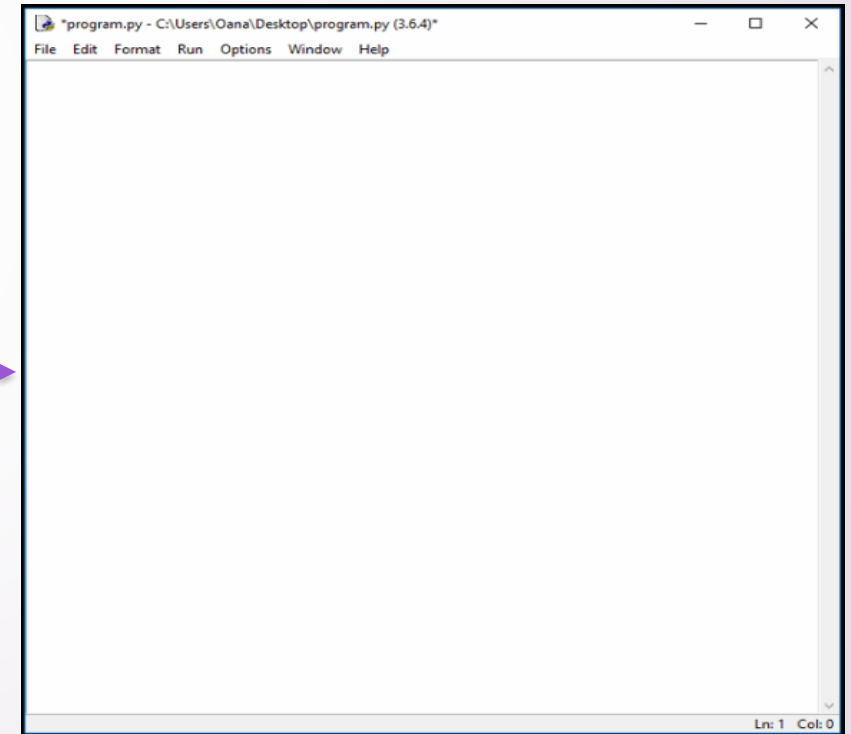
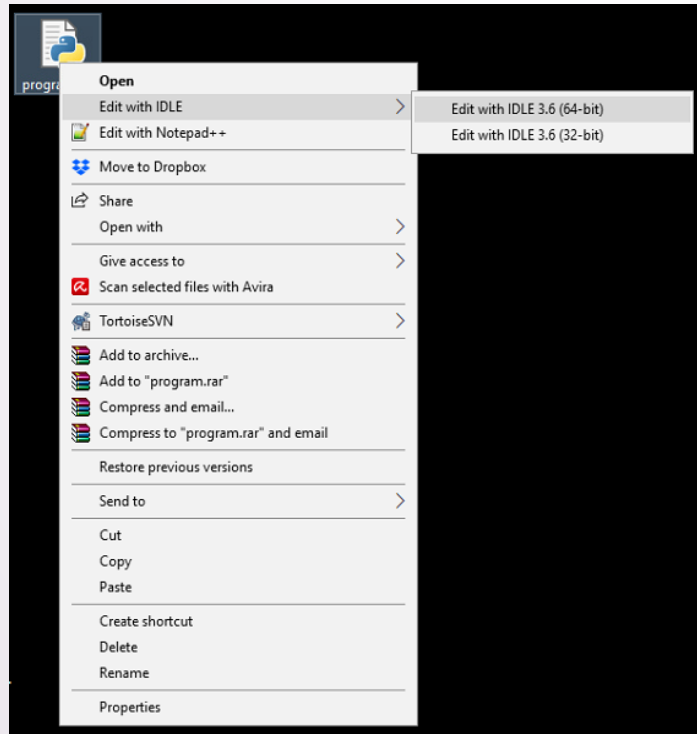
Crearea unui nou fișier PYTHON (Metoda 2)

- Pentru a crea un nou fișier python, se crează un nou fișier text cu **extensia .py**



Deschiderea fișierului Python pentru editare

- Pentru a deschide fișierul nou creat se va executa ***clic dreapta -> Edit with IDLE -> Edit with IDLE 3.6, 3.7***, sau ce versiune aveți instalată:







Instalare Python în Visual Studio





Modifying — Visual Studio Community 2022 — 17.4.5

Workloads Individual components Language packs Installation locations

Web & Cloud (4)






-  **ASP.NET and web development** ☒
Build web applications using ASP.NET Core, ASP.NET, HTML/JavaScript, and Containers including Docker supp...
-  **Python development** ☒
Editing, debugging, interactive development and source control for Python.
-  **Azure development**
Azure SDKs, tools, and creating resources
-  **Node.js development**
Build scalable network asynchronous event-driven applications

Desktop & Mobile (5)

-  **.NET Multi-platform App UI development** ☐
Build Android, iOS, Windows, and Mac apps from a single codebase using C# with .NET MAUI.
-  **.NET desktop development**
Build WPF, Windows Forms, and Universal Windows Platform apps using C#, Visual Basic, and F#
-  **Desktop development with C++** ☐
Build modern C++ apps for Windows using tools of your choice including MSVC, Clang, CMake, or MSBuild
-  **Universal Windows Platform development**
Create applications for Windows 10 and Windows 11 with C# or VB

Location
C:\Program Files\Microsoft Visual Studio\2022\Community

Python All platforms All project types

-  **Python Application**
A project for creating a command-line application
Python Windows Linux macOS Console
-  **Classifier Project (cookiecutter)**
A project for testing scikit-learn classifiers.
This project requires the pandas, numpy, scikit-learn, and matplotlib packages.
Opens Cookiecutter window where you can create the project from a cookiecutter template available online.
Python Windows Machine Learning
-  **Clustering Project (cookiecutter)**
A project for testing scikit-learn clustering learners.
This project requires the pandas, numpy, scikit-learn, and matplotlib packages.
Opens Cookiecutter window where you can create the project from a cookiecutter template available online.
Python Windows Machine Learning
-  **Regression Project (cookiecutter)**
A project for testing scikit-learn regression learners.
This project requires the pandas, numpy, scikit-learn, and matplotlib packages.
Opens Cookiecutter window where you can create the project from a cookiecutter template available online.
Python Windows Machine Learning
-  **From Existing Python code**
Create a new project using code files that are already in a folder hierarchy
Python Linux macOS Windows Console Web

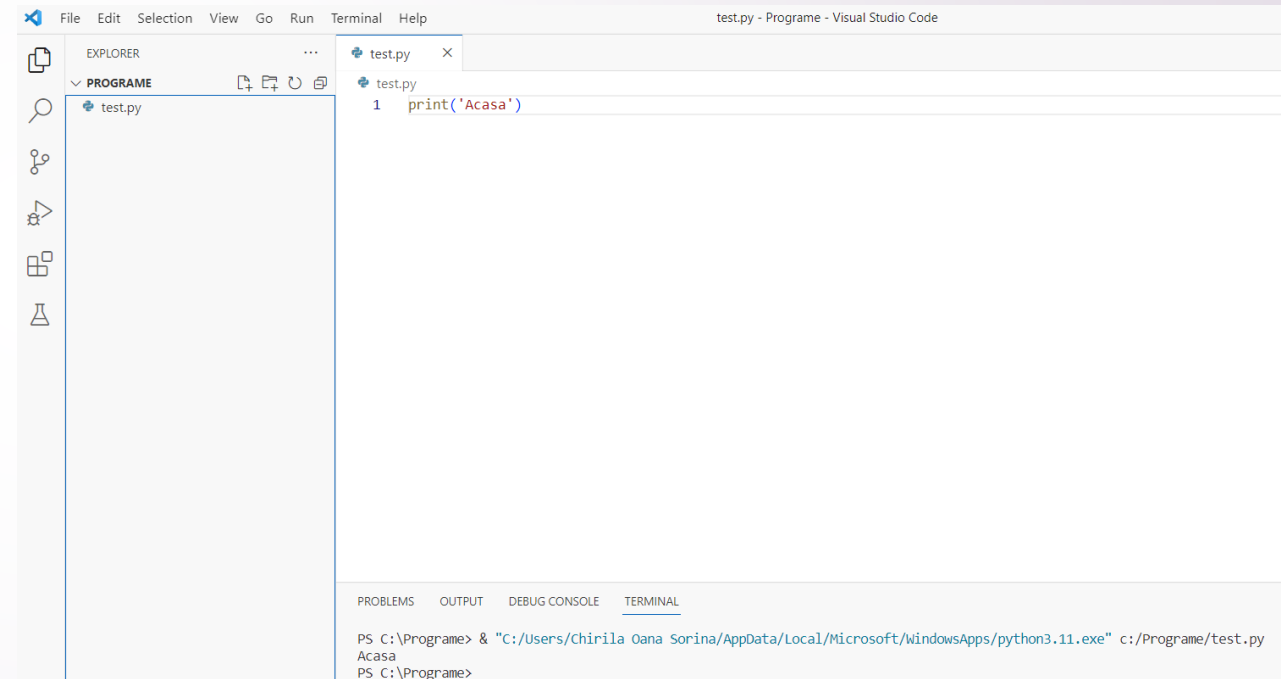
Instalare Python în VSCode

► <https://code.visualstudio.com/docs/python/python-tutorial>

► Pentru a crea aplicații Python cu Visual Studio Code trebuie să instalați:

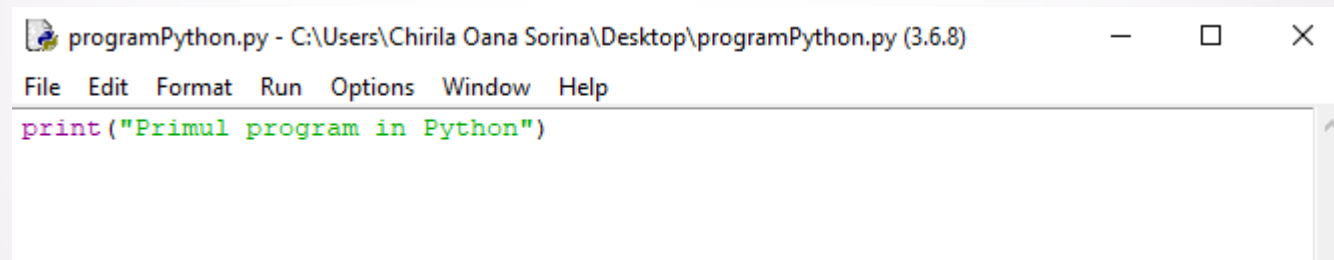
- Python 3
- VS Code
- VS Code Python extension

După instalarea VSCode și Python vom crea un nou proiect VSCode, vom alege un folder în care vom crea aplicațiile și vom crea fișierele .py în care se va scrie codul sursă.



Primul program în Python

- În fereastra de editare deschisă vom scrie următorul cod sursă care ne va afișa pe ecran un text:

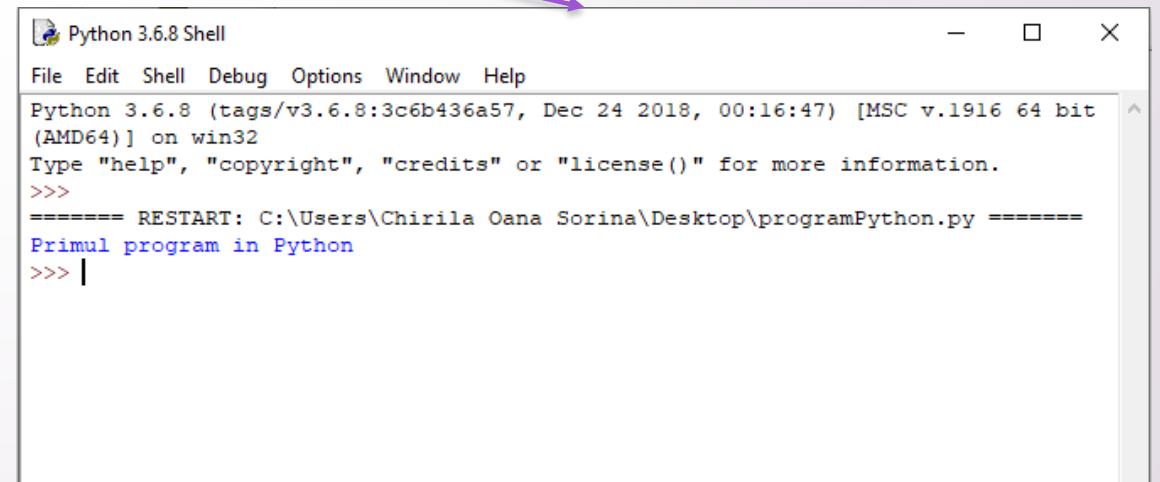
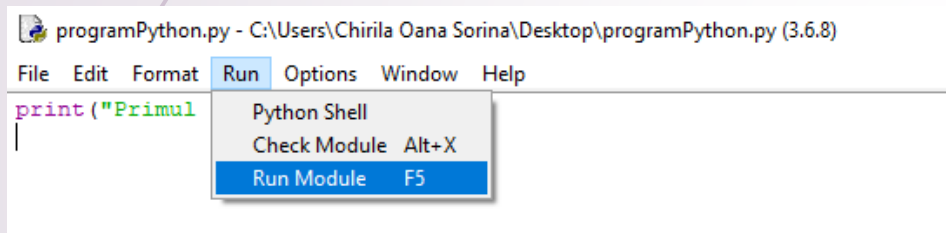


The image shows a screenshot of a Python IDE window. The title bar reads "programPython.py - C:\Users\Chirila Oana Sorina\Desktop\programPython.py (3.6.8)". The menu bar includes "File", "Edit", "Format", "Run", "Options", "Window", and "Help". The code editor contains a single line of Python code: `print("Primul program in Python")`. The code is color-coded: `print` is purple, the opening parenthesis is green, the string is in red, and the closing parenthesis is green.

```
print("Primul program in Python")
```

Rularea unui program Python

- Pentru a rula un program în Python:
- ***Run - > Run Module***



Python virtual environment (venv)


Aplicațiile create de dezvoltatorii de software necesită adesea instalarea unui ansamblu de module externe suplimentare.

Python oferă conceptul de 'Python virtual environment' sau mediu virtual Python.

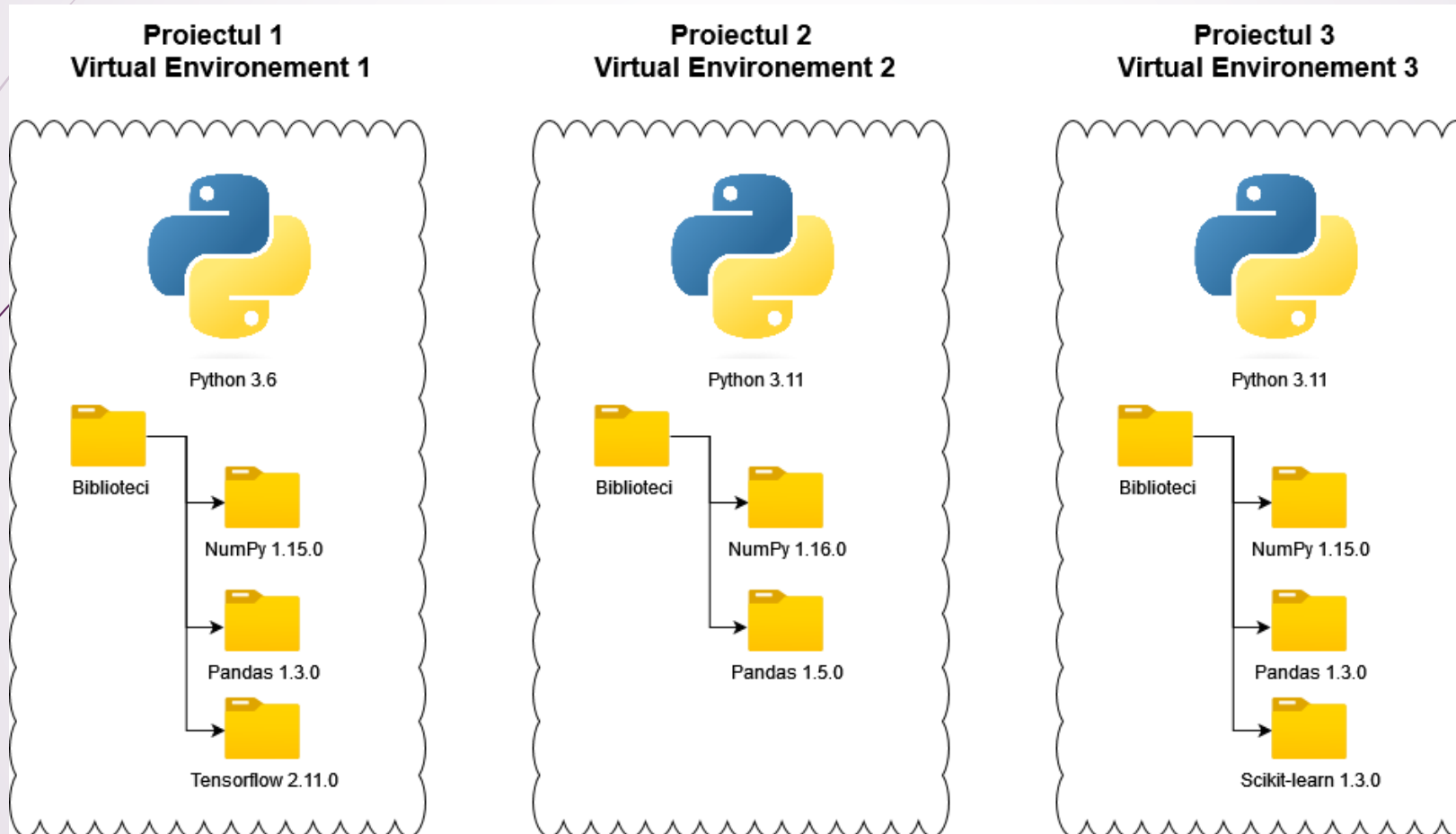




Python virtual environment (venv)

- - Library 1.0.0
 - - Library 1.2.0
 - - Library 1.2.4
 - - Instalăm biblioteca...vom avea versiunea 1.2.4
 - - Library 1.2.0
 - - Library 1.3.2
 - - Instață biblioteca, se va actualiza la versiunea 1.3.2
- 

Python virtual environment (venv)



Python virtual environment (venv) - Exemplu

- Pentru crearea mediului virtual se pot folosi 2 metode:
- 1. **venv** este un modul încorporat care vine împreună cu Python 3.3 și versiunile ulterioare. Este ușor de utilizat și creează medii virtuale pentru proiectele Python. venv este folosit prin executarea următoarei comenzi în terminal:

- `python -m venv NumeProiect`

După crearea mediului virtual acesta trebuie activat prin comanda:

- `NumeProiect\Scripts\activate.bat`

Pentru a dezactiva mediul virtual folosim comanda:

- `deactivate.bat`

```
Command Prompt
Microsoft Windows [Version 10.0.19044.2965]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Chirila Oana Sorina>cd..

C:\Users>cd..

C:\>cd programe

C:\Program>cd proiect1
The system cannot find the path specified.

C:\Program>cd virtualenv

C:\Program\VirtualEnv>python -m venv NumeProiect

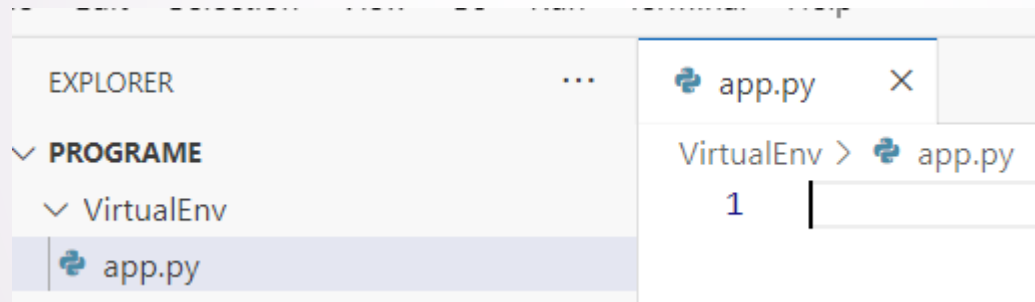
C:\Program\VirtualEnv>NumeProiect\Scripts\activate.bat

(NumeProiect) C:\Program\VirtualEnv>deactivate.bat
C:\Program\VirtualEnv>
```

Python virtual environment (Virtualenv)

- Exemplu

- 2. **virtualenv** este un pachet extern care oferă funcționalități similare cu venv. Este compatibil atât cu Python 2, cât și cu 3, și poate fi instalat folosind pip:
 - `pip install virtualenv`
- Creem un folder VirtualEnv în care adăugăm un fișier .py



- În linia de comandă vom scrie următoarea comandă pentru a crea mediul virtual:
 - `virtualenv NumeProiect`

Python virtual environment (Virtualenv)

- Exemplu

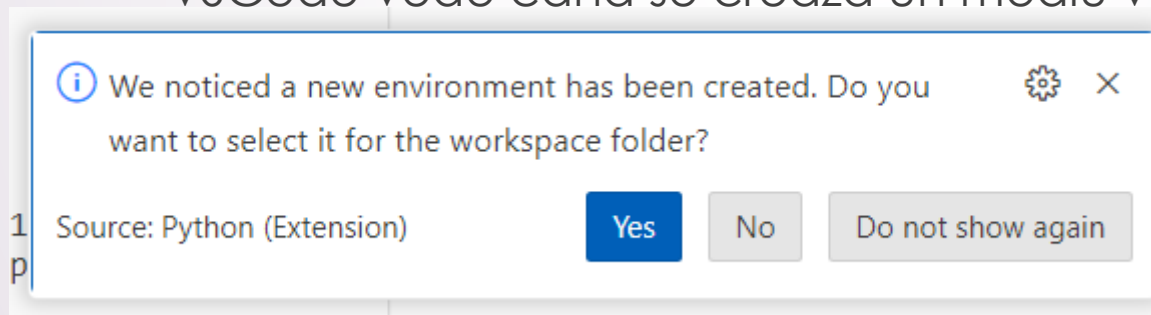
► Pentru a-l activa vom folosi comanda:

- Scripts\activate.bat

Pentru a-l dezactiva vom folosi comanda:

- deactivate.bat

VSCode vede când se crează un mediu virtual și dă posibilitatea de a-l folosi



Python virtual environment (Virtualenv)

- Exemplu

```
Select Command Prompt

C:\Users\Chirila Oana Sorina>cd\

C:\>cd programe

C:\Programe>virtualenv VirtualEnv
created virtual environment CPython3.11.4.final.0-64 in 619ms
  creator CPython3Windows(dest=C:\Programe\VirtualEnv, clear=False, no_vcs_ignore=False, global=False)
  seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy, app_data_dir=C:\Users\Chirila Oana Sorina\AppData\Local\pypa\virtualenv)
  added seed packages: pip==23.2.1, setuptools==68.0.0, wheel==0.41.0
  activators BashActivator,BatchActivator,FishActivator,NushellActivator,PowerShellActivator,PythonActivator

C:\Programe>cd VirtualEnv

C:\Programe\VirtualEnv>Scripts\activate.bat

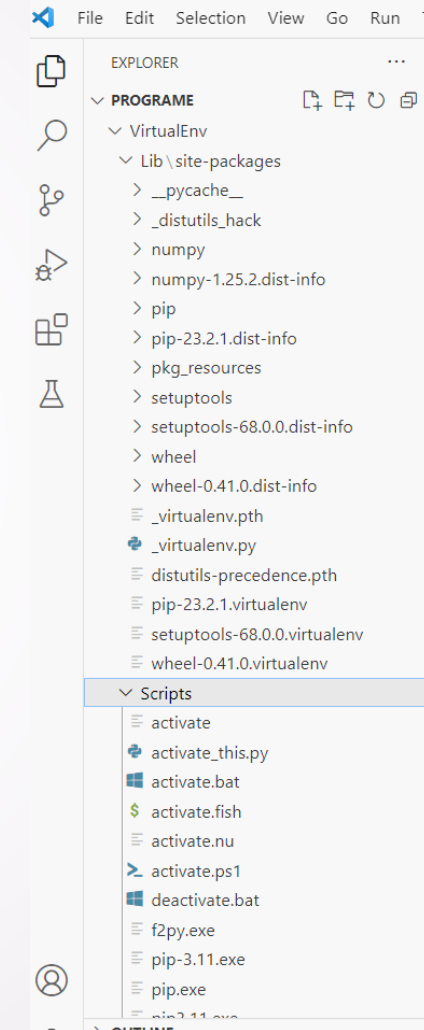
(VirtualEnv) C:\Programe\VirtualEnv>pip list
Package      Version
-----
pip          23.2.1
setuptools   68.0.0
wheel        0.41.0
```

```
Command Prompt

Successfully installed numpy-1.25.2

(VirtualEnv) C:\Programe\VirtualEnv>pip list
Package      Version
-----
numpy        1.25.2
pip          23.2.1
setuptools   68.0.0
wheel        0.41.0

(VirtualEnv) C:\Programe\VirtualEnv>deactivate
C:\Programe\VirtualEnv>
```



Python virtual environment

- Pentru a distribui o aplicație, trebuie să specificăm setul de pachete și versiunile acestora necesare pentru a rula aplicația. În cazul utilizării unui mediu virtual, această listă de pachete poate fi obținută foarte ușor folosind următoarea comandă:

- `pip freeze > requirements.txt`

Ulterior, destinatarul final al aplicației poate instala automat toate aceste pachete prin implementarea listei de module specificată în fișierul creat:

- `pip install -r requirements.txt`

```
(VirtualEnv) C:\Programe\VirtualEnv>pip freeze
```

```
numpy==1.25.2
```

```
(VirtualEnv) C:\Programe\VirtualEnv>pip freeze > requirements.txt
```


Comentarii în Python

- În limbajul de programare Python, comentariile pot fi adăugate pentru a explica codul sau pentru a face anumite note pentru cei care citesc codul. Există două tipuri principale de comentarii în Python:
 1. Comentarii pe o singură linie: Acestea sunt utilizate pentru a adăuga comentarii scurte pe o singură linie.
 2. Comentarii pe mai multe linii (comentarii de bloc): Acestea sunt utilizate pentru a adăuga comentarii care se întind pe mai multe linii.

```
# Acesta este un comentariu pe o singură linie
```

```
"""
```

```
Acesta este un comentariu  
pe mai multe linii.  
Poate fi folosit pentru a documenta  
mai detaliat un bloc de cod.
```

```
"""
```

Variabile în Python

- ▶ Variabilele în Python nu au nevoie de declarație explicită pentru a rezerva spațiu de memorie.
- ▶ Declarația se întâmplă automat atunci când se atribuie o valoare unei variabile. Semnul egal (=) este utilizat pentru a atribui valori la variabile.
- ▶ Operandul din stânga operatorului = este numele variabilei, iar operandul din dreapta operatorului = este valoarea stocată în variabilă.
- ▶ De exemplu:

```
numar1 = 105           # atribuirea unei valori intregi  
numar2 = 1023.36       # o valoare reala  
nume = "Oana Sorina"   # un string
```

Variabile în Python

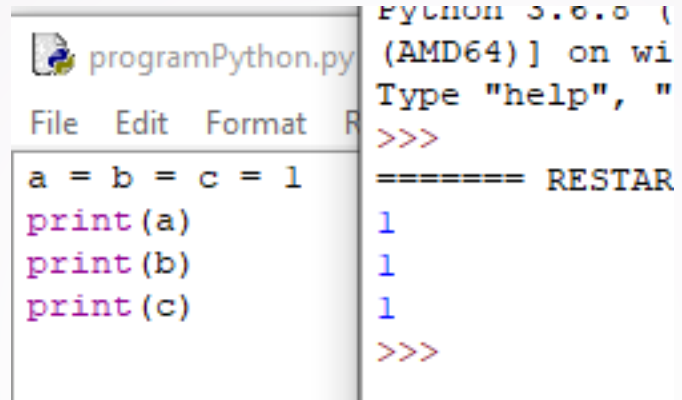
- Dacă vom afișa variabilele la care am asignat valori rezultatul va fi următorul:

programPython.py - C:\Users\O	Python 3.6.8 Shell
<pre>File Edit Format Run Option numar1 = 105 numar2 = 1023.36 nume = "Oana Sorina" print(numar1) print(numar2) print(nume)</pre>	<pre>File Edit Shell Debug Python 3.6.8 (tags (AMD64)] on win32 Type "help", "cop >>> ===== RESTART: (105 1023.36 Oana Sorina >>></pre>

Alocare multiplă

- Python vă permite să atribuiți o singură valoare mai multor variabile simultan.

- De exemplu



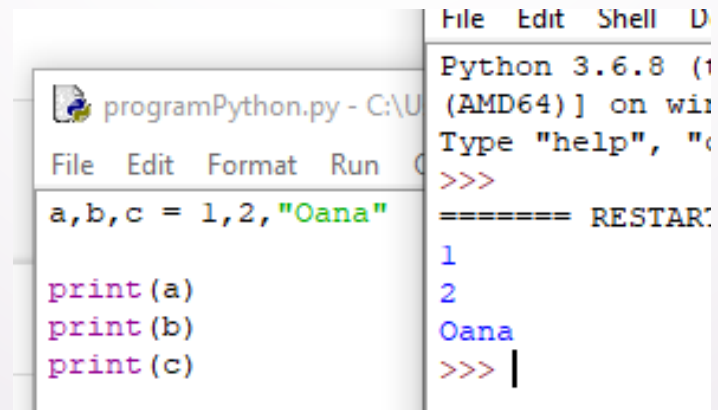
```
programPython.py
File Edit Format Run
a = b = c = 1
print(a)
print(b)
print(c)

Python 3.6.8 (
(AMD64)] on wi
Type "help", "
>>>
===== RESTAR
1
1
1
>>>
```

- Aici, este creat un obiect întreg cu valoarea 1 și toate cele trei variabile sunt atribuite aceleași locații de memorie.

- De asemenea, puteți atribui mai multe obiecte mai multor variabile.

- De exemplu



```
programPython.py - C:\U
File Edit Shell D
Python 3.6.8 (1
(AMD64)] on wi
Type "help", "(
>>>
===== RESTAR
1
2
Oana
>>> |
```

Tipuri de date standard

- ▶ Datele stocate în memorie pot fi de mai multe tipuri. De exemplu, vârsta unei persoane este stocată ca valoare numerică, iar adresa sa este stocată sub formă de caractere alfanumerice.
- ▶ Python are cinci tipuri de date standard:
 - Numere
 - Șir
 - Listă
 - Tuplu
 - Dicționar

Python Numbers

- ▶ Tipurile de date număr permit stocarea de valori numerice.
- ▶ Obiectele de tip număr sunt create atunci când li se atribuie o valoare.

- ▶ De exemplu: *valoare1=1*

- valoare2=200*

- Python acceptă patru tipuri numerice diferite:
 - **int** (numere întregi semnate)
 - **long** (întregi lungi, ele pot fi, de asemenea, reprezentate în octal și hexadecimal)
 - **float** (valorile reale)
 - **complex** (numere complexe)

Șiruri de caractere în Python

- Șirurile din Python sunt identificate ca un set contiguu de caractere reprezentate în ghilimele.
- Python permite perechi de ghilimele simple sau duble. Subșiruri de șiruri pot fi luate cu ajutorul operatorului [] și [:] începând cu poziția 0
- Semnul plus (+) este operatorul de concatenare șiruri, iar asteriscul (*) este operatorul de repetare.

► Exemplu

```
str = 'Informatica medicala'

print (str)           # Afiseaza tot sirul
print (str[0])        # Afiseaza primul caracter din sir
print (str[2:5])      # Afiseaza caracterele incepand de la caracterul 3 pana la caracterul 5
print (str[2:])       # Afiseaza caracterele incepand cu caracterul 3
print (str * 2)       # Afiseaza sirul de 2 ori
print (str + " in servicii de sanatate") # Afiseaza sirurile concatenate
```

```
----- RESTART: C:\Users\Chirila Dana Sorina
Informatica medicala
I
for
formatica medicala
Informatica medicalaInformatica medicala
Informatica medicala in servicii de sanatate
I
```

Listele în Python

- Listele sunt cele mai versatile dintre tipurile de date ale lui Python.
- O listă conține elemente separate prin virgule și închise între paranteze pătrate ([]). Într-o oarecare măsură, listele sunt similare cu tablourile din C. O diferență între ele este că toate elementele aparținând unei liste pot fi de tipul de date diferite.
- Valorile stocate într-o listă pot fi accesate utilizând operatorul ([]) și [:]) cu indexuri care încep de la 0.
- Semnul plus (+) este operatorul de concatenare a listei, iar asteriscul (*) este operatorul de repetare.

■ Exemplu

```
lista1 = [ 'abcd', 786 , 2.23, 'oana', 70.2 ]
lista2 = [123, 'oana']

print (lista1)           # Afiseaza toata lista
print (lista1[0])        # Afiseaza primul element din lista
print (lista1[1:3])      # Afiseaza elementele de la pozitia 2 pana la al 4-lea
print (lista1[2:])       # Afiseaza lista incepand cu elementul de pe pozitia 3
print (lista2 * 2)        # Afiseaza lista de 2 ori
print (lista1 + lista2)  # Afiseaza lista concatenata
```

```
----- RESTART: C:\Users\Chirila Oana\AppData\Local\Microsoft\Windows\Shell\
['abcd', 786, 2.23, 'oana', 70.2]
abcd
[786, 2.23]
[2.23, 'oana', 70.2]
[123, 'oana', 123, 'oana']
['abcd', 786, 2.23, 'oana', 70.2, 123, 'oana']
>>> |
```

Tupluri în Python

- Un tuple este un alt tip de date similar cu lista. Un tuple constă dintr-un număr de valori separate prin virgule. Spre deosebire de liste, tuplurile sunt incluse în paranteze.
- Principalele diferențe între liste și tuple sunt:
 - Listele sunt închise între paranteze [] și elementele și dimensiunea lor pot fi schimbate
 - tuplurile sunt închise între paranteze () și nu pot fi actualizate.
 - Tuplurile pot conține doar elemente în citire (read-only)

Exemplu

```
tuple1 = ( 'abcd', 786 , 2.23, 'oana', 70.2 )
tuple2 = (123, 'oana')

print (tuple1)           # Afiseaza tot tuplul
print (tuple1[0])        # Afiseaza primul element din tuplu
print (tuple1[1:3])      # Afiseaza elementele de la pozitia 2 pana la al 4-lea
print (tuple1[2:])       # Afiseaza tuple incepand cu elementul de pe pozitia 3
print (tuple2 * 2)        # Afiseaza tuple de 2 ori
print (tuple1 + tuple2)  # Afiseaza tuple concatenata
```

```
( 'abcd', 786, 2.23, 'oana', 70.2)
abcd
(786, 2.23)
(2.23, 'oana', 70.2)
(123, 'oana', 123, 'oana')
('abcd', 786, 2.23, 'oana', 70.2, 123, 'oana')
>>>
```

Dictionar în Python

- Dicționarele din Python sunt un tip de tabel hash. Ele consta din perechi cheie-valoare. O cheie de dicționar poate fi aproape orice tip de date Python, dar de obicei sunt numere sau șiruri. Pe de altă parte, valorile pot fi orice obiect Python arbitrar.
- Dicționarele sunt închise cu acolade {}, iar valorile pot fi atribuite și accesate folosind parantezele pătrate [].

Exemplu

```
dictionar1 = {}  
dictionar1['unu'] = "Acesta este unu"  
dictionar1[2] = "Acesta este 2"  
  
dictionar2 = {'nume': 'Oana', 'cod': 6734, 'departament': 'AIA'}  
  
print (dictionar1['unu'])      # Afiseaza valoarea pentru cheia ['unu']  
print (dictionar1[2])         # Afiseaza valoarea pentru cheia 2  
print (dictionar2)            # Afiseaza tot dictionarul  
print (dictionar2.keys())     # Afiseaza toate cheile  
print (dictionar2.values())   # Afiseaza toate valorile
```

```
Acesta este unu  
Acesta este 2  
{'nume': 'Oana', 'cod': 6734, 'departament': 'AIA'}  
dict_keys(['nume', 'cod', 'departament'])  
dict_values(['Oana', 6734, 'AIA'])  
>>> [
```

Conversii ale tipurilor de date

- Uneori, poate fi necesar să efectuați conversii între tipurile de date. Pentru a converti între tipuri, utilizați pur și simplu numele tipului ca funcție.
- Există mai multe funcții pentru a efectua conversia de la un tip de date la altul. Aceste funcții returnează un obiect nou reprezentând valoarea convertită.

Function & Description
int(x [,base]) Converts x to an integer. base specifies the base if x is a string.
long(x [,base]) Converts x to a long integer. base specifies the base if x is a string.
float(x) Converts x to a floating-point number.
complex(real [,imag]) Creates a complex number.
str(x) Converts object x to a string representation.

Etc.

Tipul variabilelor

- Pentru a putea vedea tipul unei variabile vom apela funcția **type()** pe numele variabilei:

```
a=10
b = {'nume': 'Oana', 'cod':6734, 'departament': 'AIA'}
c=(1,2,3)
d=23.56
e="Oana"
```

```
print (type(a))
print (type(b))
print (type(c))
print (type(d))
print (type(e))
```

Python 3.6.8 Shell

File Edit Shell Debug Options Window Help

Python 3.6.8 (tags/v3.6.8:3c6b436a57, Dec 22 2018, 19:34:48) on win32

Type "help", "copyright", "credits" or "quit()"

>>>

==== RESTART: C:\Users\Chirila Oana S

<class 'int'>

<class 'dict'>

<class 'tuple'>

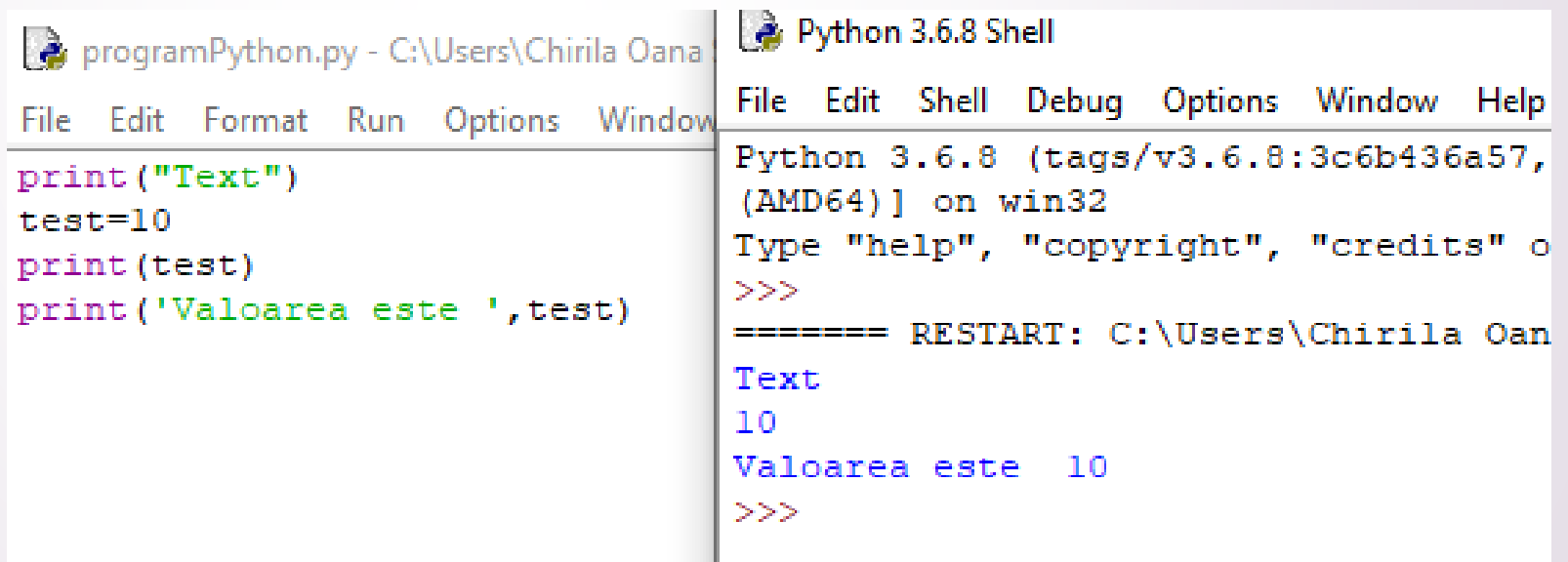
<class 'float'>

<class 'str'>

>>>

Afișare pe ecran

- Dacă dorim să afișăm mesaje pe ecran vom folosi instrucțiunea **print**
- Exemplu:



The image shows a side-by-side comparison of a Python script and its execution output. On the left is a text editor window titled 'programPython.py - C:\Users\Chirila Oana' with a menu bar (File, Edit, Format, Run, Options, Window). The script contains the following code:

```
print("Text")
test=10
print(test)
print('Valoarea este ',test)
```

On the right is a 'Python 3.6.8 Shell' window with a menu bar (File, Edit, Shell, Debug, Options, Window, Help). It displays the output of the script after execution:

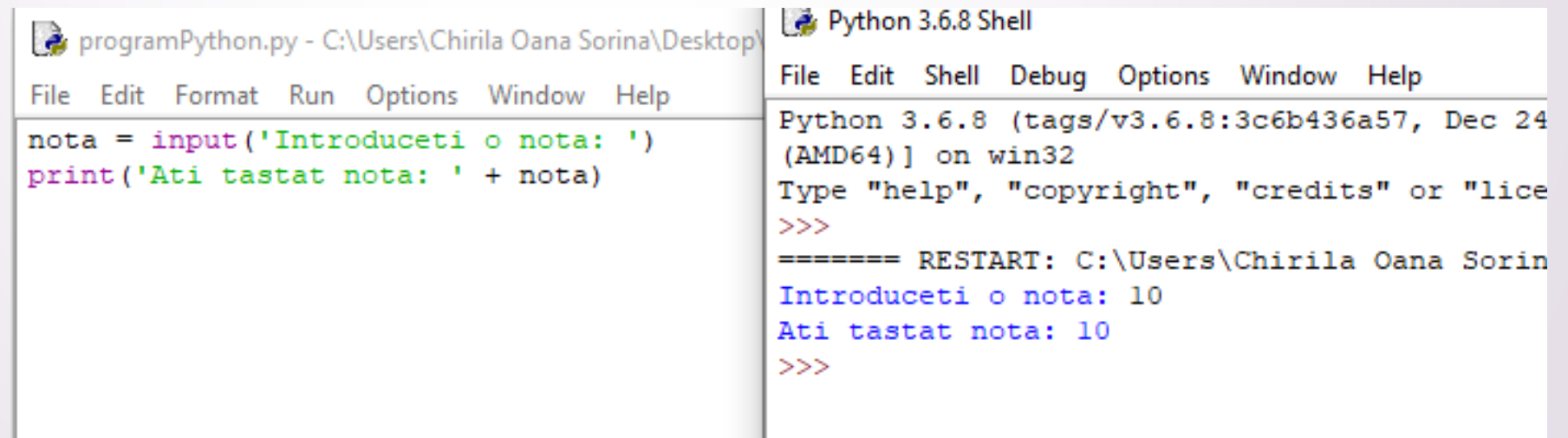
```
Python 3.6.8 (tags/v3.6.8:3c6b436a57,
(AMD64)] on win32
Type "help", "copyright", "credits" o
>>>
===== RESTART: C:\Users\Chirila Oan
Text
10
Valoarea este  10
>>>
```

Citire de la tastatură

- Dacă dorim să citim o variabilă de la tastatură sintaxa este următoarea:

`variabila = input('Introduceti o valoare: ')`

Exemplu:



The screenshot displays two windows from a Python IDE. The left window, titled 'programPython.py - C:\Users\Chirila Oana Sorina\Desktop\...', contains the following Python code:

```
nota = input('Introduceti o nota: ')
print('Ati tastat nota: ' + nota)
```

The right window, titled 'Python 3.6.8 Shell', shows the execution of the script. It includes a menu bar (File, Edit, Shell, Debug, Options, Window, Help) and the following output:

```
Python 3.6.8 (tags/v3.6.8:3c6b436a57, Dec 24
(AMD64)] on win32
Type "help", "copyright", "credits" or "lice
>>>
===== RESTART: C:\Users\Chirila Oana Sorin
Introduceti o nota: 10
Ati tastat nota: 10
>>>
```

Instrucțiunea IF

Sintaxă:

(1)

```
if Expresie_Logica:  
    Bloc de cod indentat
```

(2)

```
if Expresie_Logica:  
    Bloc de cod indentat 1  
else:  
    Bloc de cod indentat 2
```

(3)

```
if Expresie_Logica_1 :  
    Bloc de cod indentat1  
elif Expresie_Logica_2 :  
    Bloc de cod indentat2  
elif Expresie_Logica_3 :  
    Bloc de cod indentat3  
...  
else :  
    Bloc de cod indentatN
```

Instrucțiunea IF - exemplu

```
raspuns = int(input("Cate zile sunt intr-un an bisect? "))
print("Raspunsul dvs a fost:", raspuns)

if raspuns == 366 :
    print("Ati raspuns corect la prima intrebare")
    raspuns = input("Ce luna are o zi in plus in anul bisect? ").lower()
    if raspuns == "februarie" :
        print("Ati raspuns corect")
    else :
        print("Ati raspuns gresit")
else :
    print("Ati raspuns gresit la prima intrebare")
```

Instrucțiunea FOR

Sintaxa:

(1)

for variabila in secventa:

operatii cu variabila – indentat

(2)

for x in range(start, stop):

operatii cu x

Instrucțiunea FOR - exemplu

```
int_list = [1, 2, 3, 4, 5, 6]
sum = 0
for iter in int_list:
    sum = sum + iter
print("Sum =", sum)
print("Avg =", sum/len(int_list))
```


Instrucțiunea WHILE

Sintaxa:

while condiție (sau expresie) :

bloc de cod – indentat

```
count = 0
while count < 5 :
    num = int(input("Introduceti un numar intre 1 si 100?"))
    if (num < 0) or (num > 100):
        print("Ati introdus un numar invalid")
        break
    else:
        count += 1
        print("Numar corect!")
```

Funcții în Python

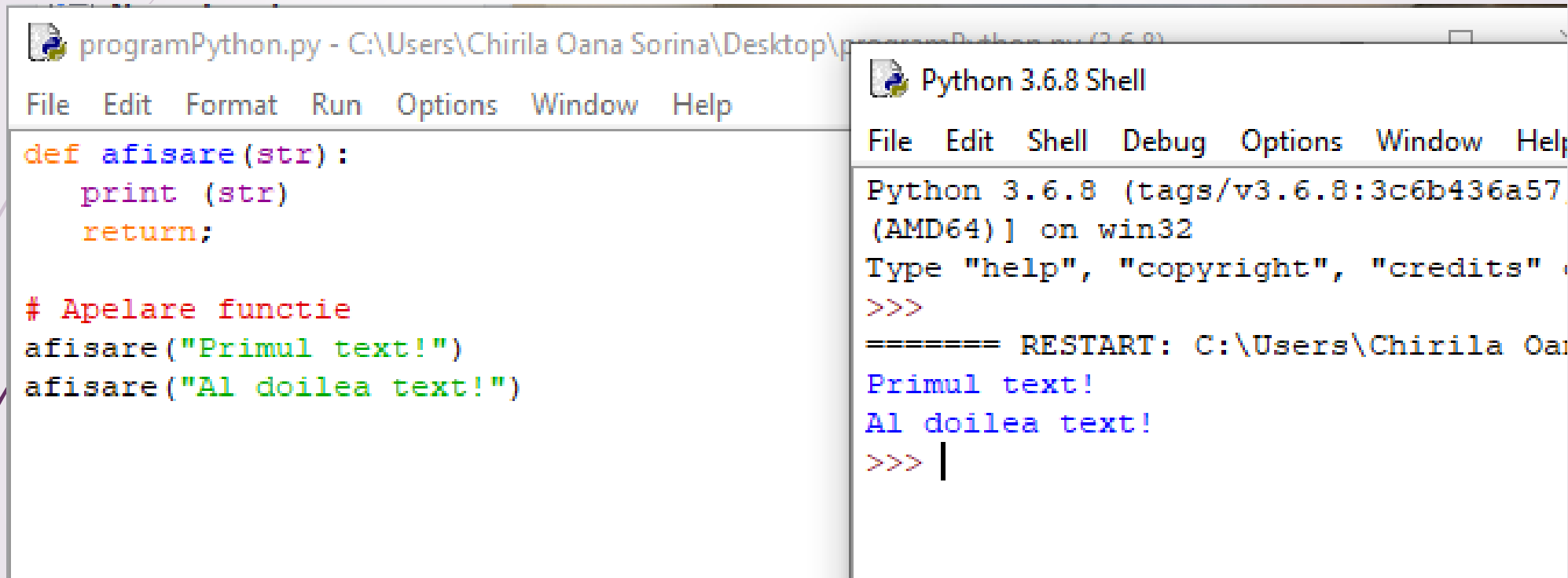
Iată câteva reguli simple pentru a defini o funcție în Python.

- Blocurile funcționale încep cu cuvântul cheie **def** urmat de numele funcției și paranteze ().
- Orice parametri sau argumente de intrare ar trebui să fie plasate în aceste paranteze.
- De asemenea, puteți defini parametrii în aceste paranteze.
- Blocul de cod din fiecare funcție începe cu două puncte (:) și este indentat.
- Declarația `return [expresie]` returnează valoarea funcției.

Syntax

```
def functionname( parameters ):  
    "function_docstring"  
    function_suite  
    return [expression]
```

Funcții - exemplu



The image shows a Python IDE window titled 'programPython.py' and a 'Python 3.6.8 Shell' window. The IDE window contains the following code:

```
def afisare(str):  
    print (str)  
    return;  
  
# Apelare functie  
afisare("Primul text!")  
afisare("Al doilea text!")
```

The shell window shows the output of the code:

```
Python 3.6.8 Shell  
File Edit Shell Debug Options Window Help  
Python 3.6.8 (tags/v3.6.8:3c6b436a57, (AMD64)] on win32  
Type "help", "copyright", "credits" or "quit()" for more  
>>>  
===== RESTART: C:\Users\Chirila Oana Sorina\Python368\Python368 Shell  
Primul text!  
Al doilea text!  
>>> |
```

Funcții - exemplu

```
programPython.py - C:\Users\Chirila Oana Sorina\Desktop\programPyth
File Edit Format Run Options Window Help
def sum( arg1, arg2 ):
    total = arg1 + arg2
    print ("Suma este in functie: ", total)
    return total;

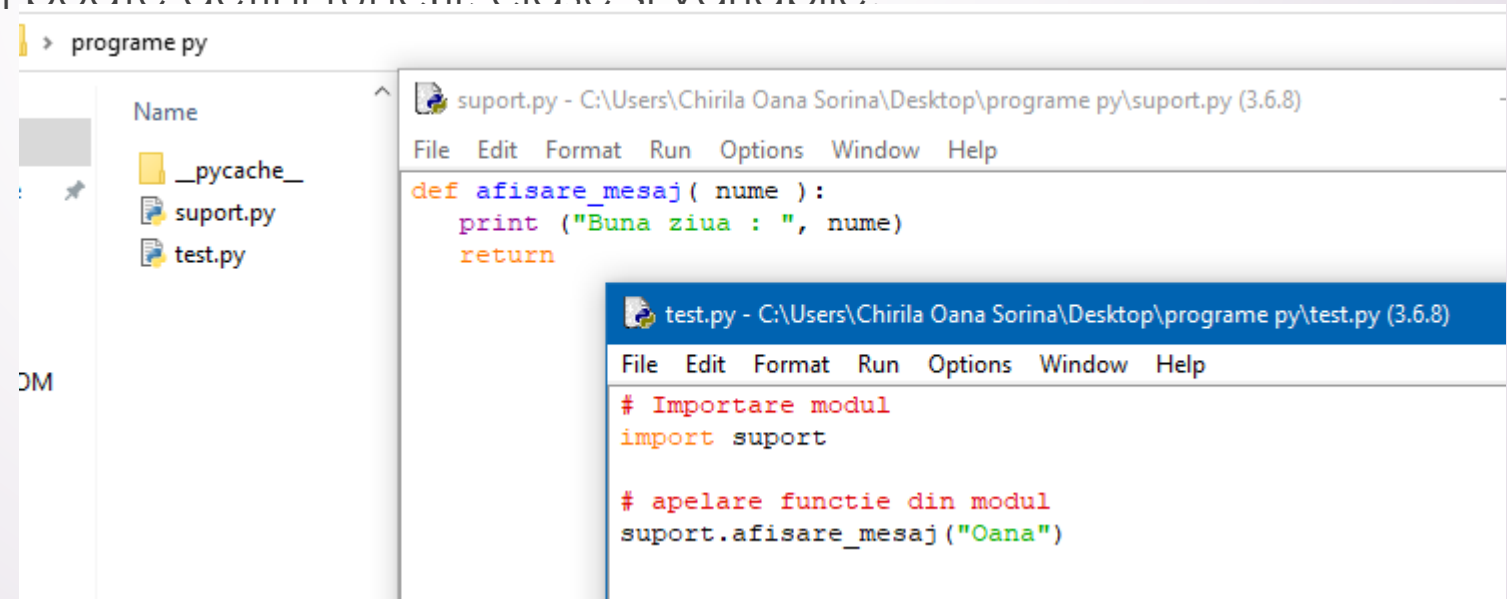
#apelare
total = sum( 10, 20 );
print ("Suma este in afara functiei: ", total)
```

```
Python 3.6.8 Shell
File Edit Shell Debug Options Window Help
Python 3.6.8 (tags/v3.6.8:3c6b436a57, De
(AMD64)] on win32
Type "help", "copyright", "credits" or "
>>>
===== RESTART: C:\Users\Chirila Oana S
Suma este in functie: 30
Suma este in afara functiei: 30
>>> |
```

Module în Python

- Un modul vă permite să vă organizați codul Python într-un mod logic.
- Gruparea codului aferent într-un modul facilitează înțelegerea și utilizarea codului.
- Un modul este un fișier format din cod Python.
- Un modul poate defini funcții, clase și variabile.

► Exemplu:



```
> programe py
Name
  _pycache_
  suport.py
  test.py

suport.py - C:\Users\Chirila Oana Sorina\Desktop\programe py\suport.py (3.6.8)
File Edit Format Run Options Window Help
def afisare_mesaj ( nume ) :
    print ("Buna ziua : ", nume)
    return

test.py - C:\Users\Chirila Oana Sorina\Desktop\programe py\test.py (3.6.8)
File Edit Format Run Options Window Help
# Importare modul
import suport

# apelare functie din modul
suport.afisare_mesaj("Oana")
```

Desenare grafice

- ▶ **Biblioteca matplotlib()** permite construirea de grafice.
- ▶ Pentru instalarea bibliotecii, în cmd vom folosi comanda:
 - ▶ **py -m pip install matplotlib**
- ▶ Pentru includerea bibliotecii matplotlib() se scrie următoarea linie de cod:
 - ▶ ***import matplotlib.pyplot as plt***

Desenare grafice – funcții

- ▶ Funcții folosite în construirea unui grafic:
- ▶ **title()** – funcție pentru setarea numelui (titlului) graficului
 - ▶ ex.: `plt.title('Grafic')`

În acest exemplu s-a dat titlul unui grafic “Grafic”

- ▶ Se mai pot transmite ca și parametri variabile care se referă la culoarea textului scris în titlul graficului.
 - ▶ Ex.:
 - ▶ `plt.title('Grafic',color='green')`
 - ▶ sau
 - ▶ `plt.title('Grafic',color='g')`
 - ▶ În ambele cazuri culoarea titlului va fi verde.

Desenare grafice – funcții

- ▶ **xlabel()** – funcție pentru setarea numelui axei x a graficului.
- ▶ Funcționalitatea acestei funcții se aseamănă cu funcționalitatea funcției *title()*
 - ▶ Ex.:
 - ▶ `plt.xlabel('axa x',color='red')`
 - ▶ sau
 - ▶ `plt.xlabel('axa x',color='r')`
- ▶ Setarea numelui axei x a graficului “axa x” cu culoarea roșu.
- ▶ **ylabel()** – funcție pentru setarea numelui axei y a graficului.
- ▶ Funcționalitatea acestei funcții se aseamănă cu funcționalitatea funcției *xlabel()*
 - ▶ Ex.:
 - ▶ `plt.ylabel('axa y',color='blue')`
 - ▶ sau
 - ▶ `plt.ylabel('axa y',color='b')`
- ▶ Setarea numelui axei y a graficului “axa y” cu culoarea albastru.

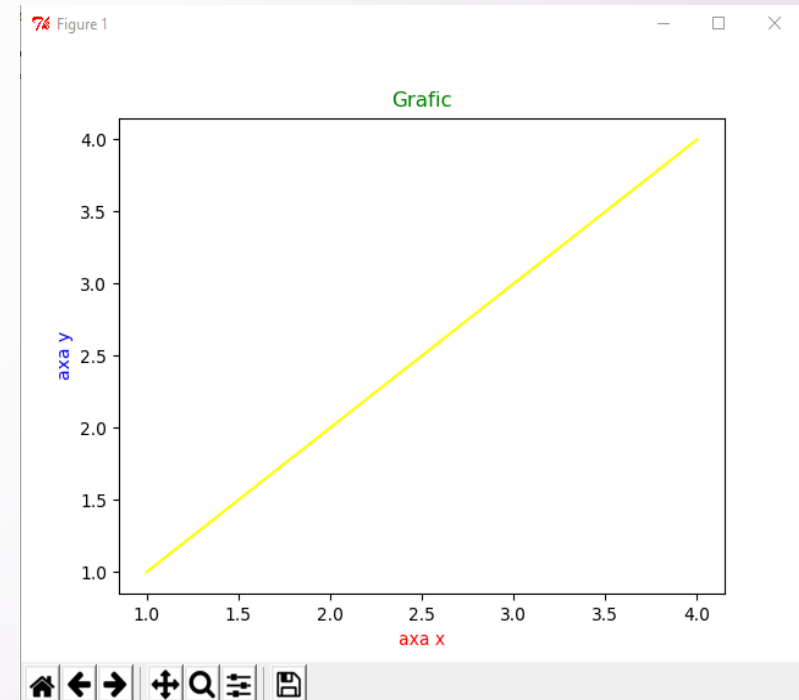
Desenare grafice – funcții

- **plot()** – funcție pentru desenarea graficului
- se transmit ca și parametrii coordonatele axei x și coordonatele axei y, iar dacă se dorește se poate seta și culoarea graficului sau formatul punctelor pe grafic
- Ex.:
 - `plt.plot([1,2,3,4],[1,2,3,4], color='yellow')`
 - sau
 - `plt.plot([1,2,3,4],[1,2,3,4], color='y')`
- **show()** – funcție folosită pentru deschiderea graficului.

Desenare grafice - exemplu

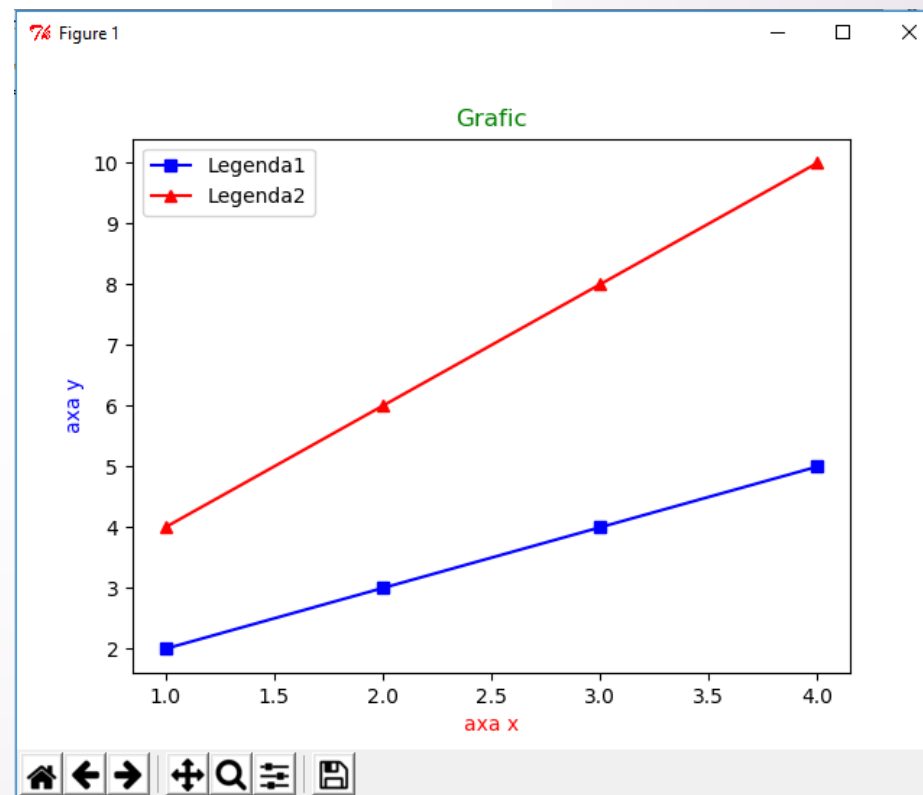
- Programul următor combină cele 5 funcții menționate pentru desenarea unui grafic:

```
import matplotlib.pyplot as plt
plt.plot([1,2,3,4],[1,2,3,4],color='yellow')
plt.title('Grafic',color='g')
plt.xlabel('axa x',color='red')
plt.ylabel('axa y',color='blue')
plt.show()
```



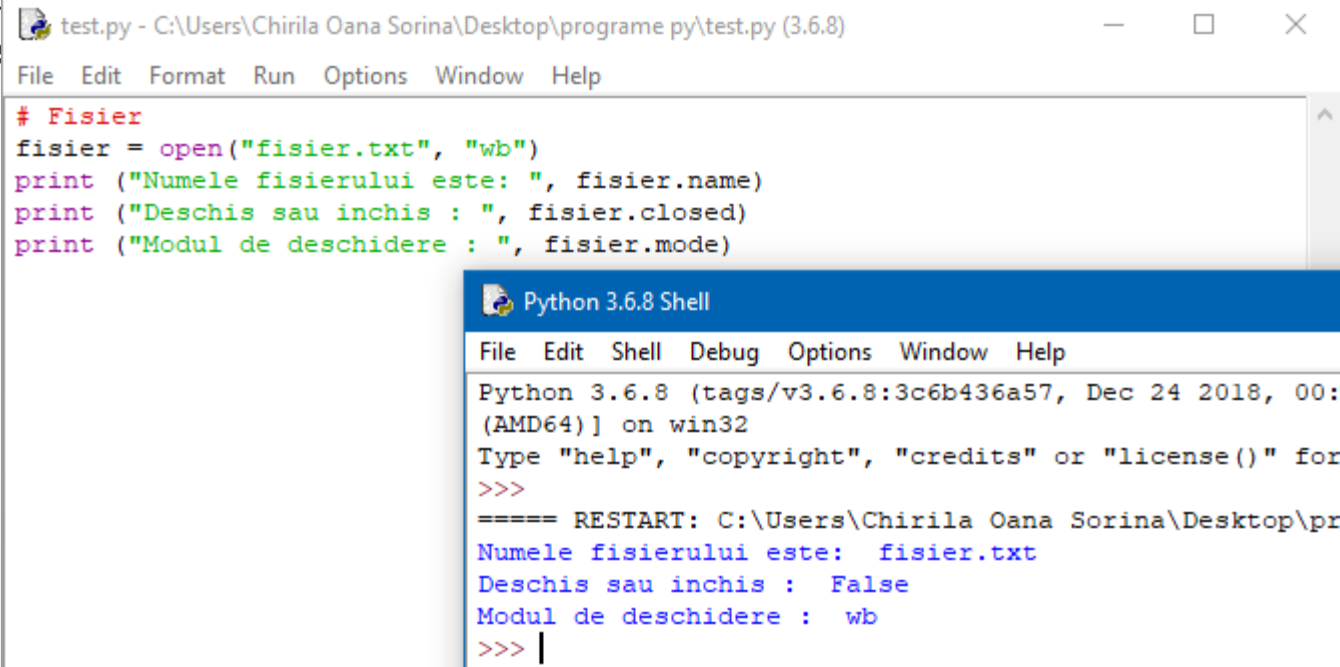
Desenare grafice - exemplu

```
import matplotlib.pyplot as plt
plt.plot([1,2,3,4],[2,3,4,5], 'bs-', [1,2,3,4],[4,6,8,10], 'r^--')
plt.legend(['Legenda1', 'Legenda2'])
plt.title('Grafic', color='g')
plt.xlabel('axa x', color='red')
plt.ylabel('axa y', color='blue')
plt.show()
```



Fișiere

- Python oferă funcții și metode de bază necesare pentru a manipula fișierele.
- Pentru a putea lucra cu fișiere, va trebui să folosiți obiecte de tip file.
- Înainte de a putea citi sau scrie în un fișier trebuie să îl deschideți folosind funcția open
- Exemplu:

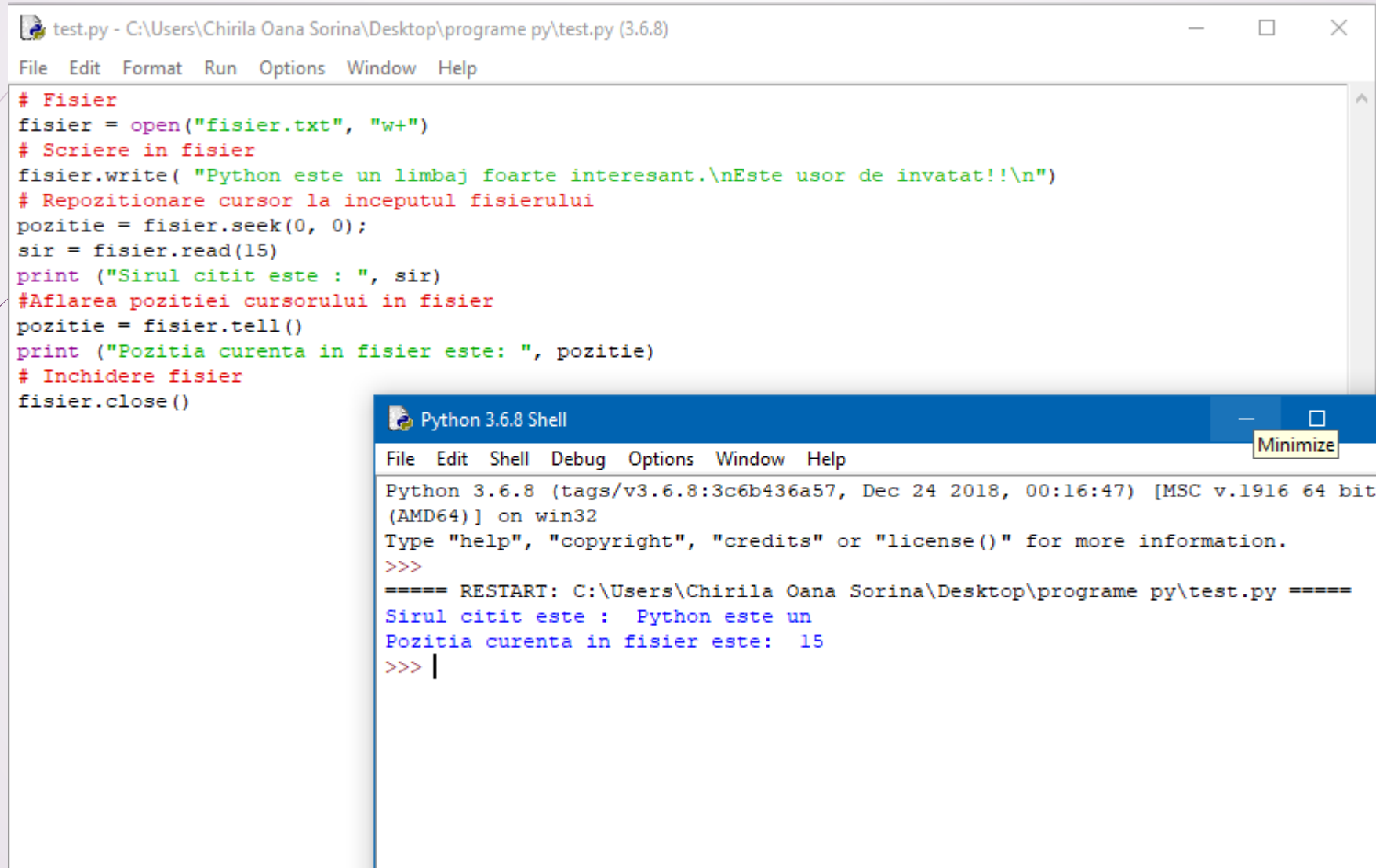


```
test.py - C:\Users\Chirila Oana Sorina\Desktop\programe py\test.py (3.6.8)
File Edit Format Run Options Window Help

# Fisier
fisier = open("fisier.txt", "wb")
print ("Numele fisierului este: ", fisier.name)
print ("Deschis sau inchis : ", fisier.closed)
print ("Modul de deschidere : ", fisier.mode)

Python 3.6.8 Shell
File Edit Shell Debug Options Window Help
Python 3.6.8 (tags/v3.6.8:3c6b436a57, Dec 24 2018, 00:
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for
>>>
===== RESTART: C:\Users\Chirila Oana Sorina\Desktop\pr
Numele fisierului este:  fisier.txt
Deschis sau inchis :  False
Modul de deschidere :  wb
>>> |
```


Fișiere



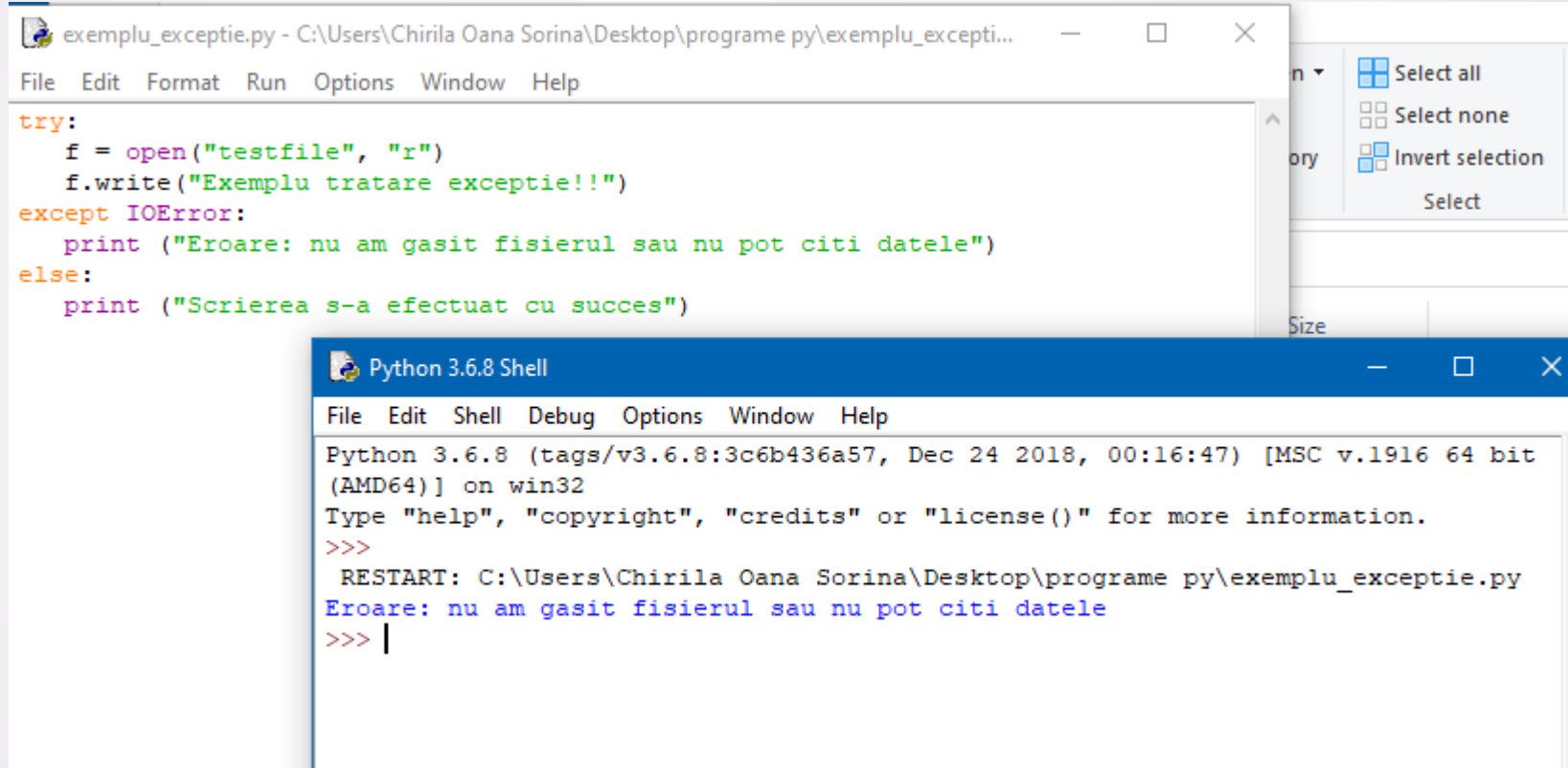
The image shows a Python IDE window titled "test.py - C:\Users\Chirila Oana Sorina\Desktop\programe py\test.py (3.6.8)". The code in the editor is as follows:

```
# Fisier
fisier = open("fisier.txt", "w+")
# Scriere in fisier
fisier.write( "Python este un limbaj foarte interesant.\nEste usor de invatat!!\n")
# Repozitionare cursor la inceputul fisierului
pozitie = fisier.seek(0, 0);
sir = fisier.read(15)
print ("Sirul citit este : ", sir)
#Aflarea pozitiei cursorului in fisier
pozitie = fisier.tell()
print ("Pozitia curenta in fisier este: ", pozitie)
# Inchidere fisier
fisier.close()
```

Below the editor is a "Python 3.6.8 Shell" window. It shows the execution of the script, with the output of the print statements:

```
Python 3.6.8 (tags/v3.6.8:3c6b436a57, Dec 24 2018, 00:16:47) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Chirila Oana Sorina\Desktop\programe py\test.py =====
Sirul citit este :  Python este un
Pozitia curenta in fisier este:  15
>>> |
```

Exceptii



The image shows a Python IDE window titled 'exemplu_exceptie.py - C:\Users\Chirila Oana Sorina\Desktop\programe py\exemplu_excepti...' with a menu bar (File, Edit, Format, Run, Options, Window, Help). The code in the editor is as follows:

```
try:
    f = open("testfile", "r")
    f.write("Exemplu tratare exceptie!!")
except IOError:
    print ("Eroare: nu am gasit fisierul sau nu pot citi datele")
else:
    print ("Scrierea s-a efectuat cu succes")
```

Below the editor is a 'Python 3.6.8 Shell' window with a menu bar (File, Edit, Shell, Debug, Options, Window, Help). It displays the following text:

```
Python 3.6.8 (tags/v3.6.8:3c6b436a57, Dec 24 2018, 00:16:47) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:\Users\Chirila Oana Sorina\Desktop\programe py\exemplu_exceptie.py
Eroare: nu am gasit fisierul sau nu pot citi datele
>>> |
```

Baze de date in Python

- Standardul Python pentru interfețele bazei de date este Python DB-API. Majoritatea interfețelor de baze de date Python aderă la acest standard.
- Puteți alege baza de date potrivită pentru aplicația dvs. API-ul Python Database acceptă o gamă largă de servere de baze de date, cum ar fi:
 - GadFly
 - mSQL
 - MySQL
 - PostgreSQL
 - Microsoft SQL Server 2000
 - Informix
 - Interbase
 - Oracle
 - Sybase
- Trebuie să descărcați un modul API DB separat pentru fiecare bază de date pe care trebuie să o accesați. De exemplu, dacă trebuie să accesați o bază de date Oracle, precum și o bază de date MySQL, trebuie să descărcați atât modulele de baze de date Oracle, cât și MySQL

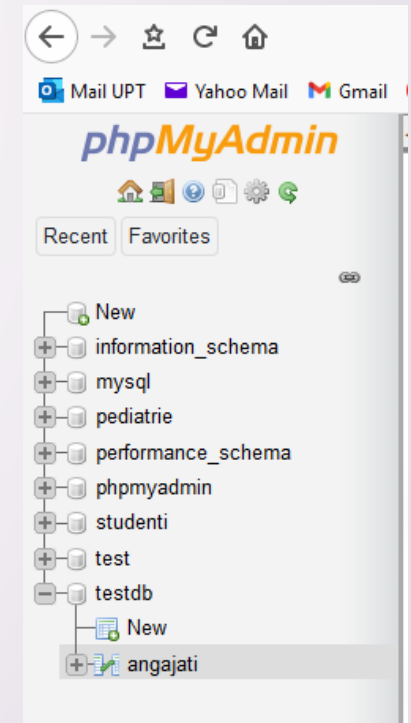
Baze de date in Python

- MySQLdb este o interfață pentru conectarea la un server de baze de date MySQL de la Python.
- Pentru a instala modulul MySQLdb, utilizați următoarea comandă:

- `pip install mysqlclient`

Conexiunea la baza de date:

- Înainte de a vă conecta la o bază de date MySQL:
 - Creați o bază de date cu numele TESTDB
 - Creați un tabel ANGAJATI în TESTDB.



Baze de date in Python

```
import MySQLdb

# Deschidem conexiunea la baza de date
db = MySQLdb.connect("localhost","root","","TESTDB" )

#pregătim un obiect cursor prin folosirea metodei cursor()
cursor = db.cursor()

# executăm interogarea SQL folosind metoda execute()
cursor.execute("SELECT VERSION()")

#Selectăm un singur rând folosind metoda fetchone()
data = cursor.fetchone()
print ("Versiunea bazei de date este: %s " %data)

# Stergem tabelul in cazul in care acesta există
cursor.execute("DROP TABLE IF EXISTS ANGAJATI")

#Creem tabelul Angajati
sql = """CREATE TABLE ANGAJATI (
    NUME CHAR(20) NOT NULL,
    PRENUME CHAR(20),
    VARSTA INT,
    FUNCTIA CHAR(20),
    SALAR FLOAT )"""
cursor.execute(sql)
```

```
# Pregatim interogarea SQL pentru a insera o noua inregistrare
sql = """INSERT INTO ANGAJATI(NUME,
    PRENUME, VARSTA, FUNCTIA, SALAR)
    VALUES ('Popescu', 'Oana', 34, 'Programator', 7000)"""

try:
    # Executam comanda SQL
    cursor.execute(sql)
    # Confirmam modificările în baza de date
    db.commit()
except:
    # Revenire în caz de eroare
    db.rollback()

sql = "SELECT * FROM ANGAJATI \
    WHERE SALAR > '%d'" % (1000)
try:
    # Executam comanda SQL
    cursor.execute(sql)
    # Selectam toate randurile care indeplinec conditia
    results = cursor.fetchall()
    for row in results:
        nume = row[0]
        pren = row[1]
        varsta = row[2]
        func = row[3]
        sal = row[4]
        # Afisam rezultatul
        print ("Numele= %s, Prenumele= %s, Varsta= %d, Functia= %s, Salar= %d" % \
            (nume, pren, varsta, func, sal ))
except:
    print ("Eroare")
```

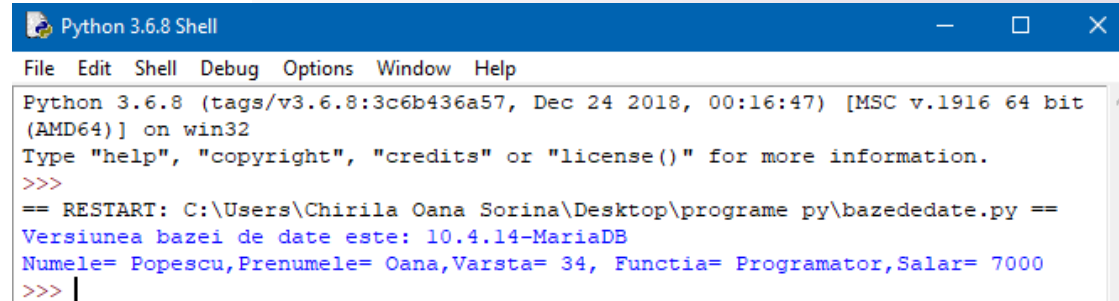
Baze de date in Python

```
# pregatire interogare SQL pentru UPDATE
sql = "UPDATE ANGAJATI SET VARSTA = VARSTA + 1"

try:
    # executam comada SQL
    cursor.execute(sql)
    # Confirmam modificările în baza de date
    db.commit()
except:
    # Revenire în caz de eroare
    db.rollback()

# pregatire interogare SQL pentru DELETE
sql = "DELETE FROM ANGAJATI WHERE VARSTA > '%d'" % (20)
try:
    # executam comada SQL
    cursor.execute(sql)
    # Confirmam modificările în baza de date
    db.commit()
except:
    # Revenire în caz de eroare
    db.rollback()

# deconectare de la baza de date
db.close()
```



```
Python 3.6.8 Shell
File Edit Shell Debug Options Window Help
Python 3.6.8 (tags/v3.6.8:3c6b436a57, Dec 24 2018, 00:16:47) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
== RESTART: C:\Users\Chirila Oana Sorina\Desktop\programe py\bazededate.py ==
Versiunea bazei de date este: 10.4.14-MariaDB
Numele= Popescu, Prenumele= Oana, Varsta= 34, Functia= Programator, Salar= 7000
>>> |
```


Link-uri utile pentru studiere mai avansata PYTHON

- <https://www.tutorialspoint.com/python/index.htm>
- <https://www.w3schools.com/python/>
- <https://www.youtube.com/watch?v=uQrJ0TkZlc>
- <https://docs.python.org/3/tutorial/>
- <https://www.learnpython.org/>