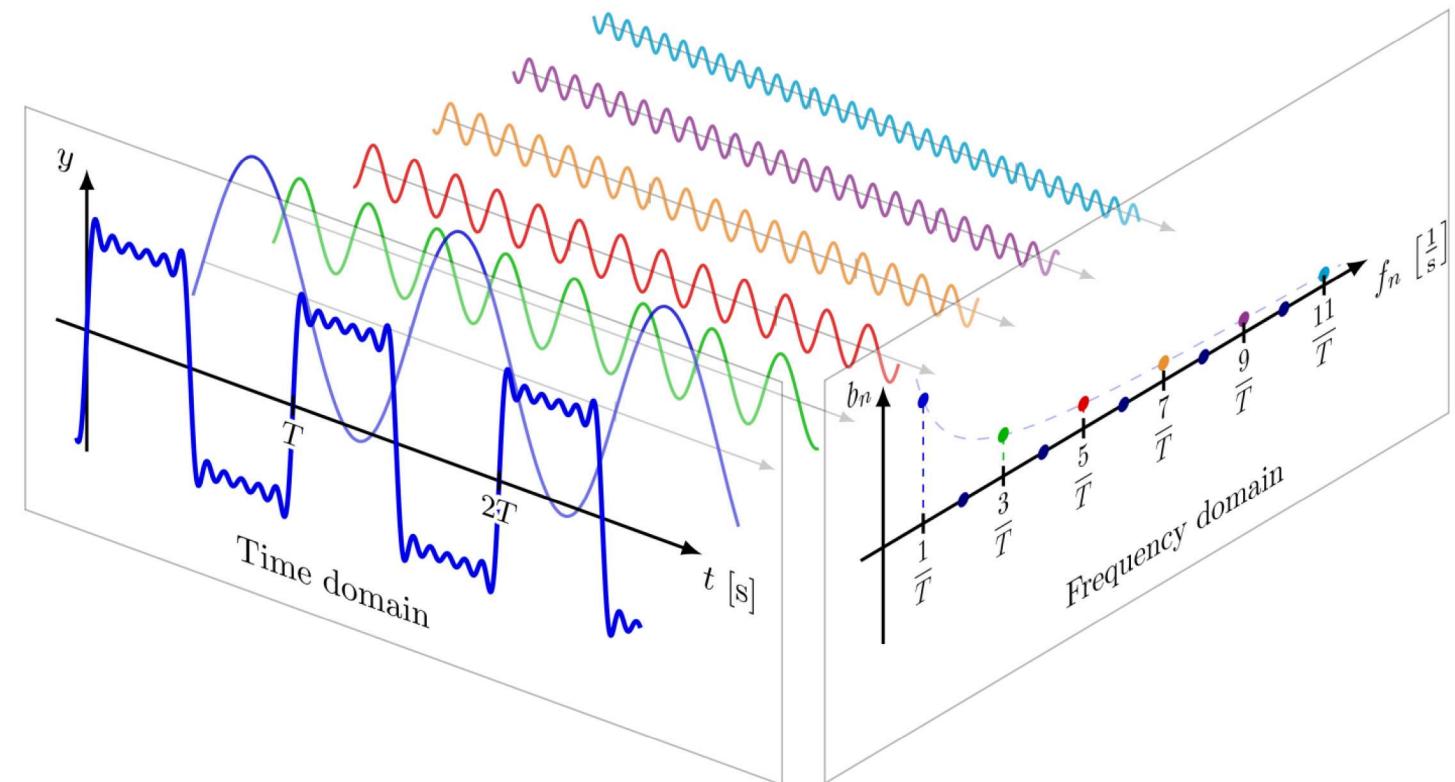


# CURS 5 PD

## Introducere în limbajul Python

### Librăria Matplotlib



**Introducere**

**Reprezentări grafice**

**Adnotarea graficelor**

**Afișarea imaginilor**

1. Introducere

2. Reprezentări grafice

3. Adnotarea graficelor

4. Afişarea imaginilor

Reprezentări grafice

Adnotarea graficelor

# Introducere

Afişarea imaginilor

## Definiție

**Matplotlib** este o librărie **Python** utilizată pentru generarea și afișarea graficelor. Această librărie conține un sub-modul numit **pyplot** ce oferă un mod de lucru și o sintaxă asemănătoare **MATLAB-ului**

- Librăria **Matplotlib** necesită instalare (aşa cum s-a văzut la primul curs).
- Întrucât vom folosi doar sub-modulul **pyplot**, îl vom importa doar pe acesta scriind instrucțiunea `import` urmată de numele librăriei (**matplotlib**), simbolul punct și numele sub-modului (**pyplot**)
- Pentru a economisi timp, putem să utilizăm un alias. În mod uzual, pentru acest sub-modul se foloseşte alias-ul **plt**.

## Importare cu alias

```
import matplotlib.pyplot as plt
```

1. Introducere

2. Reprezentări grafice

3. Adnotarea graficelor

4. Afişarea imaginilor

Reprezentări grafice

Adnotarea graficelor

# Introducere

Afişarea imaginilor

1. Introducere

2. Reprezentări grafice

3. Adnotarea graficelor

4. Afisarea imaginilor

Introducere

Adnotarea graficelor

# Reprezentări grafice

Afisarea imaginilor

- Pentru reprezentări grafice ale datelor, **Matplotlib** pune la dispoziție o multitudine de funcții, din care, cea mai uzuală este funcția **plt.plot** ce are o sintaxă asemănătoare cu funcția **plot** din **MATLAB**.
- Această funcție primește ca parametrii două matrice unidimensionale reprezentând valorile de pe abscisă, respectiv valorile de pe ordonată, și, optional, un parametru de formatare de tip sir de caractere ce poate specifica atât culoarea liniei, cât și tipul acesteia și tipul de marcator:

Culoare		Tip linie		Tip marcaj			
Simbol	Descriere	Simbol	Descriere	Simbol	Descriere	Simbol	Descriere
y	galben	-	continuă	+	plus	^	triunghi <b>▲</b>
m	magenta	--	întreruptă	o	cerc	v	triunghi <b>▼</b>
c	turcoaz	:	punctată	*	steluță	>	triunghi <b>►</b>
r	roșu	- .	linie-punct	.	punct	<	triunghi <b>◀</b>
g	verde			x	x	p	pentagon
b	albastru			s	pătrat	h	hexagon
w	alb			d	romb		
k	negră						

- După ce s-au adăugat toate elementele necesare pe grafic, pentru a putea vedea rezultatul, trebuie apelată funcția **plt.show()**.
- Atenție**, codul scris după apelarea funcției **plt.show()** se va executa doar după închiderea graficului.

## EXEMPLU

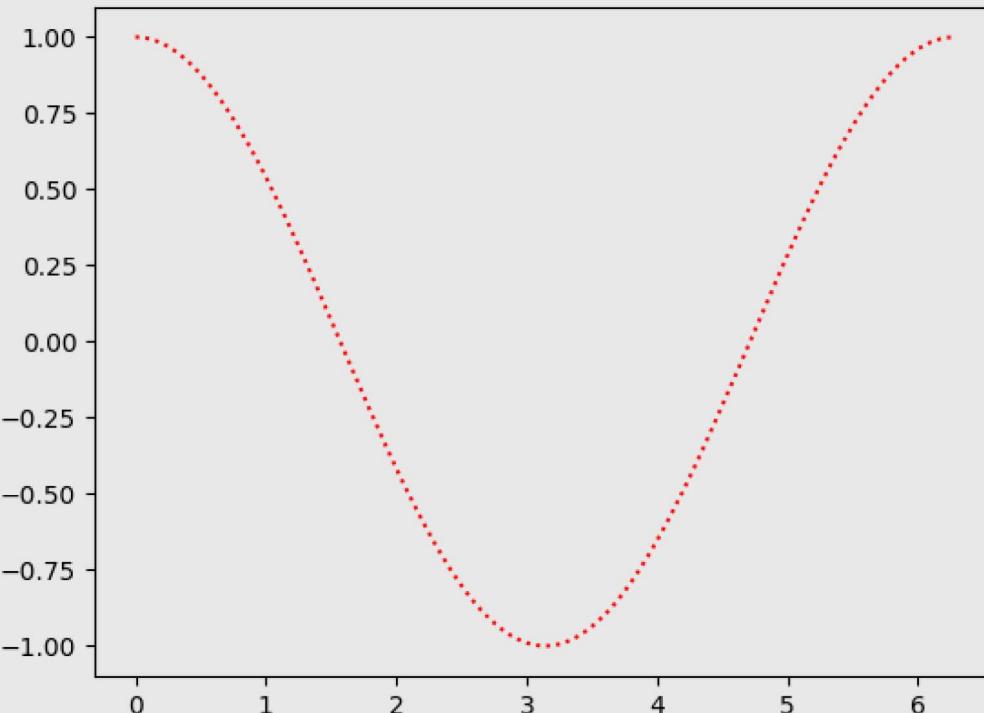
## main.py

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

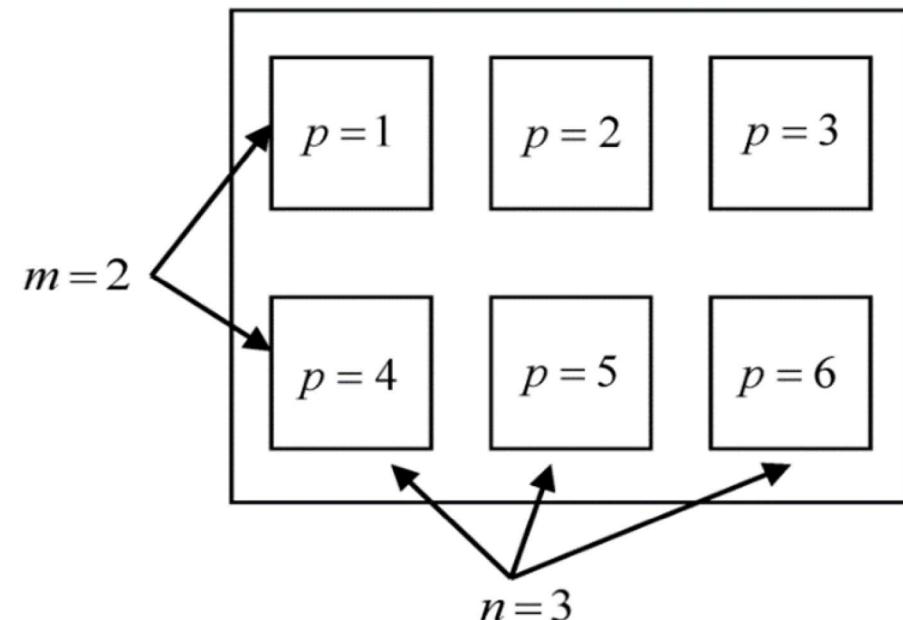
x = np.arange(0, 2 * np.pi, 0.01)
y = np.cos(x)

plt.plot(x, y, "r:")
plt.show()
```

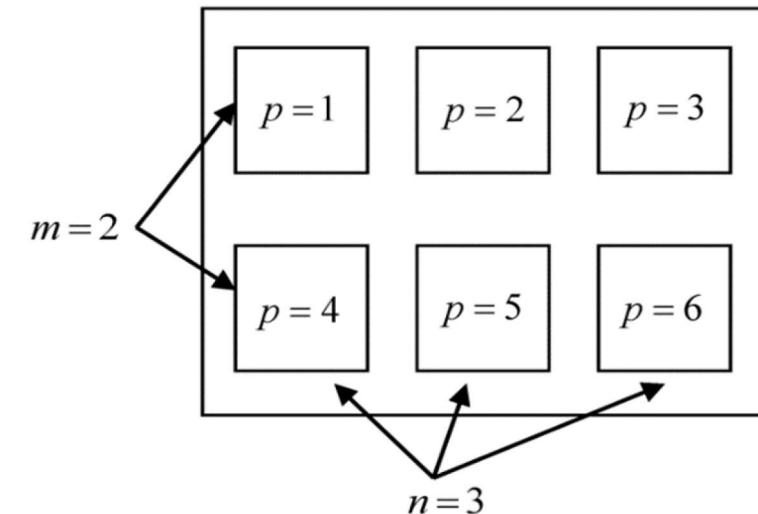
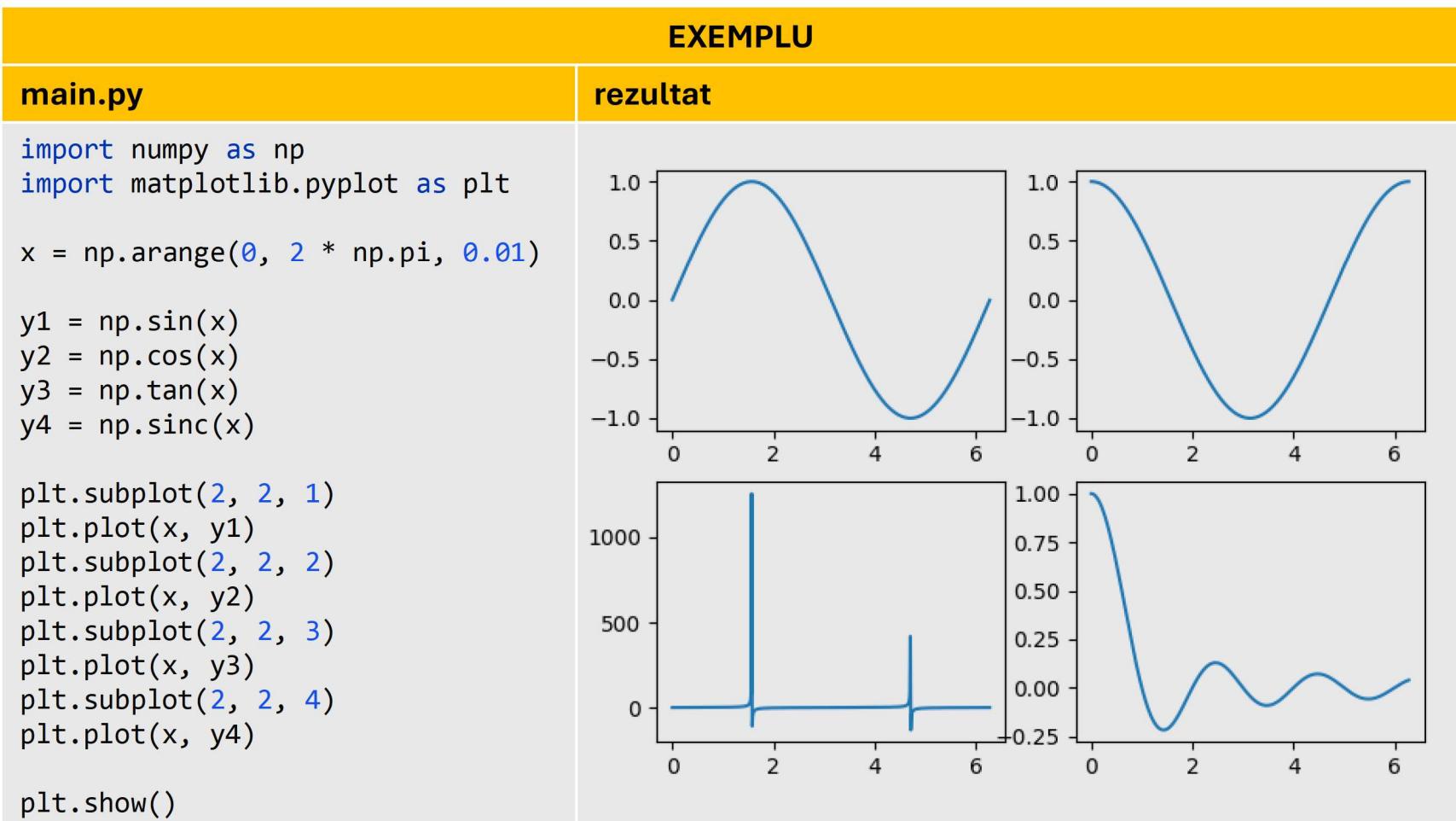
## rezultat



- Afișarea mai multor grafice, se poate face fie în ferestre diferite, fie în cadrul aceleiași ferestre:
  - Pentru afișare în ferestre diferite, este necesară apelarea funcției **plt.figure(n)** (unde **n** reprezintă numărul figurii / ferestrei) înainte de apelarea funcției **plt.plot**.
  - Pentru a afișa mai multe grafice în aceeași fereastră există trei posibilități:
    - se poate apela funcția **plt.plot** de mai multe ori, pentru fiecare set de date ce se dorește reprezentat grafic; nu este necesară utilizarea unui echivalent pentru **hold on**.
    - se pot da mai mulți parametrii funcției **plt.plot**:
      - plt.plot(x1, y1, x2, y2, ...)**
      - plt.plot(x1, y1, format1, x2, y2, format2, ...)**
    - se poate utiliza funcția **plt.subplot(m, n, p)** ce va împărți fereastra într-o matrice de sub-ferestre dispuse pe **m** rânduri și **n** coloane, și va selecta sub-fereastra cu numărul **p** pentru a afișa ulterior în ea. Numerotarea sub-ferestrelor se face la fel ca în MATLAB, începând cu numărul 1, de la stânga la dreapta și de sus în jos.



3. se poate utiliza funcția `plt.subplot(m, n, p)` ce va împărți fereastra într-o matrice de sub-ferestre dispuse pe **m** rânduri și **n** coloane, și va selecta sub-fereastra cu numărul **p** pentru a afișa ulterior în ea. Numerotarea sub-ferestrelor se face la fel ca în MATLAB, începând cu numărul 1, de la stânga la dreapta și de sus în jos.



- Asemănător mediului **MATLAB**, **Matplotlib** permite reprezentarea grafică și în coordonate logaritmice sau semilogaritmice:

Funcția	Descriere
<code>plt.semilogx(x, y)</code>	grafic cu axa x logaritmică
<code>plt.semilogy(x, y)</code>	grafic cu axa y logaritmică
<code>plt.loglog(x, y)</code>	grafic cu ambele axe logaritmice

### EXEMPLU

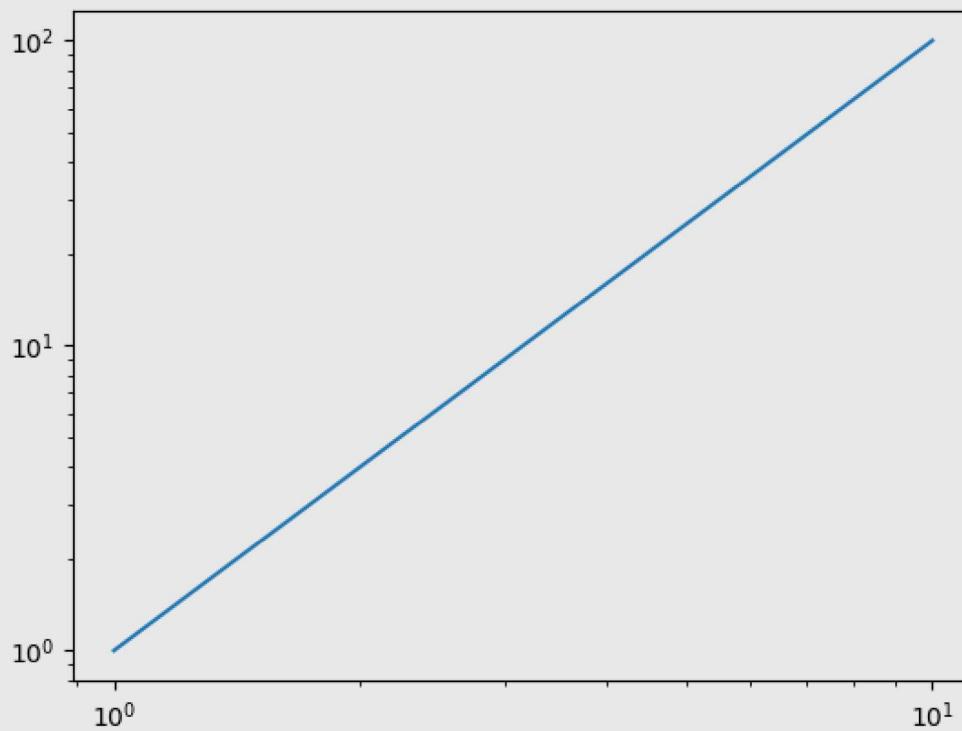
**main.py**

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

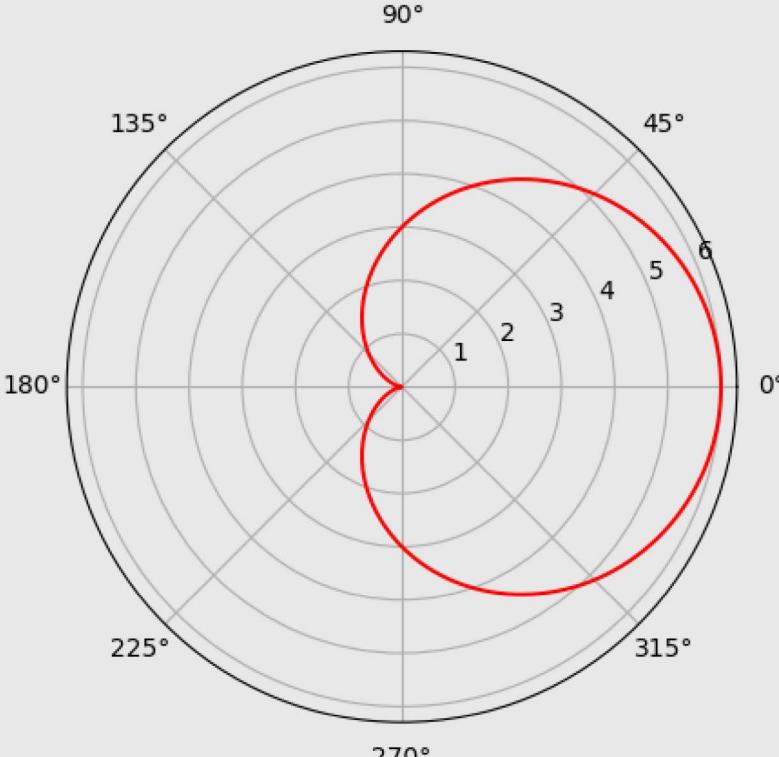
x = np.linspace(1, 10, 100)
y = x**2

plt.loglog(x, y)

plt.show()
```

**rezultat**

- **Matplotlib** mai permite și reprezentarea grafică în coordonate polare utilizând funcția `plt.polar(theta, r)`, unde `theta` și `r` sunt matrice unidimensionale specificând unghiurile, respectiv distanțele

EXEMPLU	
main.py	rezultat
<pre>import numpy as np import matplotlib.pyplot as plt  theta = np.arange(0, 2 * np.pi, 0.01) r = 3 * (np.cos(theta) + 1)  plt.polar(theta, r, 'r') plt.show()</pre>	

- Alte funcții specializede pentru reprezentări grafice:

Funcția	Descriere
<code>plt.scatter(x, y)</code>	reprezentare grafică ca puncte
<code>plt.bar(x, y)</code>	reprezentare grafică cu bare
<code>plt.barh(y, x)</code>	reprezentare grafică cu bare orizontale
<code>plt.fill(x, y)</code>	reprezentare grafică ca poligon
<code>plt.stem(x, y)</code>	reprezentare grafică ca eșantioane
<code>plt.step(x, y)</code>	reprezentare grafică în trepte
<code>plt.pie(x, labels)</code>	reprezentare grafică tip „pie chart”
<code>plt.errorbar(x, y, err)</code>	reprezentare grafică cu erori

1. Introducere

2. Reprezentări grafice

3. Adnotarea graficelor

4. Afisarea imaginilor

Introducere

Adnotarea graficelor

# Reprezentări grafice

Afișarea imaginilor

1. Introducere

2. Reprezentări grafice

3. Adnotarea graficelor

4. Afişarea imaginilor

Introducere

Reprezentări grafice

# Adnotarea graficelor

Afişarea imaginilor

- De obicei este necesar ca pe grafic să apară mai multe detalii despre reprezentare, aşa că **Matplotlib** implementează o serie de funcții pentru cosmetizarea graficului. Cele mai importante sunt descrise în tabelul următor:

Funcția	Descriere
<code>plt.title(s)</code>	setează titlul graficului la valoarea șirului de caractere <b>s</b>
<code>plt.suptitle(s)</code>	setează titlul superior al graficului la valoarea șirului de caractere <b>s</b>
<code>plt.xlabel(s)</code>	setează eticheta axei x la valoarea șirului de caractere <b>s</b>
<code>plt.ylabel(s)</code>	setează eticheta axei y la valoarea șirului de caractere <b>s</b>
<code>plt.grid()</code>	adaugă grila pe grafic
<code>plt.xlim(xmin, xmax)</code>	setează intervalul de afișare al axei x
<code>plt.ylim(ymin, ymax)</code>	setează intervalul de afișare al axei y
<code>plt.legend(lista)</code>	adaugă o legendă, fiecărui grafic atribuindu-i-se o etichetă din lista primită ca parametru
<code>plt.text(x, y, s)</code>	adaugă textul <b>s</b> la coordonatele <b>(x, y)</b>
<code>plt.annotate(s, xy, xy2, arrowprops = {})</code>	adaugă textul <b>s</b> la coordonatele specificate de tuplul <b>xy2</b> și săgeată spre coordonatele date de tuplul <b>xy</b>

## EXEMPLU

## main.py

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

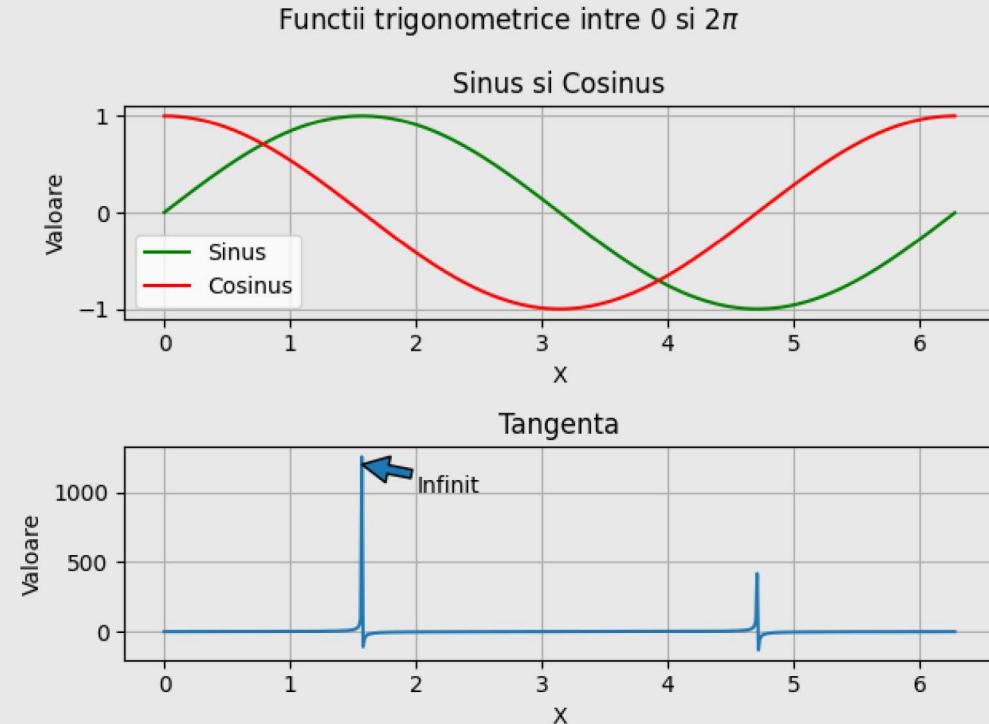
x = np.arange(0, 2 * np.pi, 0.01)

y1 = np.sin(x)
y2 = np.cos(x)
y3 = np.tan(x)

plt.subplot(2, 1, 1)
plt.plot(x, y1, 'g', x, y2, 'r')
plt.xlabel("X")
plt.ylabel("Valoare")
plt.title("Sinus si Cosinus")
plt.grid()
plt.legend(["Sinus", "Cosinus"])
```

```
plt.subplot(2, 1, 2)
plt.plot(x, y3)
plt.xlabel("X")
plt.ylabel("Valoare")
plt.title("Tangenta")
plt.annotate("Infinit", (np.pi / 2, 1200),
            (2, 1000), arrowprops={})
plt.grid()
plt.suptitle("Functii trigonometrice intre 0 si 2$\pi$")
plt.tight_layout()
plt.show()
```

## rezultat



- Toate funcțiile pentru cosmetizarea graficelor ce primesc ca parametru un sir de caractere permit utilizare, în sirul de caractere, a sintaxei **TeX** între două simboluri \$. La adresa <https://matplotlib.org/stable/tutorials/text/mathtext.html> se poate consulta o descriere detaliată a acestei sintaxe.

1. Introducere

2. Reprezentări grafice

3. Adnotarea graficelor

4. Afişarea imaginilor

Introducere

Reprezentări grafice

# Adnotarea graficelor

Afişarea imaginilor

1. Introducere

2. Reprezentări grafice

3. Adnotarea graficelor

4. Afişarea imaginilor

Introducere

Reprezentări grafice

Adnotarea graficelor

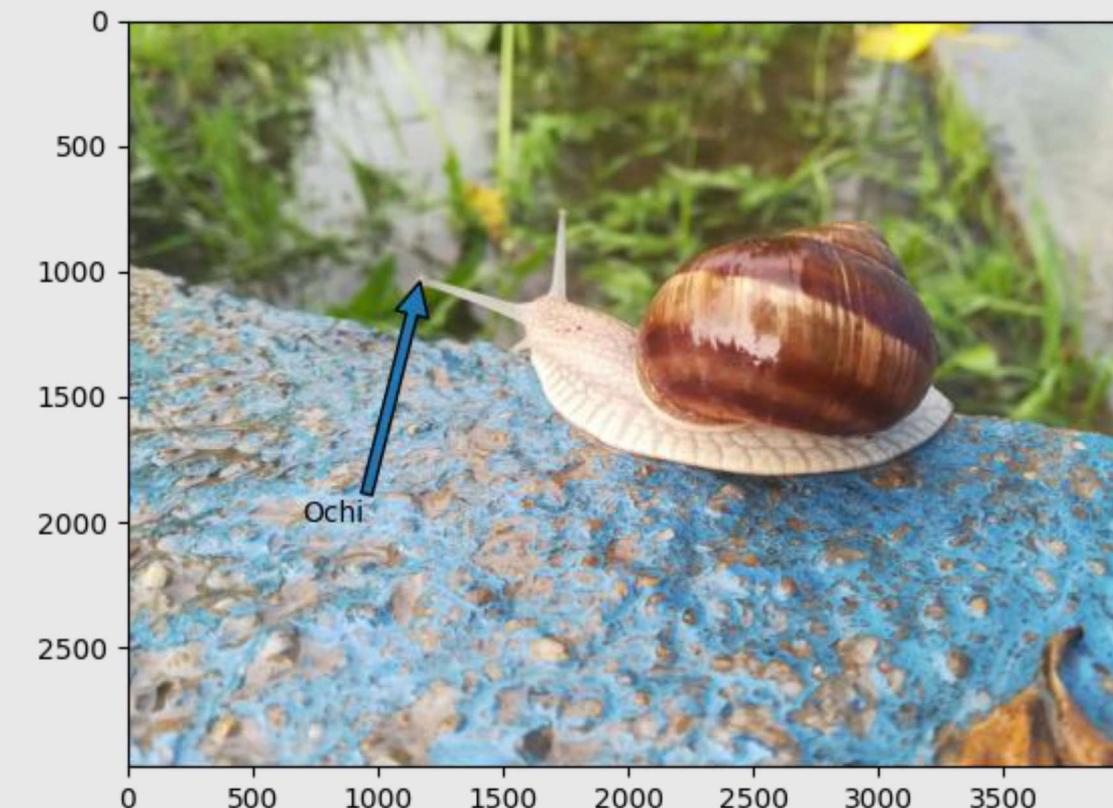
# Afișarea imaginilor

- **Matplotlib** implementează, în sub-modulul **pyplot**, funcțiile:
  - **imread** ce primește ca parametru calea spre o imagine și returnează o matrice **NumPy** cu eșantioanele imaginii
  - **imshow** ce primește ca parametru o astfel de matrice și o afișează grafic.

**EXEMPLU****main.py**

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

x = plt.imread("melc.jpg")
plt.imshow(x)
plt.annotate("Ochi", (1170, 1040), (700, 2000), arrowprops={})
plt.show()
```

**rezultat**

1. Introducere

2. Reprezentări grafice

3. Adnotarea graficelor

4. Afişarea imaginilor

Introducere

Reprezentări grafice

Adnotarea graficelor

# Afișarea imaginilor

1. Introducere

2. Reprezentări grafice

3. Adnotarea graficelor

4. Afişarea imaginilor

**Introducere**

**Reprezentări grafice**

**Adnotarea graficelor**

**Afişarea imaginilor**