Laborator 2

Saptamana 9-13 martie 2020

Continut:

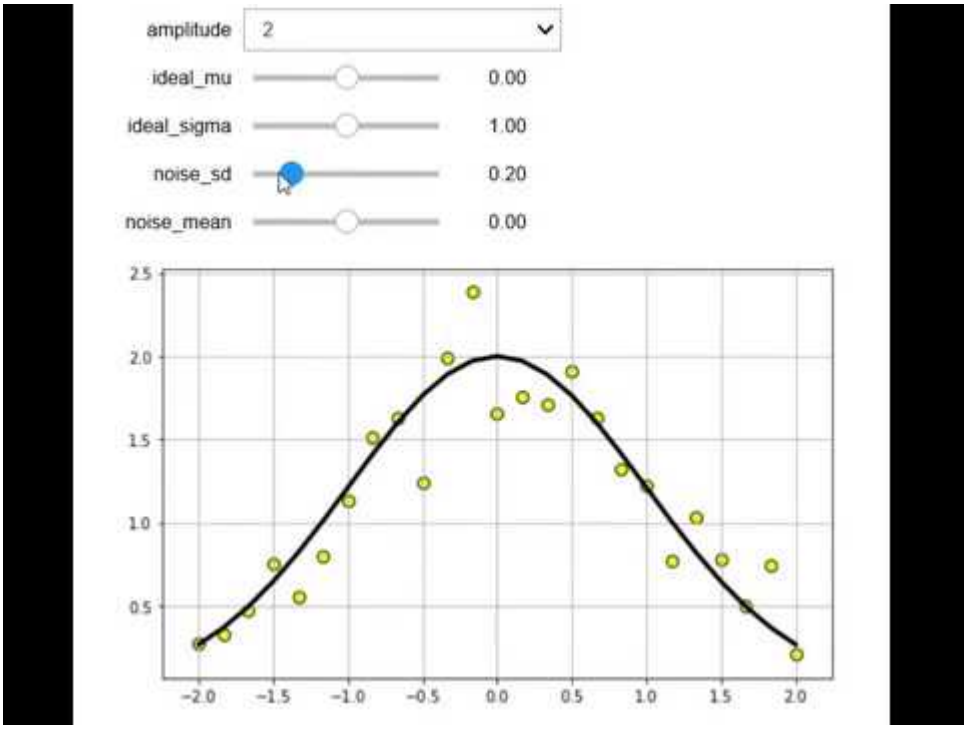
1. Biblioteca ipywidgets
2. Exercitii cu ipywidgets

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 manuala a 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:

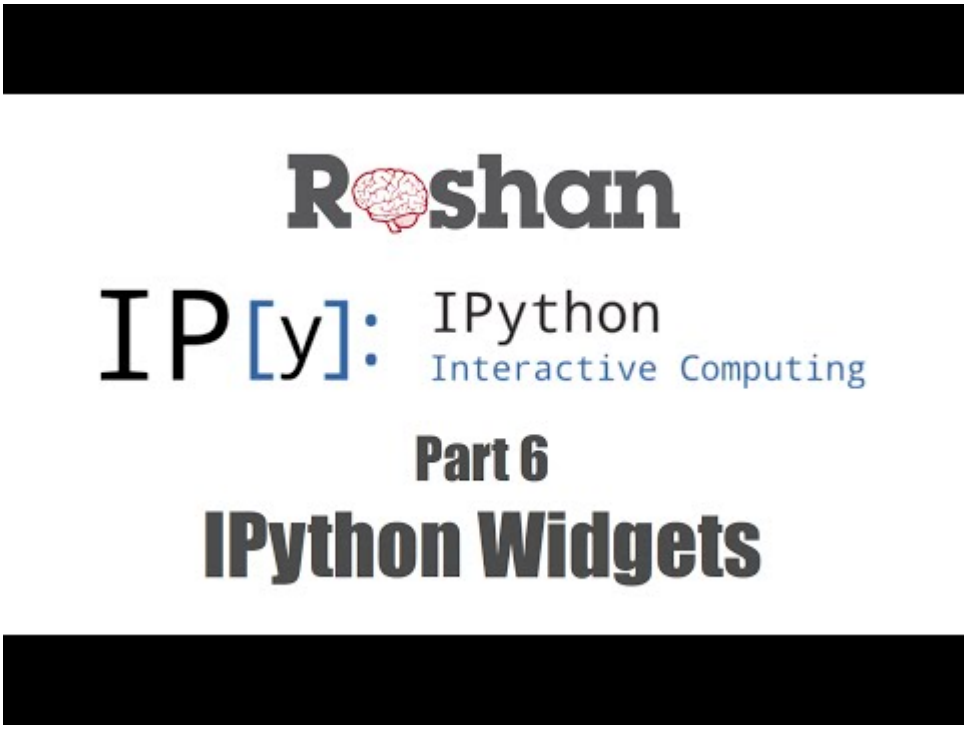


(https://www.youtube.com/watch?v=nRmkS_6ngCU)

- Demo 2:

(https://www.youtube.com/watch?v=j5d7vOQBttI)

- Demo 3:



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

- Demo 4:

Exemple de utilizare

Documentatia completa si exemple sunt date aici (https://ipywidgets.readthedocs.io/en/stable/user_guide.html).

Incarcarea pachetului de `ipywidgets` se face prin:

In [1]:
```
import ipywidgets as widgets
```

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

In [2]:
```
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 [3]:
```
def n_factorial(n:int) -> int:
    """Calculeaza n factorial
    :param n: intreg >= 0 pt care se calculeaza factorialul
    :return: valoarea lui n!
    """
    p = 1
    for i in range(1, n+1):
        p *= i
    return str(n) + "!= " + str(p)
```

In [4]:
```
interact(n_factorial, n=100)
```

Out[4]:
```
<function __main__.n_factorial(n: int) -> int>
```

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

In [5]:
```
interact(n_factorial, n=(0, 100));
```

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).

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

In [6]:
```
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[6]:
```
<function __main__.g(x, y, z, t)>
```

Exemplu: Sa se deseneze graficul functiei $f : [left, right] \to \mathbb{R}$, $f(x) = a \cdot x^2 + b \cdot x + c$, cu a, b, c coeficienti reali.

Rezolvare:

In [7]:
```
# import de pachete numerice si grafice

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

In [10]:
```
def f_square(a=10, b=20, c=-10,left=-10, right=20) -> None:
    '''Afiseaza graficul unei functii de gradul al doilea de forma:
    f(x)=a*x**2 + b*x + c. Valorile lui x sunt luate din intervalul
    [left, right] prin discretizare.
    :param a: coeficientul lui x**2
    :param b: coeficientul lui x
    :param c: termenul liber
    :param left: capatul din stanga al intervalului peste care se face
    reprezenatrea
    :param right: capatul din dreapta al intervalului peste care se face
    reprezenatrea
    :return: None
    '''
    assert left < right
    range_x = np.linspace(left, right, 100)
    values_f = a * range_x ** 2 + b * range_x + c
    plt.figure(figsize=(10, 8))
    plt.xlabel('x')
    plt.ylabel(str(a) + '$\cdot x^2 + $' + str(b) + '$\cdot x + $' + str(c))
    plt.plot(range_x, values_f, color='red')
    plt.grid(axis='both')
    plt.axhline(y=0, color='k')
    plt.axvline(x=0, color='k')
    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));
```

In [11]:
```
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.grid(axis='both')
    plt.axhline(y=0, color='k')
    plt.axvline(x=0, color='k')
    plt.plot(range_x, values_f)

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

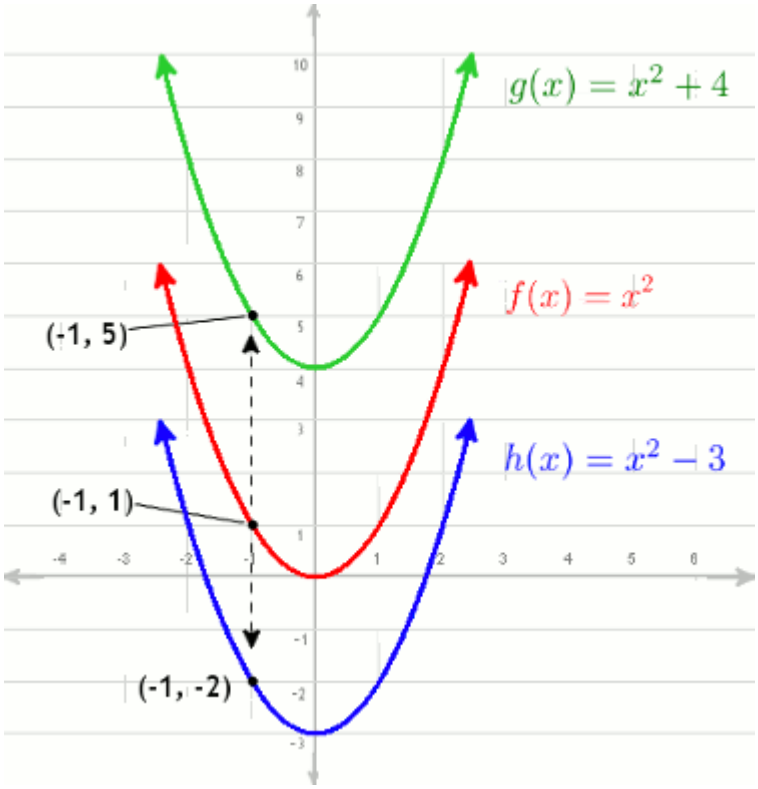
Out[11]:
```
<function __main__.sinusoid(f=10)>
```

In [13]:
```
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 la grafic
    x_segment = np.linspace(x-10, x+10, 100)
    # panta tangentei la grafic este derivata functiei in pctul 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))
```

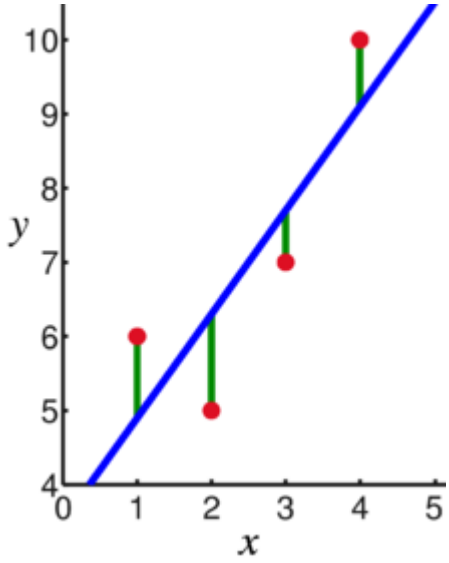
Out[13]:
```
<function __main__.graph_f_and_derived(x, left=-30, right=30)>
```

Exercitii `ipywidgets`:

1. (la clasa) Consideram functia $f(x) = x^2$ si punctul $P = (5, 3)$. Folosind controale grafice ipywidgets, sa se modifice functia f prin translatarea ei pe orizontala si pe verticala, a.i. minimul functiei sa sa afle in punctul P; a se vedea mai jos exemplu de translatie pe verticala. Se cere desenarea axelor Ox si Oy cu reprezentarea puntului 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.



2. (3 puncte) Generati o lista de 20 de perechi de valori $\{x_i, y_i\}_{i=0,19}$ in intervalul [0, 10), 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) si punctul de intersectie a verticalei duse prin x_i cu dreapta data. Dreapta trebuie sa fie controlabila din widgets, prin cei doi coeficienti. Constatati modificarea histogramei in functie de pozitia dreptei si calculati suma: $\sum_{i=0}^{19}(y_i - (a \cdot x_i + b))^2$, adica suma patratelor lungimilor segmentelor verzi de mai jos.



Indicatii:

A. Pentru generare de valori distribuite uniform in intervalul [0, 1) puteti folosi functia numy.random.uniform (https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.uniform.html) iar vectorul obtinut sa il inmultiti cu 10; in felul acesta, numerele generate vor fi uniform distribuite in intervalul $[0, 10)$.

B. Puteti opta sa returnati cele 20 de puncte sub forma `vector_x, vector_y`.

3. (5 puncte) Incarcati fisierul `data/carbon_nanotubes.csv` (adaptare dupa Carbon Nanotubes Data Set (http://archive.ics.uci.edu/ml/datasets/Carbon+Nanotubes)). In functie de alegerile exprimate de un utilizator, afisati intr-un grafic 2D coloanele numerice alese (de exemplu, coloana 0 si coloana 2). *Indicatii/optiuni*:

   A. Incarcarea de date se poate face cu numpy, functia loadtxt (https://docs.scipy.org/doc/numpy/reference/generated/numpy.loadtxt.html). Specificati faptul ca se sare peste prima linie din fisier (header). Alternativ, puteti folosi pandas.read_csv (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_csv.html).

   B. Numarul de coloane din setul de date se poate afla cu `data.shape[1]`.

   C. Pentru cele doua alegeri puteti sa instantiati doua obiecte Dropdown

In [ ]: