CODE 1: (code 1.py)

- **Concept used:** NumPy (Numerical Python) to create and work with arrays.
- **OBSERVATION:**
- `np.array([1, 2, 3, 4, 5])`: Makes a 1D array , a simple row of numbers
- `np.array([[1, 2, 3], [4, 5, 6]])` :Makes a 2D array ,liike a table or matrix with rows and columns.
- `np.zeros((3, 3))`: Creates a 3x3 grid filled with zeros.
- `np.ones((2, 4))`: Makes a 2x4 grid filled with ones**.**
- `np.arange(0, 10, 2)`: Creates numbers from 0 up to 10, but with a step size of 2.
- `np.linspace(0, 1, 5)`: Creates 5 numbers evenly spaced between 0 and 1.

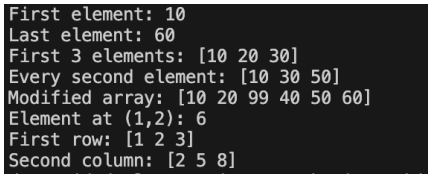
```
1D Array: [1 2 3 4 5]
2D Array:
 [[1 2 3]
 [4 5 6]]
3x3 Zero Matrix:
 [[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
2x4 Ones Matrix:
 [[1. 1. 1. 1.]
 [1. 1. 1. 1.]]
Range Array: [0 2 4 6 8]
Linearly spaced values: [0.   0.25 0.5  0.75 1.  ]
```

CODE 2 : (code 2.py)

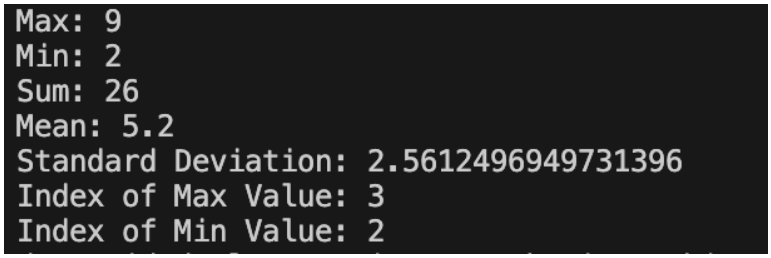
- **Concept used:** NumPy array operations
- **OBSERVATION:**
- Using numpy function we can easily perform addition, subtraction, multiplication, division, square roots, and powers on the whole array at once.
-

```
Addition: [11 22 33 44]
Subtraction: [ 9 18 27 36]
Multiplication: [ 10  40  90 160]
Division: [10. 10. 10. 10.]
Square root of a: [3.16227766 4.47213595 5.47722558 6.32455532]
a squared: [ 100  400  900 1600]
```

CODE 3 : (code 3.py)

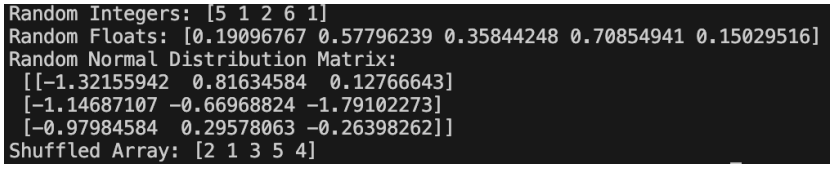
- **Concept used:** Indexing and Slicing in NumPy arrays.
- **OBSERVATION:**
- This code teaches us how to access elements by index, slice sub-parts, and modify elements directly.
- 

CODE 4: (code 4.py)

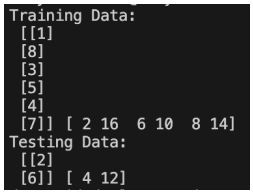
- **Concept used:** basic statistical operations in NumPy
- **OBSERVATION:**
- This code tells us how arrays can quickly give summary statistics (max, min, sum, average, std deviation) and even positions of those values.
- 

CODE 5 (code 5.py)

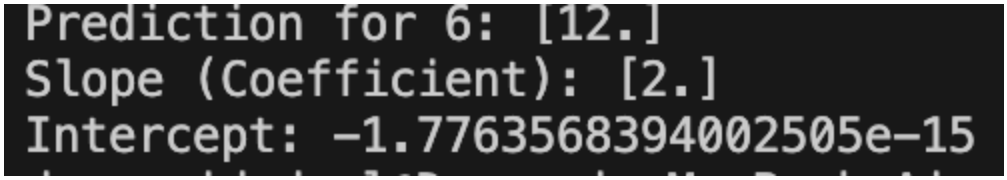
- **Concept used:** random number generation and shuffling with NumPy.
- **OBSERVATION:**
- Through thai code we understand how the outputs vary each time : showing randomness. One learns how to generate random integers, floats, random normal-distribution values, and shuffle arrays. This is useful in simulations or ML training.

- 

CODE 6 (code 6.py)

- **Concept used:** how to split a dataset into training and testing sets using `train_test_split` from scikit-learn.
- **OBSERVATION:**
- How the dataset splits into training and testing sets. This teaches why we don't train and test on the same data → helps in fair model evaluation.
- 

CODE 7 : (code 7.py)

- **Concept used:** Linear Regression using scikit-learn.
- **OBSERVATION**
- The model predicts correctly (like `x=6 → y=12`). One can learn that regression finds the best straight-line relationship between input (X) and output (y).
- 

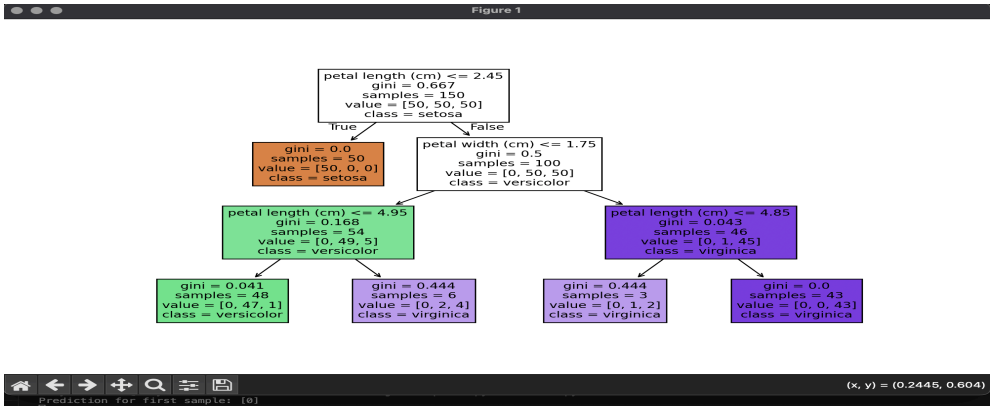
CODE 8: (code 8.py)

- **Concept used:** Logistic Regression using scikit-learn
- **OBSERVATION:**
- The output shows 0 or 1 predictions depending on hours studied. One learns that logistic regression helps in classification problems, and `predict_proba` shows confidence levels.
-

```
Prediction for 2.5 hours: [0]
Prediction for 6 hours: [1]
Probabilities for 2.5 hours: [[0.75496813 0.24503187]]
```

CODE 9: (code 9.py)

- **Concept used:** trained and visualized a Decision Tree Classifier on the Iris dataset
- **OBSERVATION:**
- The tree shows decisions like "if petal length < X → class = setosa." Students learn decision trees make predictions by asking a series of yes/no questions.
-



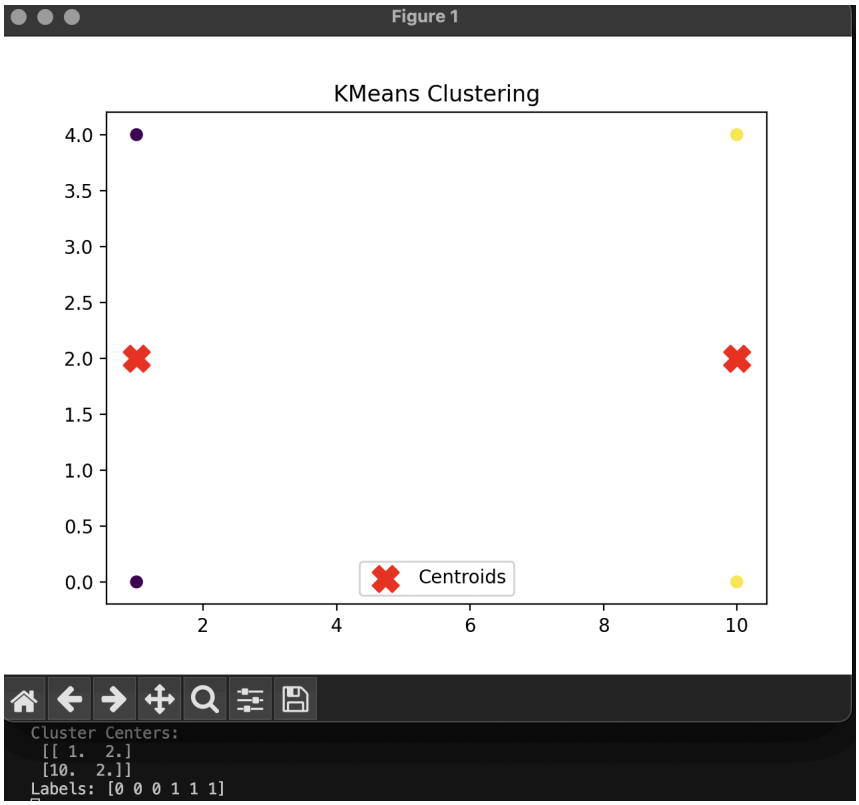
CODE 10: (code 10.py)

- **Concept used:** Standardization (Z-score Normalization)
- **OBSERVATION:**
- How original values are transformed into values with mean = 0 and std = 1. One learns that scaling helps ML models work better.

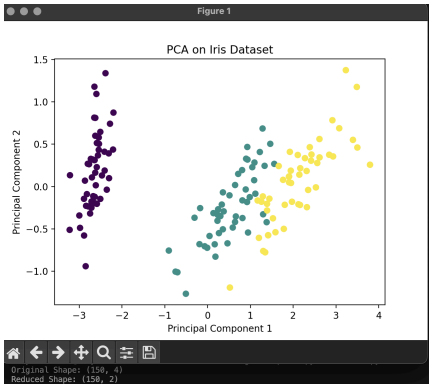
-
```
Original Data:
 [[ 10 100]
 [ 20 200]
 [ 30 300]
 [ 40 400]]
Standardized Data:
 [[-1.34164079 -1.34164079]
 [-0.4472136  -0.4472136 ]
 [ 0.4472136   0.4472136 ]
 [ 1.34164079  1.34164079]]
```

CODE 11: (code 11.py)

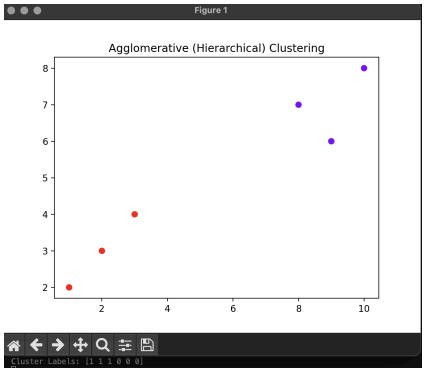
- **Concept used:** K-Means Clustering
- **OBSERVATION:**
- Data points are grouped into clusters, with red “X” marking cluster centers. One learns that unsupervised learning finds hidden patterns without labels.

- 

CODE 12: (code 12.py)

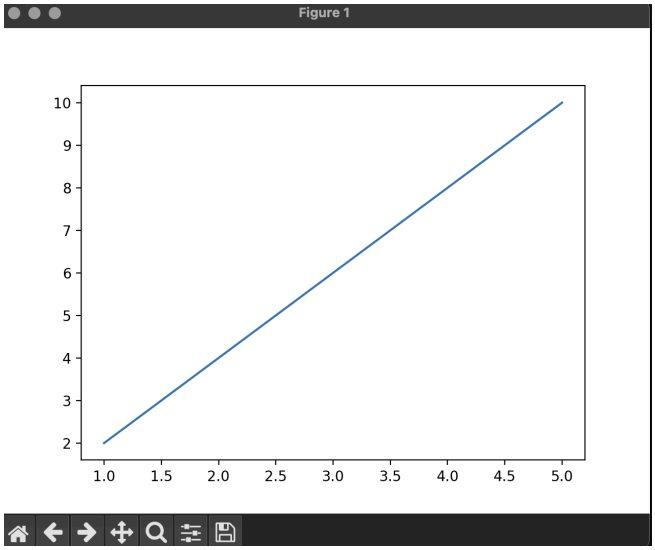
- **Concept used:** Principal Component Analysis (PCA) for dimensionality reduction
- **OBSERVATION:**
- From 4D data reduced to 2D → one can see we can visualize high-dimensional data while keeping important patterns.
- 

CODE 13: (code 13.py)

- **Concept used:** Agglomerative (Hierarchical) Clustering for unsupervised learning
- **OBSERVATION:**
- One can learn how the points are grouped into clusters using hierarchy. Unlike KMeans, no predefined centroids → clusters form bottom-up step by step.
- 

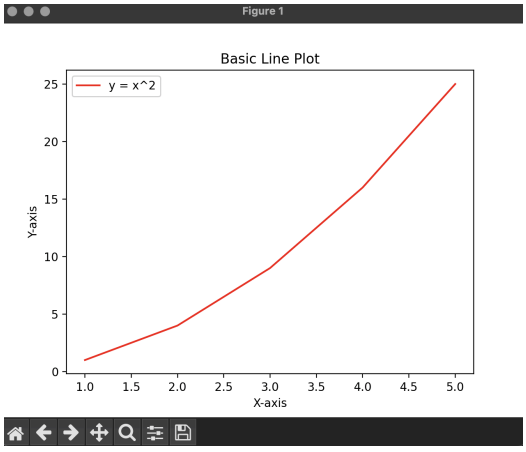
CODE 14: (plot1.py)

- **Concept used:** Line plotting using Matplotlib**.**
- **OBSERVATION:**
- A simple line graph shows relationship between X and Y , one can learn how to visualize data trends.



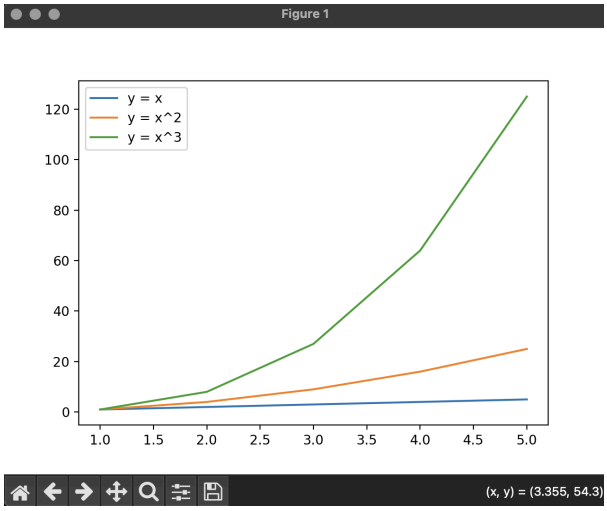
CODE 15: (plot2.py)

- **Concept used:** Line plotting with customization using Matplotlib.
- **OBSERVATION:**
- One can learn how to add labels, legends, and titles , making plots informative and professional.



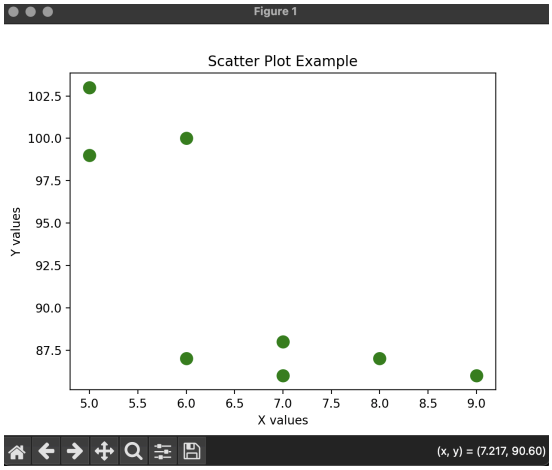
CODE 16: (plot3.py)

- **Concept used:** Multiple line plots in a single figure using Matplotlib
- **OBSERVATION:**
- One learns how to compare multiple functions in the same plot. Useful for comparing relationships.



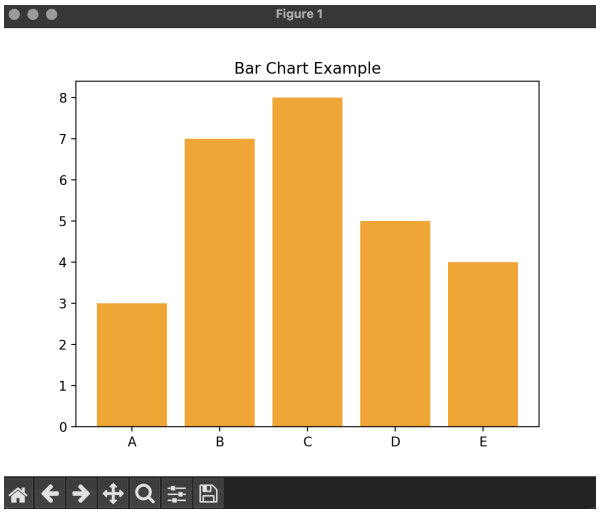
CODE 17: (plot4.py)

- **Concept used:** Scatter plot using Matplotlib
- **OBSERVATION:**
- One can learn how individual data points are plotted. This shows relationships, clusters, or trends in data.



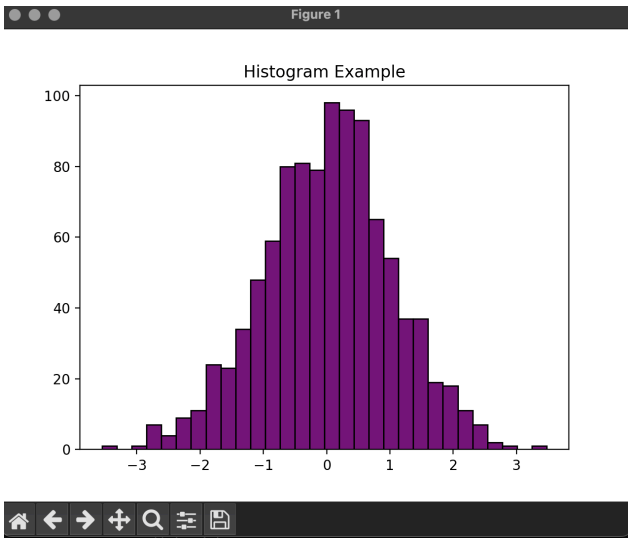
CODE 18: (plot5.py)

- **Concept used:**Bar chart using Matplotlib
- **OBSERVATION:**
- One learns how bar charts compare categorical data (A, B, C) with numerical values.



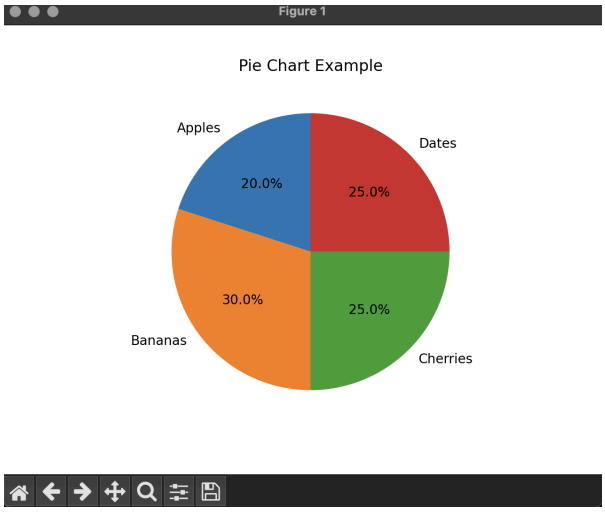
CODE 19: (plot6.py)

- **Concept used:** Histogram using Matplotlib
- **OBSERVATION:**
- One can learn the data distribution . This helps understand frequency of values in datasets.



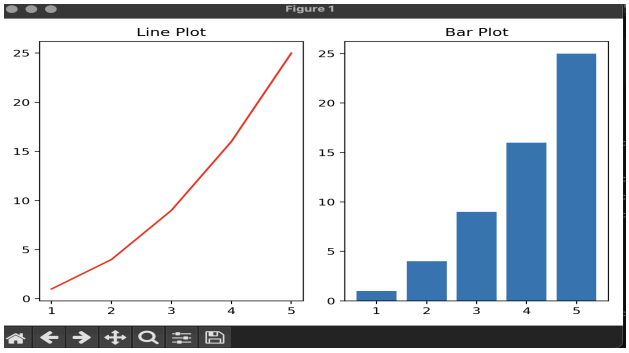
CODE 20: (plot7.py)

- **Concept used:** Pie chart using Matplotlib
- **OBSERVATION:**
- One learns how percentages of categories (like fruits) can be visualized. Good for proportions.



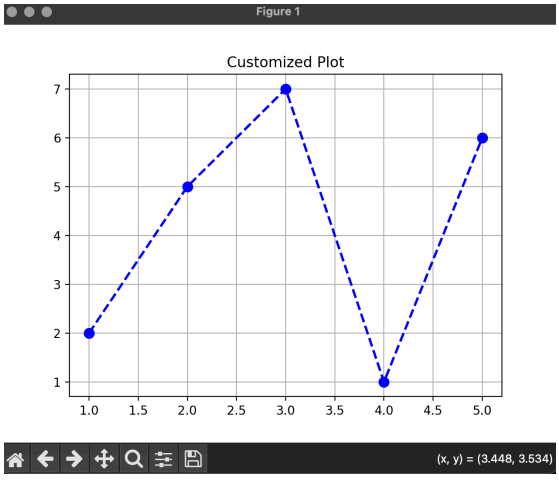
CODE 21: (plot8.py)

- **Concept used:** Multiple plots in a single figure using Matplotlib's `subplot`
- **OBSERVATION:**
- One learns multiple plots side-by-side. They learn to compare data in different views in one figure.



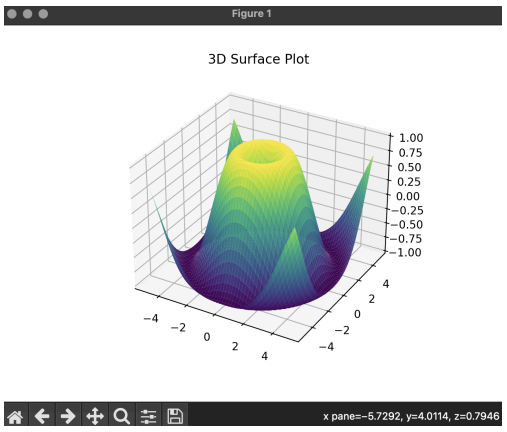
CODE 22: (plot9.py)

- **Concept used:** Customized line plot using Matplotlib
- **OBSERVATION:**
- One learns how customizing style (markers, colors, grid) makes graphs clearer and more professional.



CODE 23: (plot10.py)

- **Concept used:** 3D surface plotting using Matplotlib
- **OBSERVATION:**
- One sees a colorful 3D surface of a mathematical function. They learn how to visualize functions of two variables.



CODE 24: (scikitlearn_linear regression.py)

- **Concept used:** Simple Linear Regression using scikit-learn**.**
- **OBSERVATION:**
- Scatter plot shows noisy data + red line shows regression fit. One learns that linear regression can estimate trends despite noise.
-