

# 1 Setarea mediului de lucru

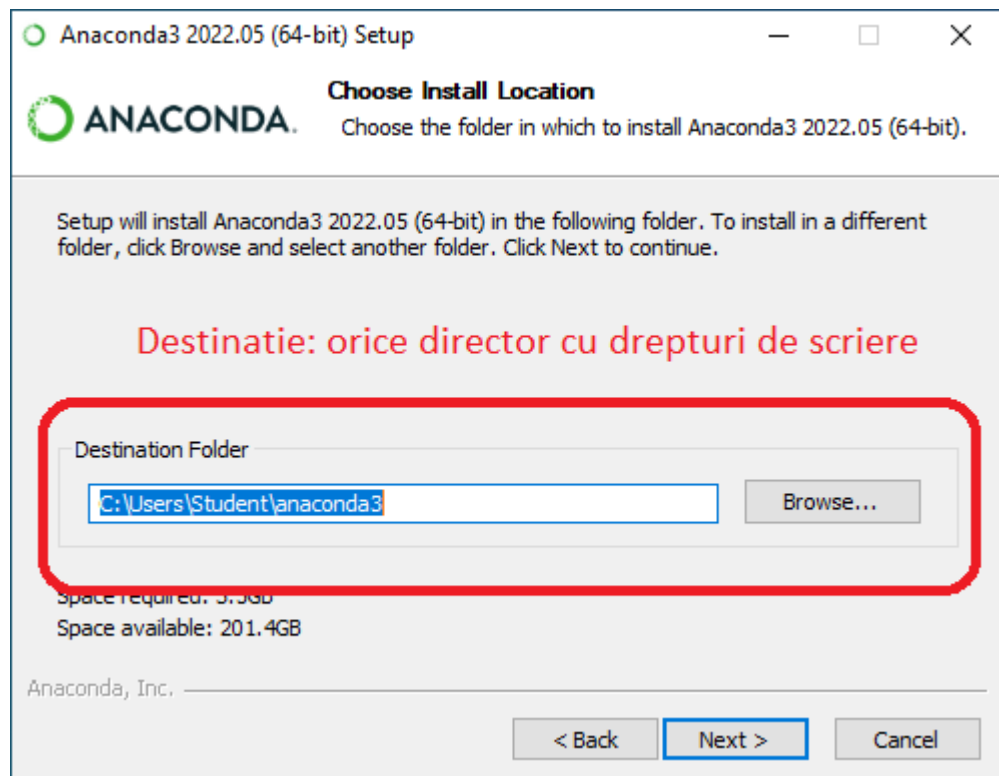
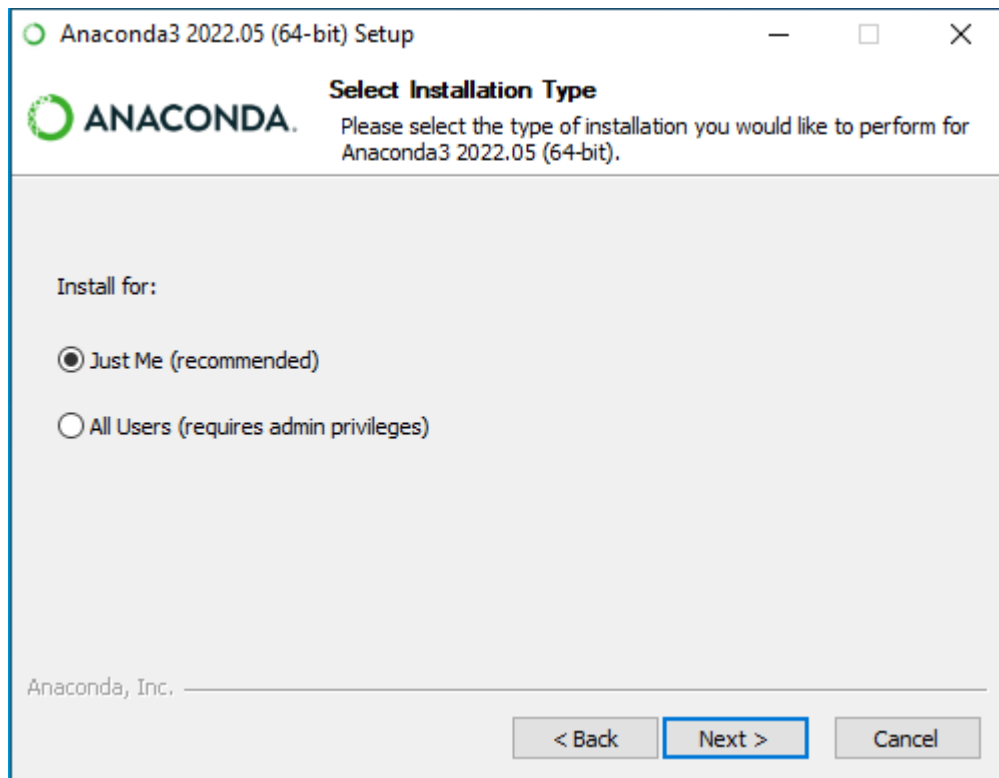
- Instalare Python si auxiliare
  - Instalare Anaconda
  - Update Anaconda
  - Creare de mediu virtual
  - Instalare de pachete cu pip
- Medii de lucru
  - PyCharm
  - Jupyter notebook

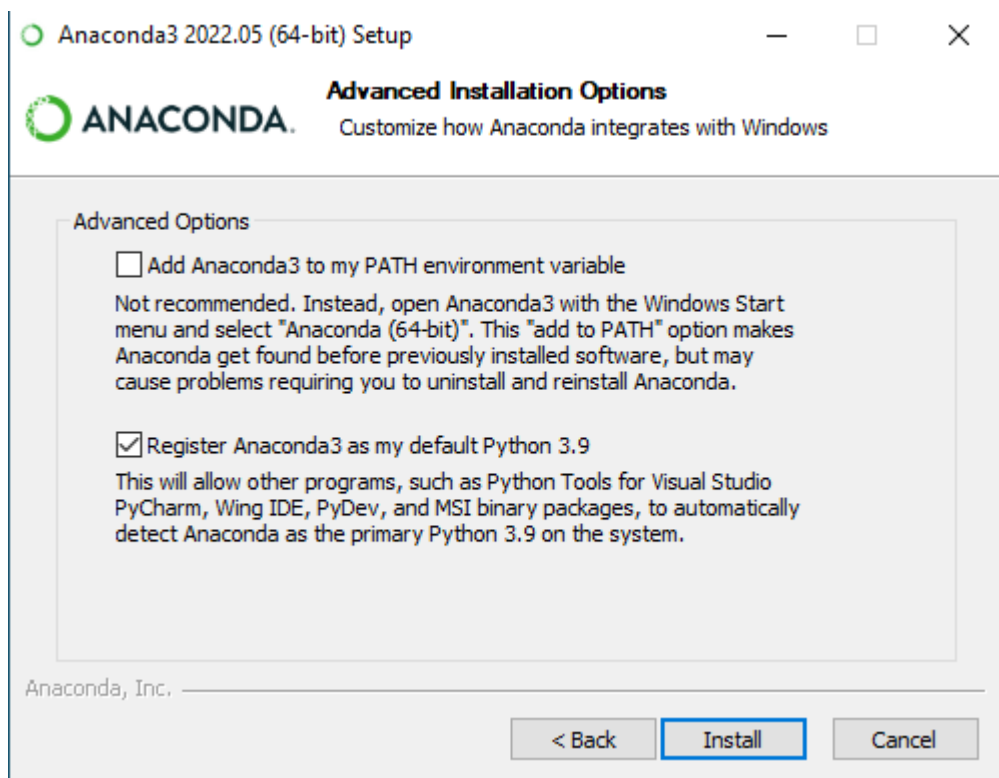
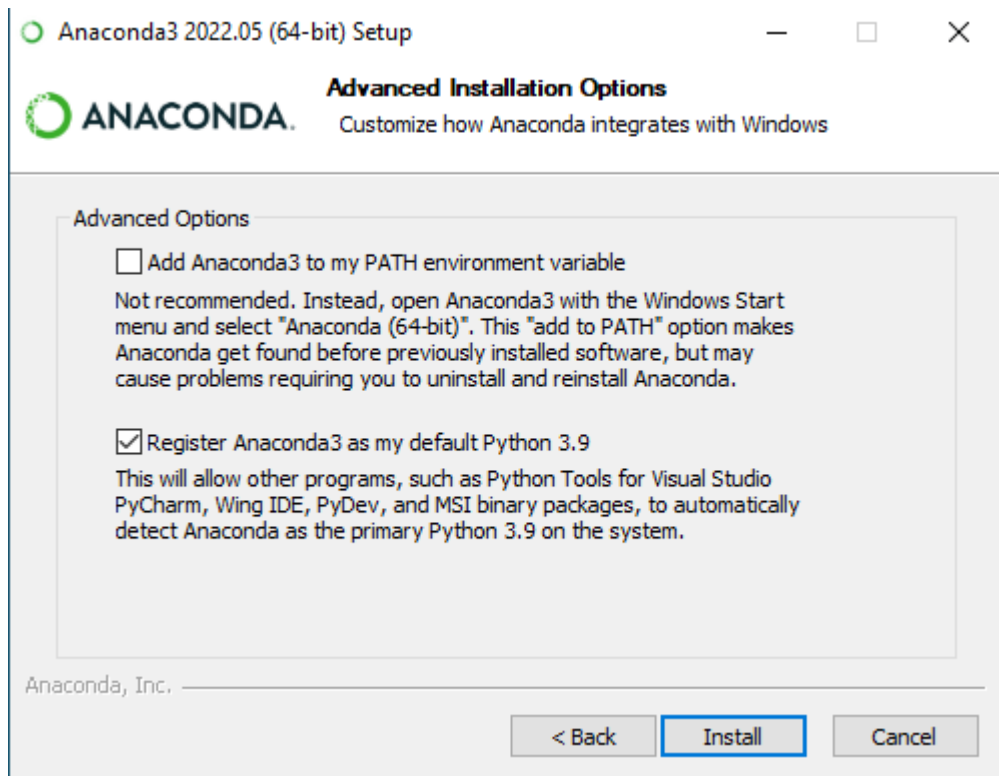
## 1.1 Instalare Anaconda

In cadrul laboratorului se va folosi distributia de Python Anaconda. Kitul Anaconda se instaleaza de la adresa <https://www.anaconda.com/distribution/> (<https://www.anaconda.com/distribution/>).

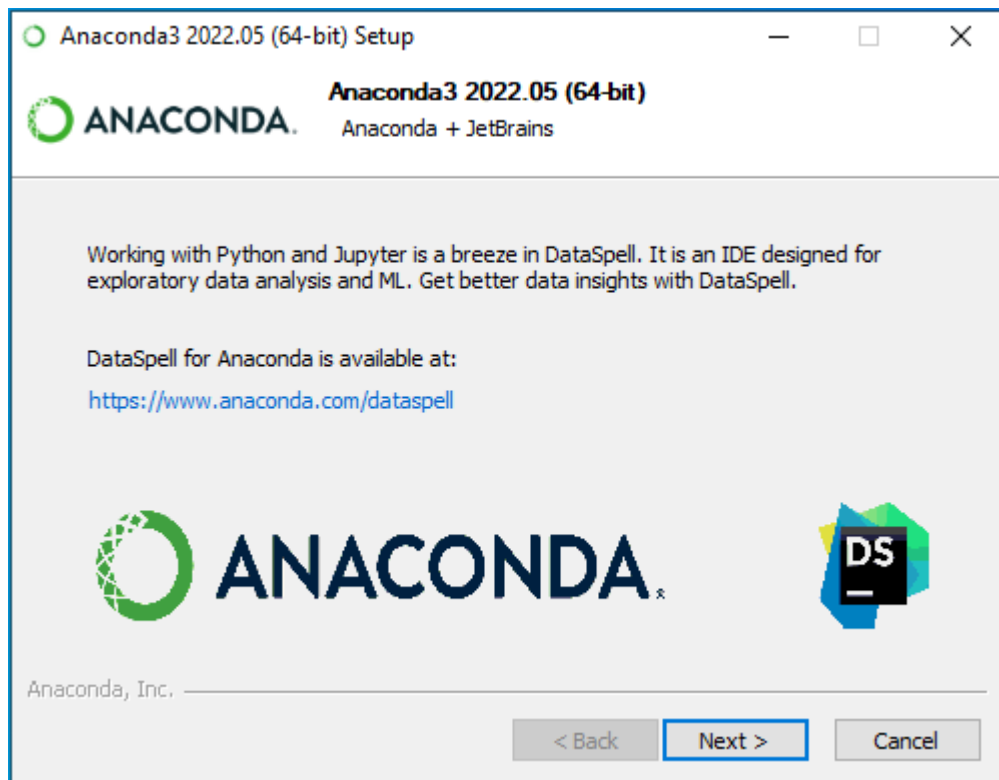
Se poate face instalarea Anaconda fara drepturi de administrator. Pasii sunt dati in capturile de ecran de mai jos. Daca sistemul de operare este pe 64 de biti, folositi installer de 64 de biti.







Se poate ca in final sa se instaleze VS Code sau PyCharm, ce ofera suport pentru scrierea de proiecte Python:



Pentru PyCharm, contul de student cu credentiale [...@student.unitbv.ro](mailto:...@student.unitbv.ro) (<mailto:...@student.unitbv.ro>) permite descarcarea de PyCharm Professional Edition. Altfel, este disponibilă versiunea Community.

## ▼ 1.2 Update Anaconda

Interfața în linia de comandă, particularizată pentru Anaconda, se obține din Start->Anaconda->Anaconda prompt.

În linia de comandă, actualizarea pachetelor se face folosind utilitarul `conda`. Se recomandă actualizarea pachetului `conda` în mod regulat, prin:

```
conda update conda --yes
```

unde opțiunea `--yes` inhibă interogarea utilizatorului pentru acord de instalare. Lista pachetelor instalate se obține cu:

```
conda list
```

```
Administrator: Anaconda Prompt
Solving environment: done

# All requested packages already installed.

(base) C:\Users\Lucian>conda list
# packages in environment at C:\Anaconda3:
#
# Name                    Version            Build    Channel
ipyw_jlab_nb_ext_conf    0.1.0              py37_0
jlabaster                 0.7.12             py37_0
anaconda                  custom             py37_0
anaconda-client           1.7.2              py37_0
anaconda-navigator        1.9.6              py37_0
anaconda-project          0.8.2              py37_0
asn1crypto                0.24.0             py37_0
astroid                   2.1.0              py37_0
astropy                   3.1.1              py37he774522_0
atomicwrites              1.3.0              py_0
attrs                     18.2.0             py37h28b3542_0
babel                     2.6.0              py37_0
backcall                  0.1.0              py37_0
backports                 1.0                py37_1
backports.os              0.1.1              py37_0
backports.shutil_get_terminal_size 1.0.0              py37_2
beautifulsoup4            4.7.1              py37_1
```

În dreptul fiecărui pachet este scrisă versiunea sa.

Pachetele se actualizează folosind:

```
conda update --all --yes
```

Dacă se dorește actualizarea doar a unui pachet, atunci se specifică similar cu:

```
conda update scikit-learn
```

Dacă se dorește aducerea unui pachet la o anumită versiune, se folosește:

```
conda install pandas=0.13.1
```

Uneori se dorește instalarea unui pachet care nu e în canalul oficial de pachete Anaconda. Prin opțiunea `--channel` :

```
conda install pytorch torchvision cudatoolkit=9.0 -c pytorch
```

În decursul instalării se stochează pe disc pachetele aduse. Pentru a elibera spațiul ocupat, se folosește:

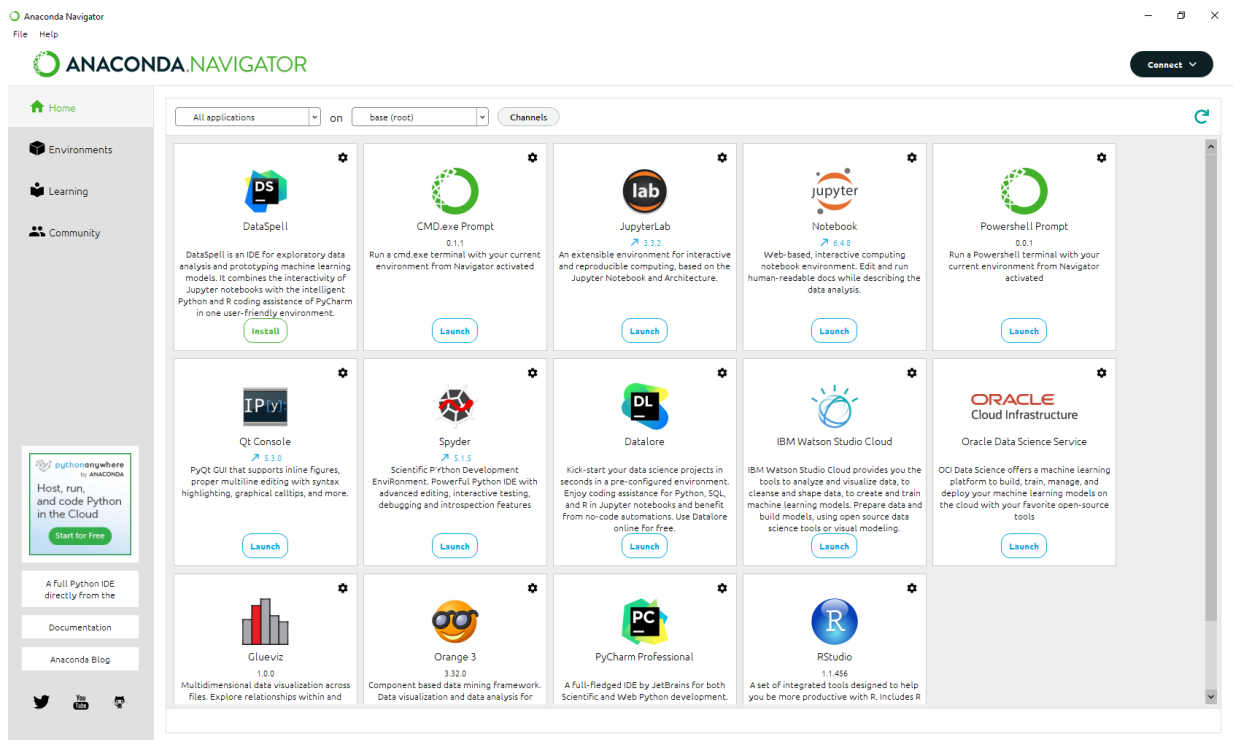
```
conda clean --all --yes
```

Alte opțiuni ce pot fi date utilitatului `conda` sunt descrise la

<https://conda.io/projects/conda/en/latest/user-guide/tasks/index.html>  
(<https://conda.io/projects/conda/en/latest/user-guide/tasks/index.html>).

**Nota:** în loc de `conda` se poate folosi managerul de pachete [mamba](https://mamba.readthedocs.io/en/latest/)  
(<https://mamba.readthedocs.io/en/latest/>).

Mentionăm că există și utilitar grafic - Anaconda Navigator - livrat odată cu instalarea Anaconda prin care se poate face gestiunea pachetelor, configurarea mediilor virtuale și altele:



## 1.3 Creare de mediu virtual

Frecvent este nevoie de crearea unui mediu virtual, in care sa fie instalate anumite pachete, corespunzatoare unei anumite versiuni a limbajului Python. De exemplu, pot exista pachete care functioneaza numai cu Python 3.6, iar versiunea de baza instalata este 3.7. Altele e nevoie de folosirea unor pachete de versiune mai vechi si nu dorim sa face downgrade in pachetul de baza. Se prefera in aceste cazuri crearea de medii virtuale specifice.

Crearea unui mediu virtual numit "machine-learning", continuand toate pachetele din distributia anaconda se face cu comanda `conda create` :

```
conda create --name machine-learning anaconda --yes
```

Daca se doreste ca versiunea de limbaj Python sa fie una anume, e.g. 3.6, se foloseste:

```
conda create --name machine-learning anaconda python=3.6 --yes
```

Lista mediilor virtuale se obtine cu:

```
conda info --envs
```

Mark Anaconda Prompt

```
(base) C:\Users\IEUser>conda info --envs
# conda environments:
#
base                    * C:\work\Anaconda3
machine-learning        C:\work\Anaconda3\envs\machine-learning

(base) C:\Users\IEUser>
```

Activarea mediului virtual creat anterior se face cu:

```
conda activate machine-learning
```

Numele mediului virtual este dat la inceputul prompt-ului:

Anaconda Prompt

```
(base) C:\Users\IEUser>conda info --envs
# conda environments:
#
base                    * C:\work\Anaconda3
machine-learning        C:\work\Anaconda3\envs\machine-learning

(base) C:\Users\IEUser>conda activate machine-learning
(machine-learning) C:\Users\IEUser>
```

Se recomanda calduros rulara periodica de comanda de update a pachetelor din interiorul mediilor virtuale, prin `conda update` .

Deactivarea mediului virtual se face cu `conda deactivate` .

Stergerea completa a mediului virtual `machine-learning` se face cu:

```
conda remove --name machine-learning --all --yes
```

Alte comenzi utile pentru gestiunea mediilor virtuale (de exemplu clonarea unui mediu virtual) se gasesc la <https://conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html> (<https://conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html>).

## ▼ 1.4 Instalare de pachete cu pip

Traditional, instalarea de pachete in Python se face cu comanda `pip` . In Anaconda este utila comanda pentru cazul in care pachetul dorit nu poate fi instalat cu comanda `conda`.

Exemplu:

```
pip install dtw
```

```
Anaconda Prompt (machine-learning)

(machine-learning) C:\Users\IEUser>pip install dtw
Collecting dtw
  Downloading dtw-1.3.3.tar.gz
Requirement already satisfied: numpy in c:\work\anaconda3\envs\machine-learning\lib\site-packages (from dtw) (1.15.4)
Requirement already satisfied: scipy in c:\work\anaconda3\envs\machine-learning\lib\site-packages (from dtw) (1.1.0)
Building wheels for collected packages: dtw
  Running setup.py bdist_wheel for dtw ... done
  Stored in directory: C:\Users\IEUser\AppData\Local\pip\Cache\wheels\3\ec\0b\91460dfe9064731fe97be1d868ccaa23e0f99bfae
a7c688f5a
Successfully built dtw
Installing collected packages: dtw
Successfully installed dtw-1.3.3

(machine-learning) C:\Users\IEUser>
```

## 2 Limbajul Python

- Limbaj open source, gratuit
- Doua versiuni majore: Python 2 (ultima versiune: 2.7) si 3 (versiunea curenta: 3.9).
- Python este limbaj interpretat si netipizat
- Permite programare imperativa, orientata pe obiect, [functionala](https://docs.python.org/3/library/functional.html) (<https://docs.python.org/3/library/functional.html>)
- Exista o paleta larga de biblioteci deja implementate pentru Python, usurand considerabil dezvoltarea de prototipuri sau chiar de aplicatii comerciale larg raspandite.

### 2.1 De ce Python?

Iata cateva motive solide pentru a utiliza Python, conform [The unexpected effectiveness of Python in science](https://lwn.net/Articles/724255/) (<https://lwn.net/Articles/724255/>).

1. Limbajul vine "cu toate bateriile incluse": multitudinea de biblioteci il face perfect pentru o gama larga de proiecte. Cercetatorii care il folosesc activeaza in astronomie, fizica, bioinformatica, chimie, stiinte cognitive etc. - si pentru toate aceste domenii exista biblioteci care simplifica mult dezvoltarea de prototipuri. Prin comparatie, limbaje traditionale precum C/C++/Java/C# vin cu mai putin suport disponibil, iar uneori includerea unui pachet/biblioteci poate fi o experienta consumatoare de timp.
2. Capacitatea de interoperare cu alte limbaje este iarasi un plus. Exista puncte de comunicare (bridges) care permit apelul de cod Python din alte limbaje sau invers. Desigur, aceasta facilitate se regaseste si in alte limbaje.
3. Natura dinamica a limbajului poate fi inteligent speculata: o functie poate sa preia o multitudine de parametri (lista, tuplu, dictionar) si cateva linii de cod functioneaza la fel de bine peste toate acestea.
4. Python incurajeaza un stil de lucru experimental, via REPL: poate e nevoie de cateva incercari pentru a ajunge la instructiunile potrivite pentru o anumita functionalitate. Incercarea unei alte

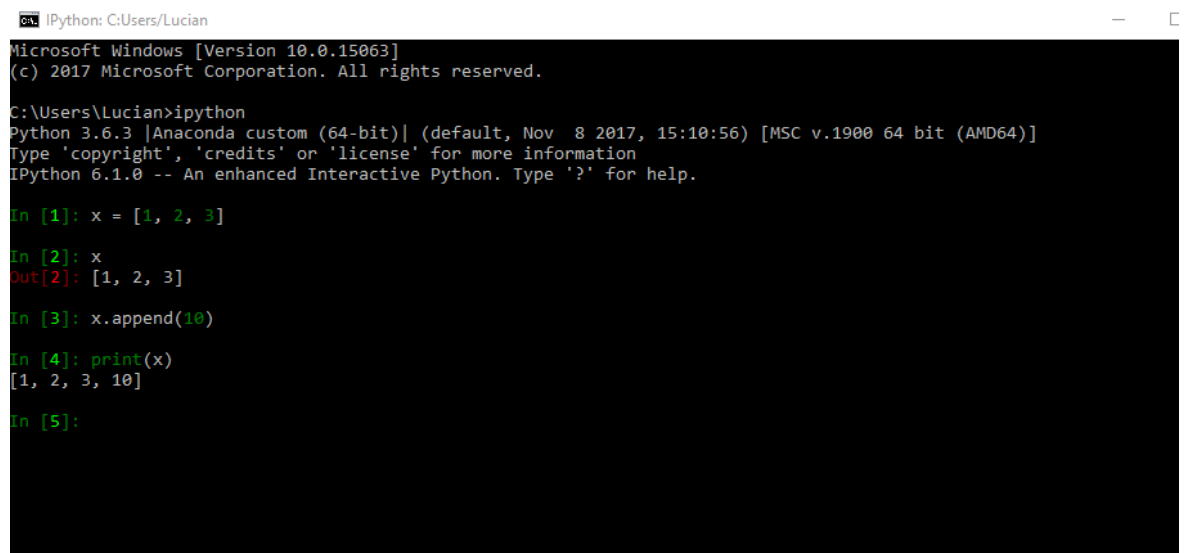


instrucțiuni nu necesită recompilarea sau rerularea codului anterior, ci vine în completare - până când ajungi la ce dorești să obții.

## 3 Modalități și medii de lucru

### 3.1 Moduri de utilizare

1. Codul poate fi scris într-un fișier text cu extensia `.py`. Lansarea codului se face cu comanda `python nume_script.py`
2. Se poate folosi interpretorul `python` sau `ipython`, în care modul de lucru este REPL: read-evaluate-print-loop. REPL permite o prototipizare rapidă și scrierea codului în mod incremental. Interpretorul `ipython` este o variantă îmbunătățită a interpretorului `python`.



```
Python: C:\Users\Lucian
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\Lucian>python
Python 3.6.3 [Anaconda custom (64-bit)] (default, Nov 8 2017, 15:10:56) [MSC v.1900 64 bit (AMD64)]
Type 'copyright', 'credits' or 'license' for more information
IPython 6.1.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: x = [1, 2, 3]

In [2]: x
Out[2]: [1, 2, 3]

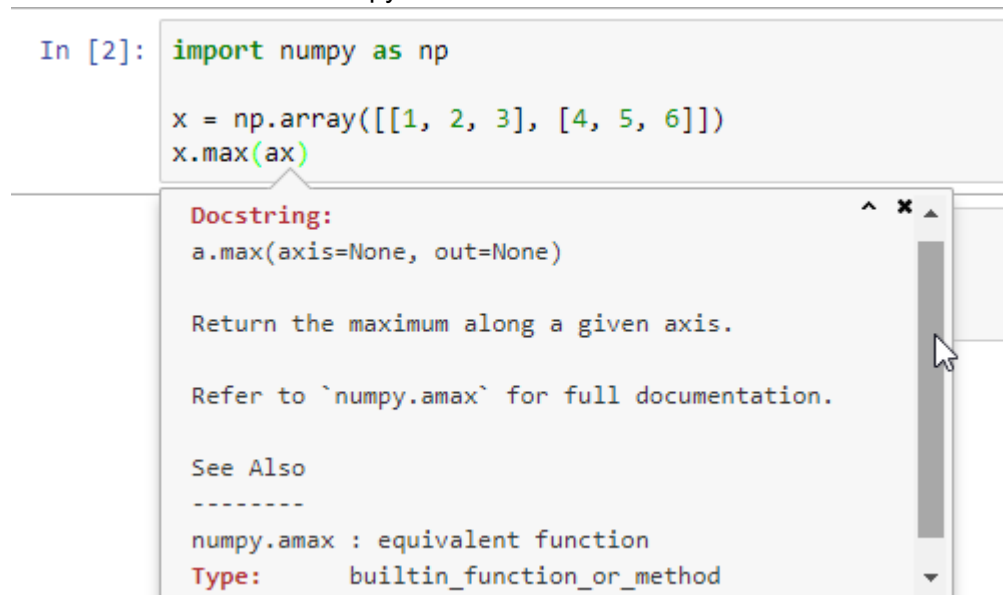
In [3]: x.append(10)

In [4]: print(x)
[1, 2, 3, 10]

In [5]:
```

Mai multe detalii pentru lucrul cu interpretorul `ipython` se găsesc în resursa bibliografică [1], capitolul 1: debug, magic commands, lucru cu interpretorul de comenzi etc.

3. Jupyter notebook - codul este scris în browser, cu suport pentru completare automată de cod. Fișierele rezultate sunt cu extensia `ipynb`.



```
In [2]: import numpy as np

x = np.array([[1, 2, 3], [4, 5, 6]])
x.max(ax)
```

**Docstring:**  
`a.max(axis=None, out=None)`

Return the maximum along a given axis.

Refer to ``numpy.amax`` for full documentation.

See Also  
-----  
`numpy.amax` : equivalent function

**Type:** builtin\_function\_or\_method

O lista de notebooks se gaseste [aici](http://nbviewer.jupyter.org/) (<http://nbviewer.jupyter.org/>).

## ▼ 3.2 PyCharm

Anaconda vine cu un editor minimal, Spyder. Se poate folosi acesta sau unul mai evoluat, de exemplu PyCharm sau VSCode.

PyCharm se poate instala [de aici](https://www.jetbrains.com/pycharm/) (<https://www.jetbrains.com/pycharm/>), de exemplu versiunea Community Edition. Studentii pot beneficia gratuit de licenta academica pentru versiunea Professional. In versiunea Community se permite debug, utilizarea de medii virtuale specifice proiectelor, lucru cu git etc.

Demo:

- program simplu, code completion, refactoring, suport pentru docstrings, ....
- debug
- ...

## ▼ 3.3 Jupyter notebook

Jupyter notebook (numit inainte ipython notebook) permite scrierea de cod in browser si rularea in stil Read-Evaluate-Print-Loop. Este un mod util pentru prototipizare, cand succesiunea de pasi nu e clar cunoscuta aprioric.

Nativ, in Anaconda sunt suportate nuclee de Python si R. Se pot adauga alte nuclee, ceea ce inseamna ca in acelasi notebook pot fi folosite secvente de cod in diferite limbaje. O lista de posibilitati este data [aici](https://github.com/jupyter/jupyter/wiki/Jupyter-kernels) (<https://github.com/jupyter/jupyter/wiki/Jupyter-kernels>).

Lansarea serverului Jupyter se face in doua moduri:

1. din command line interface de Anaconda, se scrie comanda `jupyter notebook`
2. din Anaconda Navigator

Ca rezultat se porneste un server web, local, de regula pe portul 8888 (sau alt port disponibil; se pot lansa mai multe servere Jupyter simultan, acestea vor folosi porturile 8889, 8890 etc.)

- Demo Jupyter notebook
- Help din docstring
- Operatii cu kernelul de Python - repornire, oprire etc.
- Export de script din Jupyter notebook

### ▼ 3.3.1 Plugin-uri de Jupyter notebook

Se pot instala - in mediul de baza sau alte medii virtuale - pachete cu functionalitati auxiliare. Recomandam instalarea celor mentionate [aici](https://towardsdatascience.com/jupyter-notebook-extensions-517fa69d2231) (<https://towardsdatascience.com/jupyter-notebook-extensions-517fa69d2231>).

## 4 Bibliografie

1. [How to Use Jupyter Notebook: A Beginner's Tutorial \(https://www.dataquest.io/blog/jupyter-notebook-tutorial/\)](https://www.dataquest.io/blog/jupyter-notebook-tutorial/)
2. [Jupyter Notebook Tutorial: The Definitive Guide \(https://www.datacamp.com/tutorial/tutorial-jupyter-notebook\)](https://www.datacamp.com/tutorial/tutorial-jupyter-notebook)
3. [Jupyter Notebook Tutorial - A Complete Beginners Guide \(https://www.projectpro.io/data-science-in-python-tutorial/jupyter-notebook-tutorial\)](https://www.projectpro.io/data-science-in-python-tutorial/jupyter-notebook-tutorial)