Facultatea de Inginerie Electrică

UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

EUROPEAN UNIVERSITY OF TECHNOLOGY

Facultatea de Inginerie Electrica

PCLP 2

Programarea calculatoarelor si limbaje de programare 2

PCLP2

An I semestrul II

"Coding is easy when you C it in action."

Cap. 14

Aplicatii in Python

14.1. Introducere in Python

14.2. Aplicatii in Python

14.1. Introducere in Python

Limbajul Python

DEFINIRE

Python: Limbaj de programare de nivel înalt (high level programming language), interpretat (nu compilat).

Interpretor Python: un program care permite rularea/interpretarea programelor scrise in limbajul Python.

Biblioteci Python: funcții, module, tipuri de date disponibile în Python, scrise de alți programatori

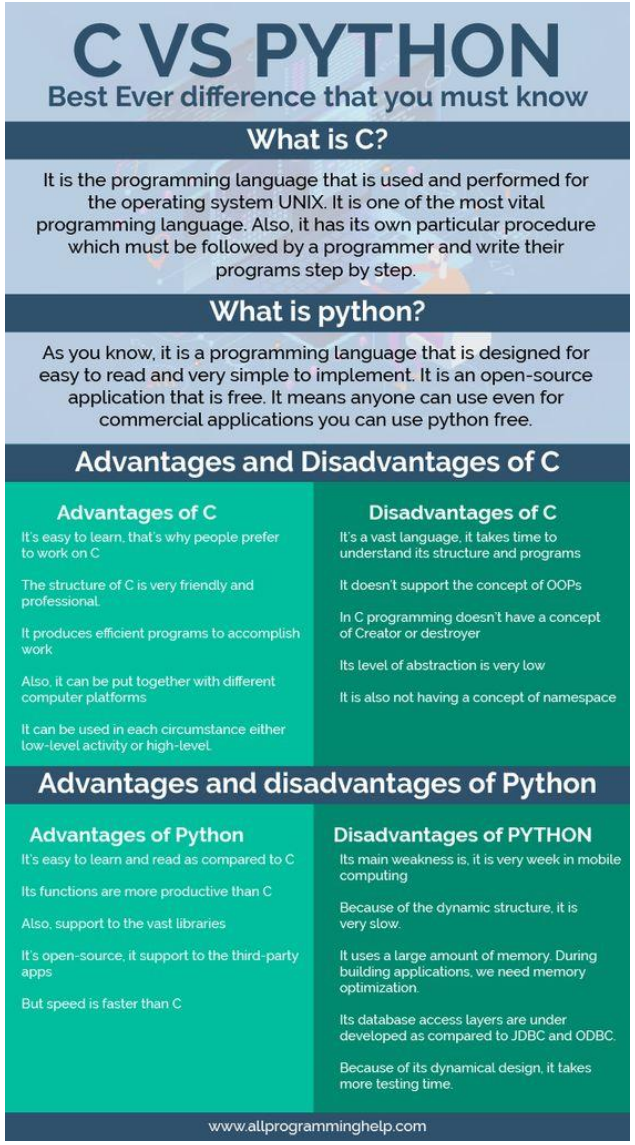
A fost creat de Guido van Rossum și lansat în 1991.

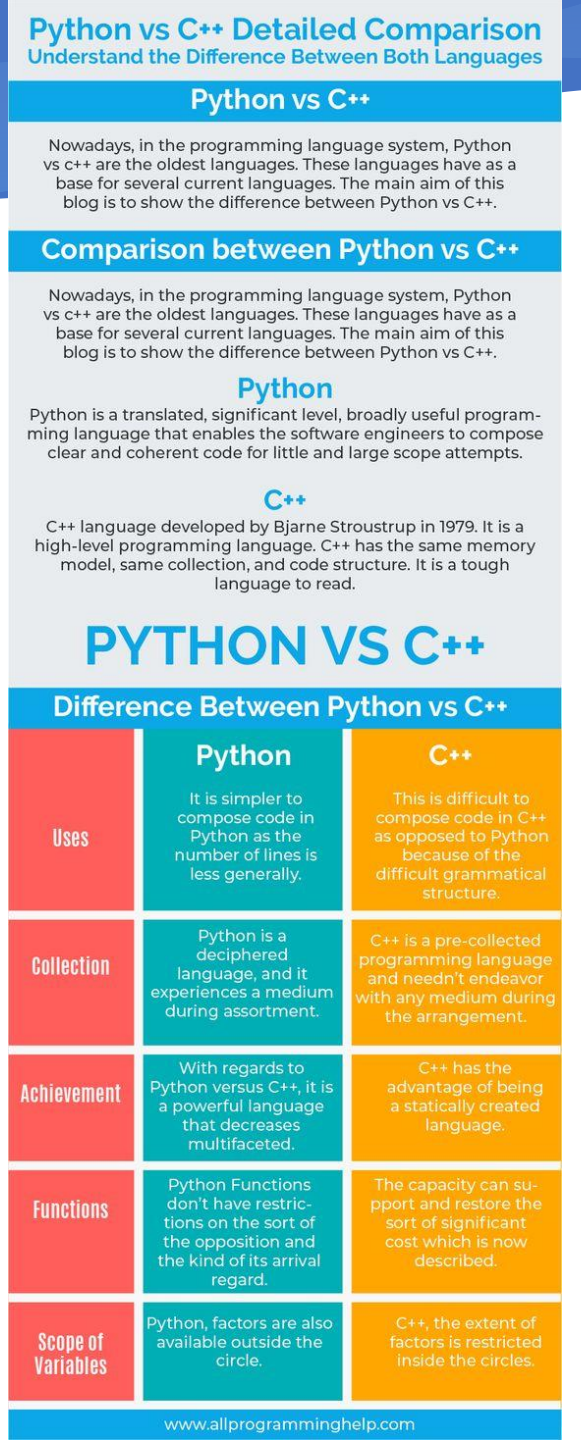
Python	C++
Python is an interpreted language.	C++ is a compiled language.
Python is not portable.	C++ is portable.
Python supports garbage collectors.	C++ does not support garbage collectors.
Slower speed of execution.	Faster speed of execution.
Python does not provides restrictions on the parameters types and return value.	C++ provides restrictions on the parameters types and return value.

14.1. Introducere in Python

Limbajul C/C++ vs Python

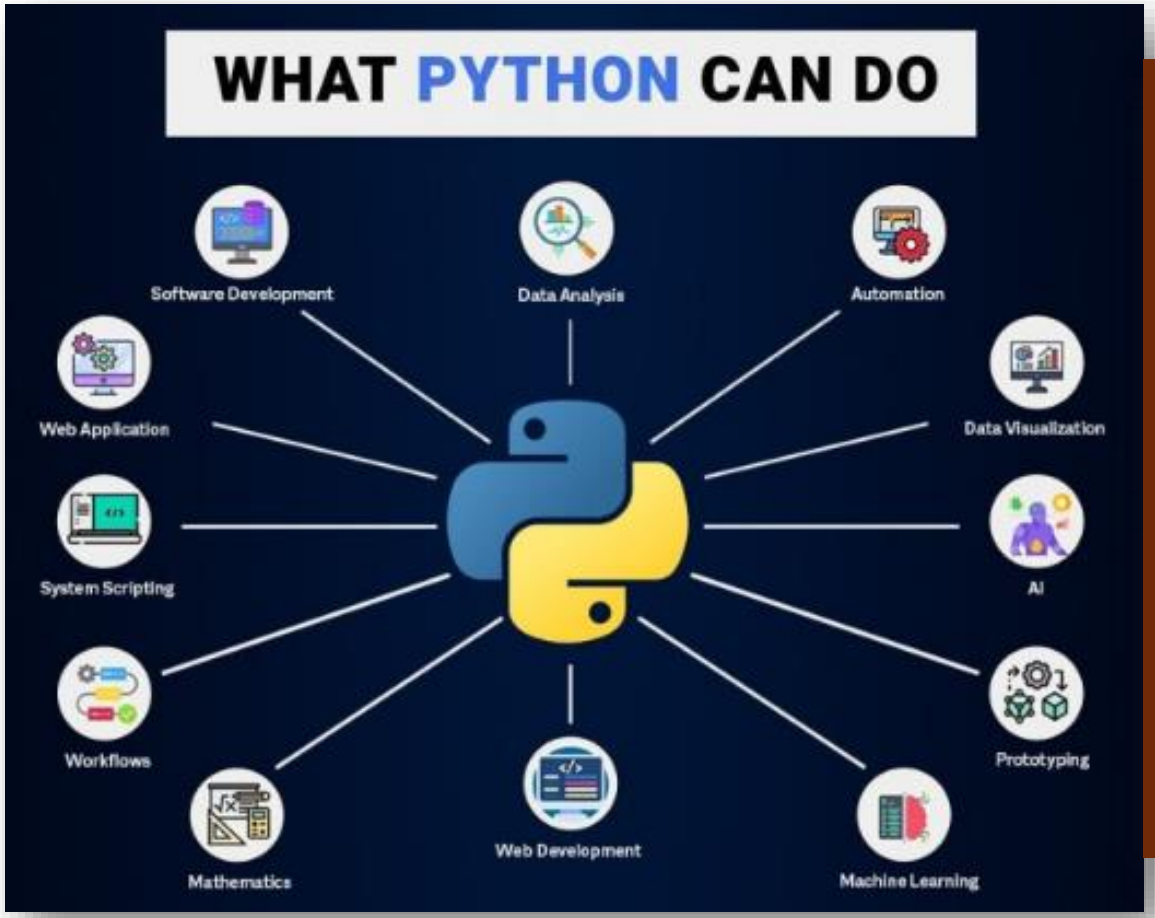
allprogramminghelp.com

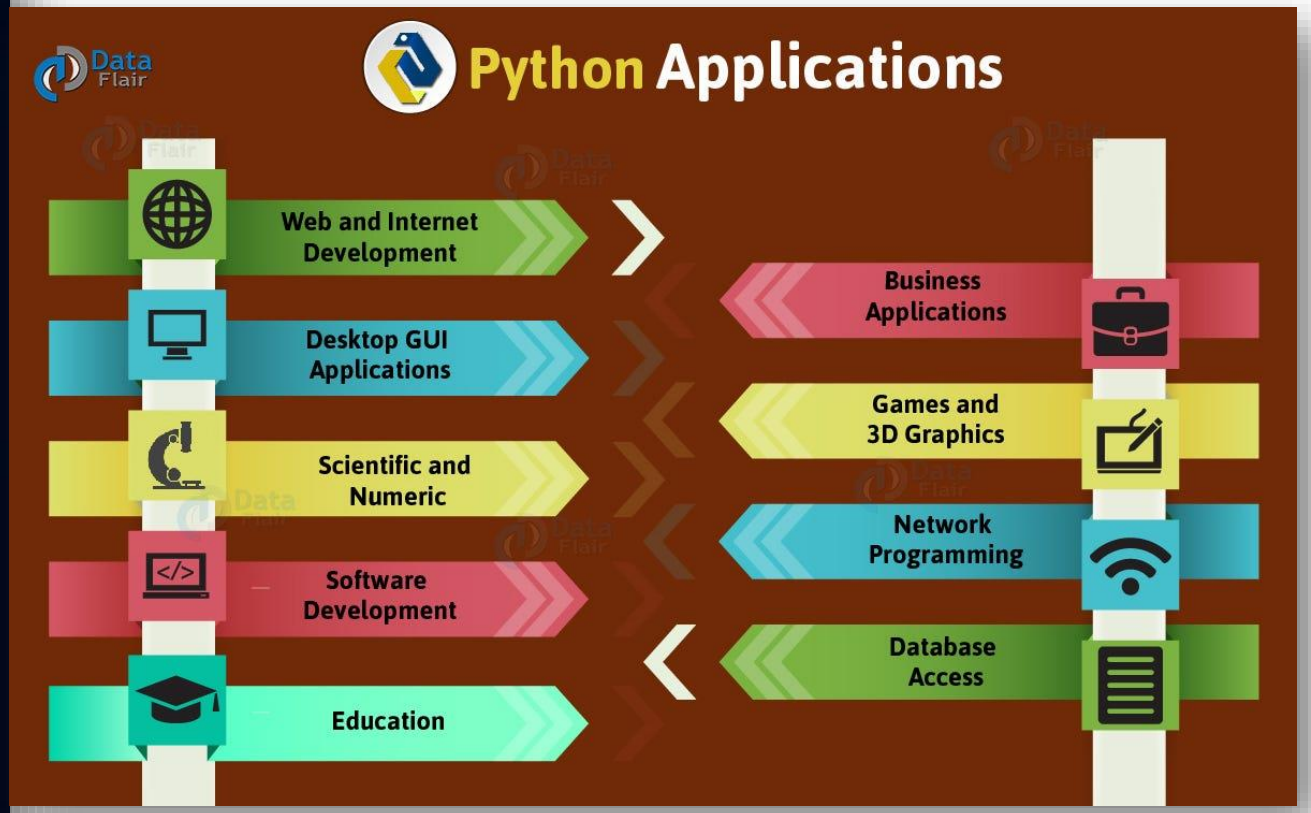




14.1. Introducere in Python

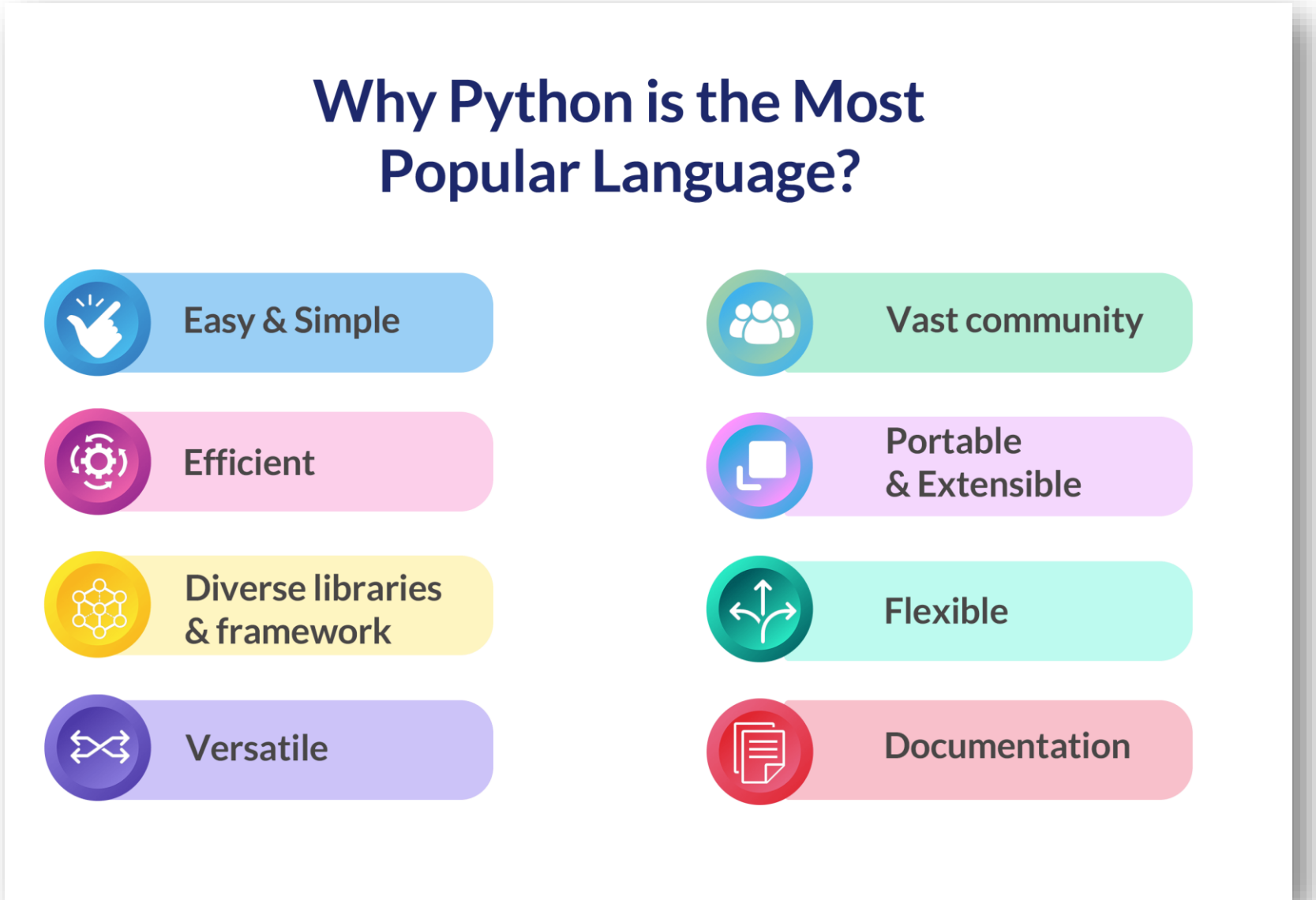
Utilizare, aplicatii





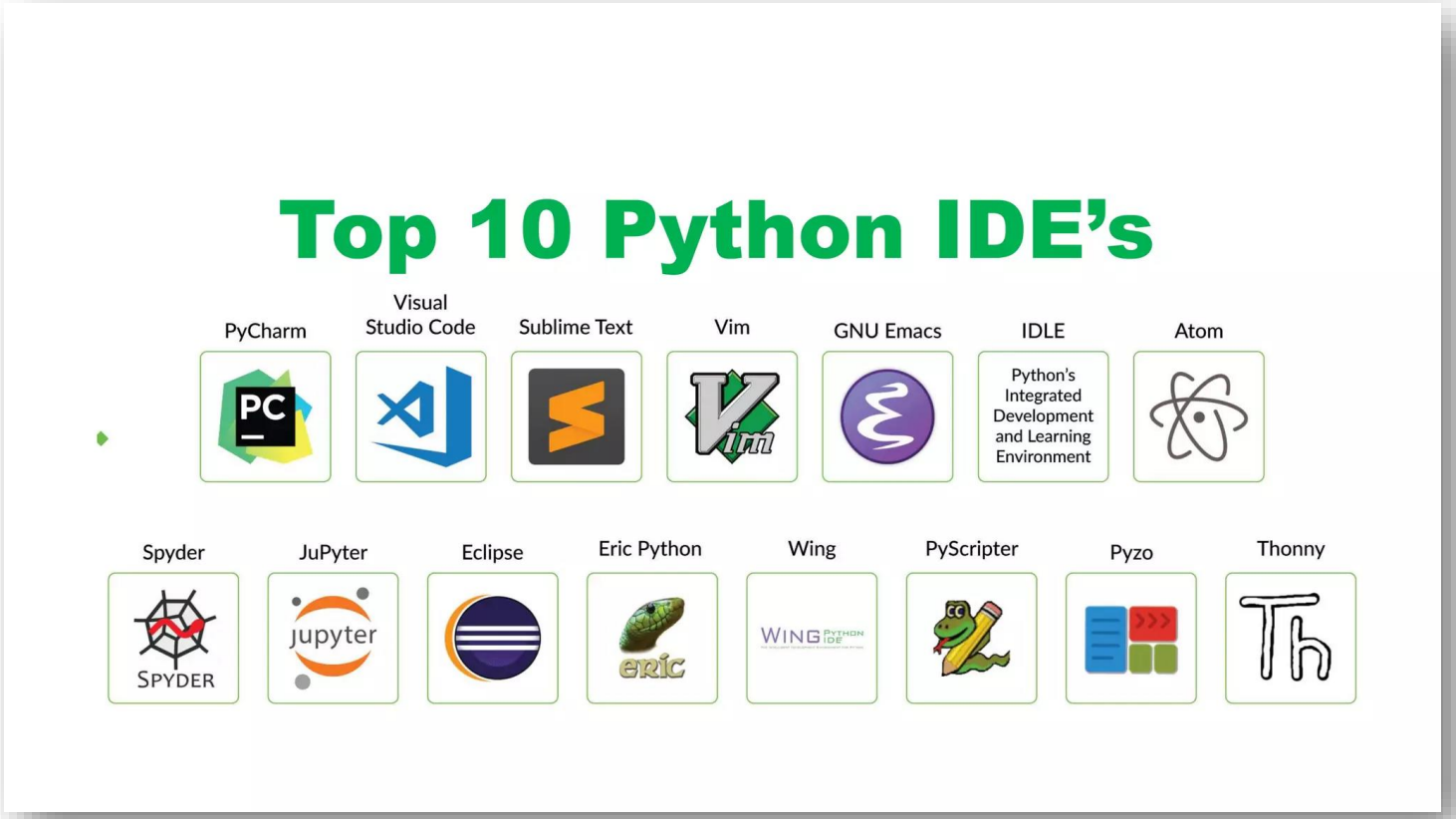
14.1. Introducere in Python

De ce Python?



14.1. Introducere in Python

Cum utilizam Python? Exemple IDE



- Codul Python poate fi executat direct in linia de comanda:

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

- Intr-un fisier cu extensia .py

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

14.1. Introducere in Python

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

Python Cheat Sheet: Keywords

"A puzzle a day to learn, code, and play" → Visit finxter.com

Keyword	Description	Code example
`False, True`	Data values from the data type Boolean	`False == (1 > 2), True == (2 > 1)`

`and, or, not`	Logical operators:
	`(x and y)` → both x and y must be True
	`(x or y)` → either x or y must be True
	`(not x)` → x must be false
		`x, y = True, False`
		`(x or y) == True  # True`
		`(x and y) == False  # True`
		`(not y) == True  # True`

`break`	Ends loop prematurely	`while(True):`
		`  break # no infinite loop`
		`print("hello world")`

`continue`	Finishes current loop iteration	`while(True):`
		`  continue`
		`  print("43") # dead code`

`class`
 	
`def`
	Defines a new class → a real-world concept (object oriented programming)
	
	Defines a new function or class method. For latter, first parameter ("self") points to the class object. When calling class method, first parameter is implicit.
		`class Beer:`
		`  def __init__(self):`
		`  self.content = 1.0`
		`  def drink(self):`
		`  self.content = 0.0`
		
		`becks = Beer() # constructor - create class`
		`becks.drink() # beer empty: b.content == 0`

`if, elif, else`	Conditional program execution: program starts with "if" branch, tries the "elif" branches, and finishes with "else" branch (until one branch evaluates to True).	`x = int(input("your value: "))`
		`if x > 3: print("Big")`
		`elif x == 3: print("Medium")`
		`else: print("Small")`

`for, while`	`# For loop declaration`
	`for i in [0,1,2]:`
	`  print(i)`
		`# While loop - same semantics`
		`j = 0`
		`while j < 3:`
		`  print(j)`
		`  j = j + 1`

`in`	Checks whether element is in sequence	`42 in [2, 39, 42] # True`

`is`	Checks whether both elements point to the same object	`y = x = 3`
		`x is y # True`
		`[3] is [3] # False`

`None`	Empty value constant	`def f():`
		`  x = 2`
		`f() is None # True`

`lambda`	Function with no name (anonymous function)	`(lambda x: x + 3)(3) # returns 6`

`return`	Terminates execution of the function and passes the flow of execution to the caller. An optional value after the return keyword specifies the function result.	`def incrementor(x):`
		`  return x + 1`
		`incrementor(4) # returns 5`

finxter

Python Cheat Sheet: Basic Data Types

"A puzzle a day to learn, code, and play" → Visit finxter.com

	Description	Example

Boolean	The Boolean data type is a truth value, either `True` or `False`.
	
	The Boolean operators ordered by priority:
	`not x` → "if x is False, then x, else y"
	`x and y` → "if x is False, then x, else y"
	`x or y` → "if x is False, then y, else x"
	
	These comparison operators evaluate to `True`:
	`1 < 2 and 0 <= 1 and 3 > 2 and 2 >=2 and 1 == 1 and 1 != 0 # True`
		`## 1. Boolean Operations`
		`x, y = True, False`
		`print(x and not y) # True`
		`print(not x and y or x) # True`
		
		`## 2. If condition evaluates to False`
		`if None or 0 or 0.0 or '' or [] or {} or set():`
		`  # None, 0, 0.0, empty strings, or empty`
		`  # container types are evaluated to False`
		`  print("Dead code") # Not reached`

Integer, Float	An integer is a positive or negative number without floating point (e.g. `3`). A float is a positive or negative number with floating point precision (e.g. `3.14159265359`).
	
	The '`//`' operator performs integer division. The result is an integer value that is rounded toward the smaller integer number (e.g. `3 // 2 == 1`).
		`## 3. Arithmetic Operations`
		`x, y = 3, 2`
		`print(x + y) # = 5`
		`print(x - y) # = 1`
		`print(x * y) # = 6`
		`print(x / y) # = 1.5`
		`print(x // y) # = 1`
		`print(x % y) # = 1s`
		`print(-x) # = -3`
		`print(abs(-x)) # = 3`
		`print(int(3.9)) # = 3`
		`print(float(3)) # = 3.0`
		`print(x ** y) # = 9`

String	Python Strings are sequences of characters.
	
	The four main ways to create strings are the following.
	
	1. Single quotes
	`'Yes'`
	2. Double quotes
	`"Yes"`
	3. Triple quotes (multi-line)
	`"""Yes`
	`We Can"""`
	4. String method
	`str(5) == '5' # True`
	5. Concatenation
	`"Ma" + "hatma" # 'Mahatma'`
	
	These are whitespace characters in strings.
	• Newline `\n`
	• Space `\s`
	• Tab `\t`
		`## 4. Indexing and Slicing`
		`s = "The youngest pope was 11 years old"`
		`print(s[0])  # 'T'`
		`print(s[1:3])  # 'he'`
		`print(s[-3:-1])  # 'ol'`
		`print(s[-3:])  # 'old'`
		`x = s.split()  # creates string array of words`
		`print(x[-3] + " " + x[-1] + " " + x[2] + "s")`
		`  # '11 old popes'`
		
		`## 5. Most Important String Methods`
		`y = "  This is lazy\t\n  "`
		`print(y.strip()) # Remove Whitespace: 'This is lazy'`
		`print("DrDre".lower()) # Lowercase: 'drdre'`
		`print("attention".upper()) # Uppercase: 'ATTENTION'`
		`print("smartphone".startswith("smart")) # True`
		`print("smartphone".endswith("phone")) # True`
		`print("another".find("other")) # Match index: 2`
		`print("cheat".replace("ch", "m")) # 'meat'`
		`print(','.join(["F", "B", "I"])) # 'F,B,I'`
		`print(len("Rumpelstiltskin")) # String length: 15`
		`print("ear" in "earth") # Contains: True`

finxter

14.1. Introducere in Python

Instalare/utilizare Python

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

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

Online Python Interpreters= cloud-based Python interpreters :

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

14.1. Introducere in Python

Biblioteci Python

Top 10 Python Libraries

Pandas
Data analysis and manipulation

NumPy
Mathematical functions

Matplotlib
Data visualisations

SeaBorn
Data visualisations

Tensorflow
Machine Learning

Keras
Deep Learning

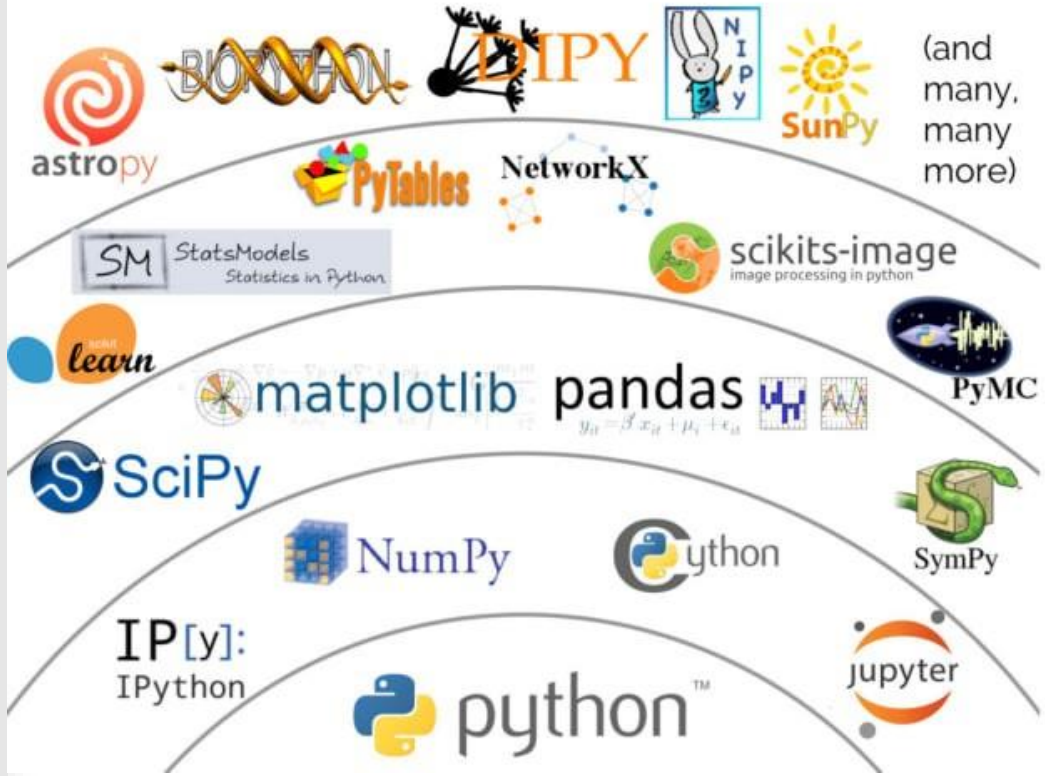
SciPy
Scientific computing

PyTorch
Machine Learning

Scrapy
Web crawling

SQLModel
Interact with SQL databases

DATA RUNDOWN



14.1. Introducere in Python

Tutoriale Python

- **Using Python for Data Analysis: https://realpython.com/python-for-data-analysis/**
- **Create a Tic-Tac-Toe Python Game Engine With an AI Player: https://realpython.com/courses/python-tic-tac-toe-ai/**
- **Python Basics Exercises: Reading and Writing Files:** https://realpython.com/courses/python-exercises-reading-writing-files/
- **Build a Hangman Game With Python and PySimpleGUI: https://realpython.com/hangman-python-pysimplegui/**

14.1. Introducere in Python

Sintaxa

- **Program Python: linii de cod** (nu se termina cu ;), nu se folosesc accolade la if/while/for => indentare
- **Comentarii** : cu # pe o linie , sau cu “ ” pe mai multe rânduri
- **Literali**: notații pentru valorile constante sau pentru tipuri definite de utilizator
- **Variabile:** sunt case-sensitive.

EXEMPLE

```
#cu indentare
if 5 > 2:
  print("5 e mai mare decat 2")
```

```
PS D:\laura\py> & d:/laura/py/.venv/Scripts/python.exe d:/laura/py/test1.py
5 e mai mare decat 2
```

```
#fara indentare
if 5 > 2:
print("5 e mai mare decat 2")
```

```
PS D:\laura\py> & d:/laura/py/.venv/Scripts/python.exe d:/laura/py/test1.py
  File "d:\laura\py\test1.py", line 3
    print("5 e mai mare decat 2")
    ^
IndentationError: expected an indented block after 'if' statement on line 2
```

14.1. Introducere in Python

Modelul de date

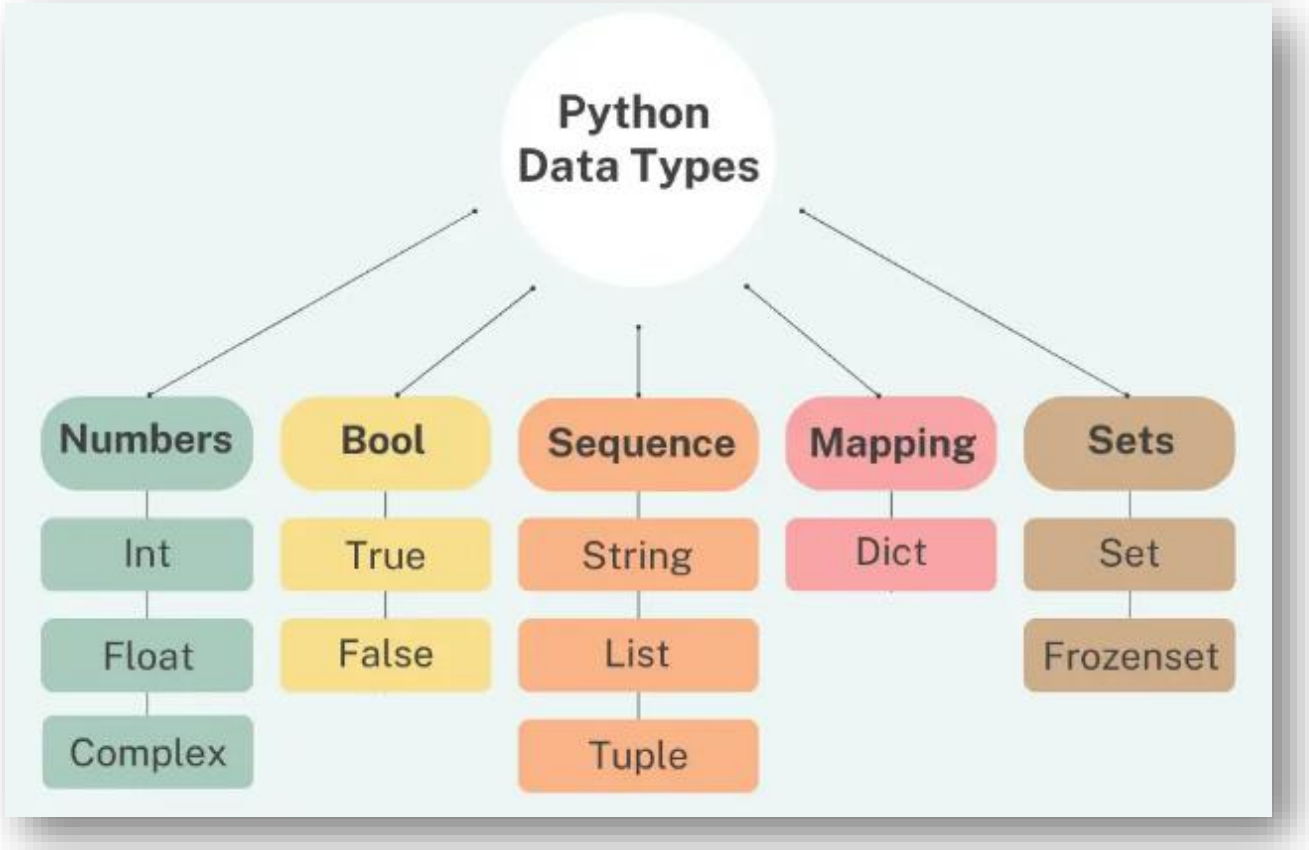
- **Toate datele într-un program Python = obiecte**
- **Obiect** are :
  - **o identitate** = adresa de memorie
  - **un tip =** care determină operațiile posibile precum și valorile pe care le poate lua obiectul \
  - **o valoare**.

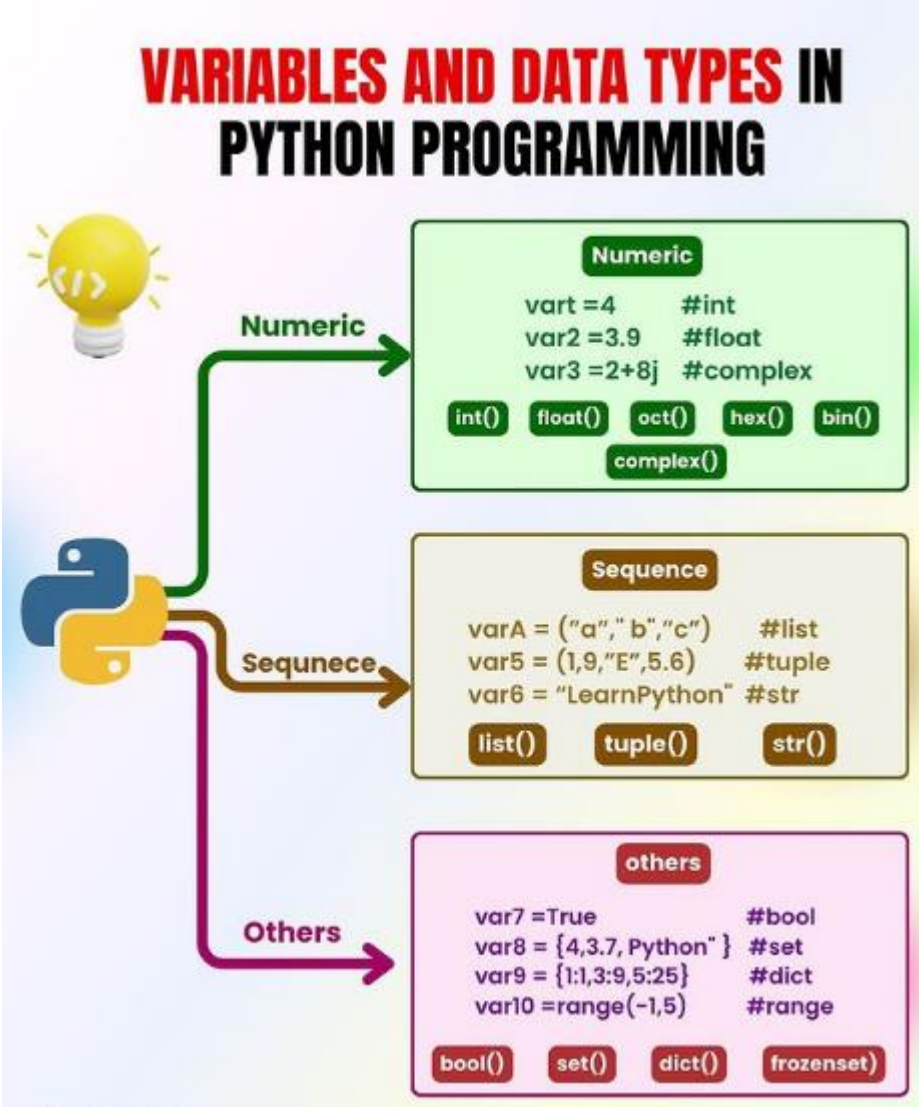
Dupa creare obiect : **identitatea și tipul obiectului nu mai pot fi modificate**, **valoarea** unor obiecte **se poate modifica** :

- **Obiecte mutabile** - se poate modifica
- **Obiecte nemutabile/inmutabile** – nu se poate modifica

14.1. Introducere in Python

Tipuri de date





14.1. Introducere in Python

Tipuri de date standard

1. Numerice : numerele sunt nemutabile = odată create valoarea nu se mai poate schimba (operațiile crează noi obiecte).

- **int** (numere întregi):
  - **Valori** : numere întregi (pozitive și negative)
  - **Operații**: +, -, *, /, //, **, % comparare:==,!=, operații pe biți: |, ^, &, <>,
  - **Literali**: 1, -3
- **float** (numere reale):
  - **Valori**: numerele reale (dublă precizie)
  - **Operatii**: +, -, *, / comparare:==,!=,
  - **Literali**: 3.14
- **complex**
  - **Valori**: numerele complexe
  - **Operatii**: specific nr complexe
  - **Literali**: 3j

2. Logice bool (boolean):

- **Valori:** True și False.
- **Operații**: and, or, not
- **Literali**: False, True; 0, 1

EXEMPLE

```
x = 1
y = 2.8
z = 1j
a = True
print(type(x))
print(type(y))
print(type(z))
print(type(a))
```

```
<class 'int'>
<class 'float'>
<class 'complex'>
<class 'bool'>
```

14.1. Introducere in Python

Tipuri de date standard

3. Secvente (similar cu tablouri in C): Mulțimi finite și ordonate, indexate . Ex. a[0], a[1], …,

- Nr de elemente: len(a);
- Elemente: a[0], a[1], …, a[len(a)-1]
- Ex: [1, ‘a’]

4. String: o secvență nemutabilă;

- Literali: ‘abc’, “abc”

5.Liste: secvență mutabilă

- Operatii:
  - creare,accesare valori,lungime (index, len),
  - modificare valori (listele sunt mutabile)
  - verificare daca un element este in lista
  - stergere
  - inserare valori (append,insert,pop)
  - slicing

```
# create
a = [1, 2, 'a']
print (a)
x, y, z = a
print(x, y, z)

# indices: 0, 1, ..., len(a) - 1
print a[0]
print ('last element = ', a[len(a)-1])

# lists are mutable
a[1] = 3
print a
```

```
# slicing
print a[:2]
b = a[:]
print (b)
b[1] = 5
print (b)
a[3:] = [7, 9]
print(a)
a[:0] = [-1]
print(a)
a[0:2] = [-10, 10]
print(a)
```

```
# lists as stacks
stack = [1, 2, 3]
stack.append(4)
print stack
print stack.pop()
print stack
```

```
# nesting
c = [1, b, 9]
print (c)
```

```
#generate lists using range
l1 = range(10)
print l1
l2 = range(0,10)
print l2
l3 = range(0,10,2)
print l3
l4 = range(9,0,-1)
print l4
```

```
#list in a for loop
l = range(0,10)
for i in l:
    print i
```

14.1. Introducere in Python

Tipuri de date standard

6.Tuple: secvențe nemutabile.

- Operatii:
  - Crearea - packing (23, 32, 3)
  - eterogen
  - poate fi folosit in for
  - unpacking

```
# Tuples are immutable sequences
# A tuple consists of a number of
values separated by commas

# tuple packing
t = 12, 21, 'ab'
print(t[0])

# empty tuple (0 items)
empty = ()
```

```
# tuple with one item
singleton = (12,)
print (singleton)
print (len(singleton))

#tuple in a for
t = 1,2,3
for el in t:
    print el
```

```
# sequence unpacking
x, y, z = t
print (x, y, z)
```

```
# Tuples may be nested
u = t, (23, 32)
print(u)
```

14.1. Introducere in Python

Tipuri de date standard

7.Dictionar: o multime de perechi (cheie, valoare). Cheile trebuie sa fie nemutabile.

Operatii:

- creare {} sau {'num': 1, 'denom': 2}
- accesare valoare pe baza unei chei
- adaugare/modificare pereche (cheie, valoare)
- ștergere pereche (cheie, valoare)
- verificare dacă cheia există

```
#create a dictionary
a = {'num': 1, 'denom': 2}
print(a)

#get a value for a key
print(a['num'])
```

```
#set a value for a key
a['num'] = 3
print(a)
print(a['num'])
```

```
#delete a key value pair
del a['num']
print (a)
```

```
#check for a key
if 'denom' in a:
    print('denom = ', a['denom'])
if 'num' in a:
    print('num = ', a['num'])
```

14.1. Introducere in Python

Instructiuni- atribuire/ atribuire multipla

EXEMPLE

```
x = y = z = "osciloscop"
print(x)
print(y)
print(z)
```

```
osciloscop
osciloscop
osciloscop
```

```
x, y, z = “osiloscop", “multimetru", “ampermetru"
print(x)
print(y)
print(z)
```

```
osiloscop
multimetru
ampermetru
```

Instructiuni- if, elif,else

EXEMPLE

```
x = int(input( "your value: " ))
if x > 3 : print( "Big" )
elif x == 3 : print( "Medium" )
else : print( "Small" )
```

```
your value: 3
Medium
```

if,elif, else au in sintaxa :

14.1. Introducere in Python

Instructiuni- while

EXEMPLE

```
j = 0
while j < 5:
 print(j)
 j = j + 1
```

```
0
1
2
3
4
```

EXEMPLE

```
list1 = ["C", "C++","Java", "Python", "Javascript"]
i = 0
print("Printing list items using while loop")
size = len(list1)
while(i < size):
    print(list1[i])
    i = i+1
```

```
Printing list items using while loop
C
C++
Java
Python
Javascript
```

while are in sintaxa :

14.1. Introducere in Python

Instructiuni- for

EXEMPLE

```
for i in [0,1,2,5,10]:
 print(i)
```

```
0
1
2
5
10
```

Functii- lambda

EXEMPLE

```
#lambda=anonymous function
print((lambda x: x + 3)(4))
```

```
7
```

Functii- def

EXEMPLE

```
def f(fname):
  print(fname + " UTCN")
f("FIE")
f("ETTI")
f("AC")
```

```
FIE UTCN
ETTI UTCN
AC UTCN
```

14.1. Introducere in Python

Instructiuni- print variabile multiple

EXEMPLE

```
x = "Python"
y = "este"
z = "cool"
print(x, y, z)
```

```
Python este cool
```

Instructiuni- print + (concatenare variabile)

EXEMPLE

```
x = "Python"
y = "este"
z = "cool"
print(x + y + z)
```

```
Pythonestecool
```

14.1. Introducere in Python

Instructiuni- print Multiline Strings

EXEMPLE

```
a = """Limbajul Python este interpretat,
adică se execută codul linie cu line,
spre deosebire de Pascal ori C/C++,
unde este necesar un compilator care
să genereze un fișier executabil"""
print(a)
```

sau

```
a = '''Limbajul Python este interpretat,
adică se execută codul linie cu line,
spre deosebire de Pascal ori C/C++,
unde este necesar un compilator care
să genereze un fișier executabil'''
print(a)
```

- Limbajul Python este interpretat,
  adică se execută codul linie cu line,
  spre deosebire de Pascal ori C/C++,
  unde este necesar un compilator care
  să genereze un fișier executabil

14.1. Introducere in Python

Siruri : operatii slicing

EXEMPLE

```
# selectare caractere in interval
a = 'Limbajul Python'
print(a[9:16])
```

- Python

```
# selectare de la inceputul sirului pana la pozitia indicata
a = 'Limbajul Python'
print(a[:6])
```

- Limbaj

```
# selectare de la sfarsitul sirului pana la pozitia indicata
a = 'Limbajul Python'
print(a[6:])
```

- ul Python

```
# selectare de la sfarsitul sirului cu indici negativi
a = 'Limbajul Python'
print(a[-5:-1])
```

- ytho

https://www.w3schools.com/python/python_ref_string.asp

14.1. Introducere in Python

Siruri : alte operatii

EXEMPLE

```
# modificare sir majuscule/litere mici
a = 'Limbajul Python'
print(a.upper())
print(a.lower())
```

```
LIMBAJUL PYTHON
limbajul python
```

```
# modificare sir
a = 'Limbajul Python'
print(a.replace("L", "l"))
```

```
limbajul Python
```

```
# scindare sir
a = 'Limbajul , Python'
print(a.split(","))
```

```
['Limbajul ', ' Python']
```

```
# concatenare sir
a = 'Limbajul'
b = ' Python'
c = a + b
print(c)
```

```
Limbajul Python
```

14.1. Introducere in Python

Siruri : alte operatii

EXEMPLE

```
# concatenare sir
a = 'Limbajul'
b = ' Python'
c = a + b
d = b + ' este '+ a +' nr.1'
print(c,d)
```

```
Limbajul Python  Python este Limbajul nr.1
```

```
# inserare valori (sir, integer) in string
nume = 'Ana'
age = 20
addr ='Cluj Napoca'
txt = "Numele meu este {}, sunt din {} si am {} ani"
print(txt.format(nume,addr,age,))
```

```
Numele meu este Ana, sunt din Cluj Napoca si am 20 ani
```

14.1. Introducere in Python

Liste : operatii

EXEMPLE

```
# creare lista -sunt admise elemente duplicate
lista = ["osciloscop", "ampermetru", "multimetru", "osciloscop"]
print(lista)
print(len(lista))
```

```
['osciloscop', 'ampermetru', 'multimetru', 'osciloscop']
4
```

```
# creare lista -sunt admise elemente de tipuri diferite
lista = ["osciloscop", 2024, 20.5, "Cluj"]
print(lista)
```

```
['osciloscop', 2024, 20.5, 'Cluj']
```

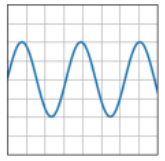
```
# creare lista - constructor -atentie paranteze duble
listanoua = list(("osciloscop", 2024, 20.5, "Cluj"))
print(listanoua)
```

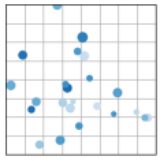
```
['osciloscop', 2024, 20.5, 'Cluj']
```

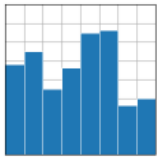
14.2. Grafice in Python

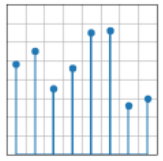
Tipuri de grafice : date perechi

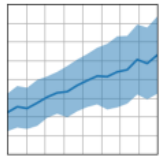
Plots of pairwise (x, y), tabular $(var\_0, \cdots, var\_n)$, and functional $f(x) = y$ data.

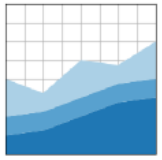

plot(x, y)

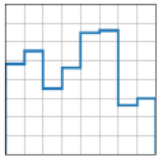

scatter(x, y)


bar(x, height)


stem(x, y)

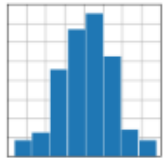

fill_between(x, y1, y2)

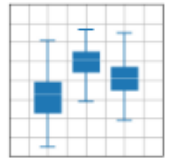

stackplot(x, y)

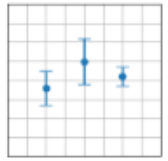

stairs(values)

Tipuri de grafice : distributie statistica

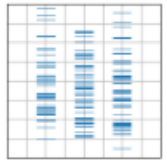
Plots of the distribution of at least one variable in a dataset. Some of these methods also compute the distributions.

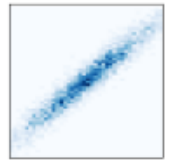

hist(x)

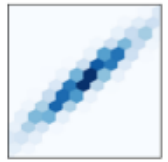

boxplot(X)


errorbar(x, y, yerr, xerr)

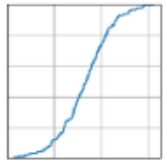

violinplot(D)


eventplot(D)


hist2d(x, y)


hexbin(x, y, C)


pie(x)


ecdf(x)

14.2. Grafice in Python

Tipuri de grafice : date gridded

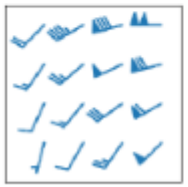
Plots of arrays and images $Z_{i,j}$ and fields $U_{i,j}, V_{i,j}$ on regular grids and corresponding coordinate grids $X_{i,j}, Y_{i,j}$.

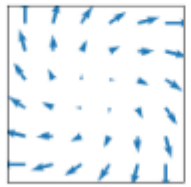

imshow(Z)

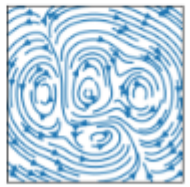

pcolormesh(X, Y, Z)


contour(X, Y, Z)


contourf(X, Y, Z)


barbs(X, Y, U, V)

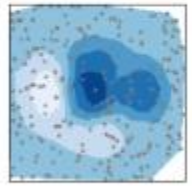

quiver(X, Y, U, V)

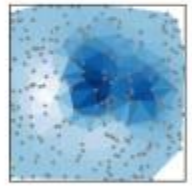

streamplot(X, Y, U, V)

Tipuri de grafice : irregularly gridded

Plots of data $Z_{x,y}$ on unstructured grids , unstructured coordinate grids (x, y), and 2D functions $f(x, y) = z$.


tricontour(x, y, z)


tricontourf(x, y, z)

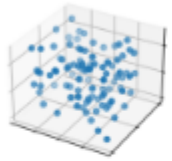

tripcolor(x, y, z)


triplot(x, y)

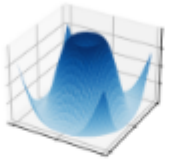
14.2. Grafice in Python

Tipuri de grafice : 3D & volumetric data

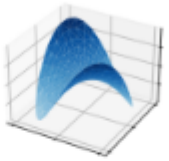
Plots of three-dimensional (x, y, z), surface $f(x, y) = z$, and volumetric $V_{x,y,z}$ data using the `mpl_toolkits.mplot3d` library.



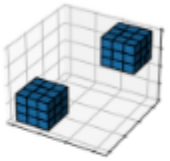
scatter(xs, ys, zs)



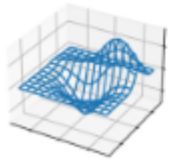
plot_surface(X, Y, Z)



plot_trisurf(x, y, z)



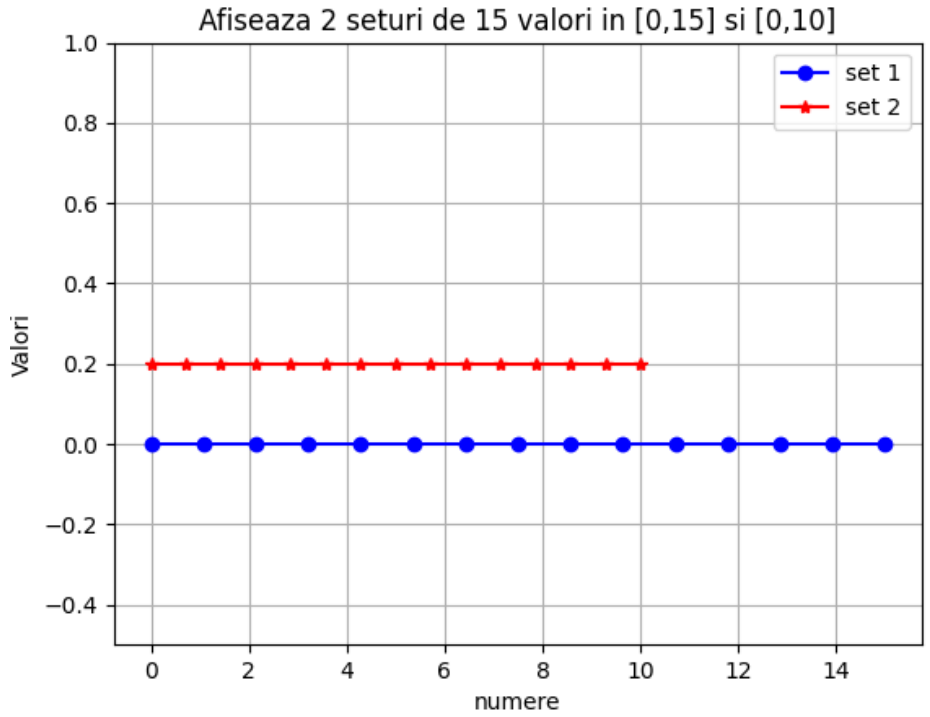
voxels([x, y, z], filled)



plot_wireframe(X, Y, Z)

14.2. Grafice in Python

Exemple: grafic 2 seturi de date



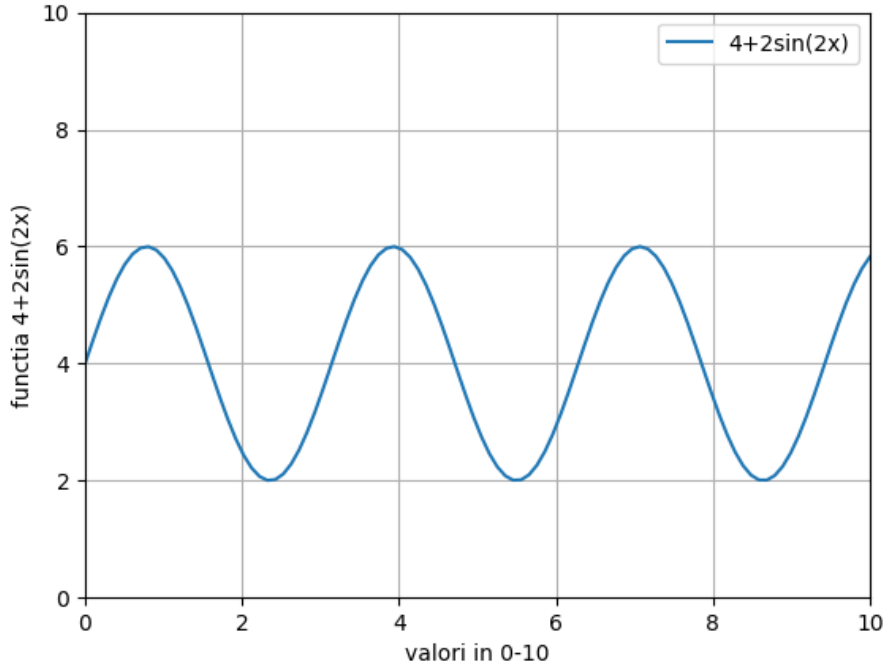
Cod Python

```
#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 axa y in interval [-0.5,1]
plt.ylim([-0.5, 1])
# afiseaza graficul
plt.show()
```

14.2. Grafice in Python

Exemple: grafic functie

*Ex: Codul Python afiseaza graficul unei functii trigonometrice : 4+2*sin(2*x).*



Cod Python

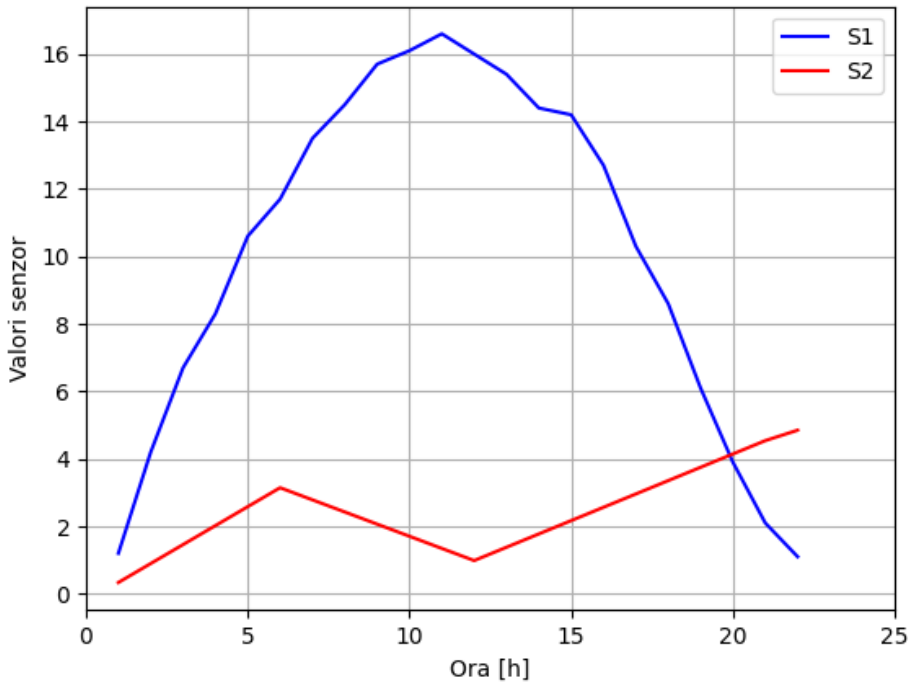
```
import matplotlib.pyplot as plt
import numpy as np

# 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()
```

14.2. Grafice in Python

Exemple: grafic date 2D

Ex: 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



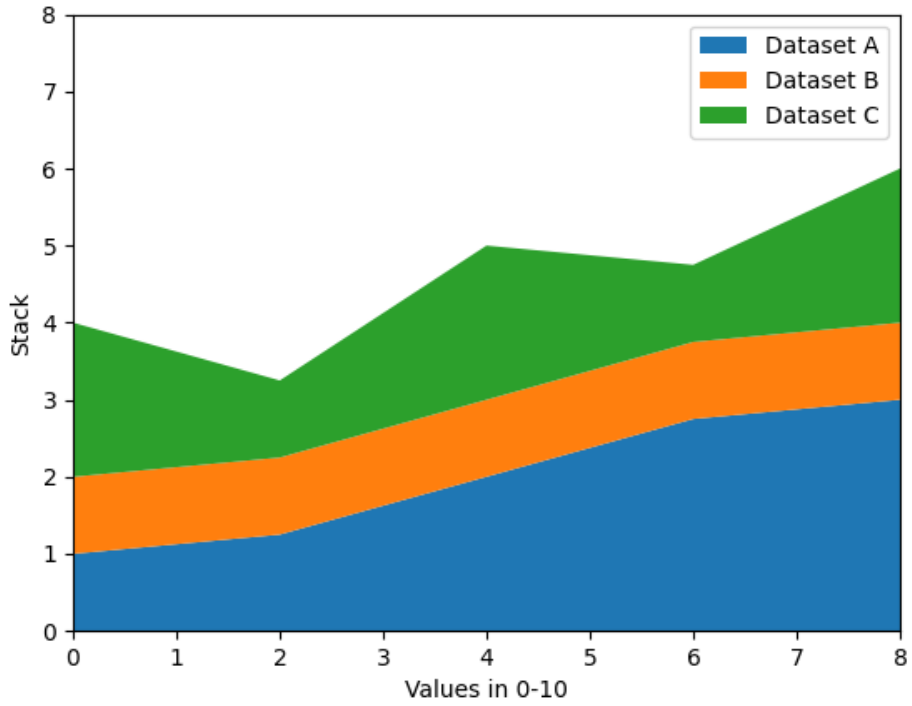
Cod Python

```
#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()
```

14.2. Grafice in Python

Exemple: grafic date stack

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



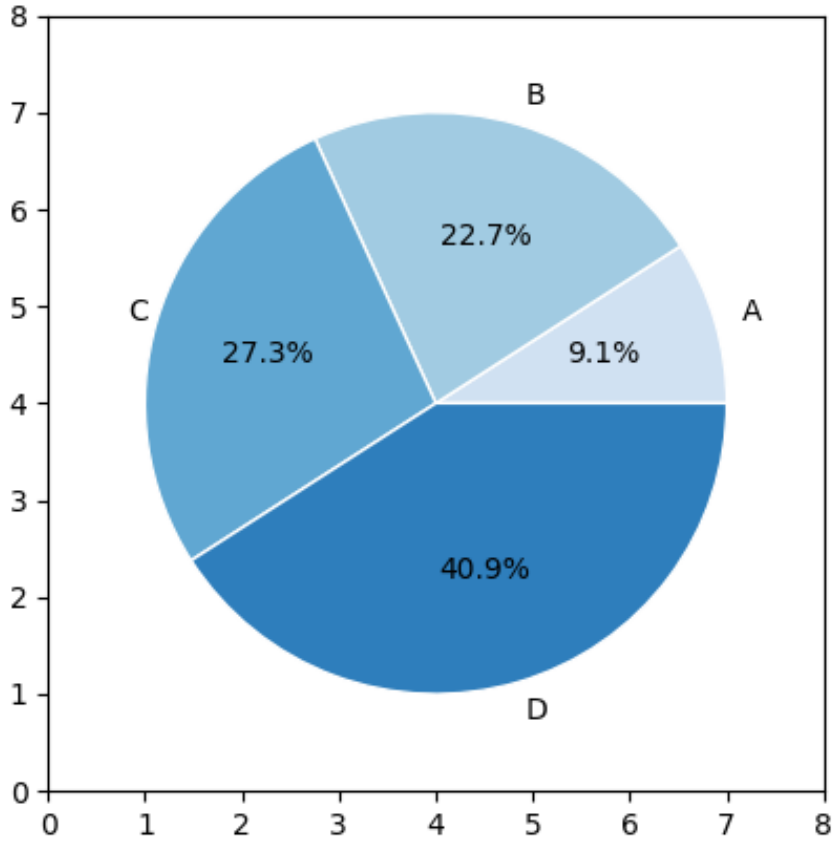
Cod Python

```
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()
```

14.2. Grafice in Python

Exemple: grafic pie

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



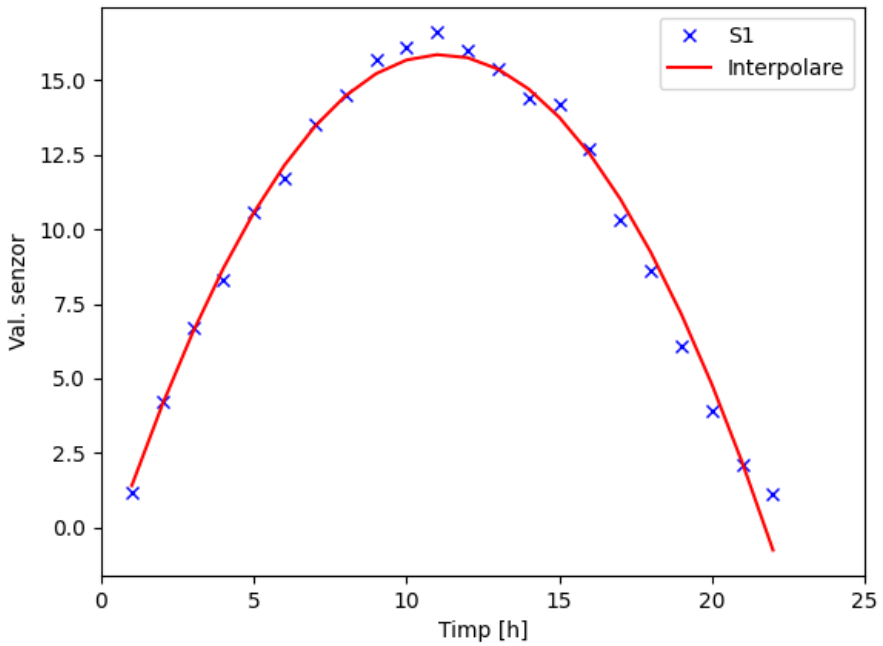
Cod Python

```
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()
```

14.2. Grafice in Python

Exemple: grafic functie interpolare

Ex: 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

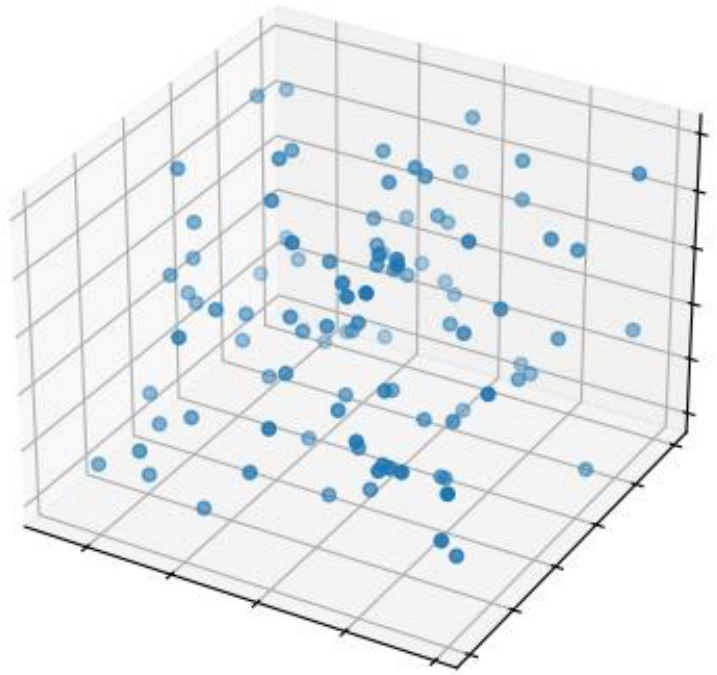
```
#import librarie op. matematice
import numpy as np
#import librarie plotare grafice
import matplotlib.pyplot as plt
#import librarie citire fisiere externe (csv, xlsx)
import pandas as pd
#import librarie fitare date experimentale
from scipy.optimize import curve_fit
#crearea unei structuri de date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

14.2. Grafice in Python

Exemple: grafic 3D scattered

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



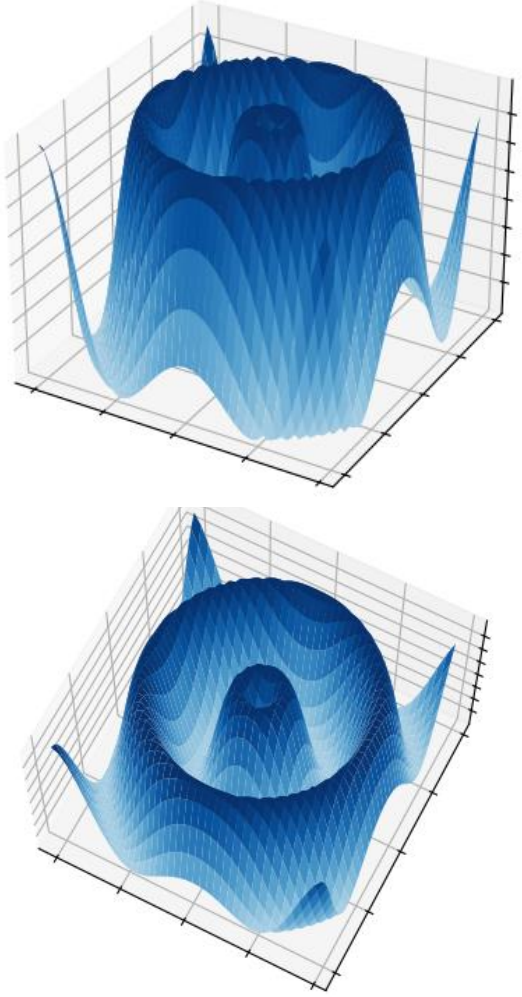
Cod Python

```
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()
```

14.2. Grafice in Python

Exemple: grafic 3D surface

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



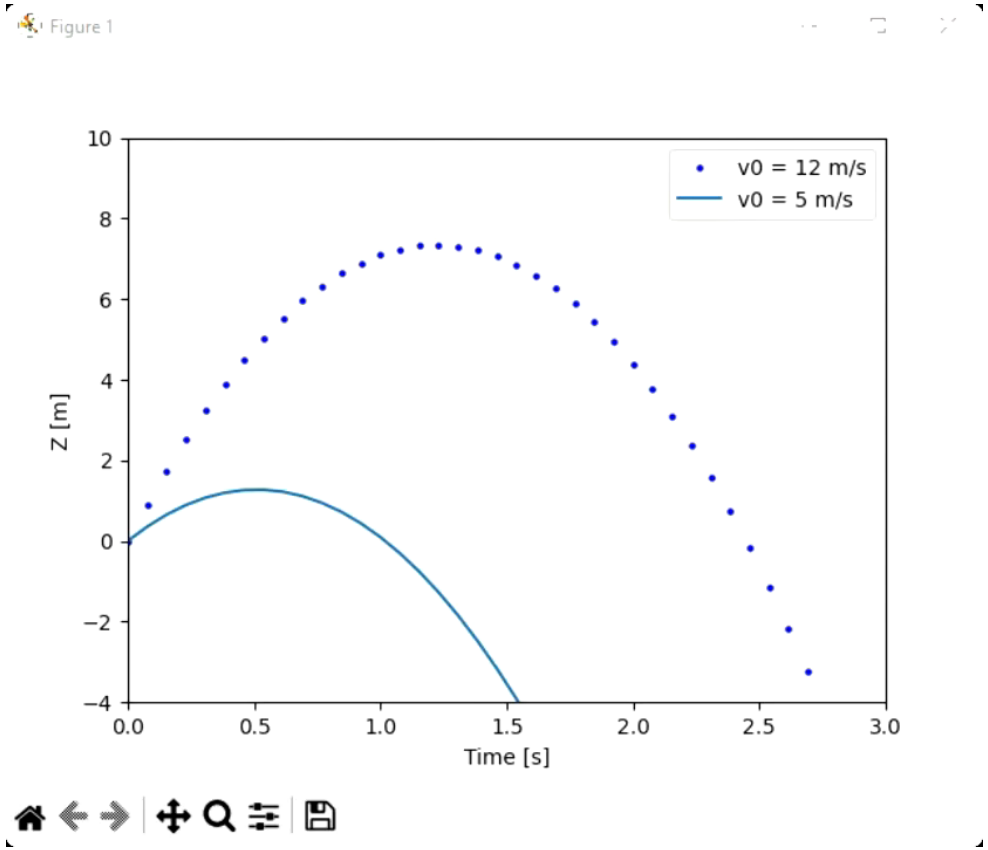
Cod Python

```
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()
```

14.2. Grafice in Python

Exemple: grafic 2D animat

Ex: animație care ilustreaza traiectoria unui proiectil cu două viteze inițiale diferite (v0 și v02).



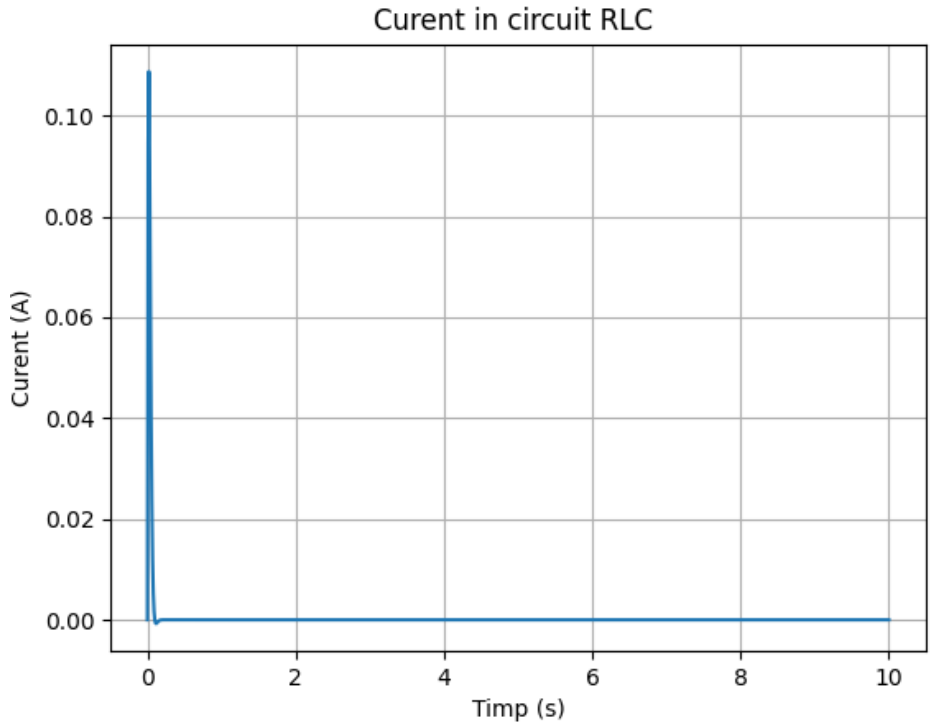
Cod Python

```
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()
```

14.2. Grafice in Python

Exemple: grafic 2D

Ex: Codul Python afiseaza graficul 2D al curentului intr-un circuit RLC



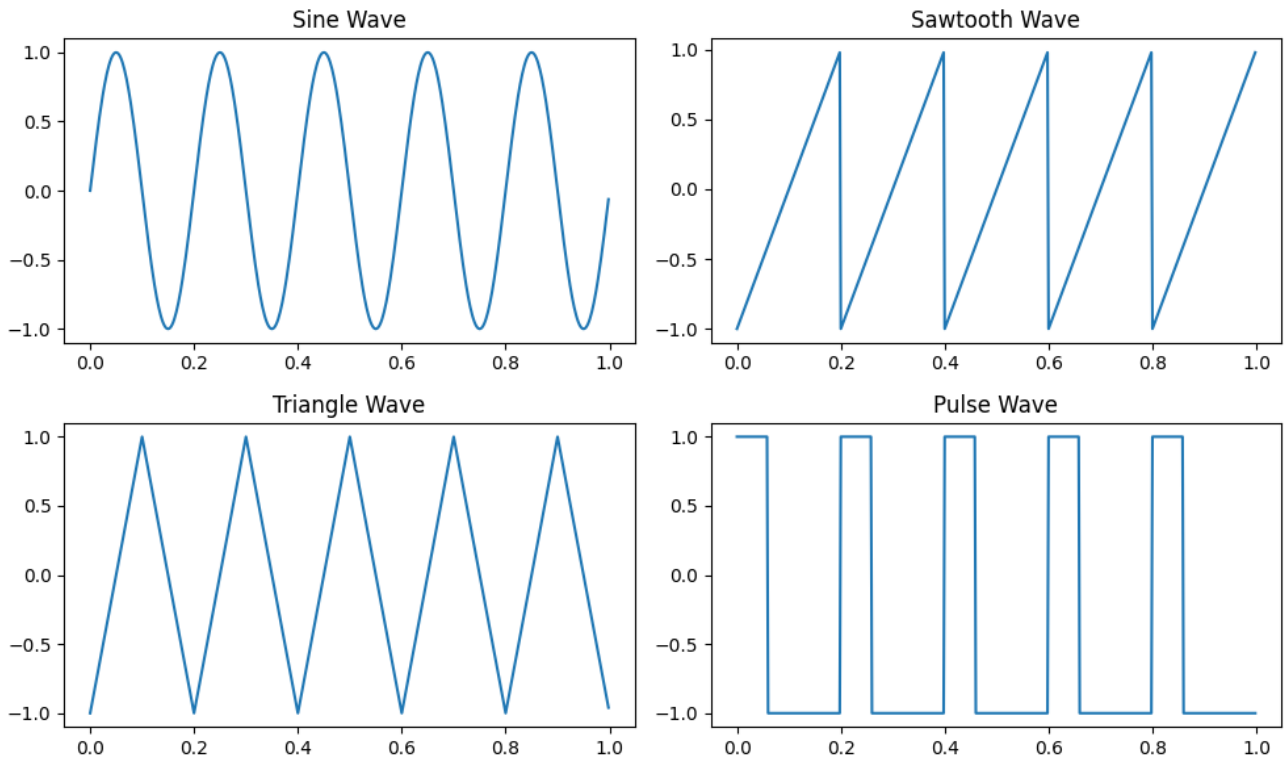
Cod Python

```
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()
```

14.2. Grafice in Python

Exemple: subgrafice 2D

Ex: Codul Python afiseaza graficele 2D ale diferitelor forme de unda



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()
```