Laborator 13
Aplicatii in Python

În acest laborator sunt prezentate cateva notiuni introductive si aplicatii in limbajul Python .

Consideraţii teoretice

Moduri de implementare aplicatii in Python

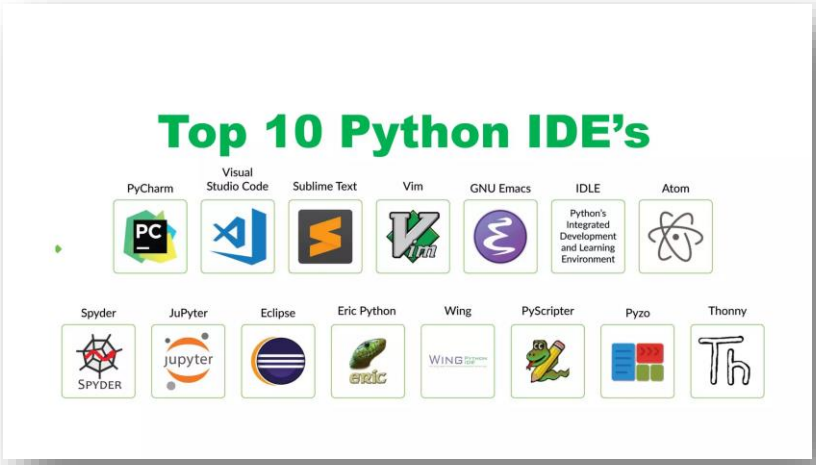
Pentru a putea rula cod in limbajul Python , se poate instala un interpretor local sau se pot utiliza IDE sau interpretoare on line.

1. **Instalare interpretor Python local**

Download: https://www.python.org/downloads/

Instalare: https://realpython.com/installing-python/

2. **Utilizare IDE (Integrated Development Environment)**



In laborator se utilizeaza Visual Studio Code (gratuit).

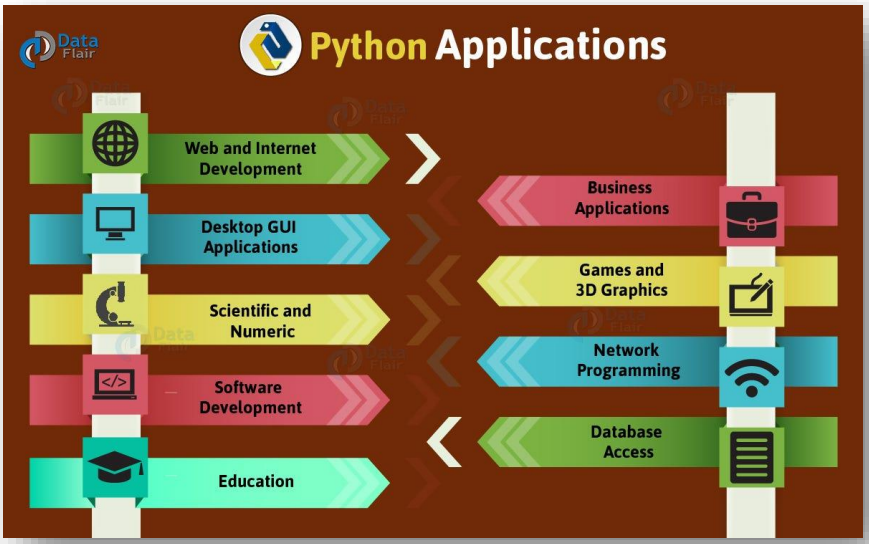
Download: https://code.visualstudio.com/download

Instalare: https://code.visualstudio.com/docs/setup/windows

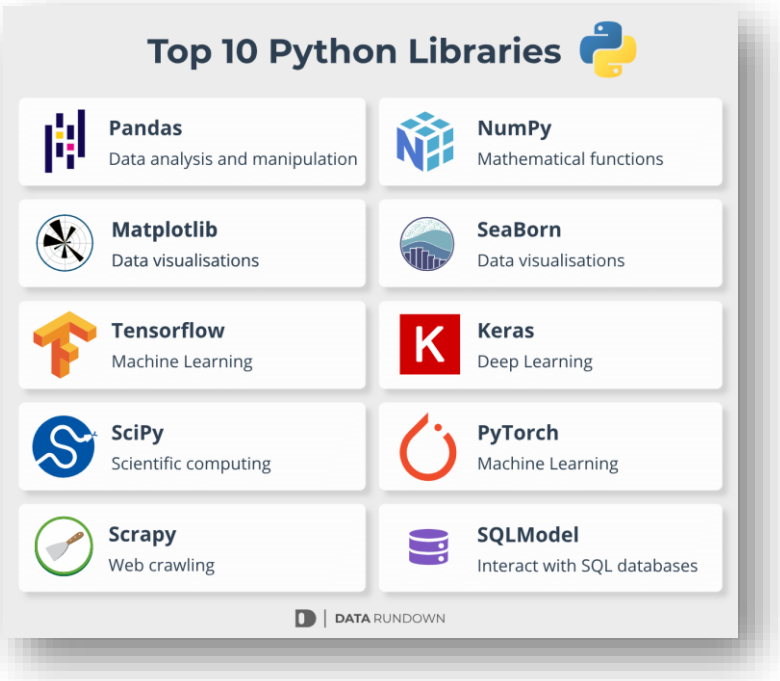
3. **Interpretoare Online Python** = interpretoare Python cloud-based:

- **Python.org Online Console**
- **Repl.it**
- **Python Fiddle**
- **Trinket**
- **Python Anywhere**

Tipuri de aplicatii ce se pot implementa in limbajul Python



Biblioteci Python



Tipuri de date, elemente de sintaxa in Python: Cheat Sheet https://blog.finxter.com/

Codul Python poate fi executat

a) direct in linia de comanda:

```
>>> print("Hello, World!")
Hello, World!
```

b) Intr-un fisier cu extensia .py

```
C:\Users>python myfile.py
```

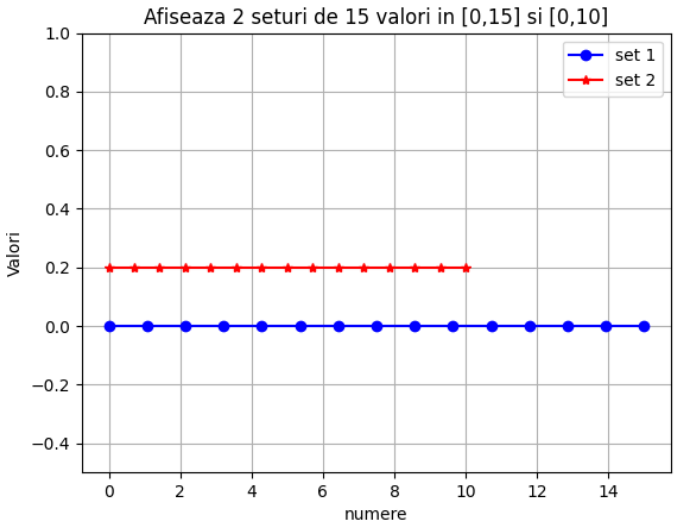
PROBLEME REZOLVATE

Ex.1: Codul Python afiseaza graficul unui sir de 15 numere generate in intervalul [0,100]

Cod Python

```
#import librarie operatii matematice cu siruri/matrici
import numpy as np
#import librarie afisare (plot) grafice
import matplotlib.pyplot as plt
# Genereaza set 1: 15 numere egal distantate in intervalul [0,15]
x1 = np.linspace(0, 15, 15)
# Genereaza set 2: 15 numere egal distantate in intervalul [0,10]
x2 = np.linspace(0, 10, 15)
# Creaza un array y de 15 numere =0 pentru a afisa grafic valorile
y = np.zeros(15)
# afiseaza setul 1
plt.plot( x1,y, marker='o', linestyle='-', color='b', label='set 1')
# afiseaza setul 2 la distanta de 0.2 de setul 1
plt.plot( x2,y+0.2, marker='*', linestyle='-', color='r', label='set 2')
# Adauga etichete , grid, legenda, titlu
plt.xlabel('numere')
plt.ylabel('Valori')
plt.title('Afiseaza 2 seturi de 15 valori in [0,15] si [0,10]')
plt.grid(True)
plt.legend()
#afiseaza graficul pe verticala in interval [-0.5,1]
plt.ylim([-0.5, 1])
# afiseaza graficul
plt.show()
```

Rezultate:



Aplicație:

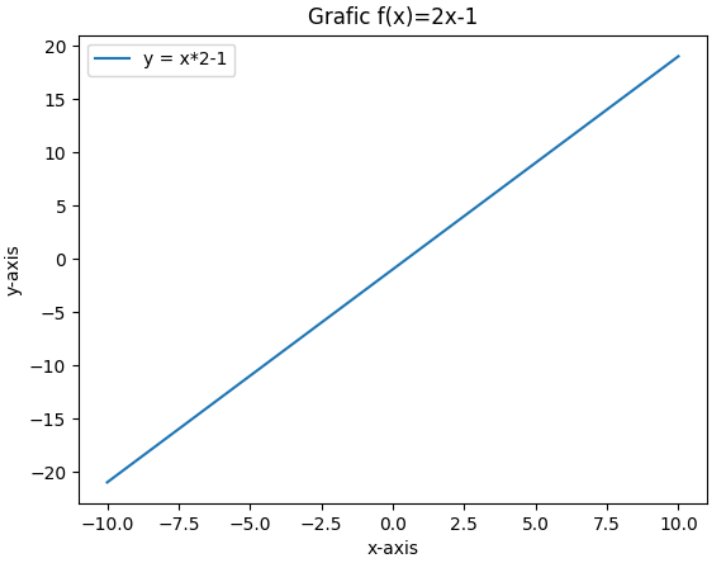
Să se modifice programul astfel pentru a afisa un set 3 cu 15 valori echidistante in intervalul [5,10]

Ex.2: Codul Python afiseaza graficul functiei f(x)=2x-1, generand 100 de valori intr-un sir (array) in intervalul [-10,10]

Cod Python

```
#import librarie operatii matematice cu siruri/matrici
import numpy as np
#import librarie afisare (plot) grafice
import matplotlib.pyplot as plt
# Defineste functia ce va fi afisata
def my_function(x):
    return x*2-1  # se poate inlocui cu orice expresie matematica
# Genereaza valorile pe axa  x in intervalul [-10,10)] de la start:-10, end:10, nr de
valori:100
x_values = np.linspace(-10, 10, 100)  # se poate ajusta intervalul sau nr valori
# Calculeaza valorile corespunzatoare axei y utilizand expresia din functia definita mai sus
y_values = my_function(x_values)
# Afiseaza graficul functiei
plt.plot(x_values, y_values, marker='o', label='y = x*2-1')
# se adauga textul pentru axe si titlul graficului
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.title('Plot of the Function')
# se adauga legenda
plt.legend()
# afiseaza graficul
plt.show()
```

Rezultate:



Aplicație:

Să se modifice programul astfel pentru a afisa graficul functiei x^3+1 pentru 50 de valori in intervalul [-5,15]

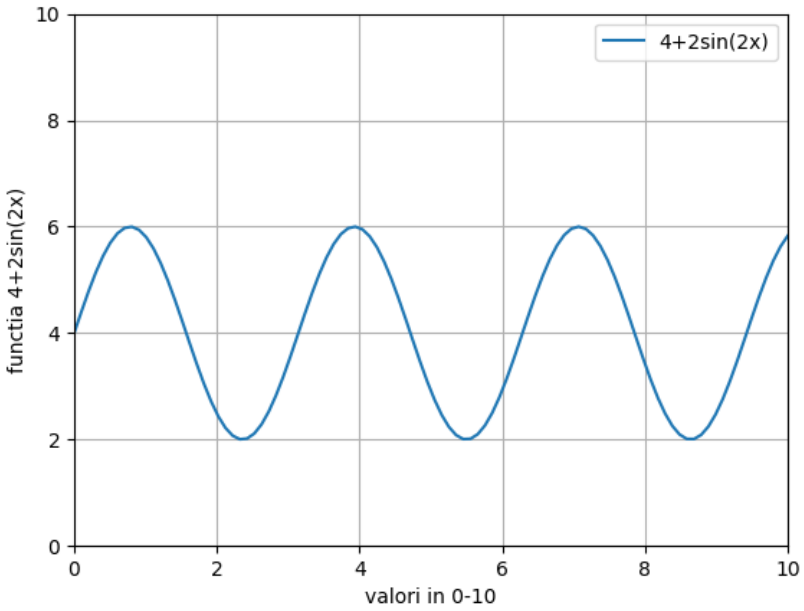
Ex.3: Codul Python afiseaza graficul functiei f(x)= 4+2sin(2x) x generat cu 100 de valori intr-un array in intervalul (0,10)

Cod Python

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

# genereaza datele din grafic
x = np.linspace(0, 10, 100) #100 de valori x in (0,10)
y = 4 + 2 * np.sin(2 * x) #valori y=f(x)=4+2sin(2x)
# plot
plt.plot(x, y,label='4+2sin(2x)')
plt.grid(True)
plt.xlim([0, 10])
plt.ylim([0,10])
plt.xlabel('valori in 0-10')
plt.ylabel('functia 4+2sin(2x)')
plt.legend()
plt.show()
```

Rezultate:



Aplicație:

Să se modifice programul astfel pentru a afisa graficul functiei 1+cos(2x) pentru 100 de valori in intervalul [0,10]

Ex.4: Codul Python citeste valorile reale x dintr-un fisier input.txt si scrie valorile functiei f(x)=2x-1 in fisierul output.txt

Cod Python

```
#import librarie operatii cu fisiere
import pandas as pd
# Definire functie f(x)=2*x-1
def f(x):
    return 2 * x - 1
#deschide fisier input.txt pentru citire
fis1=open('input.txt', 'r')
# citeste valorile din fisierul de intrare input.txt -trebuie creat inainte de rulare cod
x_values = [float(line.strip()) for line in fis1]
#float(line.strip()) "curata" liniile de text de spatii .. si le converteste din text in float
print(x_values)
# calculeaza f(x) pentru fiecare x citit din fisier
f_values = [f(x) for x in x_values]
print(f_values)
# Scrie valorile f(x) in fisierul de iesire output.txt
fis2= open('output.txt', 'w')
for value in f_values:
    fis2.write(f"{value}\n")
print("fisierul cu rezultate output.txt a fost generat cu succes!")
```

Rezultate:

```
[5.5, 4.25, 2.35, 3.19, 5.24, 6.75, 7.9, 10.3, 1.44, 9.22]
[10.0, 7.5, 3.7, 5.38, 9.48, 12.5, 14.8, 19.6, 1.88, 17.44]
fisierul cu rezultate output.txt a fost generat cu succes!
```

input.txt

1	5.5
2	4.25
3	2.35
4	3.19
5	5.24
6	6.75
7	7.9
8	10.3
9	1.44
10	9.22

output.txt

1	10.0
2	7.5
3	3.7
4	5.38
5	9.48
6	12.5
7	14.8
8	19.6
9	1.88
10	17.44

Aplicație:

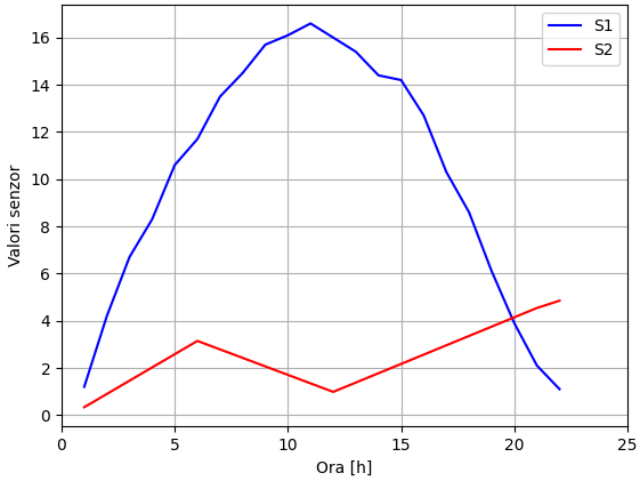
Să se modifice codul si fisierul de intrare cu valori intregi in [1,20] pentru a afisa valorile functiei $f(x)=3x^2-1$

Ex.5: Codul Python citeste dintr-un fisier .csv trei coloane de date: ora si valorile numerice de la doi senzori S1 si S2 si afiseaza graficul valorilor celor 2 senzori in functie de ora

Cod Python

```
#import librarie operatii cu fisiere
import pandas as pd
#import librarie plotare grafice
import matplotlib.pyplot as plt
#crearea unei structuri de date (Data frame) citite din fisierul extern (.csv)
date = pd.read_csv('date.csv')
#preluarea valorilor din structura date
x= date['ora']
#preluare date de la senzorul S1
y1 = date['S1']
#preluare date de la senzorul S2
y2 = date['S2']
#Afisarea graficelor valorilor citite de la senzorii S1 si S2
plt.plot(x, y1,'b-',label='S1')
plt.plot(x,y2,'r-',label='S2')
plt.grid(1)
plt.xlabel('Timp [h]')
plt.ylabel('Valori senzor')
plt.xlim(0,25)
plt.legend()
plt.show()
```

Rezultate:



date.csv

```
ora,S1,S2
1,1.2,0.335
2,4.2,0.897
3,6.7,1.459
4,8.3,2.021
5,10.6,2.583
6,11.7,3.145
7,13.5,2.785
8,14.5,2.425
9,15.7,2.065
10,16.1,1.705
11,16.6,1.345
12,16,0.985
13,15.4,1.38
14,14.4,1.775
15,14.2,2.17
16,12.7,2.565
17,10.3,2.96
18,8.6,3.355
19,6.1,3.75
20,3.9,4.145
21,2.1,4.54
22,1.1,4.85
```

Aplicație:

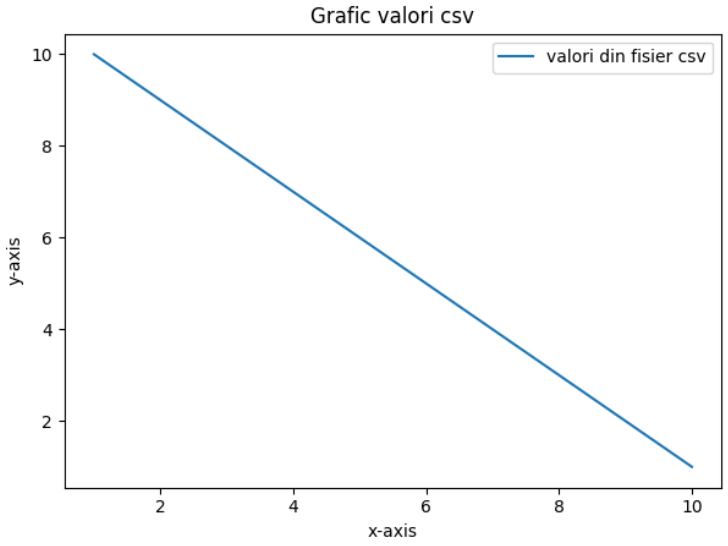
Să se modifice codul si fisierul de intrare date.csv astfel incat sa se afiseze date pentru 3 senzori

Ex.6: Codul Python creaza un fisier output.csv cu 2 coloane de valori intregi initializate prin program pe baza carora se genereaza un dataframe si afiseaza graficul valorilor.

Cod Python

```
#import librarie operatii cu fisiere
import pandas as pd
#import librarie plotare grafice
import matplotlib.pyplot as plt
#crearea a 2 seturi de valori
a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
b = [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
#definirea unei structuri de date (data frame)
date2 = pd.DataFrame(columns=['x', 'y'])
date2['x'] = a
date2['y'] = b
#scrierea in fisierul csv
date2.to_csv('output.csv', index = False)
# Afiseaza graficul functiei
plt.plot(a, b, label='valori din fisier csv')
# se adauga textul pentru axe si titlul graficului
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.title('Grafic valori csv')
# se adauga legenda
plt.legend()
plt.show()
```

Rezultate:



output.csv >

1	x,y
2	1,10
3	2,9
4	3,8
5	4,7
6	5,6
7	6,5
8	7,4
9	8,3
10	9,2
11	10,1

Aplicație:

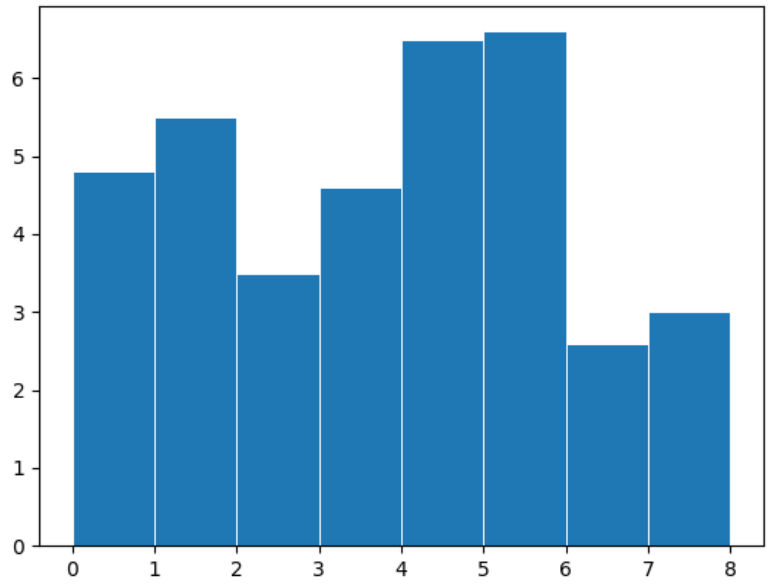
Să se modifice codul astfel incat sa se citeasca aceste valori dintr-un fisier csv si sa se afiseze graficul.

Ex.7: Codul Python afiseaza valori generate si definite in program in grafic de tip bars.

Cod Python

```
import matplotlib.pyplot as plt
import numpy as np
# make data:
x = 0.5 + np.arange(8) #genereaza 8 valori =0.5+ valori in (0,7)
y = [4.8, 5.5, 3.5, 4.6, 6.5, 6.6, 2.6, 3.0]
# plot
plt.bar(x, y, width=1, edgecolor="white", linewidth=0.7, label='bar grafic ')
plt.xlim=(0, 8)
plt.ylim=(0, 8)
plt.grid
plt.legend
plt.show()
```

Rezultate:



Aplicație:

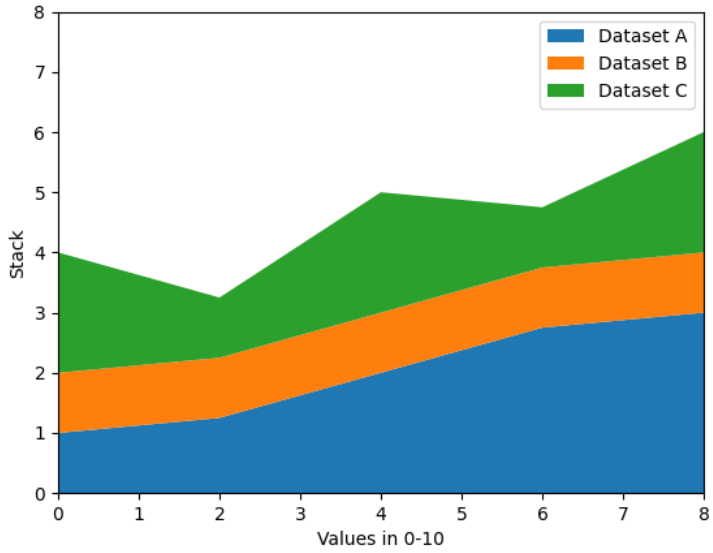
Să se modifice codul astfel incat sa se citeasca aceste valori dintr-un fisier csv si sa se afiseze graficul.

Ex.8: Codul Python afiseaza valori generate in program in grafic de tip stack.

Cod Python

```
import matplotlib.pyplot as plt
import numpy as np
# make data
#genereaza 5 valori x in intervalul (0,10) cu pas 2
#x=0,2,4,6,8
x = np.arange(0, 10, 2)
#genereaza 3 seturi de valori pentru y
ay = [1, 1.25, 2, 2.75, 3] #Dataset A
by = [1, 1, 1, 1, 1] #Dataset B
cy = [2, 1, 2, 1, 2] #Dataset C
#creeaza un set de date y tip stack cu cele 3 seturi de date
y = np.vstack([ay, by, cy])
# plot
plt.stackplot(x, y, labels=['Dataset A', 'Dataset B', 'Dataset C'])
plt.xlabel('Values in 0-10')
plt.ylabel('Stack')
plt.xlim([0, 8])
plt.ylim([0, 8])
# add legend
plt.legend()
plt.show()
```

Rezultate:



Aplicație:

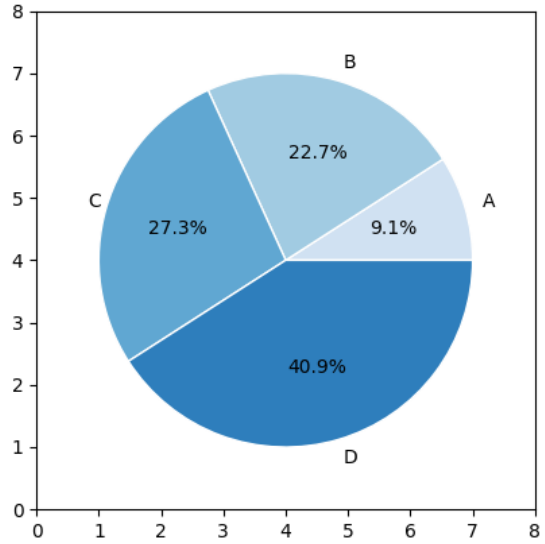
Să se modifice codul astfel incat sa se adauge un set D de date in acelasi grafic

Ex.9: Codul Python afiseaza valori generate in program in grafic de tip pie.

Cod Python

```
import matplotlib.pyplot as plt
import numpy as np
# make data
x = [10, 25, 30, 45]
#defineste culorile graficului pie: 5 nuante de albastru=len(x)
colors = plt.get_cmap('Blues')(np.linspace(0.2, 0.7, len(x)))
# plot
plt.pie(x, labels=['A', 'B', 'C', 'D'],colors=colors, radius=3, center=(4, 4),
wedgeprops={"linewidth": 1, "edgecolor": "white"}, frame=True, autopct='%1.1f%%')
plt.xlim([0, 8])
plt.ylim([0, 8])
plt.show()
```

Rezultate:



Aplicație:

Să se modifice codul astfel incat sa se reprezinte acest grafic cu date citite dintr-un fisier csv

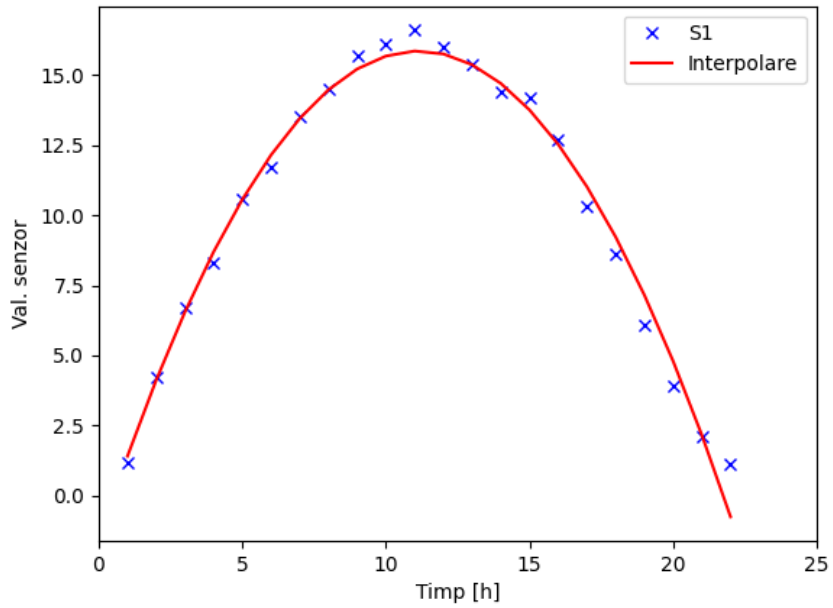
Ex.10: Codul Python creaza un fisier output.csv cu 2 coloane generate intr-un dataframe si afiseaza graficul valorilor. Interpolarea datelor preluate de la senzorul S1 se realizeaza observand ca valorile citite pentru acest senzor S1 se pot aproxima cu un polinom de ordin 2.

Cod Python

```
#import librarie op. matematice
import numpy as np
#import librarie plotare grafice
import matplotlib.pyplot as plt
#import librarie citire fisiere externe (csv, xlsx)
import pandas as pd
#import librarie fitare date experimentale
from scipy.optimize import curve_fit

#crearea unei structuri de date (Data frame) citite din fisierul extern (.csv)
date = pd.read_csv('date.csv')
#preluarea valorilor din structura date
x= date['ora']
y1 = date['S1']
y2 = date['S2']
#Interpolarea datelor preluate de la senzorul S1#
#dupa cum se poate observa, valorile citite din senzorul S1 se pot aproxima cu un polinom de ordin 2
#Definirea unei functii polinom de ordin 2
def fit(x, A, B, C):
    y = A*x**2+B*x+C
    return y
#Setarea interpolarii datelor de la S1 cu functia polinom - utilizarea curve_fit
parameters, covariance = curve_fit(fit, x, y1)
a = parameters[0]
b = parameters[1]
c = parameters[2]
#Afisarea parametrilor calculati in urma interpolarii
print('a=',a)
print('b=',b)
print('c=',c)
#Afisarea graficului valorilor citite de la S1 si interpolarea rezultatelor
plt.plot(x, y1,'bx',label='S1')
plt.plot(x, fit(x,a,b,c), 'r-', label='Interpolare')
plt.xlabel('Timp [h]')
plt.ylabel('Val. senzor')
plt.xlim(0,25)
plt.legend()
plt.show()
```

Rezultate:



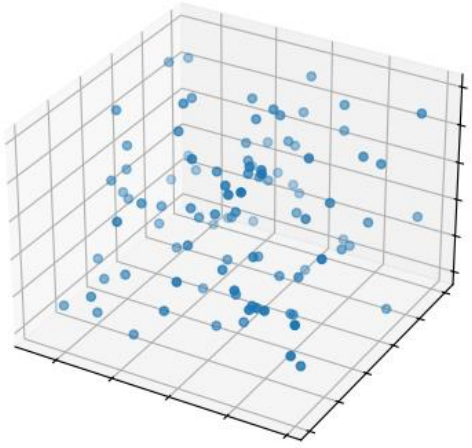

```
a= -0.14073969524813673
b= 3.133963866384361
c= -1.5857143060682413
```

Ex.11: Codul Python afiseaza graficul 3D scattered a datelor generate aleator in program.

Cod Python

```
import matplotlib.pyplot as plt
import numpy as np
# Make data : genereaza date aleator diferite la fiecare rulare 19680801 -
# arbitrar poate fi schimbat
np.random.seed(19680801)
#nr de valori generate=100
n = 100
#generate random numbers using various probability distributions.
rng = np.random.default_rng()
#generates n random numbers from a uniform distribution between 23 and 32.
xs = rng.uniform(23, 32, n)
#generates n random numbers from a uniform distribution between 0 and 100.
ys = rng.uniform(0, 100, n)
#generates n random numbers from a uniform distribution between -50 and -25.
zs = rng.uniform(-50, -25, n)
# Plot : creates a figure and a 3D axes object.
fig, ax = plt.subplots(subplot_kw={"projection": "3d"})
#plots the 3D scatter plot.
ax.scatter(xs, ys, zs)
# Remove tick labels
ax.set_xticklabels([])
ax.set_yticklabels([])
ax.set_zticklabels([])
plt.show()
```

Rezultate:



Aplicație:

Să se modifice codul astfel incat sa se reprezinte acest grafic cu date citite dintr-un fisier csv

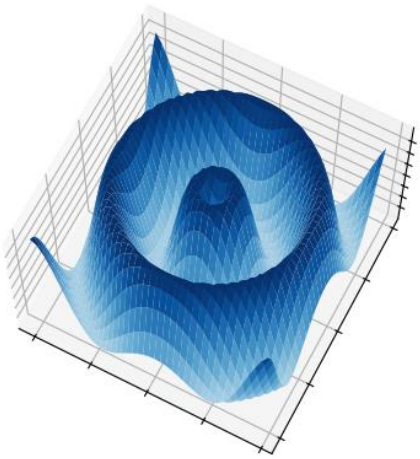
Ex.12: Codul Python afiseaza graficul 3D surface al datelor generate aleator in program.

Cod Python

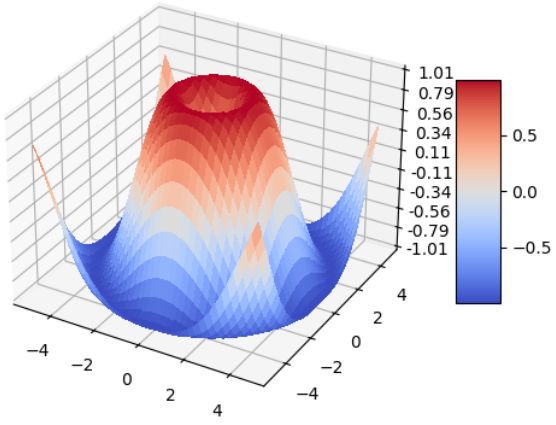
```
import matplotlib.pyplot as plt
import numpy as np
#importa cm= colormaps  din Matplotlib,   pentru reprezentare grafic culori.
from matplotlib import cm
# Make data : genereaza un array X si unul Y cu valori echidistante in (-10,10)
(exclusive) cu pas 0.5.
X = np.arange(-10, 10, 0.5)
Y = np.arange(-10, 10, 0.5)
#creaza o retea (grid) de puncte din X si Y  pentru afisarea suprafetei 3D
X, Y = np.meshgrid(X, Y)
#calculeaza  distanta fiecarui punct (X, Y) fata de origine (0, 0) utilizand
formula Euclidiana .
R = np.sqrt(X**2 + Y**2)
#Z se calculeaza ca sin(R), rezultand o suprafata sinusoidala
Z = np.sin(R)
# creaza suprafata 3D si afiseaza graficul cu Matplotlib
fig, ax = plt.subplots(subplot_kw={"projection": "3d"})
ax.plot_surface(X, Y, Z, vmin=Z.min() * 2, cmap=cm.Blues)
ax.set(xticklabels=[],
       yticklabels=[],
       zticklabels=[])
plt.show()
```

Rezultate:

`cmap=cm.Blues`



`cmap=cm.coolwarm`



Aplicație:

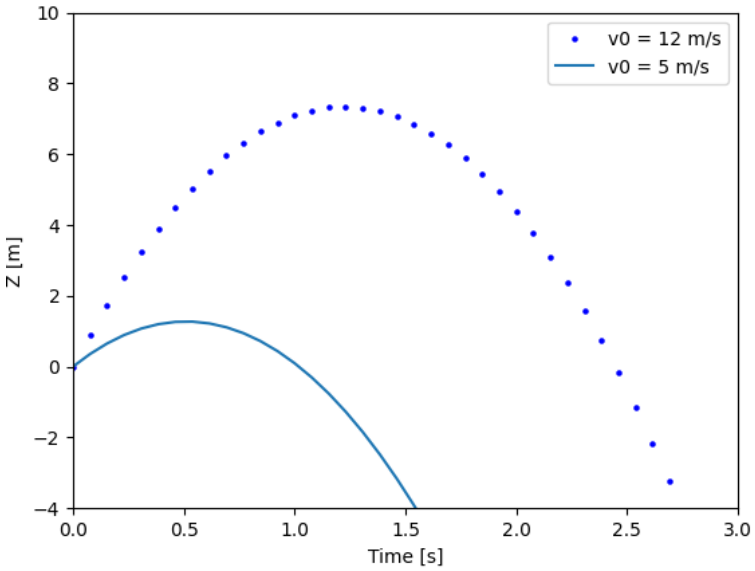
Să se modifice in cod expresia functiei din definirea coordonatei Z

Ex.13: Codul Python afiseaza o animație care ilustreaza traiectoria unui proiectil cu două viteze inițiale diferite (v0 și v02). Pe măsură ce trece timpul, se observa că traiectoria se schimbă în funcție de viteza inițială..

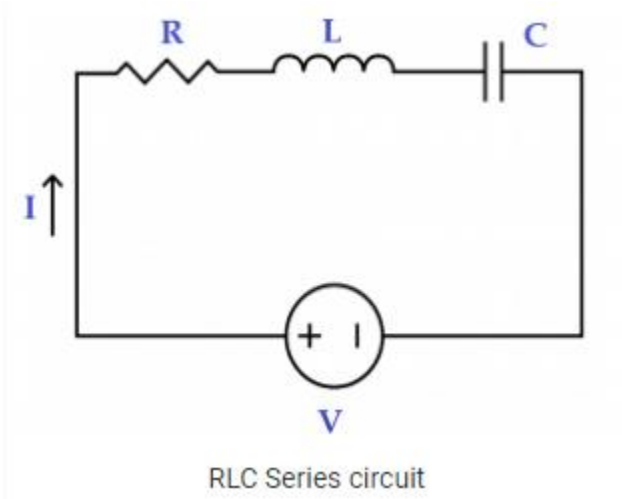
Cod Python

```
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.animation as animation
#se creeaza o figura si un suplot
fig, ax = plt.subplots()
#se genereaza variabilele ce vor fi reprezentate grafic: 40 de numere uniform
distanțate in (0,3).
t = np.linspace(0, 3, 40)
#se definesc variabile g,v0 si v02 care sunt constante
#g este accelerația datorată gravitației, v0 și v02 sunt viteze inițiale.
g = 9.81
v0 = 12
# se definesc si calculeaza z și z2= înălțimile proiectilului în
# momente diferite (t) pentru două viteze inițiale diferite.
z = -g * t**2 / 2 + v0 * t
v02 = 5
z2 = -g * t**2 / 2 + v02 * t
#se creeaza un grafic de dispersie cu un singur punct pentru pozitia initiala
#a proiectilului, de culoare albastră (c="b") si dimensiunea markerului  (s=5).
#eticheta graficului este v0=12m/s.
scat = ax.scatter(t[0], z[0], c="b", s=5, label=f'v0 = {v0} m/s')
#similar se creeaza un grafic liniar pentru cealalta viteza initiala v02
line2 = ax.plot(t[0], z2[0], label=f'v0 = {v02} m/s')[0]
#setează proprietatile axelor:limite ,etichete axe,legenda
ax.set(xlim=[0, 3], ylim=[-4, 10], xlabel='Time [s]', ylabel='Z [m]')
ax.legend()
#se creeaza o functie ce actualizeaza datele din graficul de dispersie si cel
liniar pentru fiecare cadru al animatiei.
#se actualizează pozițiile graficului de dispersie și graficului liniar pe baza
numărului de cadre.
def update(frame):
    # for each frame, update the data stored on each artist.
    x = t[:frame]
    y = z[:frame]
    # update the scatter plot:
    data = np.stack([x, y]).T
    scat.set_offsets(data)
    # update the line plot:
    line2.set_xdata(t[:frame])
    line2.set_ydata(z2[:frame])
    return (scat, line2)
#se creeaza un obiect de tip animație folosind FuncAnimation.
#acesta va anima  figura (fig), are functia de actualizare (func=update),
#iar nr de cadre (frames=50) și intervalul dintre cadre în milisecunde
(interval=40).
ani = animation.FuncAnimation(fig=fig, func=update, frames=50, interval=40)
plt.show()
```

Rezultate:



Ex.14: Codul Python afiseaza graficul 2D al curentului I intr-un circuit RLC , in care se cunosc valorile R=10 Ohmi, L=0.1 Henry, C=0.01 Farazi si V=5 Volti:



, unde

$$I(t) = \frac{V}{\sqrt{R^2 + \left(\omega_0 L - \frac{1}{\omega_0 C}\right)^2}} e^{-\alpha t} \sin(\omega_0 t)$$

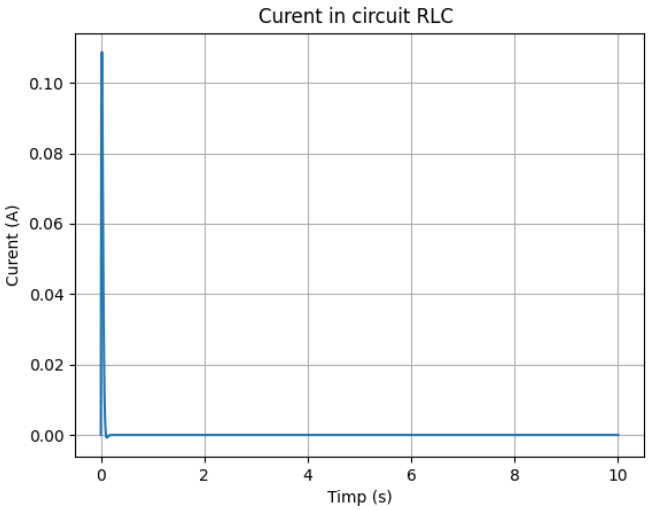
- $\omega_0 = \frac{1}{\sqrt{LC}}$
- $\alpha = \frac{R}{2L}$

Cod Python

```
import numpy as np
import matplotlib.pyplot as plt
# Definire parametri circuit
R = 10  # Rezistenta in ohmi
L = 0.1  # Inductanta in Henry
C = 0.01  # Capacitatea in Farad
V = 5  # Tensiunea electrica in Volti
# Generare sir timp (sec) in (0,10) , 1000 valori
t = np.linspace(0, 10, 1000)
# Calculeaza curent in circuit RLC
omega0 = 1 / np.sqrt(L * C)
alpha = R / (2 * L)
I=V/np.sqrt(R**2 + (omega0*L-1/(omega0*C))**2)*np.exp(-alpha*t)*np.sin(omega0*t)
# Plot curent
plt.plot(t, I)
```

```
plt.title('Curent in circuit RLC ')
plt.xlabel('Timp (s)')
plt.ylabel('Curent (A)')
plt.grid(True)
plt.show()
```

Rezultate:



Aplicație:

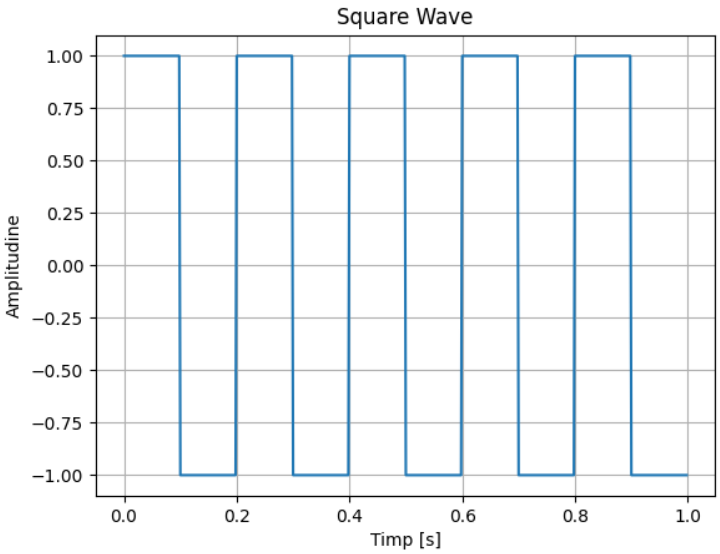
Să se modifice codul astfel incat sa se reprezinte acest grafic cu date citite dintr-un fisier txt

Ex.15: Codul Python genereaza si afiseaza o unda patrata (square wave)

Cod Python

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import signal
#genereaza o unda de forma patrata : square wave
#initial se genereaza 500 de puncte in intervalul (0,1)
#fara capetele intervalului (endpoint=False)
t = np.linspace(0, 1, 500, endpoint=False)
#apoi se genereaza unda ca un sir de valori cu freventa unghiulara=2*pi*f,f=5*t
square_wave = signal.square(2 * np.pi *5*  t)
# Plot unda patrata (square wave)
plt.plot(t, square_wave)
plt.xlabel('Timp [s]')
plt.ylabel('Amplitudine')
plt.title('Square Wave')
plt.grid()
plt.show()
```

Rezultate:



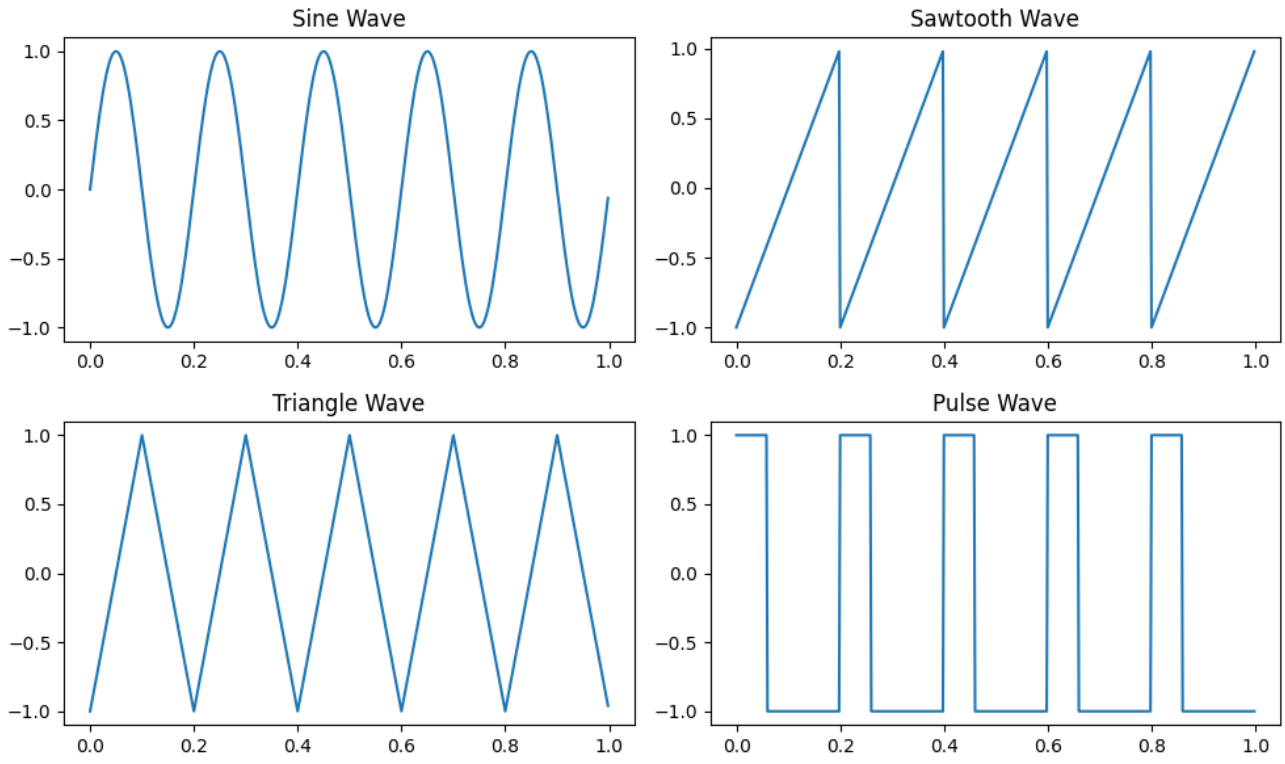
Aplicație:

Să se modifice codul astfel incat sa se reprezinte acest grafic si alte forme de unda:

Rezolvare:

Cod Python

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import signal
# Generate time vector
t = np.linspace(0, 1, 500, endpoint=False)
# Generate waveforms
sine_wave = np.sin(2 * np.pi * 5 * t)
sawtooth_wave = signal.sawtooth(2 * np.pi * 5 * t)
triangle_wave = signal.sawtooth(2 * np.pi * 5 * t, width=0.5)
pulse_wave = signal.square(2 * np.pi * 5 * t, duty=0.3)
# Plot waveforms
plt.figure(figsize=(10, 6))
plt.subplot(2, 2, 1)
plt.plot(t, sine_wave)
plt.title('Sine Wave')
plt.subplot(2, 2, 2)
plt.plot(t, sawtooth_wave)
plt.title('Sawtooth Wave')
plt.subplot(2, 2, 3)
plt.plot(t, triangle_wave)
plt.title('Triangle Wave')
plt.subplot(2, 2, 4)
plt.plot(t, pulse_wave)
plt.title('Pulse Wave')
plt.tight_layout()
plt.show()
```



PROBLEME PROPUSE

1. ***Să se scrie un cod Python care citeste si afiseaza valorile reale dintr-un fisier csv , dispuse pe o coloana, si afiseaza graficul functiei f(x) = ax^2+b, unde a si b sunt valori reale introduse de utilizator.***

2. ***Sa se scrie un cod Python care afiseaza graficul valorilor dintr-un fisier csv in mai multe variante de reprezentare (scattered, line, pie, etc). Fisier .csv exemplu: https://www.w3schools.com/python/pandas/trypandas.asp?filename=demo_pandas_plot.***