

Cum programatorii caută mereu să învețe limbaje noi care sunt open source, care se execută pe mai multe platforme și au un suport excelent al comunității, este important să ne oprim pe această cale la Python. După ce v-ați familiarizat cu informațiile de bază despre limbajul de programare Python și înainte de a începe orice script inițial, este necesar a se configura mai întâi mediul de lucru, de care ne vom ocupa, printre altele, în această lecție.

Ce este Python?

Python este un limbaj de programare orientat pe obiecte, ceea ce înseamnă că absolut totul este reprezentat ca obiect, adică include variabile, funcții, module și biblioteci. Sistemul de tipizare este dinamic, ceea ce înseamnă că verificarea tipurilor de variabile se face în timpul execuției programului și nu înainte de execuție.

Python este cel mai comun în aplicațiile web și data science și este suportat pe aproape toate platformele.



Imaginea 2.1. Logoul Python¹

Istoria

Calea de dezvoltare a limbajului Python începe la sfârșitul anilor

optzeci ai secolului XX, mai exact în 1989, când Guido van Rossum a venit cu ideea de a începe un hobby planificat de mult de a scrie un compilator pentru un limbaj nou de programare, la care se gândise de mult. Numele îl alege pe baza serialului BBC TV *Monty Python*. Limbajul de programare Python este scris în limbajul de programare C.

După puțin mai mult de un deceniu, Python primește o revizuire demult așteptată - 2.0; Python 3.9 a fost lansat în 2020, iar Python 3.10 în 2022, pe care îl vom folosi în acest curs.

Cum funcționează un program Python?

Viața programului Python depinde de implementare, iar în prezent există numeroase implementări Python (Jython, PyPy, Anaconda). Cea mai comună implementare a limbajului Python este CPython.

Fluxul de execuție a codului în Python arată așa:

1. **Analizor (parser)**
Folosește codul sursă al [scriptului](#) pentru a genera un arbore de sintaxă abstract
2. **Translator de programe (compiler)**
Convertește arborele de sintaxă abstract menționat mai sus în cod binar Python.
3. **Interpreter (interpreter)**
Execută codul linie după linie într-o manieră REPL (Read-Evaluate-Print-Loop).

Avantajele limbajului Python

Python este unul dintre cele mai populare limbaje de programare cu scop general. Se numără printre limbajele de programare cu cea mai rapidă creștere; este folosit de inginerii de software, matematicieni, analiști de date, oameni de știință și alții. Caracteristicile care îl fac

astfel sunt:

- Python este un limbaj de programare de nivel înalt interpretat, orientat pe obiecte. Deoarece codul sursă este compilat în [bytecode](#) (cod binar) care este interpretat în continuare, numim Python un limbaj de programare interpretat. CPython este responsabil de această operațiune.
- Suportă variabile dinamice și legare dinamică. De exemplu, în limbaje de programare precum Java, C și C++, o variabilă *string* nu poate fi [inițializată](#) la tipul *int*, deci trebuie să se țină cont de tipurile de variabile, ceea ce nu este cazul în Python, deoarece tipul variabilei se determină doar atunci când acea linie de cod vine la rând să se execute. Mai multe informații despre tipurile de variabile vor fi în lecțiile viitoare.
- Sintaxa lui Python este foarte simplă de citit, astfel încât și lizibilitatea codului în sine este mai mare și mai ușoară, ceea ce reduce și mai mult timpul necesar pentru menținerea scriptului sau bibliotecii curente.

Exemplu de sintaxă pentru adunarea a două numere întregi:

```
a = 10
```

```
b = 13
```

```
sum = a + b
```

```
print(sum)
```

Explicație:

În acest exemplu putem vedea o modalitate simplă de a se crea două

variabile, de a plasa rezultatul adunării lor într-o a treia variabilă și de a imprima suma la ieșirea standard.

- Python este format din module și pachete, ceea ce permite reutilizarea aceluiași cod.
- Python este un limbaj de programare open source, ceea ce înseamnă că utilizatorul poate descărca liber codul sursă și îl poate modifica după cum este necesar.
- Procesul de modificare a codului, testarea și eliminarea erorilor sunt foarte rapide, deoarece întregul cod nu este compilat deodată la începutul execuției.
- Acceptă gestionarea erorilor.
- Gestionarea automată a memoriei de lucru. Gestionarea memoriei în Python este organizată astfel încât o parte dinamică privată - *heap* conține toate obiectele și structurile de date. După cum este necesar, modulul de gestionare a memoriei alocă spațiu din heap pentru obiecte Python noi.

Versatilitatea limbajului de programare Python

Succesul și popularitatea limbajului Python se bazează, în primul rând, pe caracteristicile sale. Sprijinul puternic al comunității și o gamă largă de module și biblioteci externe au făcut din Python un instrument puternic pentru rezolvarea problemelor de programare de zi cu zi, indiferent de domeniul din care provin, motiv pentru care acest limbaj de programare este, printre altele, bun și pentru:

1. Dezvoltarea aplicațiilor web

Programatorii web care folosesc Python au destule soluții ([framework](#)) când este vorba de limbajul de programare pe partea de server. Django și Flask sunt printre cele mai populare soluții pe partea de server. Django este folosit pentru a crea atât partea frontend, cât și backend-ul site-urilor web complexe, în timp ce Flask este o soluție simplă și ușor extensibilă pentru crearea de aplicații web mai simple; este foarte ușor de învățat și este o alegere excelentă pentru începători.

Câteva exemple de implementări ale soluțiilor Django sunt YouTube, Spotify, Mozilla, Dropbox și Instagram, iar unele [implementări](#) Flask: Airbnb, Netflix, Uber și Reddit.

2. Învățare automată

Python este un limbaj foarte accesibil, care are mult suport din parte comunității atunci când este vorba de module externe, al căror scop este să ajute utilizatorul să rezolve o anumită problemă în loc să-și consume timpul scriindu-le. Este, de asemenea, un wrapper (înveliș) excelent în jurul implementărilor C/C++ mult mai eficiente din cadrul [algoritmilor](#) CUDA-cuDNN, pe care se bazează [învățarea automată](#), unde accentul este pus pe procesarea seturilor mari de date.

3. Analiza datelor

De asemenea, Python conține și destule instrumente pentru fiecare aspect al procesării și analizei datelor. Câteva exemple sunt Facebook, care folosește biblioteca *Pandas* Python pentru analiză, și National Bank of America, care utilizează acest limbaj de programare pentru a procesa datele financiare.

Există destul de multe biblioteci care se ocupă de această problemă; vom menționa cele mai cunoscute soluții:

- **NumPy** este fundamental atunci când este vorba de analiza datelor științifice; suportă șiruri mari, multidimensionale și matrice mari, cât și un repertoriu mare de funcții matematice și statistice care pot fi aplicate pe aceste obiecte;
- **SciPy** este folosit împreună cu șirurile NumPy și oferă metode mai eficiente pentru operații numerice;
- **Pandas** este încă un derivat al bibliotecii NumPy care oferă structuri de date extrem de funcționale (tabele), precum și operațiuni asupra acestora;
- **Matplotlib** este biblioteca pentru trasarea graficelor 2D și a altor vizualizări (diagrame cu bare, histograme, diagrame cu puncte etc.).

4. Jocuri

Python și PyGame împreună sunt o pereche excelentă pentru dezvoltarea rapidă a prototipurilor pentru începători și pentru crearea de jocuri mai simple. Unele dintre cele mai faimoase jocuri comerciale scrise în Python sunt Battlefield2, Civilization IV și EVE Online. Un instrument popular de animație și modelare 3D - Blender3D, care oferă posibilitatea de a crea jocuri, acceptă, de asemenea, scripting folosind Python.

5. Aplicația desktop

Tkinter, care este procesat în cadrul programului *Python Development*, în cursul *Graphic Applications Development*, face parte din setul standard de biblioteci care vin cu Python și permite codificarea aplicațiilor GUI mai simple.

Simbolurile tipurilor de fișieriere Python (extensii)

- **Java și C++** - Deși Python este mult mai lent în executarea codului decât aceste două limbaje de programare, numărul de linii necesare pentru a scrie același program în Python și în Java/C++ este de trei până la cinci ori mai mic, așa că și timpul necesar pentru a-l scrie este mult mai scurt. Unul dintre motive pentru acest lucru este și ceea ce am menționat deja, variabile dinamice.
- **Python și Perl** sunt două limbaje de programare care au rădăcini similare, iar acest lucru este [scripting în Unix](#). În timp ce în Perl accentul este pus pe citirea datelor din fișiere text, imprimarea și tipărirea rapoartelor și conversia acelor fișiere în alte formate, Python oferă suport pentru principii comune, cum ar fi programarea orientată pe obiecte, și forțează programatorii să scrie un cod ușor de citit (și de întreținut), astfel încât să ofere o sintaxă elegantă.

Simbolurile tipurilor de fișieriere Python (extensii)

Extensiile și utilizările fișierelor Python se pot găsi în tabelul următor:

Extensia	Scopul
.py	tipul de date al fișierelor sursă (scripte)
.pyc	bytecode compilat al scriptului Python
.pyd	tip de date specific platformei, destinat sistemelor de operare Windows
.pyo	fișier creat de interpretor atunci când un modul este încărcat, dar numai atunci când acel interpretor este pornit cu setările de optimizare activate
.pyw	un script Python creat special pentru a rula din pythonw.exe
.pyz	arhiva de script Python

Tabelul 2.1. Extensiile fișierelor Python și scopul lor

Cum pornesc un program Python?

Pentru ca programul Python să poată fi pornit, este necesar să aveți:

- **programul** Python scris și
- **Interpretorul** Python.

Instalarea interpretorului Python

Pentru început, trebuie să aveți un interpretor Python, la care se poate ajunge în mai multe feluri:

- Pentru utilizatorii de Windows, fișierul de instalare poate fi

descărcat de pe site-ul oficial python.org. Instrucțiuni detaliate pas cu pas pot fi găsite în multimedia pentru această lecție.

- Sistemele de operare precum [Linux](#) includ un [manager de pachete](#) încorporat cu care Python poate fi instalat. Multe distribuții Linux includ deja Python prin instalare.
- Când este vorba de [macOS](#), pentru a instala Python este mai bine mai întâi să instalați managerul de pachete suplimentar – [Homebrew](#) și să urmați instrucțiunile din acest ghid.
- Deși dispozitivele mobile precum dispozitivele Android și iOS au suport pentru dezvoltarea de scripturi Python, configurarea unor astfel de medii nu va face obiectul acestui curs, mai ales că învățarea programării este mai ușoară și mai intuitivă pe un calculator desktop decât pe ecranul unui telefon mobil.

De asemenea, o alternativă la soluțiile menționate mai sus sunt și compilatorii online Python, care sunt porniți din browserul dorit.

Unele sisteme de operare au deja instalat un interpretor Python, deci, dacă nu sunteți sigur că aveți deja un interpretor instalat pe calculator, puteți verifica tastând comanda în consolă (Command Prompt/Terminal):

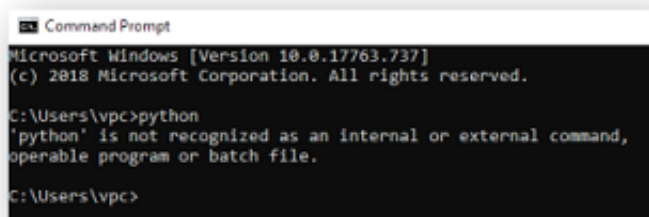
```
python -v
```

sau

```
python3 -v
```

În imaginea 2.2. este prezentat un exemplu de verificare dacă pe un calculator cu sistem de operare Windows este instalat Python.

Python **nu este** instalat →

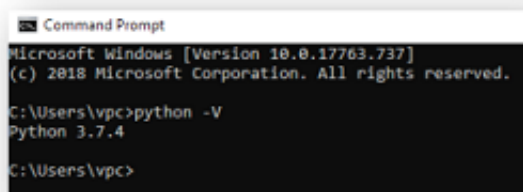


```
Command Prompt
Microsoft Windows [Version 10.0.17763.737]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\vpc>python
'python' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\vpc>
```

Python **este** instalat →



```
Command Prompt
Microsoft Windows [Version 10.0.17763.737]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\vpc>python -V
Python 3.7.4

C:\Users\vpc>
```

Imaginea 2.2 Verificarea dacă Python este instalat

Windows

Python nu vine împreună cu instalarea Windows. În continuare sunt pașii pentru instalarea lui:

Pasul 1: Descărcați fișierul de instalare Python3.x

1. Deschideți browserul dorit și accesați pagina [cu fișierele de instalare pentru Windows](#).
2. Mai jos, pe pagină, găsiți secțiunea *Stable Releases* și sub ea găsiți versiunea Python corespunzătoare sub forma **Python 3.x.x - <lună> <zi>, <an>** și deschideți acel link.
3. Derulați până la sfârșitul paginii și în tabelul cu versiuni găsiți fie *Windows x86-64 executable installer* pentru sistemele pe 64 de biți, fie *Windows x86 executable installer* pentru sistemele de operare pe 32 de biți.

Sfat

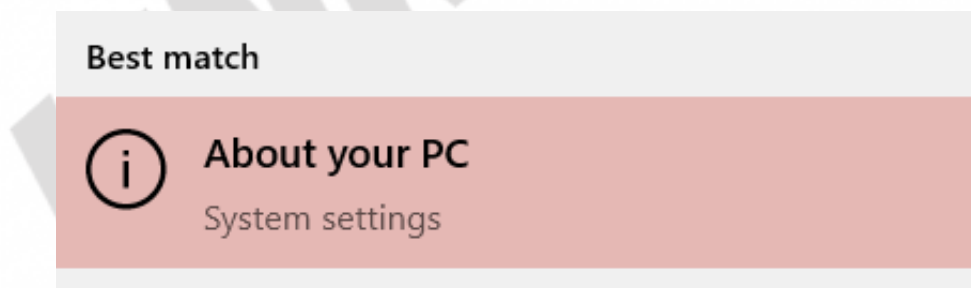
Selecția fișierului dintre *Windows x86-64 executable installer* sau *Windows x86 executable installer* depinde dacă sistemul dvs. de operare este pe 64 de biți sau pe 32 de biți, ce se poate verifica în setările pentru Windows 10 (opțiunea *About Your PC*).

- Dacă sistemul dvs. de operare este de 32 de biți, atunci ar trebui să alegeți *Windows x86 executable installer*.
- Dacă sistemul dvs. de operare este de 64 de biți, puteți instala oricare dintre aceste două versiuni, însă în acest caz este recomandat *Windows x86-64 executable installer*.

De asemenea, este important să știm dacă modulele și bibliotecile pe care dorim să le folosim acceptă versiuni pe 64 de biți sau 32 de biți.

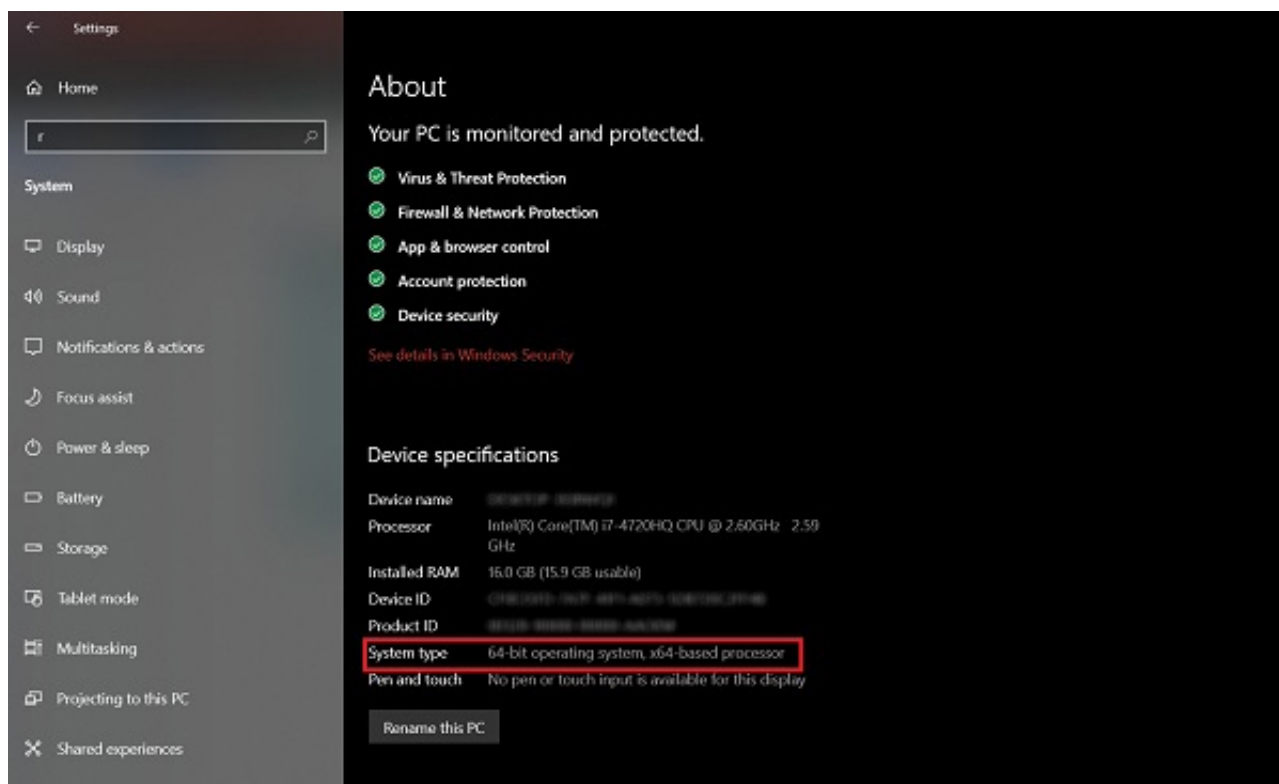
Verificarea arhitecturii

Verificarea arhitecturii sistemului de operare se face tastând „About your PC” în meniul Start, după cum vedem în imaginea 2.3.:



Imaginea 2.3. Aspectul opțiunii About your PC în meniul Start

Apoi, după ce se deschide fereastra, trebuie să găsiți label System type prezentată în imaginea 2.4.:



Imaginea 2.4. Prezentarea locației opțiunii System Type

Dacă în acest label este un procesor *64-bit, x64 based processor*, înseamnă că avem un sistem de operare pe 64 de biți. În caz contrar, avem un sistem de operare pe 32 de biți.

Files

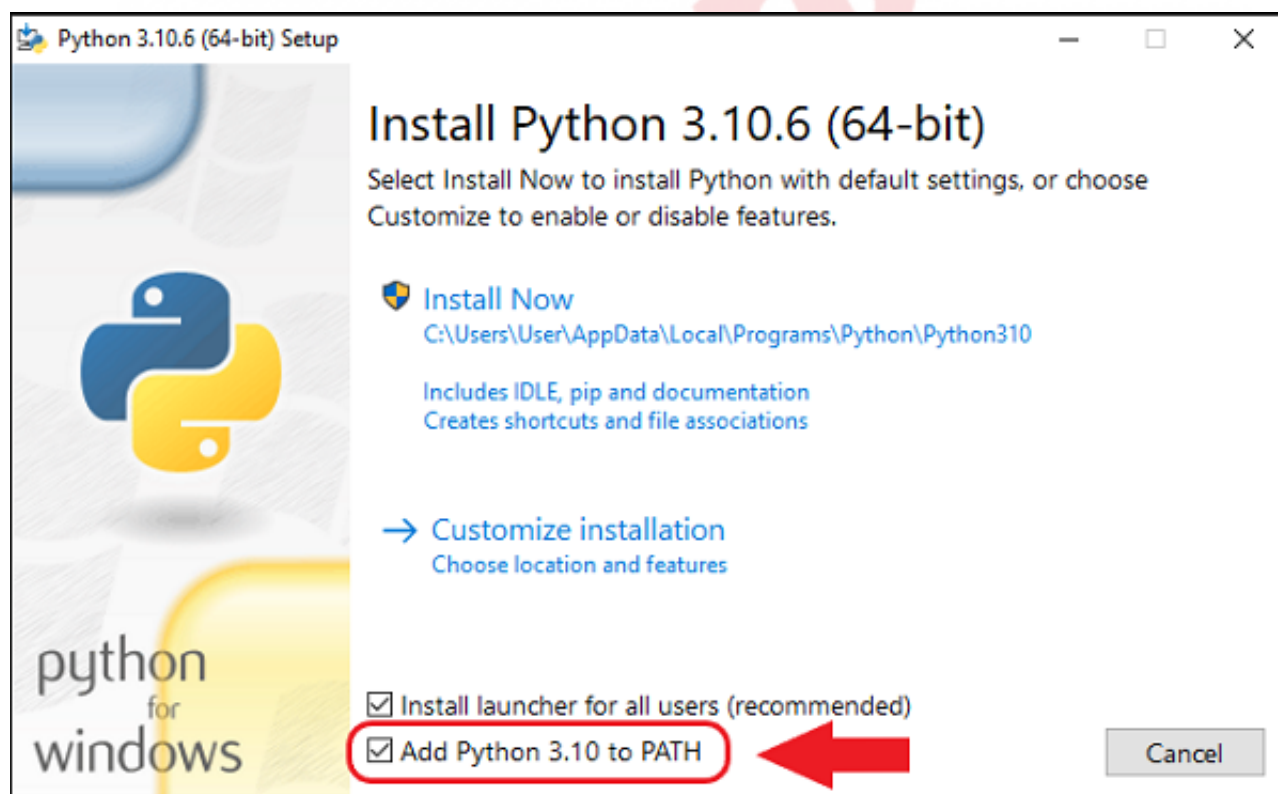
Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		d348d978a5387512fbc7d7d52dd3a5ef	23161893	SIG
XZ compressed source tarball	Source release		172c550156f7bea68ce31b2fd01fa766	17268888	SIG
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	47b06433e242c8eb848e035965a860ac	29163525	SIG
Windows help file	Windows		89bb2ea8c5838bd2612de600bd301d32	8183265	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	6aa3b1c327561bda256f2deebf038dc9	7444654	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	e0c910087459df78d827eb1554489663	26797616	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	d1d09dad50247738d8ac2479e4cde4af	1348896	SIG
Windows x86 embeddable zip file	Windows		29672b400490aa21995c6d8ae4c4e1c8	6614968	SIG
Windows x86 executable installer	Windows		e9db9cf43b4f2472d75a055380871045	25747128	SIG
Windows x86 web-based installer	Windows		8b326250252f15e19879701f5e53c76	1319912	SIG

Imaginea 2.5. Prezentarea tabelului versiunilor Windows Python

Dacă din întâmplare este instalată versiunea greșită de Python, aceasta poate fi eliminată cu ușurință prin Panoul de control.

Pasul II: Rularea fișierului de instalare

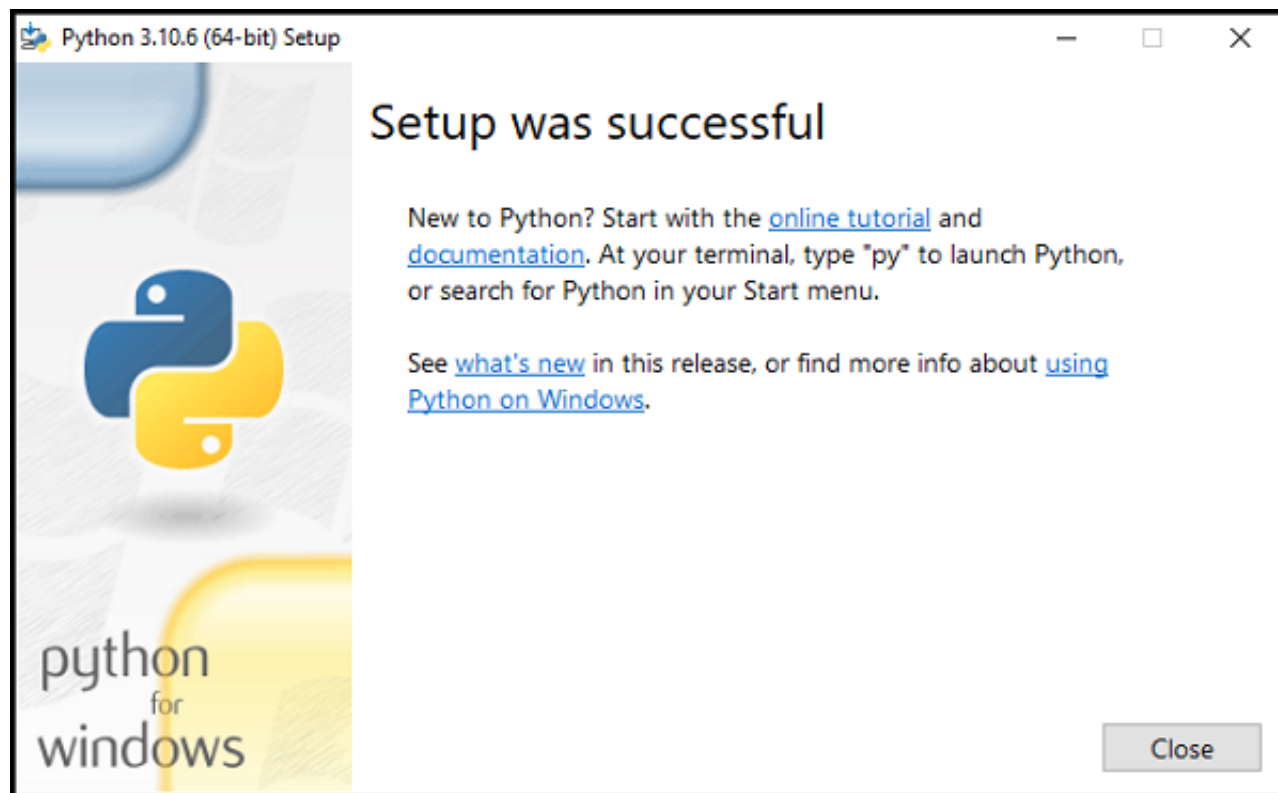
După ce am selectat și descărcat versiunea dorită, trebuie pornită cu dublu clic. În dialog va apărea câmpul *Add Python 3.10 to PATH*, care trebuie **neapărat** bifat, așa cum este prezentat în imaginea 2.6.



Imaginea 2.6. Primul pas al procesului de instalare

După ce câmpul a fost bifat, continuați procesul de instalare cu un clic pe *Install Now*.

După o instalare reușită, veți fi întâmpinat de o fereastră ca cea prezentată în imaginea 2.7.



Imaginea 2.7 Instalarea s-a finalizat cu succes

macOS

Putem instala Python pe macOS folosind managerul de pachete Homebrew. Este necesar a se deschide [Terminal](#), care se află în folderul Utilities. Folderul Utilities se află în folderul Applications. După ce se deschide Terminal, introduceți comanda:

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install.sh)"
```



```
Last login: Mon Jun 1 00:29:03 on ttys002

The default interactive shell is now zsh.
To update your account to use zsh, please run 'chsh -s /bin/zsh'.
For more details, please visit https://support.apple.com/kb/HT208050.
$ /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install.sh)"
```

Imaginea 2.8. Instalarea managerului de pachete Homebrew

După instalare, putem efectua verificarea folosind comanda:

```
brew help
```

Dacă comanda este recunoscută, Homebrew a fost instalat cu succes. Acum, când Homebrew este instalat, putem accesa instalarea Python cu comanda:

```
brew install python
```

După instalare, putem verifica versiunea de Python instalată pe calculator cu comanda:

```
python --version
```

Compilatori online

Unii dintre compilatorii online Python pe care îi puteți folosi sunt:

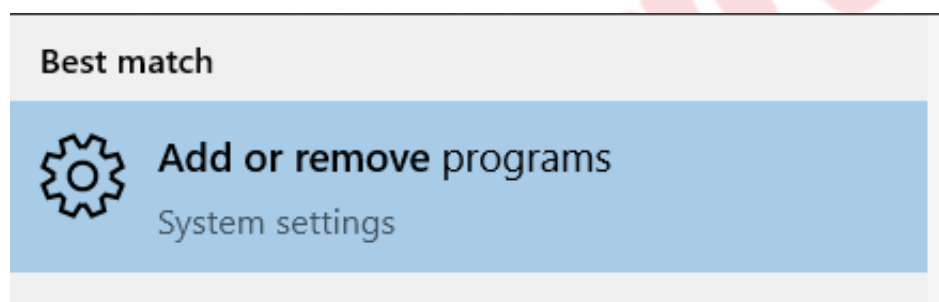
- compilator online oficial al Fundației Python – <https://www.python.org/shell/>
- Python anywhere – <https://www.pythonanywhere.com/>
- compilator online gratuit care acceptă mai multe limbaje de

programare – <https://repl.it/>

Ștergerea Python

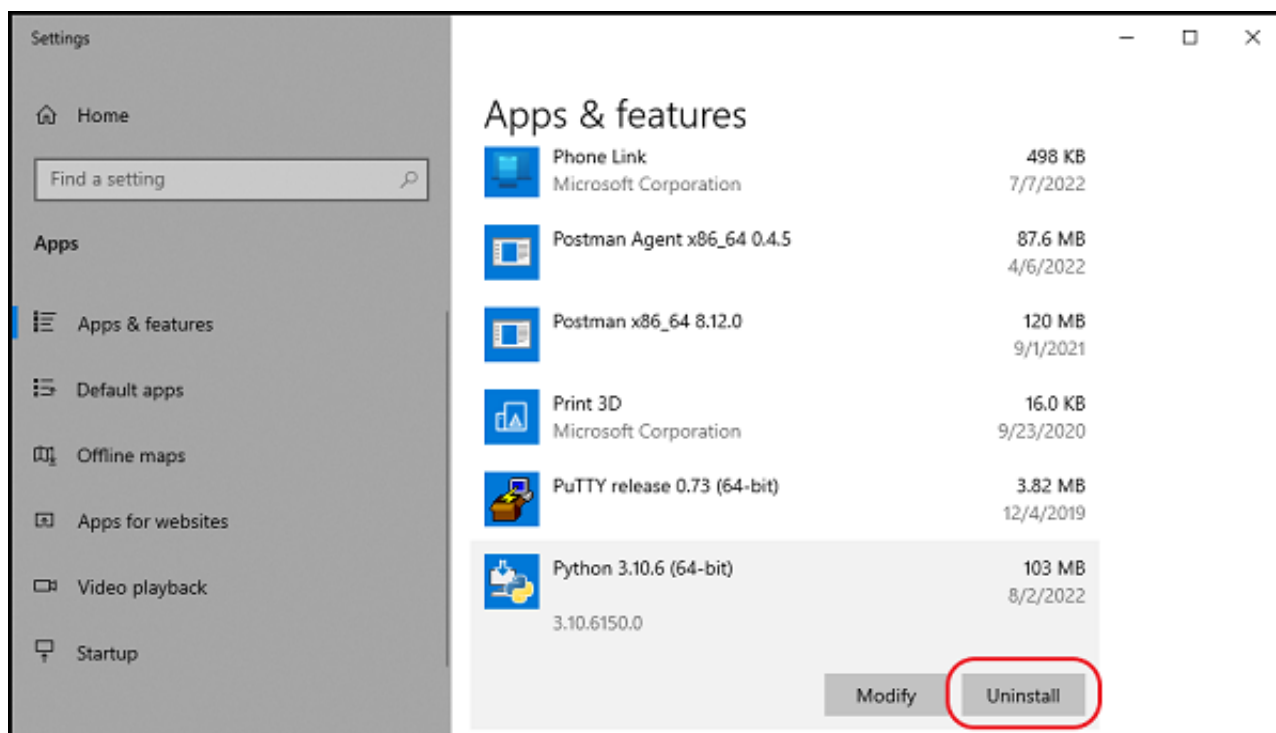
Dacă am instalat din greșeală versiunea greșită Python sau pur și simplu dorim să instalăm o versiune mai nouă/mai veche, putem face acest lucru urmând acești pași:

- tastați „Add or remove programs” în meniul Start și deschideți acea opțiune (imaginea 2.9.);



Imaginea 2.9. 'Add or remove programs' în meniul Start

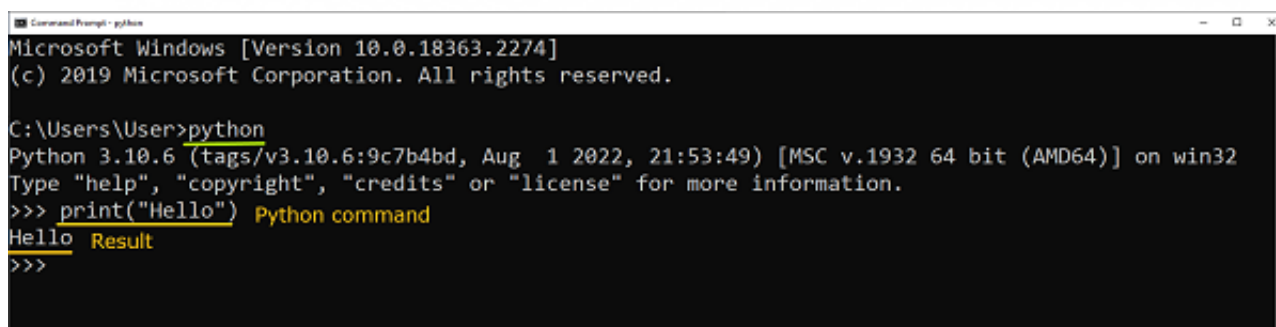
- în lista afișată de programe instalate, găsiți articolul Python cu versiunea specificată și dați clic pe Uninstall (imaginea 2.10.).



Imaginea 2.10. Ștergerea programului Python

Modul interactiv Python

Prin deschiderea consolei și tastând comanda **python**, este pornit modul interactiv Python. Acest mod implică tastarea directă a comenzilor limbajului Python în consolă și execuția lor imediată (imaginea 2.11.).



Imaginea 2.11. Modul interactiv Python și rezultatul comenzii Python executate

Comanda `print` în rezultat imprimă (tipărește) tot ceea ce i se transmite între paranteze rotunde. Textul este trecut între ghilimele duble, ca în exemplu, în timp ce numerele sunt trecute fără ghilimele, de exemplu, `print(5)`. Aceasta este doar o scurtă explicație a exemplului, iar despre regulile de codare și sintaxă vom afla mai târziu în curs.

Primul program Python

Codul sursă al programului Python este în format text, iar fișierele Python au, de obicei, extensia **py**.

Vom crea un fișier text (programul Notepad) și îl vom salva pe desktopul calculatorului.

Redenumiți fișierul și modificați-i extensia: Clic dreapta pe fișier -> **Rename** și să se numească, de exemplu, test (numele este arbitrar) cu extensia **py**: test.py

Deschideți fișierul și scrieți în el următorul conținut (comanda Python care imprimă textul Hey people!):

```
print("Hey people!")
```

După ce programul Python a fost creat, îl putem porni cu comanda consolei **python**. Este important ca în cadrul consolei să poziționați în folderul în care se află fișierul. Dacă acesta este desktop, atunci când deschideți consola, folosiți comanda `cd` (change directory) pentru a vă poziționa în folderul dorit: `cd desktop`.

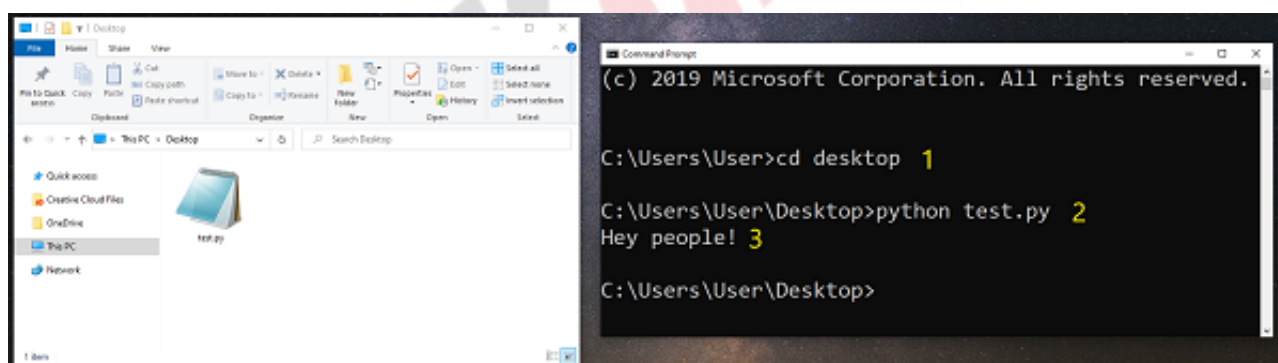
```
cd desktop
```

Pornirea unui program Python se face prin specificarea comenzii `python` și a numelui fișierului Python în care se află programul împreună cu extensia:

`python test.py`

În imaginea 2.12. sunt prezentați pașii descriși anterior. În partea dreaptă este prezentat fișierul Python creat, care este plasat pe ecranul calculatorului (desktop), iar în partea dreaptă sunt pașii:

1. poziționare în folderul dorit (`cd numele folderului`);
2. executarea fișierului Python (`python nameFile.py`);
3. prezentarea rezultatelor programului executat.



Imaginea 2.12. Crearea și executarea primului program Python

Mediul de lucru

Există diverse medii de lucru care pot fi folosite pentru scrierea și rularea programului Python; unele dintre ele sunt: PyCharm, Eclipse, Visual Studio Code.

În acest curs vom folosi mediul de lucru **Visual Studio Code**.

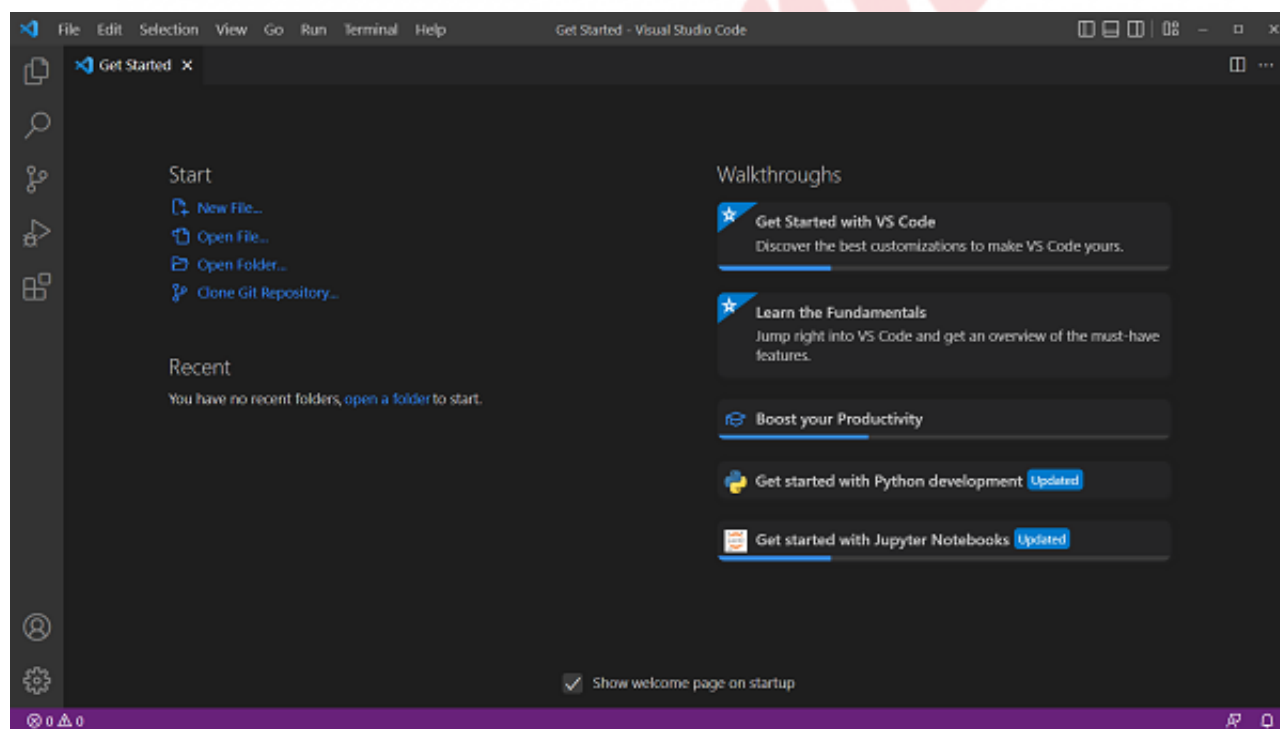
Visual Studio Code este un instrument puternic de codare care este disponibil pentru Windows, macOS și Linux. A fost dezvoltat de Microsoft. Acest mediu nu este legat doar de Python, ci poate fi folosit

și pentru a scrie programe în alte limbaje de programare.

De pe următoarea adresă puteți descărca Visual Studio Code care corespunde sistemului de operare al calculatorului dvs.

<https://code.visualstudio.com/download>

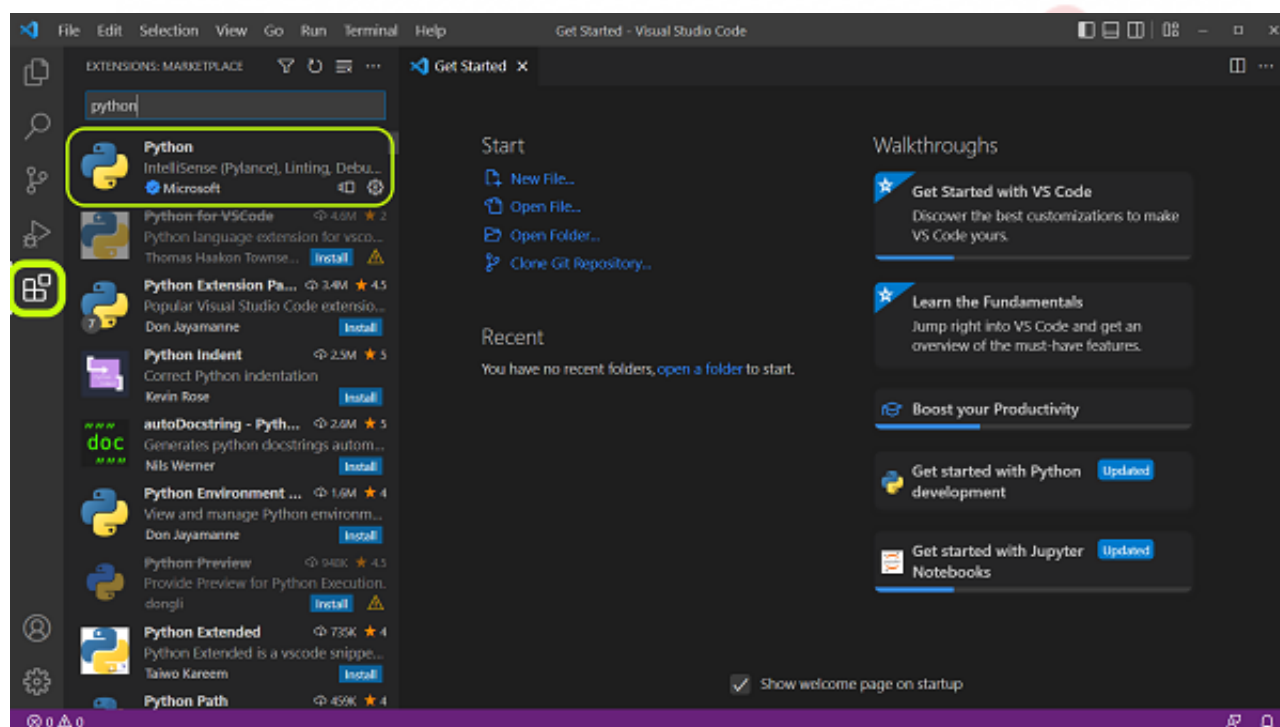
Când instalarea mediului este finalizată și este pornit pentru prima dată, apare fereastra de bun venit prezentată în imaginea 2.13.



Imaginea 2.13. Fereastra de bun venit a programului Visual Studio Code

Pentru ca Visual Studio Code să funcționeze bine cu Python, trebuie instalat suportul pentru această platformă. Cu un clic pe ultimul buton din stânga, după cum se vede în imaginea 2.14., se deschide o listă cu extensii diferite pentru diferite platforme. În câmpul de căutare puteți

introduce python, selectați prima extensie oferită, ce este afișată în imagine, și dați clic pe Install. În felul acesta, mediul va putea gestiona cu succes programele scrise în limbajul Python. În imaginea 2.14., lângă extensia menționată nu există butonul install, deoarece este deja instalat pe acel calculator, iar după instalare va fi la fel și la dvs.



Imaginea 2.14. Instalarea extensiei Python

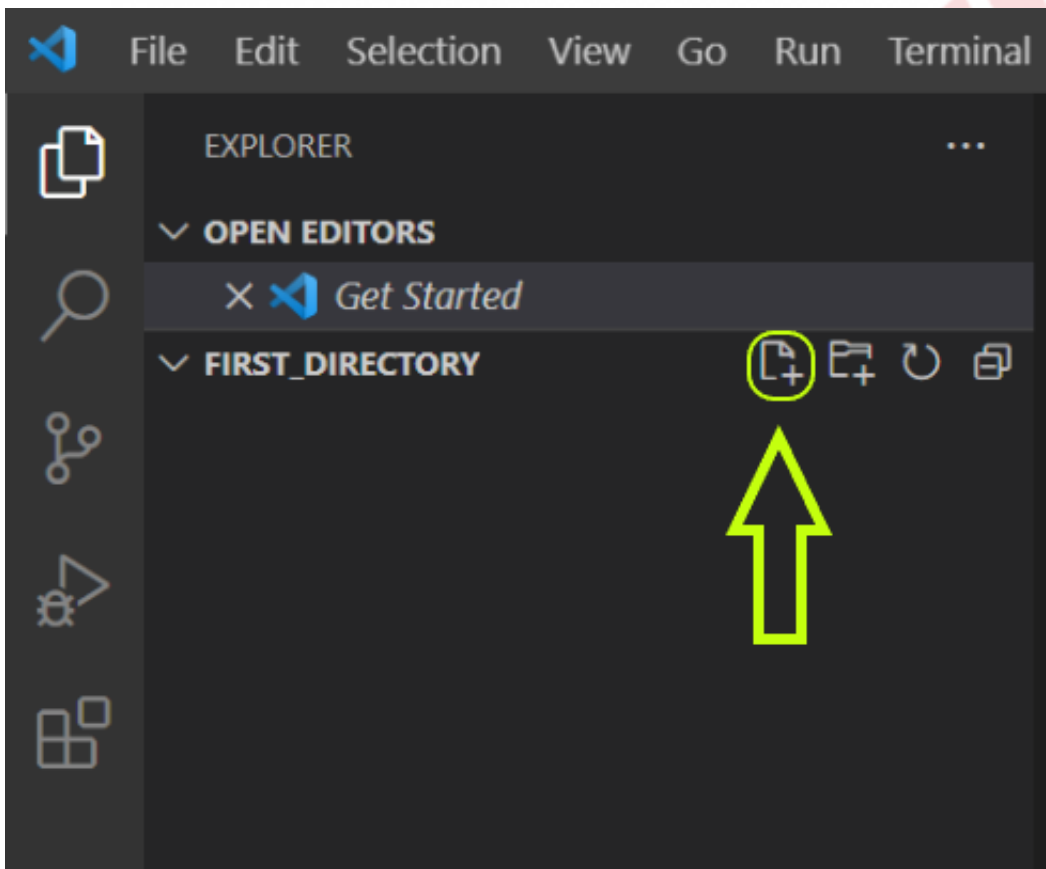
După ce extensia este instalată, putem începe să lucrăm în mediul pe care îl vom folosi în continuarea cursului.

Când este selectată opțiunea Open Folder..., mediul deschide un folder în care vor fi plasate fișierele cu care vom lucra. În scopul acesta, puteți crea un folder nou gol și îl puteți selecta după clic pe opțiunea Open Folder.

Mediul deschide apoi un folder, care este inițial gol (dacă selectați un folder existent de pe un calculator care are fișiere, de exemplu desktop, veți vedea în partea stângă exact acele fișiere). Este

convenabil să lăsați la început folderul gol pentru ca lucrul să fie mai transparent.

În imaginea 2.15. puteți vedea că a fost creat un folder nou gol numit `first_directory`. Crearea unui fișier nou Python în folder se face cu un clic pe prima opțiune prezentată în imaginea 2.15. Denumiți arbitrar fișierul și neapărat setați extensia **.py**.



Imaginea 2.15. Crearea unui fișier nou Python

În cadrul fișierului, vom scrie din nou comanda `print` care va imprima textul arbitrar.

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

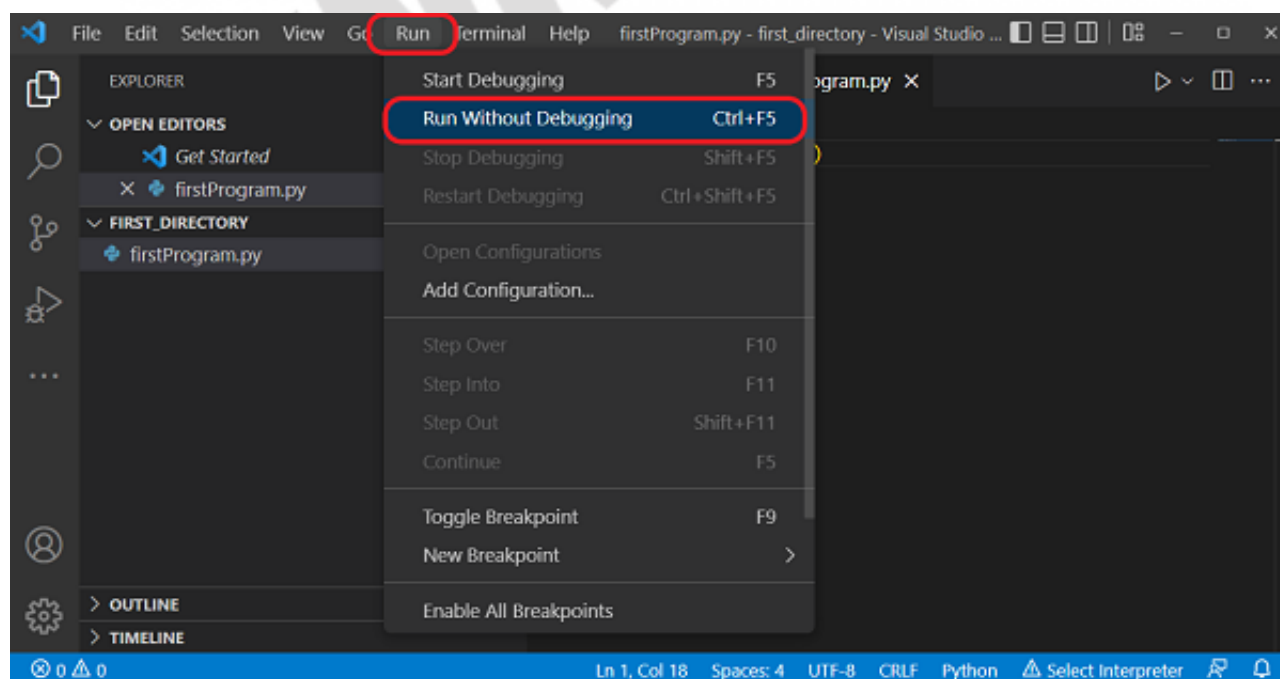
Pornirea programului

Fișierul Python creat poate fi pornit cu opțiunile:

- Run -> Start Debugging sau
- Run-> Run Without Debugging

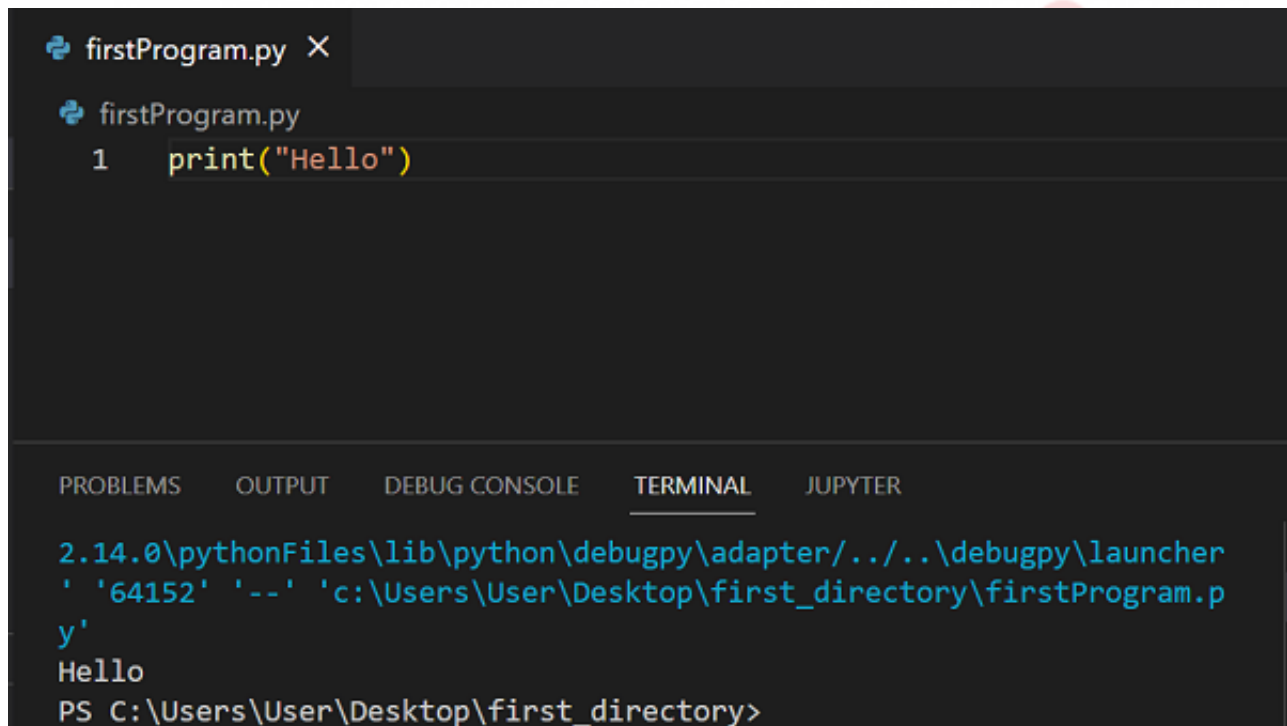
Start Debugging pornește programul, dar permite execuția programului într-un anumit moment sau să se oprească sau să continue execuția pas cu pas, pentru a verifica starea programului în acel moment anume. Acest proces de debugging este folosit cel mai des pentru localizarea mai ușoară a erorilor dintr-un program.

Run Without Debugging, de asemenea, pornește programul, dar execuția programului nu se va opri la niciun punct de întrerupere.



Imaginea 2.16. Pornirea programului Python

Deocamdată vom selecta a doua opțiune, Run Without Debugging, după cum este prezentat în imaginea 2.16., iar după selectarea acesteia în consolă, în partea de jos a ferestrei, sub fila **Terminal**, va fi prezentat rezultatul executării programului – va fi imprimat mesajul Hello, așa cum este prezentat în imaginea 2.17.



```
firstProgram.py X
firstProgram.py
1 print("Hello")

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
2.14.0\pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher
' '64152' '-- 'c:\Users\User\Desktop\first_directory\firstProgram.p
y'
Hello
PS C:\Users\User\Desktop\first_directory>
```

Imaginea 2.17. Prezentarea consolei și a rezultatelor execuției programului

În consolă, pe lângă rezultate, dacă există erori în program, sunt și ele imprimate acolo.

Exerciții

Exercițiul 1

Conform exemplului prezentat mai sus, încercați să creați un fișier nou

Python numit exercise1.py, care va imprima următorul mesaj atunci când este pornit:

Python programming language

Soluție

```
print("Python programming language")
```

[1 https://freepngimg.com/png/14702-python-logo-png-image](https://freepngimg.com/png/14702-python-logo-png-image)