

Python 101

Curs 4 - Module
10.04.2023

The background features abstract geometric shapes: a large blue circle on the left, a smaller blue circle at the bottom right, and several thin, light gray circular arcs. Small gray dots are placed at the intersections of these arcs.

Quiz time!

Module

Module

- Orice fișier ce conține cod sursă Python este un modul.

De ce mai multe?

- Modulele ajută la reutilizarea codului.
- Modulele sunt o modalitate de a împărți programele mari în componente mai mici și mai ușor de urmărit.

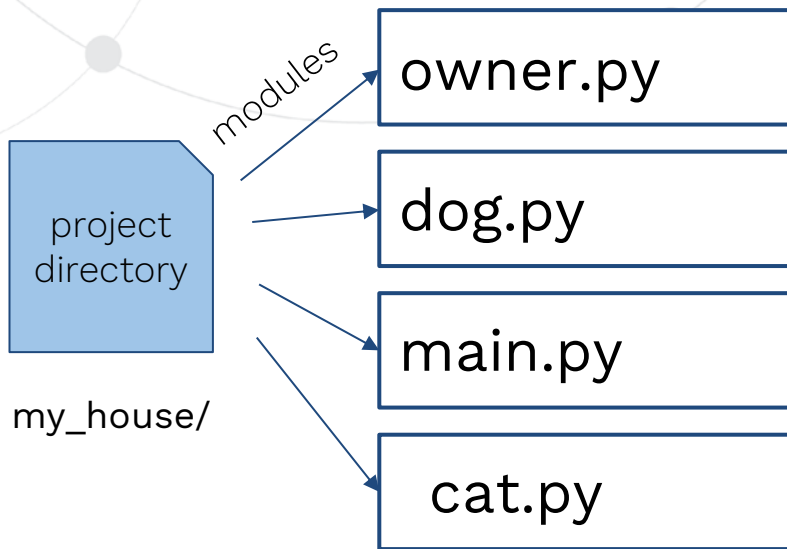
Try to not be like this guy!

But soon, you'll start to work with other people's code, and that can look like this.



Crearea unui modul

- Într-un modul putem avea definiții de funcții și/sau clase.



```
class Owner:
    def __init__(self, name):
        self.name = name

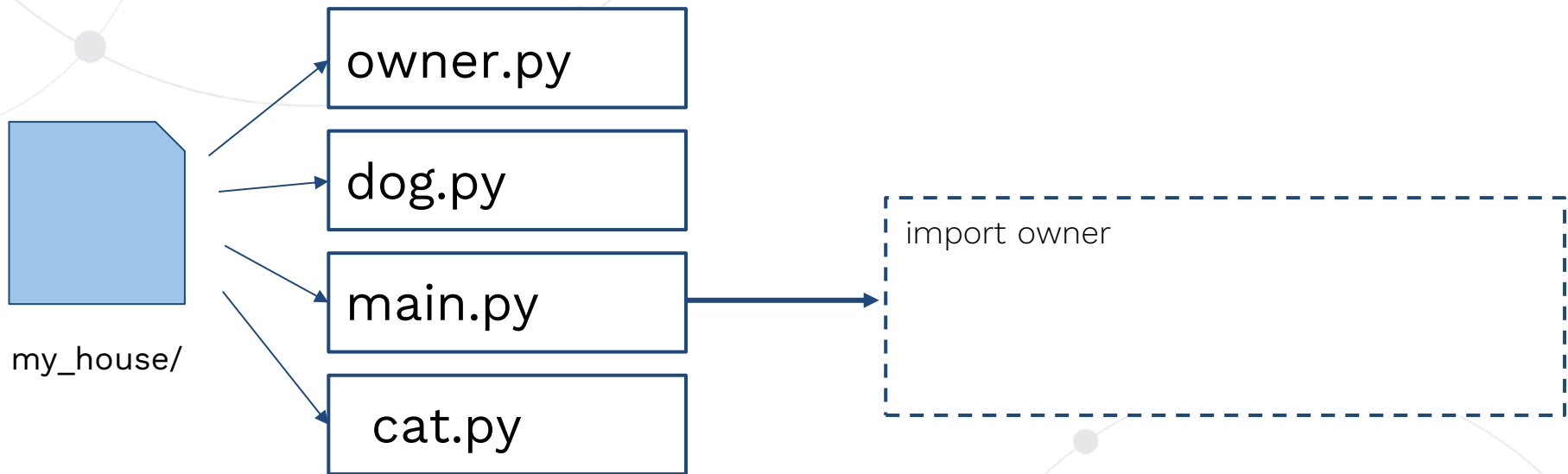
    def greet_pet(self, pet_name):
        print("Hello " +
              pet_name + "!")

#not in class scope
def hug(name_1, name_2):
    print(name_1 + " hugs " +
          name_2)
```

Importarea unui modul(1)

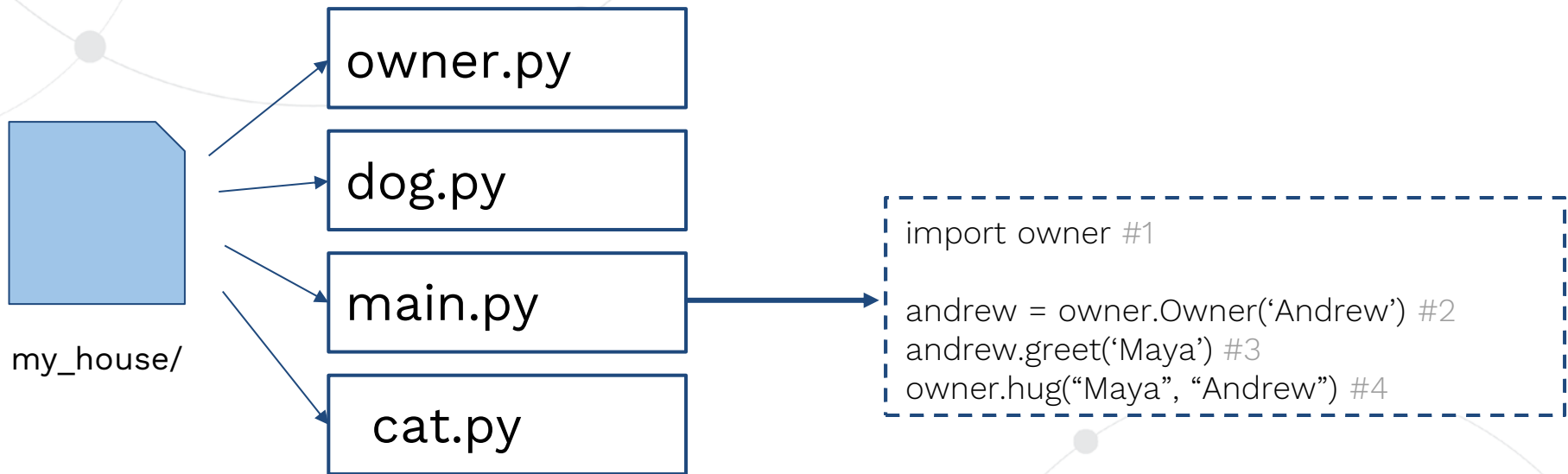
- Putem să importăm într-un alt fișier un modul folosind

import <nume_modul>



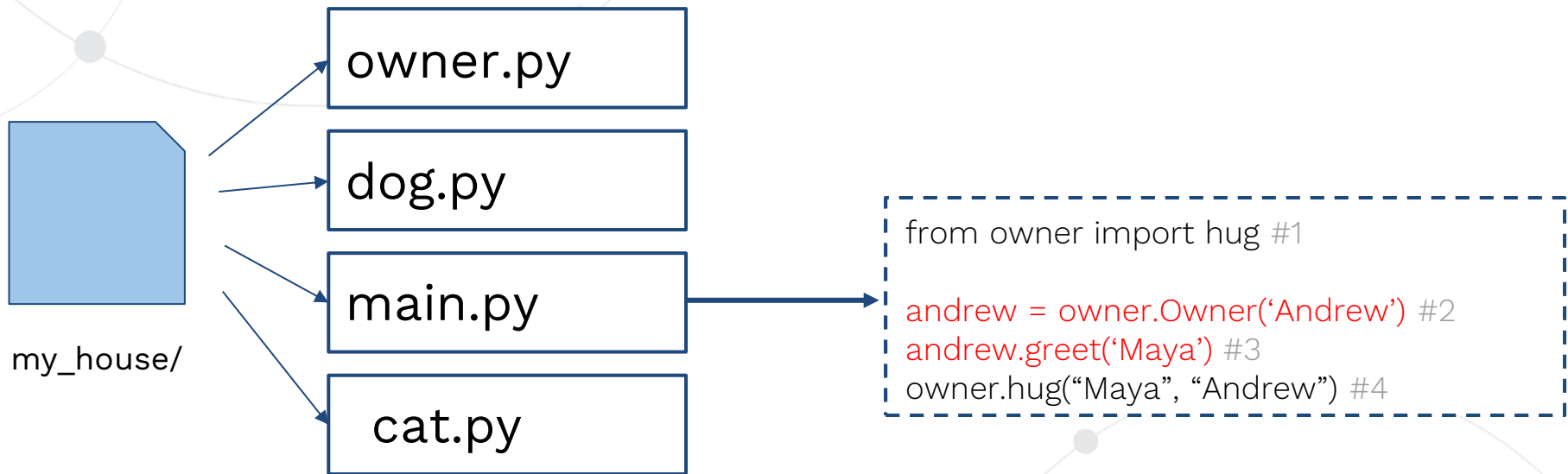
Importarea unui modul(2)

- Putem folosi funcțiile/clasele din acel modul cu **<nume_modul>.<nume_funcție/ nume_clasa>**



Importarea unui modul(3)

- Dintr-un modul se pot importa doar anumite funcții și/sau clase cu
from <nume_modul> import <nume_funcție/ nume_clasa>

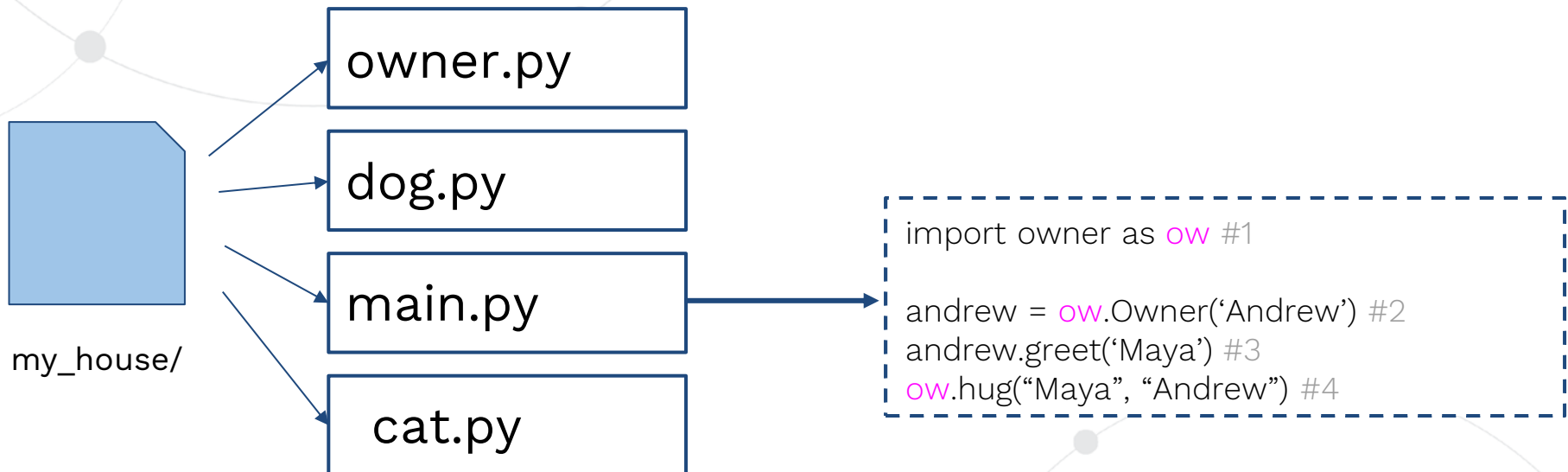


EROARE la randurile #2 și #3!
De ce?

Importarea unui modul(4)

- La importarea unui modul, se poate stabili și un **alias** pentru numele modulului.

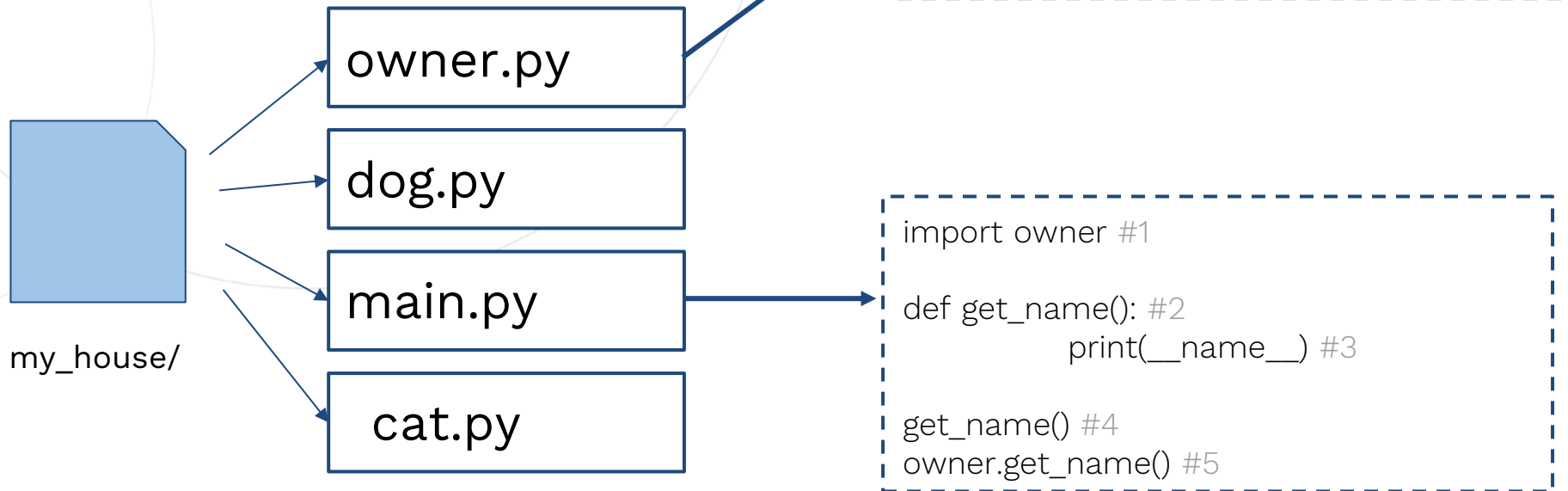
import <nume_modul> as <nume_alias>



Main

- Funcția main este punctul de început în orice limbaj de programare.
- Pentru orice program se va seta variabila **__name__** astfel:
 - **__name__** = **__main__** fișierul dat ca parametru interpretorului
 - **__name__** = nume modul pentru restul modulelor

Main - exemplu



```
student@arch:~$ python3 main.py
```

```
__main__
owner
```

Dar daca apelam `get_name()` din `owner.py` si rulam comanda “`python3 owner.py`”? Ce s-ar fi afisat?

Modulul sys

- Modulul permite interacțiunea cu interpretorul și preluarea argumentelor din linia de comandă.

test_sys.py

```
import sys

print(sys.argv)
#argumentele din linia de comanda
print(sys.platform)
#sistemul de operare
```

```
student@arch:~$ python3 test_sys.py arg1 45
['test_sys.py', 'arg1', '45']
linux
```

Modulul math

- Furnizează operații matematice:
`pow, sqrt, exp, sin, cos, ...`

test_math.p

y

```
import math  
  
print(math.sqrt(5))  
print(math.trunc(2.5))  
print(math.sin(math.pi))
```

```
student@arch:~$ python3 test_math.py  
2.23606797749979  
2  
1.2246467991473532e-16
```

Modulul random

- Generator de rezultate întâmplătoare.

random.py

```
import random as rd

x = rd.randrange(5)
#numar random intre 0 si 5
y = rd.randrange(4, 10)
#numar random intre 4 si 10
z = rd.randrange(3, 6, 2)
#numar random intre 3 si 6 cu pas 2
l = ['rock', 'jazz', 'techno']
genre = rd.choice(l)
#alegere intamplatoare dintr-o lista
print(f"{x},{y},{z},{genre}")
```

```
student@arch:~$ python3 random.py
0,7,5,rock
```

Modulul time

- Folosit la măsurarea timpului și întârzierea execuției.

test_time.py

```
from time import time, sleep  
  
t1 = time() #timpul de acum  
sleep(3) #asteapta 3 secunde  
t2 = time()  
  
print(t2 - t1)
```

```
student@arch:~$ python3 test_time.py  
3.0050642490386963
```


The background features abstract geometric elements. On the left, a large blue circle is partially visible. A thin grey arc with two small grey dots connects this circle to another blue circle in the bottom right corner. The word 'Pauză' is centered in the right half of the image.

Pauză

Modulul numpy

- Procesor matematic similar cu Matlab.
- Suport pentru operații complexe: operații vectoriale, gradient.

numpy.py

```
import numpy as np
```

Numpy array

- Echivalentul unei liste.
- Suportă operații mai avansate.

numpy.py

```
import numpy as np

c = np.zeros(4)
#array cu 4 elemente initializate cu 0
a = np.array([1,2,3,4], dtype = 'float')
#un array format din elementele listei
schimbate in tipul float
d = np.full((3, 3), 6, dtype = 'complex')
#o matrice 3x3 cu elementele initializate cu 6
sub forma numerelor complexe
print(f"c = {c}\na = {a}\nd = {d}")
```

```
student@arch:~$ python3 numpy.py
```

```
c = [0. 0. 0. 0.]
```

```
a = [1. 2. 3. 4.]
```

```
d = [[6.+0.j 6.+0.j 6.+0.j]
```

```
      [6.+0.j 6.+0.j 6.+0.j]
```

```
      [6.+0.j 6.+0.j 6.+0.j]]
```

Numpy array

- Suportă valori uniform distribuite.
- Operațiile se aplică tuturor elementelor array-ului.

numpy.py

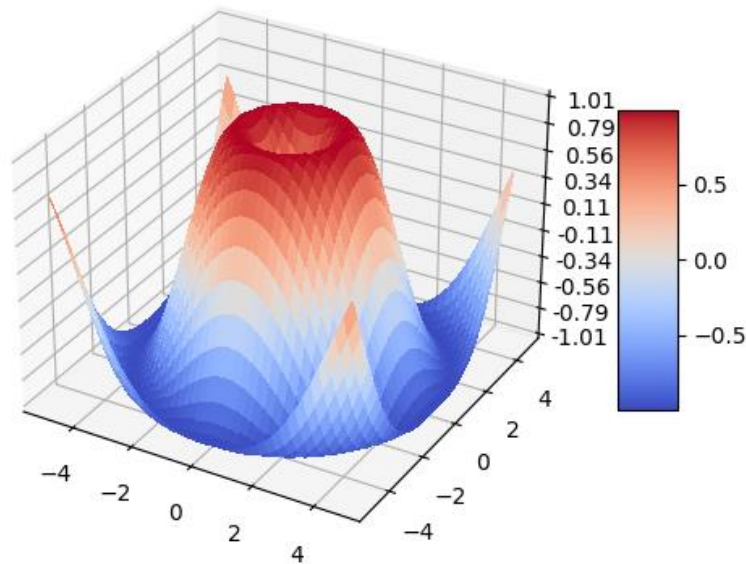
```
import numpy as np

c = np.linspace(0,10,5)
#array cu 5 elemente consecutive din intervalul
[0,10] aflate la distante egale
print(f"c = {c}")
c **= 2
print(f"c = {c}")
d = np.linspace(0,10)
#implicit 50 de elemente
print(f"d = {d}")
```

```
student@arch:~$ python3 numpy.py
c = [ 0.  2.5  5.  7.5 10.]
c = [ 0.   6.25 25.  56.25 100.]
d = [ 0.          0.20408163  0.40816327  0.6122449   0.81632653  1.02040816
 1.2244898   1.42857143  1.63265306  1.83673469  2.04081633  2.24489796
 2.44897959  2.65306122  2.85714286  3.06122449  3.26530612  3.46938776
```

Modulul matplotlib

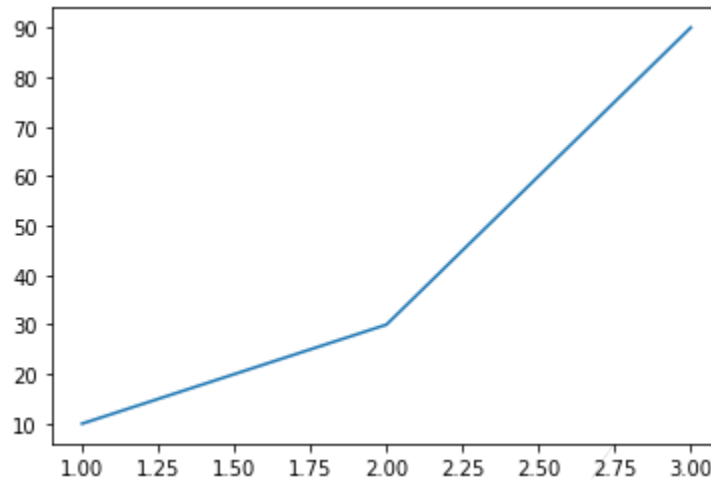
- Este un generator de grafice.
- Suport pentru 2D si 3D.



Grafic simplu

matplot.py

```
import matplotlib.pyplot as plt  
  
x = [1, 2, 3]  
y = [10, 30, 90]  
  
plt.plot(x, y)  
plt.show()
```



Mai multe grafice

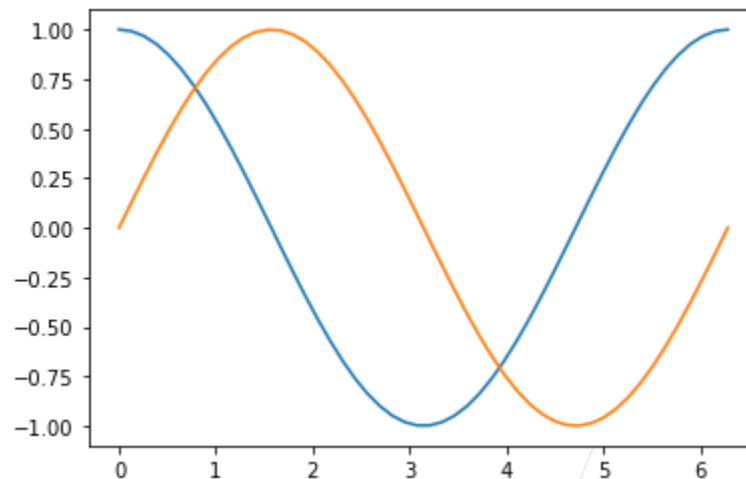
matplot.py

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

```
x = np.linspace(0, 2 * np.pi)  
y1 = np.cos(x)  
y2 = np.sin(x)
```

```
plt.plot(x, y1)  
plt.plot(x, y2)
```

```
plt.show()
```



Legenda

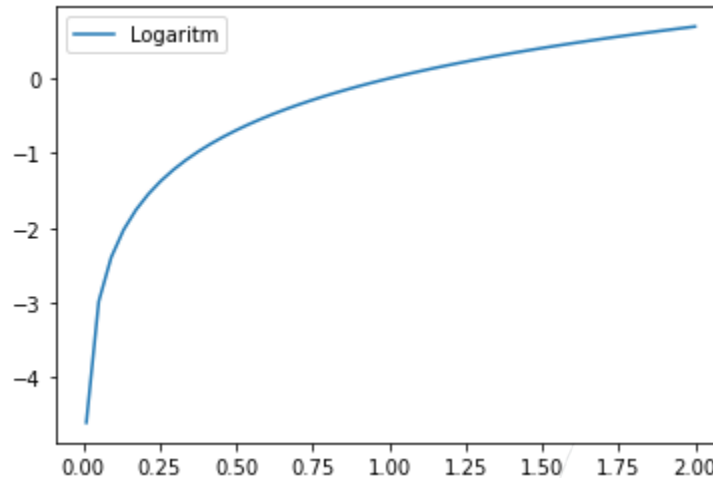
matplot.py

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

x = np.linspace(0.01, 2)
functie = np.log(x)

plt.plot(x, functie, label='Logaritm')
plt.legend()

plt.show()
```



Salvare grafic


matplot.py

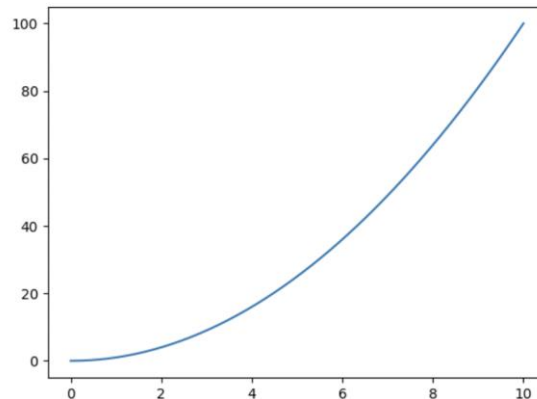
```
import matplotlib.pyplot as plt  
import numpy as np
```

```
x = np.linspace(0, 10)  
y = x ** 2
```

```
plt.plot(x, y)  
plt.savefig("name.png")
```

```
plt.show()
```

 name.png



Fișiere în Python

- În Python, lucrul cu fișiere presupune 3 pași:
 - deschidere fișier
 - operații de citire / scriere fișier
 - închidere fișier

Deschidere fișier

- Un fișier se poate deschide cu funcția **open**.
`f = open("nume_fisier", "mod")`

Mod	Explicație
r	read - se vor citi date din fișier
w	write - se vor scrie date în fișier (dacă există date în fișier, vor fi suprascrise)
a	append - se vor scrie date la finalul fișierului
t	text - fișierul este perceput ca fișier text
b	binar - fișierul este perceput ca fișier binar

Închidere fișier

- Un fișier se închide folosind funcția **close**.

test_file.py

```
f = open("example.txt")  
f.close()
```

example.txt

```
1 The quick brown fox jumps over the lazy dog.  
2 Pack my box with five dozen liquor jugs.  
3
```

Citire linii din fișier

- Putem citi toate liniile din fișier cu funcția **readlines()**, care va întoarce o listă cu liniile din fișier.

test_file.py

```
f = open("file.txt")  
lines = f.readlines()  
print(lines)  
f.close()
```

example.txt

```
1 The quick brown fox jumps over the lazy dog.  
2 Pack my box with five dozen liquor jugs.  
3
```

Citire linii din fișier(2)

- Putem citi o singură linie din fișier cu funcția **readline()**.

test_file.py

```
f = open("example.txt")  
  
line = f.readline().strip("\n")  
print(line)  
  
line = f.readline().strip("\n")  
print(line)  
f.close()
```

```
student@arch:~$ python3 test_file.py  
The quick brown fox jumps over the lazy dog.  
Pack my box with five dozen liquor jugs.
```

Scriere linii în fișier

- Putem scrie liniile în fișier cu funcția **writelines()**, care va scrie o linie în fișier.

test_file.py

```
f = open("example.txt", "w")  
f.writelines("Am scris în fișier")  
f.close()
```

example.txt

```
1 Am scris în fișier
```

With

- Putem citi mai ușor fișiere cu cuvântul cheie `with`, care va închide automat fișierul la ieșirea din blocul de cod asociat.

test_file.py

```
lines = []  
  
with open("example.txt") as file:  
    lines = file.readlines()  
    ... # operation with file  
#file este acum închis  
  
print(lines)
```

```
student@arch:~$ python3 test_file.py  
['The quick brown fox jumps over the lazy dog.\n', 'Pack my box with five dozen liquor jugs.\n']
```


Resurse utile

- sys: <https://docs.python.org/3/library/sys.html>
 - math: <https://docs.python.org/3/library/math.html>
 - random: <https://docs.python.org/3/library/random.html>
 - time: <https://docs.python.org/3/library/time.html>
 - numpy: <https://numpy.org/doc/stable/>
 - matplotlib: <https://matplotlib.org/contents.html>
-
- 40 python libraries explained in 20 minutes:
https://www.youtube.com/watch?v=-29x_deQQus&t=38s

Întrebări?