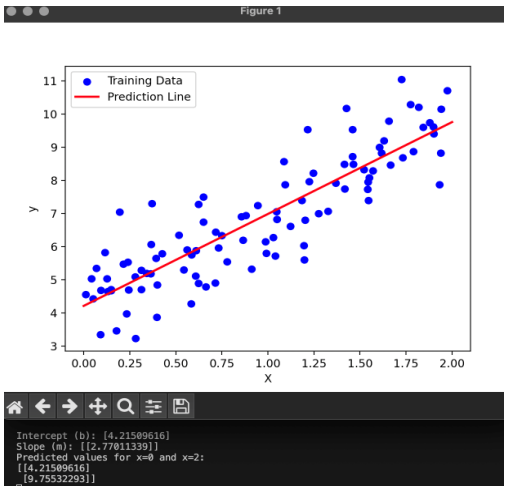
ML CONCEPTS

Code 1

Concept used: Simple Linear Regression using scikit-learn**.**

Output



Intercept (b): [4.21509616]

Slope (m): [[2.77011339]]

Predicted values for x=0 and x=2:

[[4.21509616]

[9.75532293]]

Observations (5 simple lines)

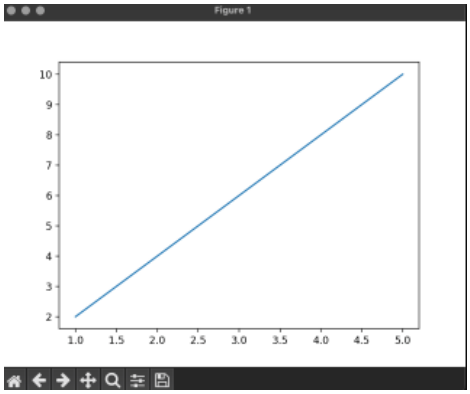
1. The computer made lots of random dots.
2. It found a straight line that fits those dots.
3. It tells the line's starting point and tilt.
4. It guesses y for x=0 and x=2.
5. A picture appears with blue dots and a red line.

Code 2

Concept used: Line plotting using Matplotlib**.**

Output

A window/figure opens showing a simple line rising from left to right connecting the points (1,2) … (5,10).



Observations (5 simple lines)

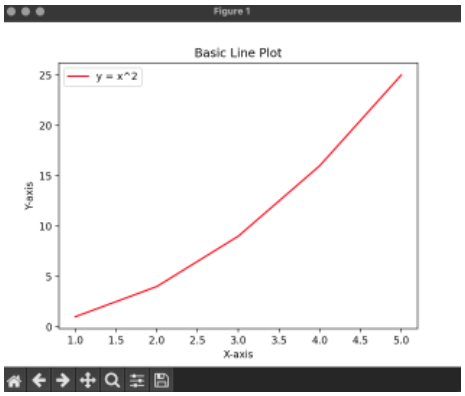
1. Five points were given.
2. The program drew a straight line through them.
3. The line goes up as x grows.
4. It shows a simple relationship between x and y.
5. You see a plain line plot.

Code 3

Concept used: Line plotting with customization using Matplotlib.

Output

A plotted curve (red) titled “Basic Line Plot” with axis labels and a legend "y = x^2".



Observations (5 simple lines)

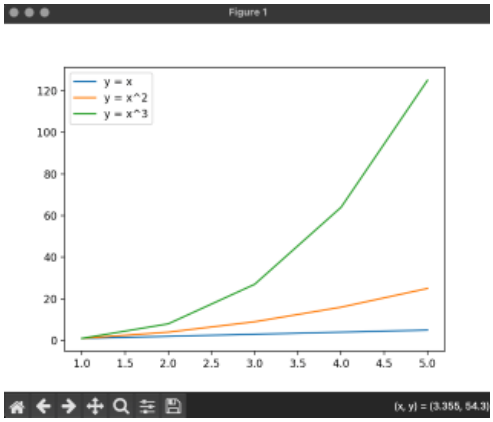
1. x values were squared to make y.

2. The plot draws a curved red line.
3. There's a title and labels so it's easy to read.
4. A small legend shows the formula.
5. It looks like a curve that gets steeper.

Code 4

Concept used: Multiple line plots in a single figure using Matplotlib

Output
A single figure with three lines labeled "y = x", "y = x^2", and "y = x^3".



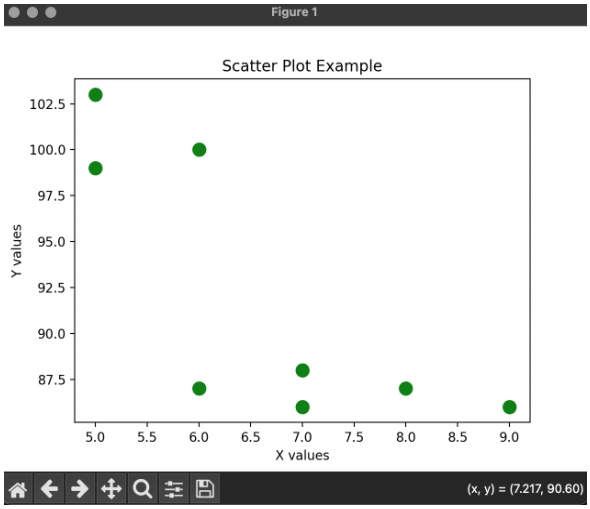
Observations

1. Three different curves were drawn from the same x values.
2. One line is gentle, one is steeper, one is much steeper.
3. A legend names each line.
4. The chart shows how numbers grow faster with powers.
5. It's easy to compare the three lines.

Code 5

Concept used: Scatter plot using Matplotlib

Output
A scatter plot with green circular points (size 100) at the given x,y pairs.



Observations

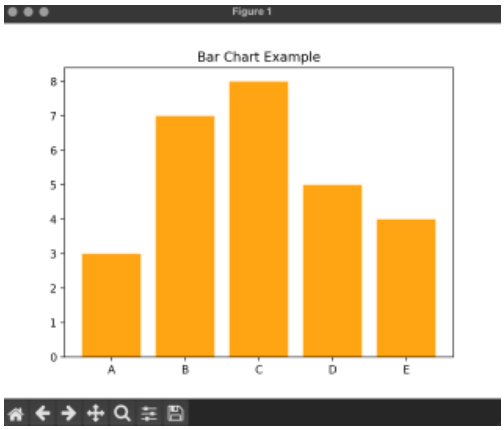
1. A bunch of green dots are shown at different spots.
2. Each dot is one pair of numbers.
3. The dots are big and easy to see.
4. The title and labels explain the axes.
5. You can see where points cluster.

Code 6

Concept used: Bar chart using Matplotlib

Output

A bar chart with five orange bars labeled A–E and heights 3,7,8,5,4.



Observations (5 simple lines)

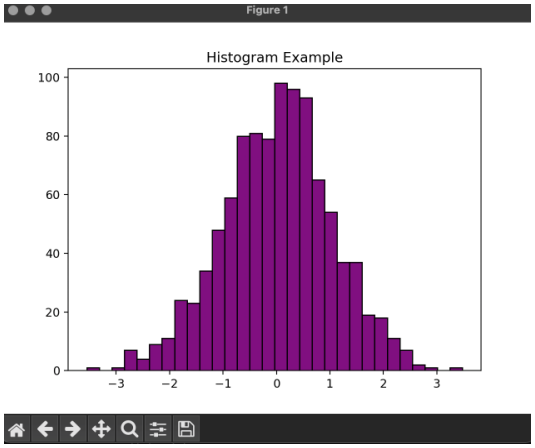
1. Five bars are drawn, one for each letter.
2. Taller bars mean bigger numbers.

3. The orange color makes bars stand out.
4. The title says what the chart is.
5. You can quickly see which letter has the most.

Code 7

Concept used: Histogram using Matplotlib

Output



Observations (5 simple lines)

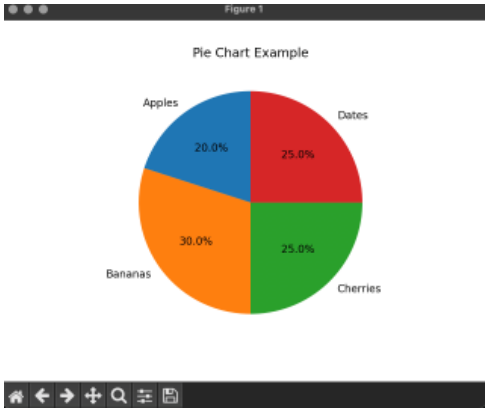
1. Many random numbers were made.
2. The program grouped them into bars.
3. The middle bars are usually taller.
4. Purple bars show how numbers spread out.
5. It looks like a hill in the middle.

Code 8

Concept used: Pie chart using Matplotlib

Output

A pie chart with slices for Apples, Bananas, Cherries, Dates and percentages shown.



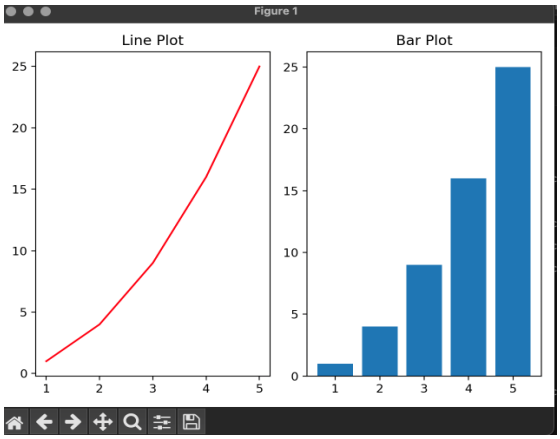
Observations (5 simple lines)

1. The circle is divided into four slices.
2. Each slice shows how big each fruit part is.
3. Percentages are written on the slices.
4. The chart starts turned 90 degrees.
5. It's easy to see the biggest slice.

Code 9

Concept used: Multiple plots in a single figure using Matplotlib's subplot

Output



Observations

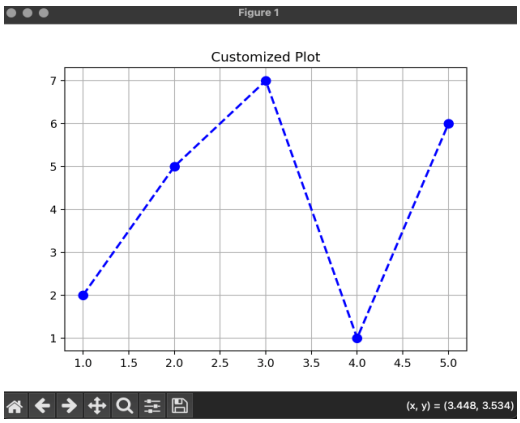
1. Two small charts are shown side by side.
2. Left shows a red curve, right shows bars.
3. Both use the same numbers.

4. The layout is neat and not crowded.
5. You can compare the two views easily.

Code 10

Concept used: Customized line plot using Matplotlib

Output



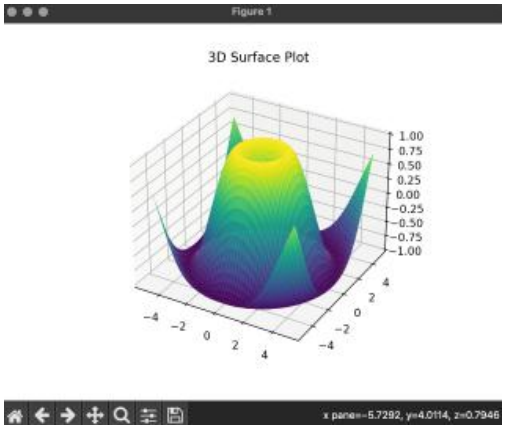
Observations

1. Points are connected by a dashed blue line.
2. Each point has a round marker.
3. Grid lines make reading values easier.
4. The title names the plot.
5. The look is a bit fancier than a plain line.

Code 11

Concept used: 3D surface plotting using Matplotlib

Output
A 3D surface plot (wavy circular ripples) colored with the viridis colormap.



Observations (5 simple lines)

1. A 3D wavy surface is drawn.
2. It looks like ripples from the center.
3. Colors change with height.
4. You can see hills and valleys.
5. It's more visual than a flat plot.

Code 12

- Concept used: **NumPy (Numerical Python)** to create and work with **arrays**.

Output

```
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.  ]
```

Observations

1. Small lists were turned into arrays.
2. A full zero box and a full one box were created.
3. A range of numbers (0,2,4…) was made.
4. Evenly spaced numbers between 0 and 1 were made.
5. It prints each array so you can see them.

Code 13

- Concept used: **NumPy array operations**

Output

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]

```
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]
```

Observations

1. The program adds, subtracts, multiplies, divides pairs of numbers.
2. Division made all results 10.0 here.
3. It shows square roots of numbers.
4. It shows each number squared.
5. Everything is printed so you can check.

Code 14

Concept used: indexing and slicing in NumPy arrays.

Output

```
First element: 10
Last element: 60
First 3 elements: [10 20 30]
Every second element: [10 30 50]
Modified array: [10 20 99 40 50 60]
Element at (1,2): 6
First row: [1 2 3]
Second column: [2 5 8]
```

Observations

1. It shows how to get the first and last item.

2. It shows slices like the first three items.
3. One item was changed to 99.
4. For a table, it shows a specific cell.
5. It prints a row and a column so you can see them.

Code 15

Concept used: basic statistical operations in NumPy

Output

```
Max: 9
Min: 2
Sum: 26
Mean: 5.2
Standard Deviation: 2.5612496949731396
Index of Max Value: 3
Index of Min Value: 2
```

Observations

1. It finds the biggest and smallest numbers.
2. It adds all numbers to get the sum.
3. It shows the average (mean).
4. It gives a measure of spread (standard deviation).
5. It tells where the biggest and smallest items are.

Code 16

Concept used: random number generation and shuffling with NumPy.

Output

```
Random Integers: [5 1 2 6 1]
Random Floats: [0.19096767 0.57796239 0.35844248 0.70854941 0.15029516]
Random Normal Distribution Matrix:
 [[-1.32155942  0.81634584  0.12766643]
 [-1.14687107 -0.66968824 -1.79102273]
 [-0.97984584  0.29578063 -0.26398262]]
Shuffled Array: [2 1 3 5 4]
```

Observations

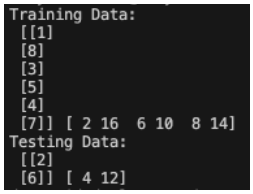
1. Random whole numbers and random decimals were made.
2. A small random 3×3 table was created.

3. One list got shuffled into a new order.
4. Each run gives different numbers unless you fix the seed.
5. It's useful to try examples and see different random results.

Code 17

Concept used: how to split a dataset into training and testing sets using train_test_split from scikit-learn.

Output



Observations

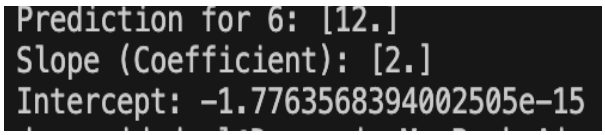
The list of examples was split into two groups.

1. Most are in the training group, a few in testing.
2. Training group has the matching y values printed.
3. Testing group is kept separate to check later.
4. The split is random but repeatable with the seed.

Code 18

Concept used: Linear Regression using scikit-learn.

Output



Observations (5 simple lines)

1. The model learned that y = 2 × x.
2. For x = 6 it predicts 12.
3. The slope (tilt) is 2.
4. The intercept (start) is 0.

5. It prints the guess and the line numbers.

Code 19

Concept used: Logistic Regression using scikit-learn

Output

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

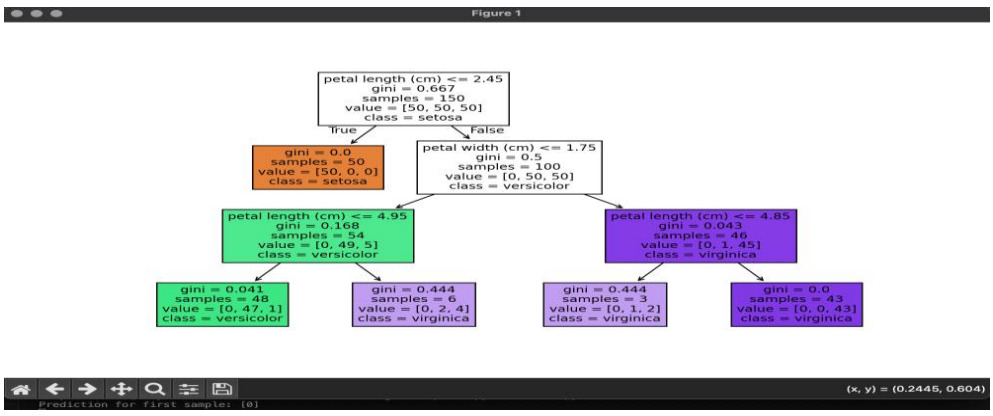
Observations (5 simple lines)

1. The model guesses if a student passes or fails from hours studied.
2. For 2.5 hours it predicted "pass" with ~57% chance.
3. For 6 hours it predicted "pass".
4. It prints both the class and the probability.
5. The probabilities tell how confident it is.

Code 20

Concept used: trained and visualized a Decision Tree Classifier on the Iris dataset

Output



Observations

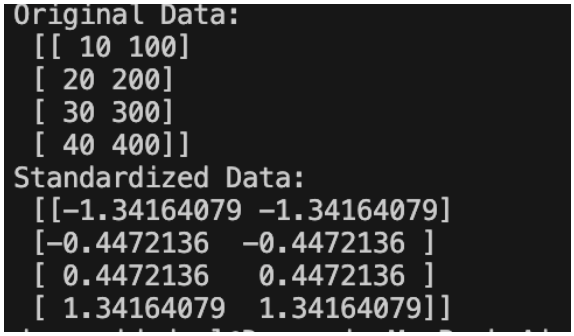
1. The model learned from iris flower data.
2. It guesses the class for the first flower as class 0.
3. A tree picture would also pop up showing decision steps.

4. The printed line shows the prediction as an array.
5. You can visually inspect the tree if you want.

Code 21

Concept used: Standardization (Z-score Normalization)

Output



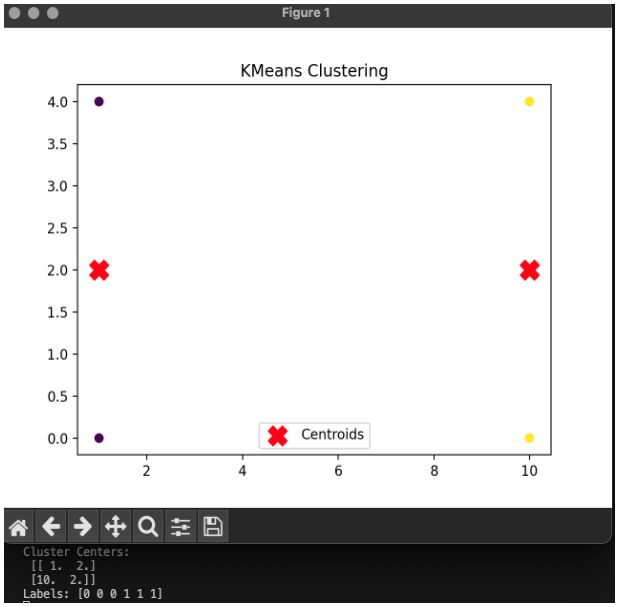
Observations

1. The original numbers are printed as a table.
2. Then the same table is changed so each column has mean 0.
3. Big numbers become small positive/negative numbers.
4. That makes different columns comparable.
5. It prints both original and changed data.

Code 22

Concept used: K-Means Clustering

Output



Observations

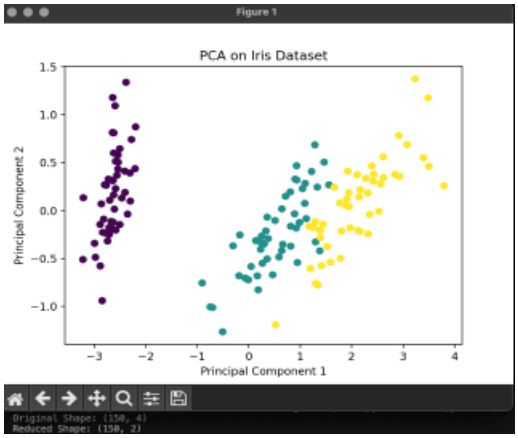
Points grouped into two clusters (left and right).

1. It prints the two center points (one near x=1, one near x=10).
2. Each data point gets a label (0 or 1).
3. A plot would color the groups and show red X for centers.
4. You can see how the points are split into two groups.

Code 23

Concept used: Principal Component Analysis (PCA) for dimensionality reduction

Output



Observations

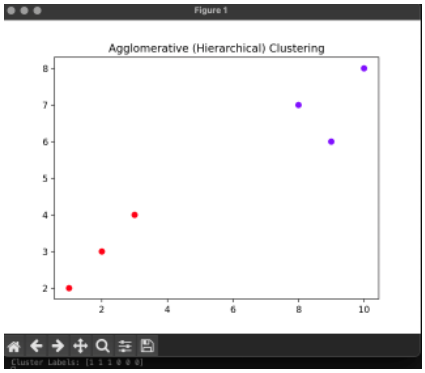
1. The iris data had 4 numbers per flower.

2. PCA squashed those into 2 numbers per flower.
3. It prints the original and new shapes.
4. A scatter plot shows the flowers in the new 2-number view.
5. You can see groups by color in that 2D view.

Code 24

Concept used: Agglomerative (Hierarchical) Clustering for unsupervised learning

Output



Observations

1. The points were grouped into two sets.
2. The program printed labels like 0 or 1 for each point.
3. The first three belong to group 0, the last three to group 1.
4. A plot would color the two groups differently.
5. You can see nearby points grouped together.