

▼ 1 Laborator 1

▼ 1.1 Instalare Anaconda

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

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

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

▼ 1.2 Manifestul Python

Lansati interpretorul interactiv Python in linia de comanda: `ipython` si apoi rulati comanda care afisa pe ecran manifestul Python [the Zen of Python](https://www.python.org/dev/pep/) (<https://www.python.org/dev/pep/>

▼ 1.3 Jupyter notebook

In linia de comanda (command prompt sau powershell) scrieti comanda:

```
| jupyter notebook
```

se va deschide automa browserul implicit la adresa localhost:8888 (daca portul 888 automat un alt port).

▼ 1.4 Jupyter lab

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

Jupyter lab vs Jupyter notebook:

- [link1](https://medium.com/@brianray_7981/jupyterlab-first-impressions-e6d70d) (https://medium.com/@brianray_7981/jupyterlab-first-impressions-e6d70d)
- [link2](https://towardsdatascience.com/jupyter-notebooks-are-breathtakingly-feab858a67b59d) (<https://towardsdatascience.com/jupyter-notebooks-are-breathtakingly-feab858a67b59d>)

Se ruleaza din linia de comanda cu:

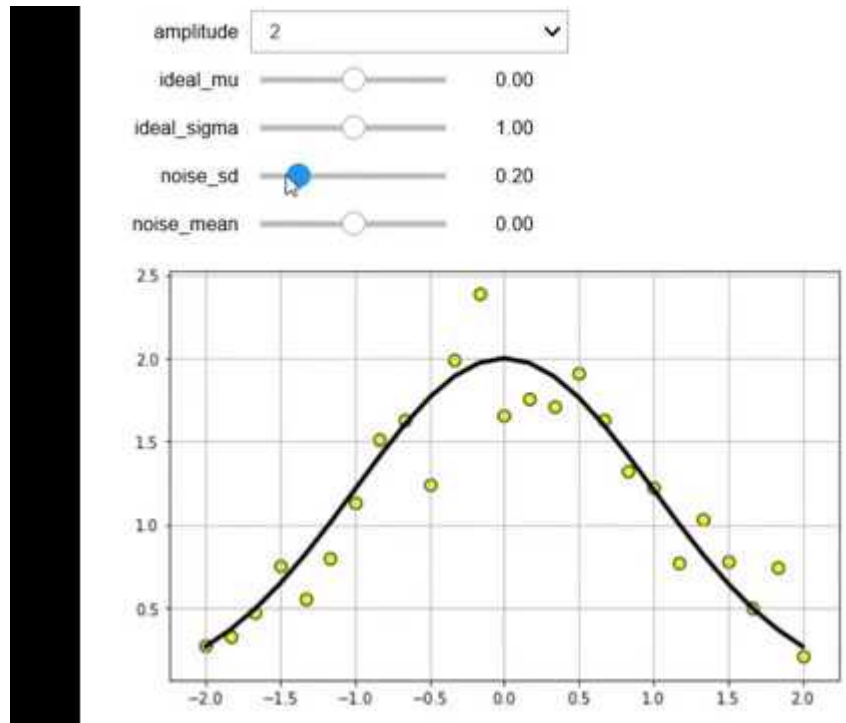
```
| jupyter lab
```

▼ 1.5 Folosire de controale grafice

Notebook-urile - indiferent ca se ruleaza in Jupyter lab sau Jupyter notebook - se pot folosi interactive. O varianta este modificarea codului in timpul demo-ului si rularea celulelor de cod rapid de facut. O alta varianta este folosirea de controale grafice care sa permita utilizarea unor valori de parametri etc.

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

- Demo 1:



(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"]
        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 [ ]: def func():
        x1 = ['']
```

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

- Demo 3:



IP[y]: IPython Interactive Computing

Part 6 IPython Widgets

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

▼ 1.5.1 Exemple de utilizare

Documentatia completa si exemple [aici](https://ipywidgets.readthedocs.io/en/stable) (<https://ipywidgets.readthedocs.io/en/stable>)

Incarcarea pachetului de `ipywidgets` se face prin:

```
In [39]: import ipywidgets as widgets
```

executed in 878ms, finished 18:07:34 2019-02-21

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

```
In [40]: from ipywidgets import interact, interactive, fixed, interact_manual
```

executed in 1.04s, finished 18:07:38 2019-02-21

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

```
In [41]: def n_factorial(n):  
    """Calculeaza n factorial"""  
    p = 1  
    for i in range(1, n+1):  
        p *= i  
    return str(n) + "!= " + str(p)
```

executed in 1.09s, finished 18:07:40 2019-02-21

```
In [42]: interact(n_factorial, n=100)
```

executed in 8.61s, finished 18:07:50 2019-02-21

```
interactive(children=(IntSlider(value=100, description='n', max=300, min=-100), Output()), _dom_classes

<function __main__.n_factorial(n)>
```

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

```
In [43]: interact(n_factorial, n=(0, 100))
```

executed in 9.05s, finished 18:08:02 2019-02-21

```
interactive(children=(IntSlider(value=50, description='n'), Output()), _dom_classes=('widget-interact',

<function __main__.n_factorial(n)>
```

Pentru a evita actualizarea sacadata a valorilor afisate, se prefera inhibaarea feedb
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 [44]: 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'})
```

execution queued 18:07:33 2019-02-21

```
interactive(children=(Checkbox(value=True, description='x'), FloatSlider(value=5.5, description='y', ma

<function __main__.g(x, y, z, t)>
```

Exemplu: Sa se deseneze graficul functiei $f : [-10, 20] \rightarrow \mathcal{R}, f(x) = a \cdot x^4 + b$
 a, b, c, d, e coeficienti reali.

Rezolvare:

In [45]: `# import de pachete numerice si grafice`

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

execution queued 18:07:33 2019-02-21

In [46]: `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))
```

execution queued 18:07:33 2019-02-21

```
interactive(children=(FloatSlider(value=10.0, description='a', min=-100.0), FloatSlider(value=20.0, des
```

```
<function __main__.f_square(a=10, b=20, c=-10, left=-10, right=20)>
```

In [47]: `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))
```

execution queued 18:07:33 2019-02-21

```
interactive(children=(FloatSlider(value=10.0, description='f', min=1.0, step=0.5), Output()), _dom_clas
```

```
<function __main__.sinusoid(f=10)>
```

```

In [50]: 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 graf_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(graf_f_and_derived, x = (-20, 20))

```

execution queued 18:07:33 2019-02-21

interactive(children=(IntSlider(value=0, description='x', max=20, min=-20), IntSlider(value=-30, descri

<function __main__.graf_f_and_derived(x, left=-30, right=30)>

- In []: