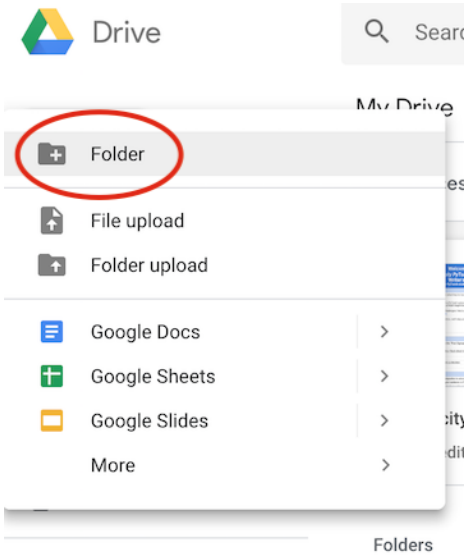
Utilizare Python 3 in cadrul laboratoarelor de AI

Folosirea Google Colab pentru a rula Python in cloud

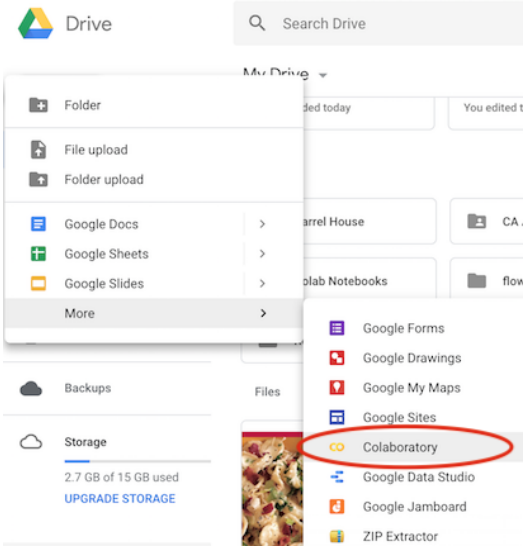
- Google Colab este un serviciu de rulare a codului Python in cloud.
- Este bazat pe Jupyter Notebooks si asigură acces gratuit la GPU (pentru aplicaţii complexe ce necesită putere mare de calcul).
- Pot fi utilizate (teoretic) toate bibliotecile disponibile pentru Python, iar fişierele se salvează în contul Google Drive.
- Este interoperabil cu Jupyter Notebook, Github, Kaggle

Setare drive

Este necesară iniţial logarea în contul Google. Cu toate că un notebook nou în Google Colab poate fi creat independent de Drive, este util să creăm un folder în care să păstrăm ceea ce lucrăm

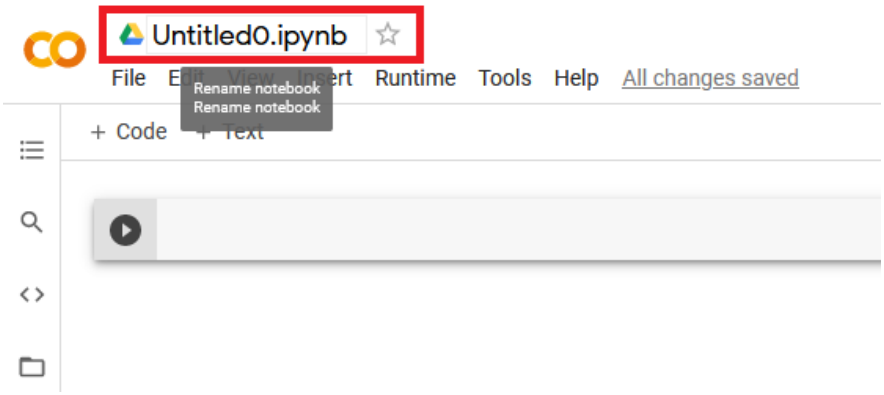


Un nou notebook poate fi creat din Drive, din meniul More → Colaboratory.



Deasemnea, poate fi lansat direct https://colab.research.google.com

Numele fişierului poate fi modificat facând click penumele notebookului sau din meniul File→ Rename.

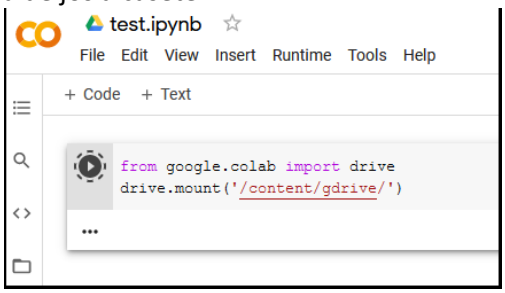


Pentru ca notebookul sa fie salvat pe drive (daca nu a fost creat acolo), alaturi de eventual alte fisiere ce rezulta din rularea unor aplicatii este nevoie ca Google Drive sa fie conectat la Colab prin rularea urmatorului cod:

```
from google.colab import drive
drive.mount('/content/gdrive/')
```

Rularea codului

In Google Colab codul Python este scris in casete pentru cod, ce pot fi rulate independent, urmand ca rezultatul sa fie afisat in partea de jos a casetei.



Pot fi introduse si casete de tip text pentru a scrie explicaţii referitoare la cod şi/sau pentru a separa codul pentru o mai bună gestionare.

Avantaje utilizare Google Colab

- Acces direct din browser, fara a fi necesare instalari locale; independenţă faţă de staţia de lucru.

- Biblioteci preinstalate. Cele mai uzuale biblioteci sunt deja instalate.
- Datele sunt salvate in Clud-ul Google Drive.
- Colaborativ. În proiecte cu mai mulţi developeri, codul poate fi partajat. Conţinutul poate fi share-uit rapid cu oricine.
- Acces gratuit la GPU si TPU, util pentru taskuri ce necesita putere mare de calcul.

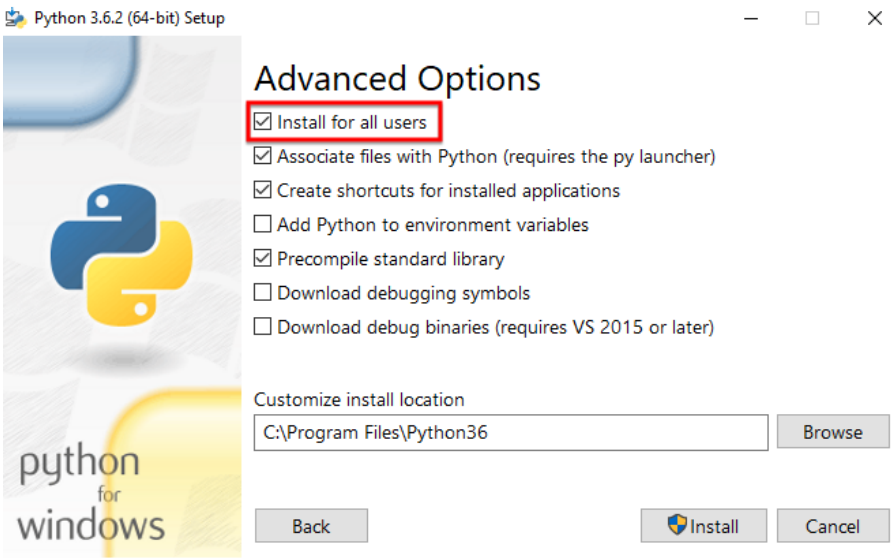
Instalare locală Python (alternativ se poate instala Anaconda [vezi mai jos])

Pas1. Se downloadează pachetul Python 3 pentru Windows: http://www.python.org/download/

Pas2. Se instalează pachetul Python 3. Vor fi instalate cel putin urmatoarele componente:

- IDLE – prompter(linie de comandă) + editor de cod sursă optimizat pentru sintaxa Python;
- pip – instalare de pachete Python
- Documentation – fisiere cu documentatie

Vor fi selectate obligatoriu optiunile „Install for all users” si „Add Python to environment variables”.



Pas3. Selectarea directorului de lucru. Acesta este folderul din care interpretorul Python va accesa fișiere în mod direct, fără să fie necesară explicitarea căilor.

În mod implicit, folderele de lucru sunt rădăcina Python3 și Python3\Lib, dar se recomandă crearea unui folder nou, de exemplu: C:\Python3\Exemple (e riscant să se lucreze pe folderele implicite, care conțin o serie de fișiere de bază pentru funcționarea limbajului și a bibliotecilor implicite).

3.a.Se creează directorul de lucru C:\Python3\Exemple
3.b.Se memorează acest folder în variabila de mediu PYTHONPATH, în felul următor:
- My Computer(This PC) – Properties – Advanced System Settings – Environment Variables (această rubrică listează variabilele de mediu setate pentru Windows)
- cu butonul New (din System Variables) se creează variabila PYTHONPATH cu valoarea C:\Python3\Exemple (pe unele sisteme poate necesita restartare)
- de la acest moment, fișierele Python trebuie salvate în acest folder.

Pas4. Verificarea instalării

Lucru în linia de comandă Python:
Se lansează mediul de lucru IDLE, apare prompterul
>>>
Se tastează o atribuire:
>>> a=2
Se afișează valoarea variabilei a:
>>> a
2
În felul acesta, la linia de comandă se pot exersa în timpul învățării diverse funcții și operații Python, precum cele prezentate în acest tutorial.

Instalare Notepad++

Notepadd++ este un editor pe care il vom utiliza la scrierea programelor in Python. In cazul in care nu il aveti deja instalat pe calculator, se descarca de la adresa: https://notepad-plus-plus.org/download
Din meniul Language se va selecta Python.
Din meniul Settings-->Preferences-->Language ne vom asigura ca Tab Size are valoarea 4 si este bifata casuta „Replace by space”.

Python este un limbaj de programare pentru care spatierea si indentarea sunt importante, intrucat indentarea tine locul instructiunii {Begin , End}.

Vom verifica functionarea Notepad++ si Python prin scrierea unui program in editorul Notepadd++:

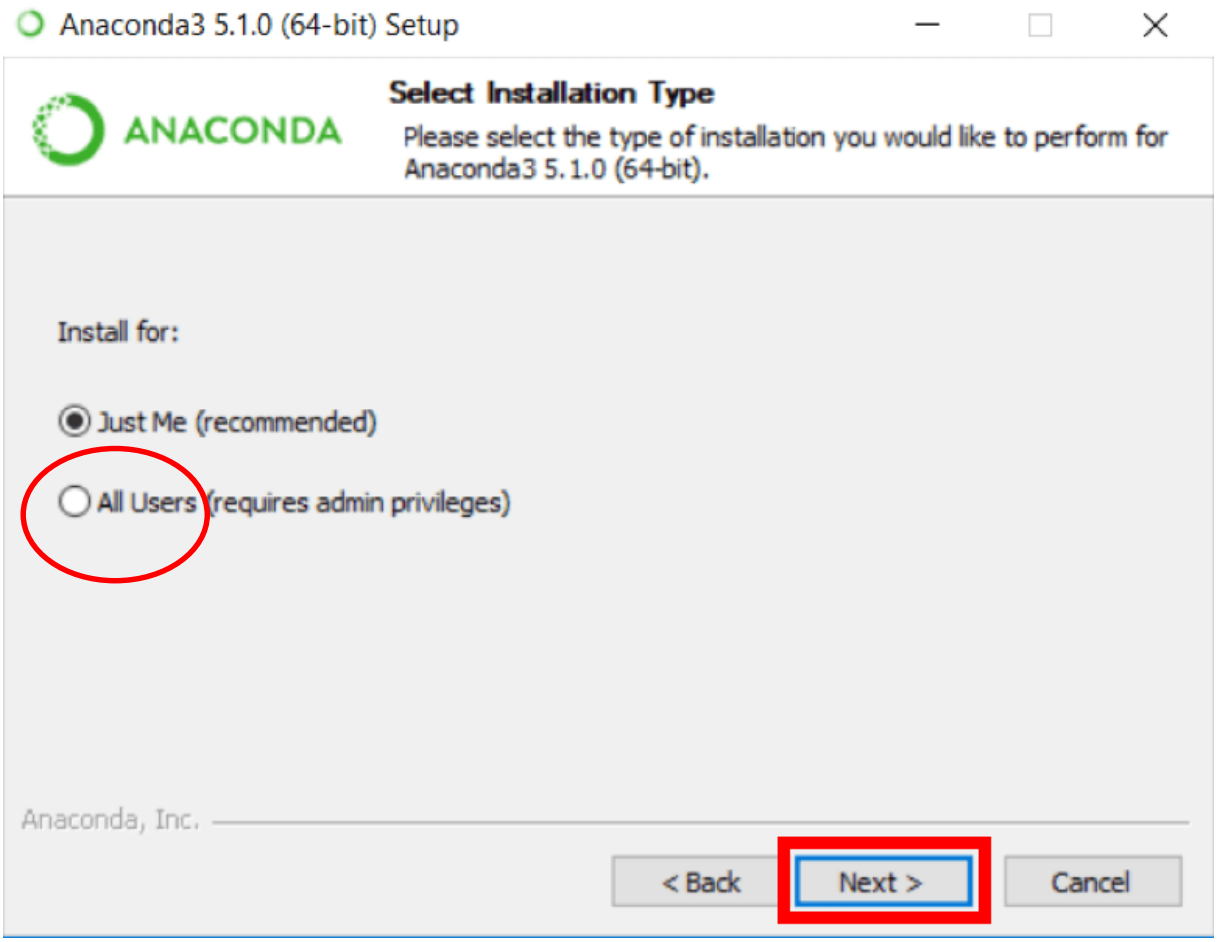
```
print ('hello world')
```

Va fi salvat cu denumirea first.py in directorul de lucru C:\Python3\Exemple\[Nume student], unde [Nume student] este directorul cu numele dumneavoastra.

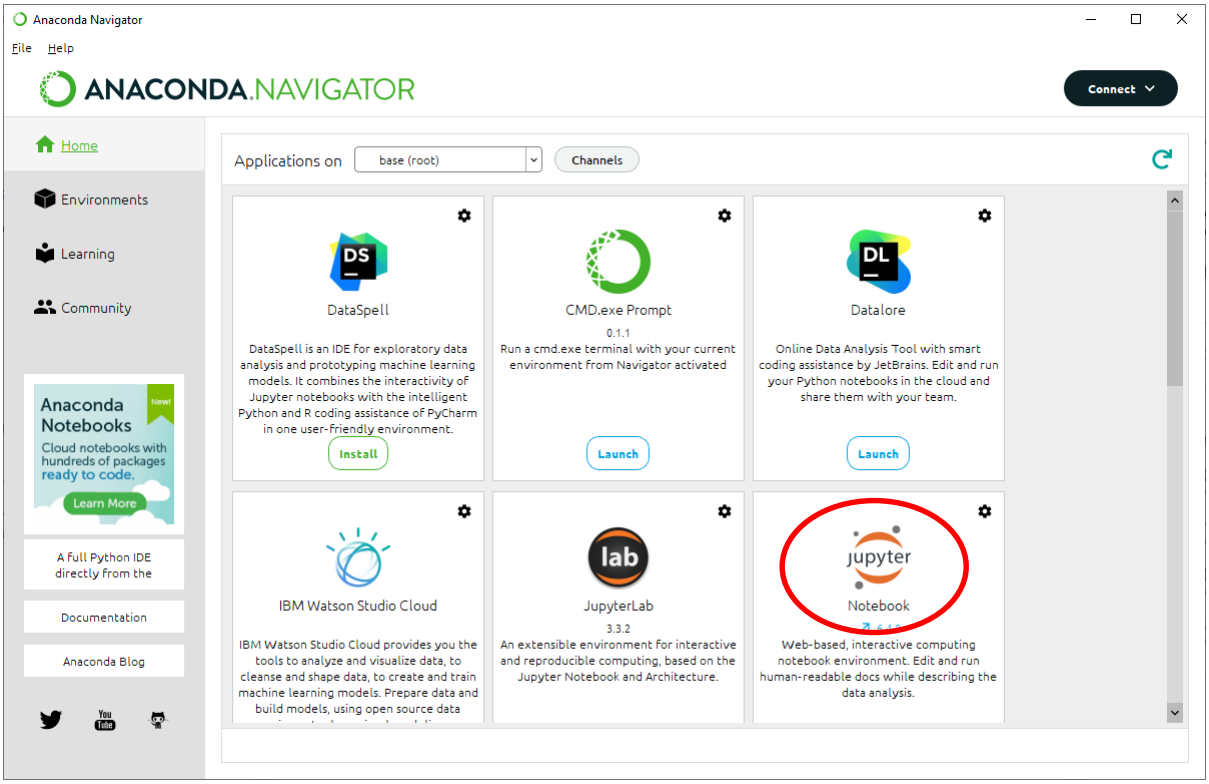
Lansarea in executie se va face din Command Prompt-ul Windows. Apasati butonul Start si tastati comanda „CMD”.
Setati directorul curent folosind comanda CD in : C:\Python3\Exemple\[Nume student]
Se lanseaza in executie programul first.py din command prompt, acesta afisand mesajul *hello world*.

Instalare locală Anaconda

1. Intraţi pe site-ul Anaconda şi descărcaţi installer-ul. Atenţie, versiunea de Python inclusă trebuie sa fie din generaţia 3.X
2. Dacă lucraţi pe calculatorul personal, puteţi utiliza varianta de instalare pentru toţi utilizatorii, altfel este recomandat să se instaleze doar pentru userul cu care sunteţi logat.



3. După instalarea Anaconda, veţi rula Jupyter Nobook din interiorul aplicaţiei Anaconda Navigator.



4. Jupyter Notebook se va deschide in browser-ul default de pe calculator.

Notă. Pentru a deschide un notebook intr-un folder preferat, puteti utiliza Powershell Prompt (tot din Anaconda Navigator) si sa utilizati comenzi ca si in Command Prompt pentru a deschide acel folder. Lansarea se va face cu comanda *jupyter notebook*

În acest mod notebookul va fi salvat în folderul indicat şi nu în folderul default.

Particularități sintactice ale limbajului Python

Python este un limbaj open source multiparadigmă (obiectual, funcțional, scriptural, aspect-oriented etc.). O caracteristică sintactică importantă este lipsa instrucțiunilor sau caracterelor de delimitare a blocurilor. În locul acestora se folosește indentarea.

Exemplu:

```
for x in colectie:
        instructiunile blocului

sau

if a==0:
        instructiuni
else:
        instructiuni
```

Blocurile subordonate structurilor for, if, etc. se indentează spre dreapta. La finalul blocului, se șterge indentarea revenind cu cursorul spre stânga. Aceasta face sintaxa mai ușor de citit, evitând construcții suplimentare ca în alte limbaje, de genul:

```
for i=1 to n
        {
        instructiuni
        }
```

sau

```
for i=1 to n
        instructiuni
endfor
```

Pe de altă parte apar unele riscuri, sunt mai dificil de detectat cu ochiul liber blocurile încadrate eronat, în special când se fac indentări multiple (blocuri în blocuri).

Variabile, expresii şi instrucţiuni

Valori si tipuri

Instructiunea print afiseaza valoarea argumentului sau:

```
print(4)
4
```

Atunci cand nu cunoastem tipul unei valori, putem apela la functia type:

```
type('Hello, World!')
<class 'str'>

 type(17)
<class 'int'>

type(3.2)
<class 'float'>
```

În cazul unor valori mari, scrise cu virgule (1,000,000), acestea nu vor fi interpretate de catre Python ca si un numar:

```
print(1,000,000)
1 0 0
```

Variabile

Variabilele sunt create prin operatia de atribuire:

```
Mesaj='Variabilele pot avea diferite valori'
n = 17
pi = 3.1415926535897931
```

Exemplul de mai sus face trei atribuiri. Pentru a afisa valorile unei variabile, vom folosi instructiunea print:

```
print(n)
17

print(pi)
3.141592653589793
```

Numele variabilelor şi cuvinte rezervate

De obicei, numele variabilelor se aleg astfel încât să aibă un înţeles pentru cei care le utilizează şi să fie sugestive pentru scopul în care au fost definite. Acestea pot să aibă o lungime arbitrară, pot conţine litere şi cifre, dar nu pot începe cu cifre.

Numele variabilelor sunt *case sensitive,* aşadar *Nume* nu este acelasi lucru cu *nume*.

Iată câteva exemple de nume greşite de variabile:

```
76trombones = 'big parade'
SyntaxError: invalid syntax

more@ = 1000000
SyntaxError: invalid syntax

class = 'Advanced Theoretical Zymurgy'
SyntaxError: invalid syntax
```

Puteti identifica motivele pentru care Python a generat mesajul de eroare?

Python are un număr de 33 cuvinte rezervate:

and	except	lambda	with
as	finally	nonlocal	while
assert	false	None	yield
break	for	not	
class	from	or	
continue	global	pass	
def	if	raise	
del	import	return	
elif	in	True	
else	is	try	

Dacă interpretatorul afişează o eroare legată de numele unei variabile, puteţi verifica dacă nu cumva se află în această listă.

Declaraţii şi instrucţiuni

O secvenţă de program conţine de obicei o serie de declaraţii şi instrucţiuni.

Secvenţa:

```
print(1)
x = 2
print(x)
```

produce următorul rezultat:

```
1
2
```

Declaraţiile nu produc rezultate.

Operatori şi operanzi

Operatorii sunt simboluri speciale care reprezintă calcule cum sunt adunarea sau înmulţirea. Valorile asupra cărora sunt aplicaţi operatorii se numesc operanzi.

Operatorii + , - , * , / şi ** realizează operaţii de adunare, scădere, multiplicare, împărţire şi exponent:

```
20+32
hour-1
hour*60+minute
minute/60
5**2
(5+9)*(15-7)
```

Expresii

O expresie este o combinaţie de valori,variabile şi operatori. O valoare în sine este considerate o expresie şi deasemnea o variabilă. Astfel, următoarele sunt valide:

```
17
x
x + 17
```

Ordinea operaţiilor

Când mai mult de un operator apare într-o expresie, ordinea evaluării depinde de regulile de precedenţă. Pentru operaţiile matematice Python urmează convenţia matematică, folosind acronimul PEMDAS.

Parantezele, Exponentul, Multiplicatorul, Divizarea, Adunarea, Scăderea.

Operatorul modulo

```
rest= 7 % 3
print(rest)
```

Operaţii cu şiruri de caractere

Operatorul + operează şi cu şiruri de caractere, dar nu în sensul mathematic, ci realizează o concatenare.

Verificaţi urmatoarele secvenţe:

```
primul=10
secund=15
print(primul+secund)
```

```
primul='100'
second='150'
print(primul+secund)
```

Operatorul * poate fi utilizat şi el cu şiruri de caractere, mutiplicând conţinutul şirului de caractere cu un întreg.

```
primul='Test '
second=3
print(primul*secund)
```

Interacţiunea cu utilizatorul

Python utilizează funcţia dedicată input pentru a citi informaţii de la tastatură, sub formă de şir de caractere.

Secvenţa:

```
inp=input()
print (inp)
```

Va permite utilizatorului să introducă un text de la tastatură:
> Ana are mere

Şi va afişa mesajul introdus:

Ana are mere

De obicei, este indicat să fie transmisă o indicaţie utilizatorului pentru a şti ce anume să introducă.

```
nume=input('Cum va numiti?\n')
print nume
```

Se va afişa:
>Cum va numiti?

Se va introduce:
>Ana

Va fi afişat:
Ana

Parametrul \n de la finalul intrebarii reprezintă marcatorul pentru o linie nouă.

Întrucât citirea de la tastatură se face sub formă de şir de carcatere, numerele introduse trebuie convertite din tipul string in int sau float.

Comentariile

Comentariile în Python se scriu folosind simbolul #

```
# calculeaza procentul dintr-o ora
percentage = (minute * 100) / 60
```

în acest caz comentariul apare pe o singura linie. Acesta poate fi asezat si la finalul unei linii:

```
percentage = (minute * 100) / 60 # procente dintr-o ora
```

Tot ce este scris după # este ignorant de interpretator.

Pentru mai multe blocuri se foloseste ‘’’:

```
'''
nume=input('Cum va numiti?\n')
print nume
'''
```

Exerciţii:

1. Scrieti un program care solicită utilizatorului introducerea de la tastatură a numărului de ore de lucru şi rata orară, calculând apoi salariul brut.

2. Scrieti un program care solicita introducerea temperaturii în grade Celsius, o converteste în Fahrenheit si afiseaza valoarea transformata.

3. Având urmatoarele atribuiri,

   ```
   latime=17
   inaltime=12.0
   ```

   scrieti valoarea expresiei si tipul rezultat:

   a. latime//2
   b. latime/2.0
   c. inaltime/3
   d. 1+2*5

   Folositi interpretatorul Python pentru a verifica răspunsurile.

Bibliografie:

Charles R. Severance, Python for Everybody – Exploring Data Using Python 3, 2016, www.py4e.com