

# Laborator 1

## Instalare Anaconda

Este explicata in detaliu la [alt laborator](#)

(<https://github.com/lmsasu/cursuri/tree/master/InteligentaArtificiala/laborator/laborator1>).

**Exercitiu:** creati un mediu virtual numit *ids*, folosind conda si actualizati-i pachetele.

## Manifestul Python

Lansati interpretorul interactiv Python in linia de comanda: `ipython` si apoi rulati comanda: `import this`. Se va afisa pe ecran manifestul Python [the Zen of Python](#) (<https://www.python.org/dev/peps/pep-0020/>).

## Jupyter notebook

In linia de comanda Anaconda prompt scrieti comanda:

```
jupyter notebook
```

se va deschide automa browserul implicit la adresa localhost:8888 (daca portul 8888 nu e liber, se va cauta automat un alt port).

## Jupyter lab

Unii prefera folosirea de Jupyter lab in loc de Jupyter notebook; cel din urma este recomandat pentru incepatori, sugerandu-se sa se treaca ulterior la Jupyter lab. Jupyter lab este deja disponibil in distributia actuala de Anaconda, sau poate fi instalat conform instructiunilor de [aici](#) ([https://jupyterlab.readthedocs.io/en/stable/getting\\_started/installation.html](https://jupyterlab.readthedocs.io/en/stable/getting_started/installation.html)).

Jupyter lab vs Jupyter notebook:

- [JupyterLab first impressions](https://medium.com/@brianray_7981/jupyterlab-first-impressions-e6d70d8a175d) ([https://medium.com/@brianray\\_7981/jupyterlab-first-impressions-e6d70d8a175d](https://medium.com/@brianray_7981/jupyterlab-first-impressions-e6d70d8a175d))
- [Jupyter Notebooks are Breathtakingly Featureless—Use Jupyter Lab](https://towardsdatascience.com/jupyter-notebooks-are-breathtakingly-featureless-use-jupyter-lab-be858a67b59d) (<https://towardsdatascience.com/jupyter-notebooks-are-breathtakingly-featureless-use-jupyter-lab-be858a67b59d>)

Se porneste Jupyter lab cu:

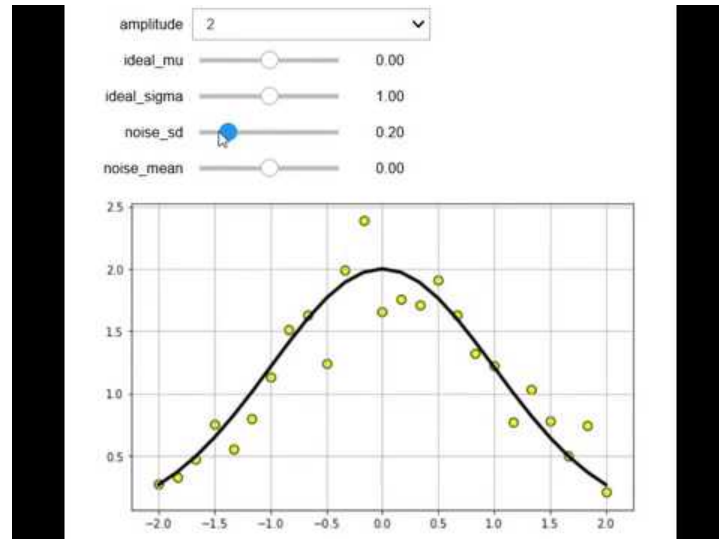
```
jupyter lab
```

## Folosire de controale grafice

Notebook-urile - indiferent ca se ruleaza in Jupyter lab sau Jupyter notebook - se pot folosi pentru demo-uri interactive. O varianta este modificarea codului in timpul demo-ului si rularea celulelor afectate - nu intotdeauna rapid de facut. O alta varianta este folosirea de controale grafice care sa permita utilizatorului sa modifice optiuni, valori de parametri etc.

`ipywidgets` (<https://ipywidgets.readthedocs.io/en/stable/>) este o biblioteca de controale grafice care permit interactiune cu utilizatorul. Mai jos sunt cateva demo-uri de urmarit.

- Demo 1:



([https://www.youtube.com/watch?v=nRmkS\\_6ngCU](https://www.youtube.com/watch?v=nRmkS_6ngCU))

- Demo 2:

```
In [1]: import ipywidgets as widgets
        from ipywidgets import interact, interact_manual, fixed

In [2]: from random import choice

In [3]: def lang():
        langSelect = ["English", "中文", "日文", "Español", "Italiano", "Deutsche"]
        print(choice(langSelect))

In [4]: lang()
        Italiano


In [5]: interact_manual(lang)
        Run lang
        Deutsche

In [6]: import ipywidgets as widgets
        from ipywidgets import interact, interact_manual, fixed
        from random import choice

In [7]: def func():
        # this time
        # ...
```

(<https://www.youtube.com/watch?v=j5d7vOQBtI>)

- Demo 3:



# Roshan

## IP[y]: IPython Interactive Computing

### Part 6

## IPython Widgets

(<https://www.youtube.com/watch?v=wxVx54ax47s>)

- Demo 4:



### Exemple de utilizare

Documentatia completa si exemple [aici](https://ipywidgets.readthedocs.io/en/stable/user_guide.html) ([https://ipywidgets.readthedocs.io/en/stable/user\\_guide.html](https://ipywidgets.readthedocs.io/en/stable/user_guide.html)).

Incarcarea pachetului de `ipywidgets` se face prin:

```
In [2]: import ipywidgets as widgets
```

De regula, e nevoie si de alte pachete, de exemplu:

```
In [3]: from ipywidgets import interact, interactive, fixed, interact_manual
```

Cel mai simplu control utilizabil este `interact`. El poate prelua ca prim parametru numele unei functii, iar al doilea parametru dicteaza forma controlului: slider, combo box, checkbox etc:

```
In [4]: def n_factorial(n):
        """Calculeaza n factorial"""
        p = 1
        for i in range(1, n+1):
            p *= i
        return str(n) + "!= " + str(p)
```

```
In [5]: interact(n_factorial, n=100)
```

```
Out[5]: <function __main__.n_factorial(n)>
```

Pentru limitarea domeniului in care  $n$  poate sa ia valori se va folosi:

```
In [6]: interact(n_factorial, n=(0, 100))
```

```
Out[6]: <function __main__.n_factorial(n)>
```

Pentru a evita actualizarea sacadata a valorilor afisate, se prefera inhibarea feedback-ului in timp real, precum in [Disabling continuous updates](https://ipywidgets.readthedocs.io/en/stable/examples/Using%20Interact.html#Disabling-continuous-updates) (<https://ipywidgets.readthedocs.io/en/stable/examples/Using%20Interact.html#Disabling-continuous-updates>).

Pentru alte tipuri de controale folosind interact, se poate folosi:

```
In [7]: def g(x, y, z, t):
        return (x, y, z, t)

        interact(g, x=True, y=(1.0, 10.0, 0.5), z='Un text',
                t={'English':'Hello', 'Romanian':'Salut', 'Spanish':'Hola'})
```

```
Out[7]: <function __main__.g(x, y, z, t)>
```

Exemplu: Sa se deseneze graficul functiei  $f : [-10, 20] \rightarrow \mathcal{R}, f(x) = a \cdot x^2 + b \cdot x + c$  cu  $a, b, c$  coeficienti reali.

Rezolvare:

```
In [8]: # import de pachete numerice si grafice
```

```
import matplotlib.pyplot as plt
import numpy as np
```

```
In [13]: def f_square(a=10, b=20, c=-10, left=-10, right=20):
        assert left < right
        range_x = np.linspace(left, right, 100)
        values_f = a * range_x ** 2 + b * range_x + c
        plt.figure(figsize=(20, 10))
        plt.xlabel('x')
        plt.ylabel('$a \cdot x^2 + b \cdot x + c$')
        plt.plot(range_x, values_f, color='red')
        plt.show()

        interact(f_square, a=(-100, 100.0), b=(-100, 100.0), c=(-100, 100.0), d=(-100, 100.0), e=(-100, 100.0))
```

```
Out[13]: <function __main__.f_square(a=10, b=20, c=-10, left=-10, right=20)>
```

```
In [14]: def sinusoid(f=10):
         range_x = np.linspace(-5, 5, 100)
         values_f = np.sin(2 * np.pi * f * range_x)
         plt.xlabel('x')
         plt.ylabel(f'$2 \cdot \pi \cdot {f} \cdot x$')
         plt.plot(range_x, values_f)

         interact(sinusoid, f = (1, 100.0, 0.5))
```

```
Out[14]: <function __main__.sinusoid(f=10)>
```

```
In [15]: def f(x):
         """calcul functie intr-un punct"""
         return x ** 2 - 10 * x + 50

         def f_values(left=-10, right=10):
             """calcul functie pe interval"""
             x = np.linspace(left, right, 100)
             return x, f(x)

         def f_prime(x):
             """Calcul derivata f
             :param x: punctul in care se calculeaza derivata
             :return: f'(x)
             """
             return 2 * x - 10

         def graph_f_and_derived(x, left=-30, right=30):
             # calcul valoare functie f
             x_range, fx = f_values(left, right)

             # intervalul pe care se reprezinta tangenta
             x_segment = np.linspace(x-10, x+10, 100)
             # panta tangentei la grafic este derivata functiei in ptul de tangenta
             slope = f_prime(x)

             #calcul puncte de tangenta
             y_segment = f(x) + slope * (x_segment - x)

             plt.figure(figsize=(20, 10))
             plt.plot(x_range, fx, color='red')
             plt.plot(x_segment, y_segment, color='blue')

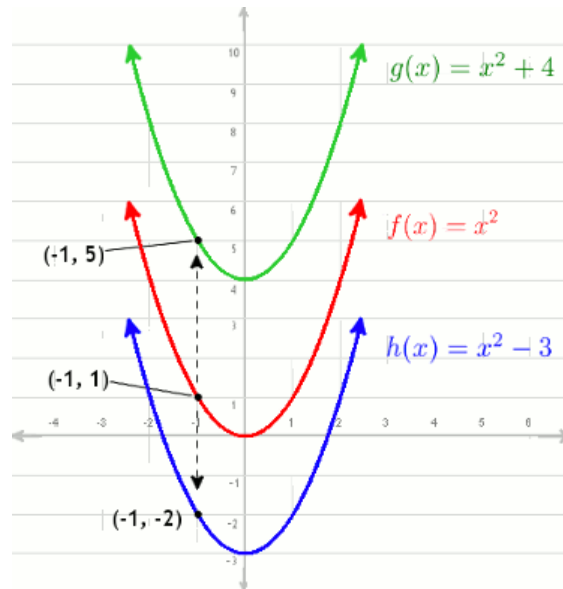
         # graf_f_and_derived(10, left=-30, right=30)

         interact(graph_f_and_derived, x = (-20, 20))
```

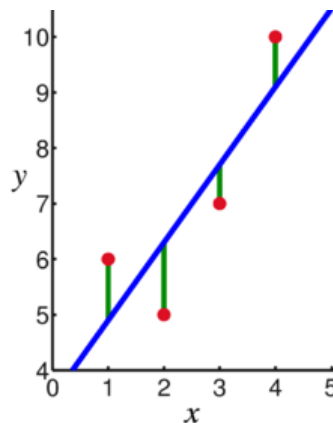
```
Out[15]: <function __main__.graph_f_and_derived(x, left=-30, right=30)>
```

## Exercitii:

- Se considera urmatorul enunt: fie functia  $f(x) = x^2$  si punctul  $P = (5, 3)$ , sa se modifice functia  $f$  prin translatie a.i. minimul functiei sa se afle in punctul  $P$ ; a se vedea mai jos exemplu de translatie pe verticala. Se cere desenarea axelor  $Ox$  si  $Oy$  cu reprezentarea punctului  $P$  printr-un dreptunghi, desenarea functiei  $f$  folosind o curba de 50 de puncte. Se vor determina coeficientii necesari mutarii functiei si se vor defini controale pentru acestea.



- Reprezentati functia  $a * x^4 + b * x^3 + c * x^2 + d * x + e$  si derivata ei intr-un punct ales de catre utilizator.
- Incarcati fisierul de date [iris](https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data) (<https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data>) si in functie de alegerile exprimate de un utilizator, afisati intr-un grafic bidimensional coloanele numerice alese (de exemplu, coloana 0 si coloana 2).
- Generati o lista de 20 de perechi de valori  $\{x_i, y_i\}_{i=0,19}$ , afisati aceste valori pe un grafic, impreuna cu o dreapta definita de o functie liniara  $y = a * x + b$ . Intr-un alt plot afisati, ca histograma, distanta dintre un punct de coordonate  $(x_i, y_i)$  fata intersectia dreptei si a verticalei duse prin punctul  $i$ . Dreapta trebuie sa fie controlabila din widgets, prin cei doi coeficienti. Constatati modificarea histogramei in functie de pozitia dreptei si calculati suma:  $\sum_{i=0}^{19} (y_i - (a * x_i + b))^2$ , adica suma patratelor lungimilor segmentelor verzi de mai jos.



## Ciclari, siruri de caractere

Se recomanda ca urmatoarele exercitii sa le lucrati in Jupyter notebook/lab. Incercati parametrizarea functiilor cu ipywidgets.

1. (fizz-buzz test) Sa se scrie numerele de la 1 la n; pentru fiecare multiplu de 3 se va scrie in locul numarului 'Fizz', pentru multiplu al lui 5 se va scrie 'Buzz'; daca numarul este multiplu de 15 se va scrie in locul lui 'FizzBuzz'.
2. Sa se verifice conjectura Collatz ([https://en.wikipedia.org/wiki/Collatz\\_conjecture](https://en.wikipedia.org/wiki/Collatz_conjecture)) pentru numerele intre 1 si 1000.
3. \* Sa se creeze o functie care preia un numar n si returneaza un alt numar pe baza cifrelor lui n, astfel: se calculeaza numarul cifrelor pare din n (fie el si 0), numarul de cifre impare (poate fi si 0), suma lor si se formeaza numarul din acestea 3; daca exista vreun zero nesemnificativ, acesta se va ignora. Numarul obtinut se va supune aceleiasi transformari. Exemplu: 3->11->22->202->303->123->123->123... Verificati ca dupa un numar finit de transformari se ajunge la numarul 123; faceti aceste verificari pentru numerele din intervalul 1, 1000.
4. \* Se pleaca de la un numar intreg. Fiecare cifra a sa se scrie cu litere, in limba engleza (de exemplu 5->five). Claculati numarul total de caractere rezultate, iar pentru numarul obtinut repetati procedura. Verificati pentru numerele de la 1 la n ca se obtine intr-un numar finit de pasi numarul 4. Exemplu: 123->onetwothree->11->oneone->6->six->3->three->5->five->4->four->4->four....
5. \* Este rezultatul de mai sus valabil si pentru transcriere in limba romana?
6. \* Se pleaca de la un numar n; se scriu toti divizorii sai, inclusiv 1 si n; se aduna *cifrele* tuturor acestor divizori; pentru numarul obtinut se aplica acelasi procedeu. Verificati ca procesul se stabilizeaza in numarul 15. Exemplu: 20->1, 2, 4, 5, 10, 20 -> suma cifrelor: 15->1, 3, 5, 15-> suma cifrelor: 15...