Lab Report 0: Introduction to Advanced Python

Course: EEEE 536.60L1 and EEEE 636.60L1 – Biorobotics/Cybernetics

Group Members: Youssief Khalifa(yak9700), Majd Abo Kalam(mma7566)

Date: 06-2-2025

---

1) Manifesto

Both team members actively worked on every section of this lab report. The structure of which collaboration was very efficient and led to high standards for all tasks being completed. We made sure that the lab questions and tasks were divided equally between both of us so that there is fairness and efficiency. The responsibility for specific set of exercises and sections of the report was taken by each of the team member.

---

2) Introduction

This lab introduces high-level Python programming with an emphasis on significant packages such as NumPy, Matplotlib, and Pandas. In general, the purpose is getting acquainted with Python capabilities for manipulating, visualization, and analysis of data. Exercises in labs involve creating and manipulating arrays with NumPy, plotting with Matplotlib, and analysis of data with Pandas. To complete the lab, Majd and I have to become proficient in creating Python script, installation of packages, and use of function in such packages for simple operations in analysis of data.

---

3) Theory

Python is a high-level, interpreted, and widely used language, mainly known for readability and simplicity. It has extensive applications in scientific computing, data analysis, and applications of artificial intelligence. The following theoretical concepts came into prominence in this lab:

- **NumPy**: for numerical computation, manipulation of arrays, including matrix operations and mathematical computations.
- **Matplotlib**: A plotting library to create plots, histograms, and bar charts.
- **Pandas**: A data analysis library that provides data structures to efficiently handle structured data, organized as DataFrames.

Application in Biorobotics/ Cybernetics

- **Data Analysis:** Python is also actively used in the analysis of physiological signals, including ECG, EMG, and EEG.
- **Robotic Systems**: It is useful in programming robotic control systems and embedded devices.
- **Machine Learning:** It powers most AI-driven applications in cybernetics for predictive modeling and pattern recognition.

---

4) Methodology

Setting Up Python Environment:

- Installed Python and set up the development environment with Visual Studio Code.
- Installed packages include NumPy, Matplotlib, and Pandas.

Exercises

1. NumPy Exercises

- Created and manipulated NumPy arrays.
- Examined attributes of created matrices: shape, size, and data type.
- Created matrices filled with zeros, ones, and identity matrices.
- Created sequences using np.linspace() and np.arange()).

2. Matplotlib Exercises

- Scatter plots to show the relation between age and height.
- Bar charts plotted representing weights of persons.
- Histograms created for analysis of weight data distribution.

3. Pandas Exercises

- A dataset from a CSV file was loaded into memory and analyzed.
- Statistical values of numerical data calculated: sum, mean, standard deviation, minimum/maximum.
- Applied the unique() method to get distinct values.
- Visualized Data Distribution of Datasets Using Pandas and Matplotlib.

Data Collection by Sensor

- The Lab Did Not Deal with Real Sensing Data Intake but Instead Relied on a Prepared Dataset.
- Data is taken from a CSV file; data analysis has been performed with Pandas.

Data Cleaning and Analysis

- Dataset preprocessing: clean missing values in the dataset; check consistency.
- Applied basic principles of programming significant in the processes of data analysis, such as:
    - Variable declaration, loops, conditionals, functions, classes, and object-oriented programming.
    -
    - Data structures including lists, tuples, and dictionaries.

Preprocessing Techniques

- **The dataset was preprocessed by the following techniques.**
    - Filtering and program flow control.
    - Iterating over sequences which is very similar to how records in a dataset are processed.
    - Manipulating and processing data in tabular format.

Feature Extraction Techniques (if applicable)

- Extracted relevant features from the dataset for analysis.
- Gained hands-on experience working with lists and dictionaries, two of the most common data structures used when working with and manipulating data.

Machine Learning (if applicable): Machine learning will not be covered in this lab.

The major goal was to know Python syntax, data types, and basic principles of programming, which work as building blocks for doing Data Science and AI.

---

5) Results

A)

Exercise 1:

```
PS C:\Users\ghena abo kalam\OneDrive\Desktop\Majd\EEEE 536\Lab 0 - advanced> & "C:/Users/ghena abo kalam/anacond
a3/envs/python/python.exe" "c:/Users/ghena abo kalam/OneDrive/Desktop/Majd/EEEE 536/Lab 0 - advanced/part 1.py"
Type/Class of this object: <class 'numpy.ndarray'>
Here is the matrix
----------
 [[1 2 3]
 [4 5 6]
 [7 8 9]]
----------
Dimension of this matrix: 2
Size of this matrix: 9
Shape of this matrix: (3, 3)
Data type of this matrix: int32
Data type of the modified matrix: float64


Even tuples can be converted to ndarrays...
We write b = np.array([(1.5,2,3), (4,5,6)])
Matrix made from tuples, not lists
---------------------------------------
[[1.5 2.  3. ]
 [4.  5.  6. ]]
A series of numbers: [ 5  6  7  8  9 10 11 12 13 14 15]
21 linearly spaced numbers between 1 and 5
-----------------------------------------
[1.  1.2 1.4 1.6 1.8 2.  2.2 2.4 2.6 2.8 3.  3.2 3.4 3.6 3.8 4.  4.2 4.4
 4.6 4.8 5. ]
Vector of zeroes
--------------------
```

```
Vector of zeroes
--------------------
[0. 0. 0. 0. 0.]
Matrix of zeroes
-------------------
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
Vector of ones
--------------------
[1. 1. 1. 1. 1.]
Matrix of ones
--------------------
[[1. 1.]
 [1. 1.]
 [1. 1.]
 [1. 1.]
 [1. 1.]]
Matrix of 5's
--------------------
[[5. 5. 5. 5. 5.]
 [5. 5. 5. 5. 5.]
 [5. 5. 5. 5. 5.]]
Empty matrix
------------
 [[5. 5. 5. 5. 5.]
 [5. 5. 5. 5. 5.]
 [5. 5. 5. 5. 5.]]
Identity matrix of dimension (4, 4)
```

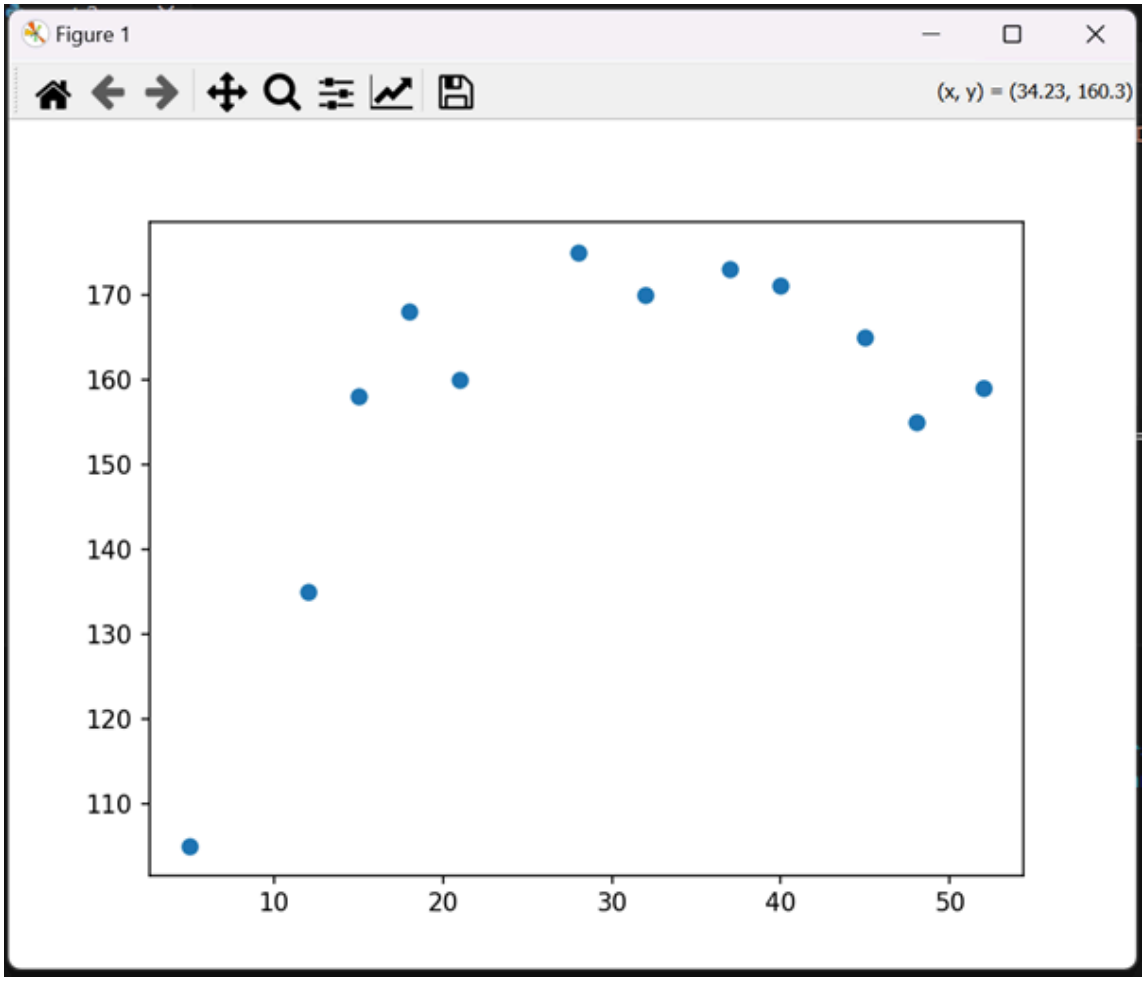
```
  [1. 1.]
  [1. 1.]]
 Matrix of 5's
 --------------------
 [[5. 5. 5. 5. 5.]
  [5. 5. 5. 5. 5.]
  [5. 5. 5. 5. 5.]]
 Empty matrix
 ------------
  [[5. 5. 5. 5. 5.]
  [5. 5. 5. 5. 5.]
  [5. 5. 5. 5. 5.]]
 Identity matrix of dimension (4, 4)
 [[1. 0. 0. 0.]
  [0. 1. 0. 0.]
  [0. 0. 1. 0.]
  [0. 0. 0. 1.]]
 PS C:\Users\ghena abo kalam\OneDrive\Desktop\Majd\EEEE 536\Lab 0 - advanced>
```

In this lab, we explored how to create and manipulate matrices using NumPy in Python. We first created a 3x3 matrix and confirmed its properties, such as its type, shape, size, and data type. Initially, the matrix contained integers, but when we introduced decimal numbers, it automatically converted to a float type. We also saw that tuples can be used to create NumPy arrays, showing that NumPy is flexible in handling different data types. Next, we generated sequences of numbers using np.arange(), which created a range of numbers from 5 to 15, and np.linspace(), which produced 21 evenly spaced values between 1 and 5.

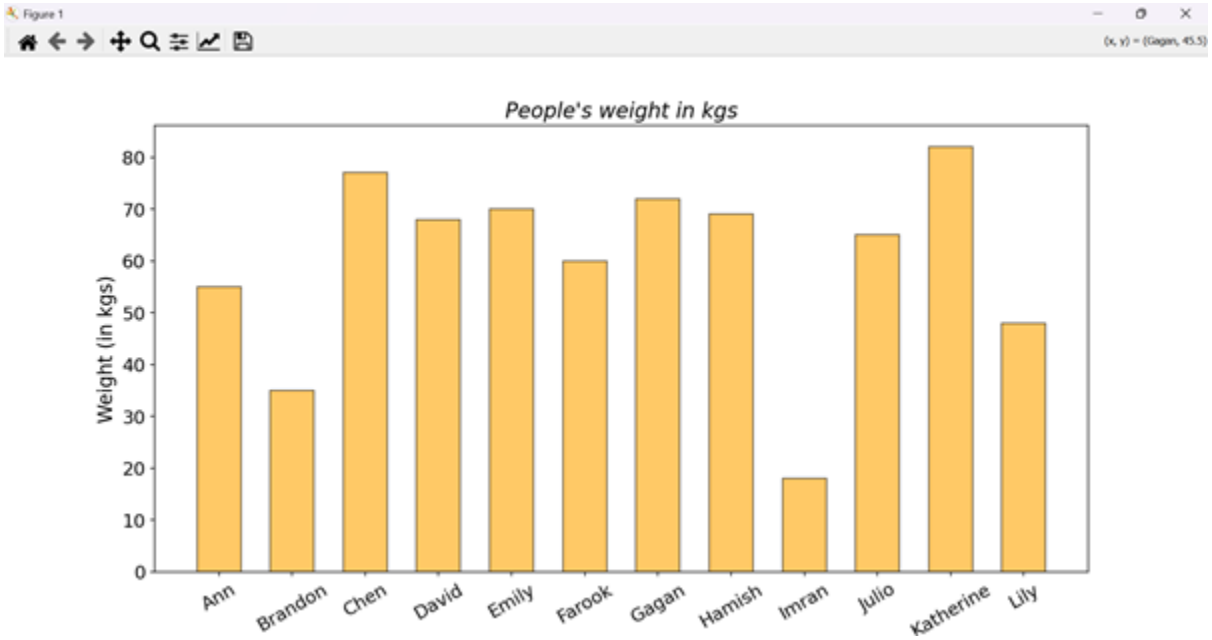
We also created special matrices, including zero matrices, ones matrices, a matrix filled with the number 5, and an empty matrix with uninitialized values. Lastly, we generated a 4x4 identity matrix with ones along the diagonal using np.eye(4), which appeared as expected.

Exercise 2:

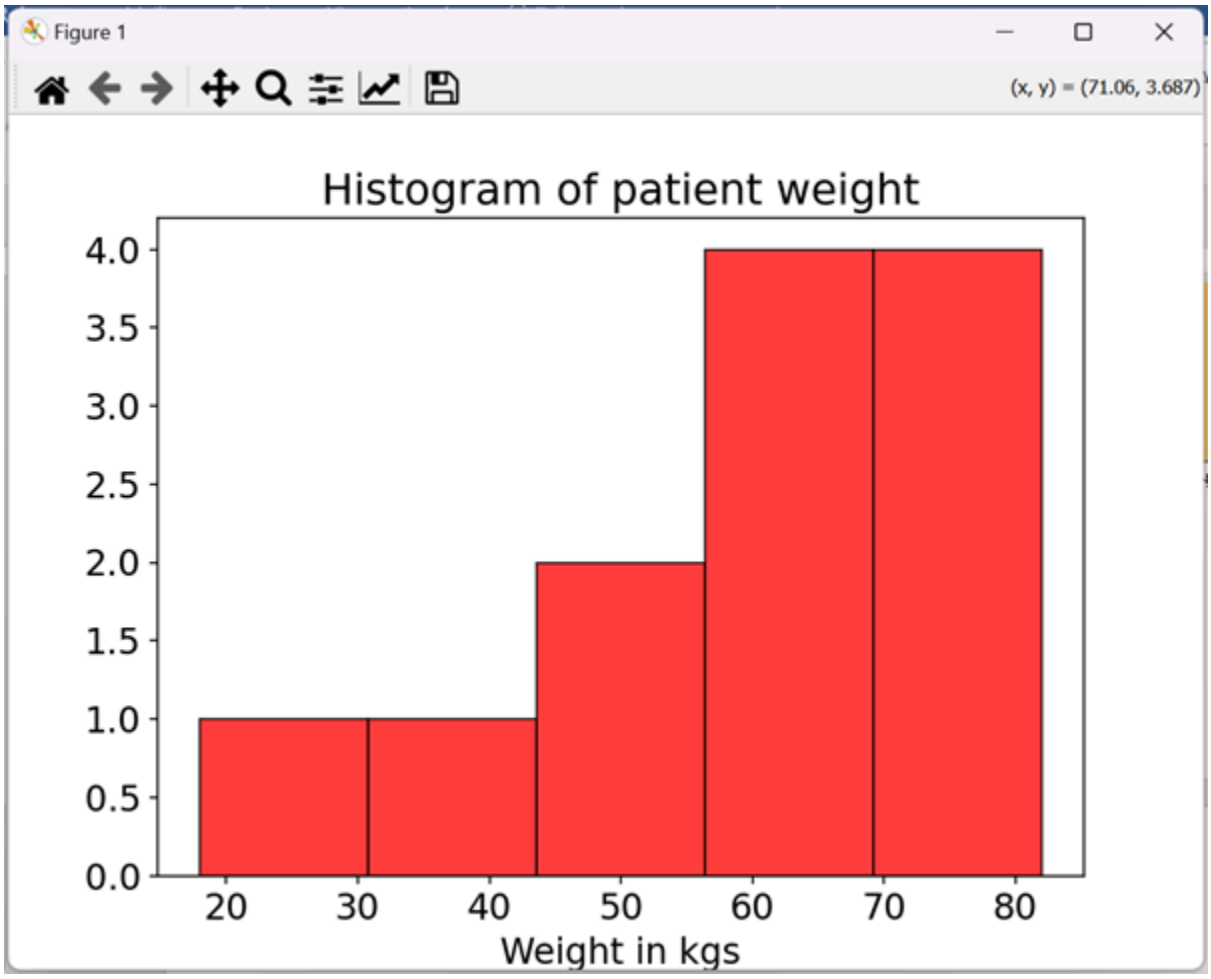
```
import matplotlib.pyplot as plt

people = ['Ann','Brandon','Chen','David','Emily','Farook',

'Gagan','Hamish','Imran','Julio','Katherine','Lily']

age = [21,12,32,45,37,18,28,52,5,40,48,15]

weight = [55,35,77,68,70,60,72,69,18,65,82,48]

height = [160,135,170,165,173,168,175,159,105,171,155,158]

plt.scatter(age, height)

plt.show()
```



```
plt.figure(figsize=(12,4))

plt.title("People's weight in kgs",fontsize=16, fontstyle='italic')

# Main plot function 'bar'

plt.bar(x=people,height=weight,
width=0.6,color='orange',edgecolor='k',alpha=0.6)

plt.xlabel("People",fontsize=15)

plt.xticks(fontsize=14,rotation=30)

plt.yticks(fontsize=14)

plt.ylabel("Weight (in kgs)",fontsize=15)

plt.show()
```



```
import numpy as np

plt.figure(figsize=(7,5))

# Main plot function 'hist'

plt.hist(weight,color='red',edgecolor='k', alpha=0.75,bins=5)

plt.title("Histogram of patient weight",fontsize=18)

plt.xlabel("Weight in kgs",fontsize=15)

plt.xticks(fontsize=15)

plt.yticks(fontsize=15)

plt.show()
```



In this lab, we explored data visualization techniques using Matplotlib. We created three different types of plots: a scatter plot, a bar chart, and a histogram to analyze the data visually.

First, we generated a scatter plot to visualize the relationship between age and height. The output showed a dispersed distribution of points, indicating that there is no clear correlation between a person's age and their height. Some younger individuals had higher heights, while some older individuals had shorter heights, confirming natural variations in human growth.

Next, we created a bar chart to represent the weight of different individuals. The bars were plotted for each person, with their weight displayed on the y-axis. The orange-colored bars provided a clear visual comparison, showing that some individuals had significantly higher or lower weights than others. The chart effectively highlighted the differences in weight distribution across the group.

Finally, we generated a histogram to analyze the distribution of weight among the individuals. The histogram grouped the weights into five bins, showing how many people fell into each weight category. The resulting output helped us identify that most individuals had weights within a certain range, while some outliers existed at the higher and lower ends of the spectrum.

Exercise 3:

```
PS C:\Users\ghena abo kalam\OneDrive\Desktop\Majd\EEEE 536\Lab 0 - advanced> & "C:/Users/ghena abo kalam/anaconda3/envs/python/p
ython.exe" "c:/Users/ghena abo kalam/OneDrive/Desktop/Majd/EEEE 536/Lab 0 - advanced/part 3.py"

Sum of the column 'Sales' is:  1377
Mean of the column 'Sales' is:  229.5
Std dev of the column 'Sales' is:  100.89945490437498
Min and max of the column 'Sales' are:  120 and 350

Method head() is for showing first few entries
------------------------------------------------

Finding unique values in 'col2'
---------------------------------------
['Sam' 'Charlie' 'Amy' 'Vanessa' 'Carl' 'Sarah']
PS C:\Users\ghena abo kalam\OneDrive\Desktop\Majd\EEEE 536\Lab 0 - advanced>
```

In this lab, we used Pandas to create and analyze a dataset containing sales data for different companies and individuals. The dataset was structured as a DataFrame with three columns: Company, Person, and Sales. Various statistical operations were performed to summarize and interpret the data.

First, we calculated key statistics on the Sales column. The sum of all sales was computed as 1377, while the mean (average) sales was 229.5, providing insight into the central tendency of the data. The standard deviation of sales was found to be 100.899, indicating the spread of sales values around the mean. The minimum and maximum sales values were 120 and 350, respectively, showing the range of sales figures in the dataset.

Next, we used the head() method to display the first few rows of the dataset, allowing a quick preview of the data structure. We also identified unique individuals in the dataset using the unique() function, which confirmed that each sales entry was associated with a distinct person.

Exercise 3.1 (print(df.to_string())):

```
PS C:\Users\ghena abo kalam\OneDrive\Desktop\Majd\EEEE 536\Lab 0 - advanced> & "C:/Users/ghena abo kalam/anaconda3/envs/python/p
ython.exe" "c:/Users/ghena abo kalam/OneDrive/Desktop/Majd/EEEE 536/Lab 0 - advanced/part 3 version 2.py"
     Duration  Pulse  Maxpulse  Calories
0          60    110       130     409.1
1          60    117       145     479.0
2          60    103       135     340.0
3          45    109       175     282.4
4          45    117       148     406.0
5          60    102       127     300.0
6          60    110       136     374.0
7          45    104       134     253.3
8          30    109       133     195.1
9          60     98       124     269.0
10         60    103       147     329.3
11         60    100       120     250.7
12         60    106       128     345.3
13         60    104       132     379.3
14         60     98       123     275.0
15         60     98       120     215.2
16         60    100       120     300.0
17         45     90       112       NaN
18         60    103       123     323.0
19         45     97       125     243.0
20         60    108       131     364.2
21         45    100       119     282.0
22         60    130       101     300.0
23         45    105       132     246.0
24         60    102       126     334.5
```

```
  146        60    107       136     400.0
  147        60    112       146     361.9
  148        30    103       127     185.0
  149        60    110       150     409.4
  150        60    106       134     343.0
  151        60    109       129     353.2
  152        60    109       138     374.0
  153        30    150       167     275.8
  154        60    105       128     328.0
  155        60    111       151     368.5
  156        60     97       131     270.4
  157        60    100       120     270.4
  158        60    114       150     382.8
  159        30     80       120     240.9
  160        30     85       120     250.4
  161        45     90       130     260.4
  162        45     95       130     270.0
  163        45    100       140     280.9
  164        60    105       140     290.8
  165        60    110       145     300.0
  166        60    115       145     310.2
  167        75    120       150     320.4
  168        75    125       150     330.4
  PS C:\Users\ghena abo kalam\OneDrive\Desktop\Majd\EEEE 536\Lab 0 - advanced>
```

Exercise 3.2 (print(df)):

```
PS C:\Users\ghena abo kalam\OneDrive\Desktop\Majd\EEEE 536\Lab 0 - advanced> & "C:/Users/ghena abo kalam/anaconda3/envs/python/p
ython.exe" "c:/Users/ghena abo kalam/OneDrive/Desktop/Majd/EEEE 536/Lab 0 - advanced/part 3 version 2.py"
     Duration  Pulse  Maxpulse  Calories
0          60    110       130     409.1
1          60    117       145     479.0
2          60    103       135     340.0
3          45    109       175     282.4
4          45    117       148     406.0
..        ...    ...       ...       ...
164        60    105       140     290.8
165        60    110       145     300.0
166        60    115       145     310.2
167        75    120       150     320.4
168        75    125       150     330.4

[169 rows x 4 columns]
PS C:\Users\ghena abo kalam\OneDrive\Desktop\Majd\EEEE 536\Lab 0 - advanced>
```

In this lab, we worked with Pandas to read and display data from a CSV file named data.csv. We used the read_csv() function to load the dataset into a Pandas DataFrame, a structured table format that allows easy manipulation and analysis of data.

First, we printed the entire dataset using df.to_string(), which ensured that all rows and columns were displayed without truncation. This provided a comprehensive view of the dataset, allowing us to observe its structure, values, and possible irregularities.

Next, we printed the DataFrame without to_string(), which displayed the dataset in a default format where large datasets might be partially shown, depending on the number of rows. This approach is useful for quick inspections of the data while preserving readability.

B)

We decided to include all the screenshots as we felt like that all figures and codes were necessary to include.

---

6) References

GeeksforGeeks. (2023). Python Programming Language.
https://www.geeksforgeeks.org/python-programming-language/

Real Python. (2023). Object-Oriented Programming (OOP) in Python.
https://realpython.com/python3-object-oriented-programming/

W3Schools. (2023). Python Tutorial.

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

NumPy Documentation:

https://numpy.org/doc/

Matplotlib Documentation:

https://matplotlib.org/stable/contents.html

Pandas Documentation:

https://pandas.pydata.org/docs/