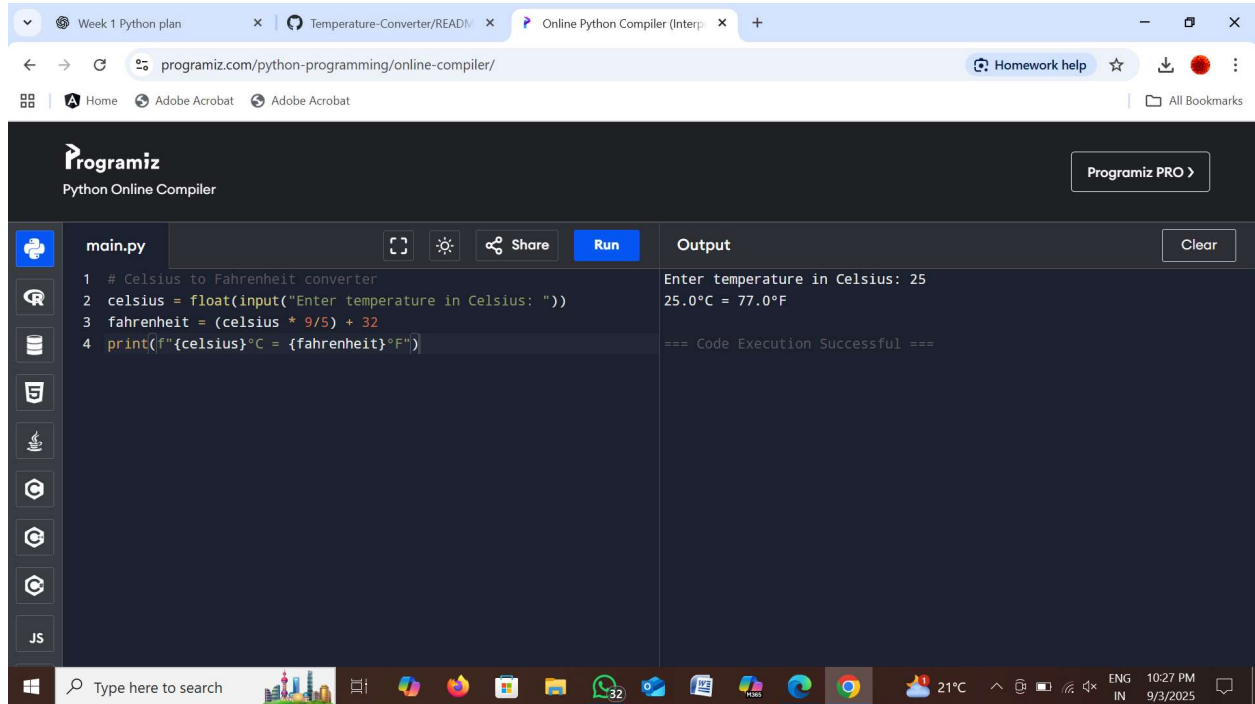


# Python Programming

## Week 1: Introduction to Python Programming

Hands-On: Write basic Python programs (e.g., temperature converter, calculator).



The screenshot shows the Programiz Python Online Compiler interface. The browser tabs include "Week 1 Python plan", "Temperature-Converter/README", and "Online Python Compiler (Interp)". The address bar shows "programiz.com/python-programming/online-compiler/". The compiler interface has a dark theme. On the left, a sidebar contains icons for Python, JavaScript, and other languages. The main editor area shows a file named "main.py" with the following Python code:

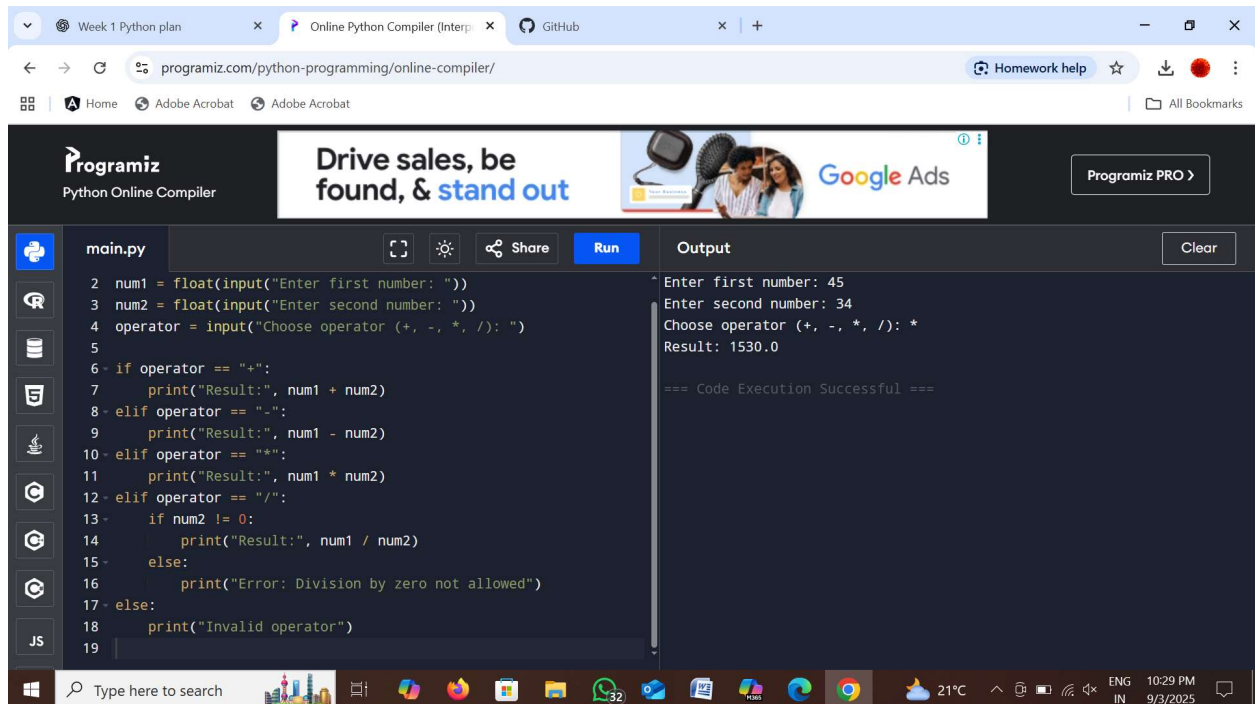
```
1 # Celsius to Fahrenheit converter
2 celsius = float(input("Enter temperature in Celsius: "))
3 fahrenheit = (celsius * 9/5) + 32
4 print(f"{celsius}°C = {fahrenheit}°F")
```

Buttons for "Run", "Share", and "Clear" are visible. The "Output" pane on the right shows the execution result:

```
Enter temperature in Celsius: 25
25.0°C = 77.0°F

=== Code Execution Successful ===
```

The Windows taskbar at the bottom shows the search bar, task view, and various application icons. The system tray indicates a temperature of 21°C and the date/time as 10:27 PM on 9/3/2025.



The screenshot shows the Programiz Python Online Compiler interface with a Google Ads banner at the top that reads "Drive sales, be found, & stand out". The browser tabs include "Week 1 Python plan", "Online Python Compiler (Interp)", and "GitHub". The address bar shows "programiz.com/python-programming/online-compiler/". The compiler interface is the same as the previous screenshot. The main editor area shows a file named "main.py" with the following Python code:

```
2 num1 = float(input("Enter first number: "))
3 num2 = float(input("Enter second number: "))
4 operator = input("Choose operator (+, -, *, /): ")
5
6 if operator == "+":
7     print("Result:", num1 + num2)
8 elif operator == "-":
9     print("Result:", num1 - num2)
10 elif operator == "*":
11     print("Result:", num1 * num2)
12 elif operator == "/":
13     if num2 != 0:
14         print("Result:", num1 / num2)
15     else:
16         print("Error: Division by zero not allowed")
17 else:
18     print("Invalid operator")
19
```

The "Output" pane on the right shows the execution result:

```
Enter first number: 45
Enter second number: 34
Choose operator (+, -, *, /): *
Result: 1530.0

=== Code Execution Successful ===
```

The Windows taskbar at the bottom is identical to the previous screenshot, showing the search bar, task view, and various application icons. The system tray indicates a temperature of 21°C and the date/time as 10:29 PM on 9/3/2025.

Client Project: Create a basic data processing script (e.g., calculating the average temperature).

The screenshot displays the Programiz Online Python Compiler interface. The browser address bar shows the URL `programiz.com/python-programming/online-compiler/`. The page header includes the Programiz logo, a navigation menu with 'Home', 'Adobe Acrobat', and 'All Bookmarks', and a 'Homework help' button. A banner for 'Great Scott! Actor Christopher Lloyd Is Selling His California Sanctuary' is visible. The main content area is divided into two panels: a code editor on the left and an output panel on the right. The code editor shows a Python script named `main.py` that calculates the average temperature over 7 days. The output panel displays the results of the script execution, including the input temperatures for each day and the calculated average.

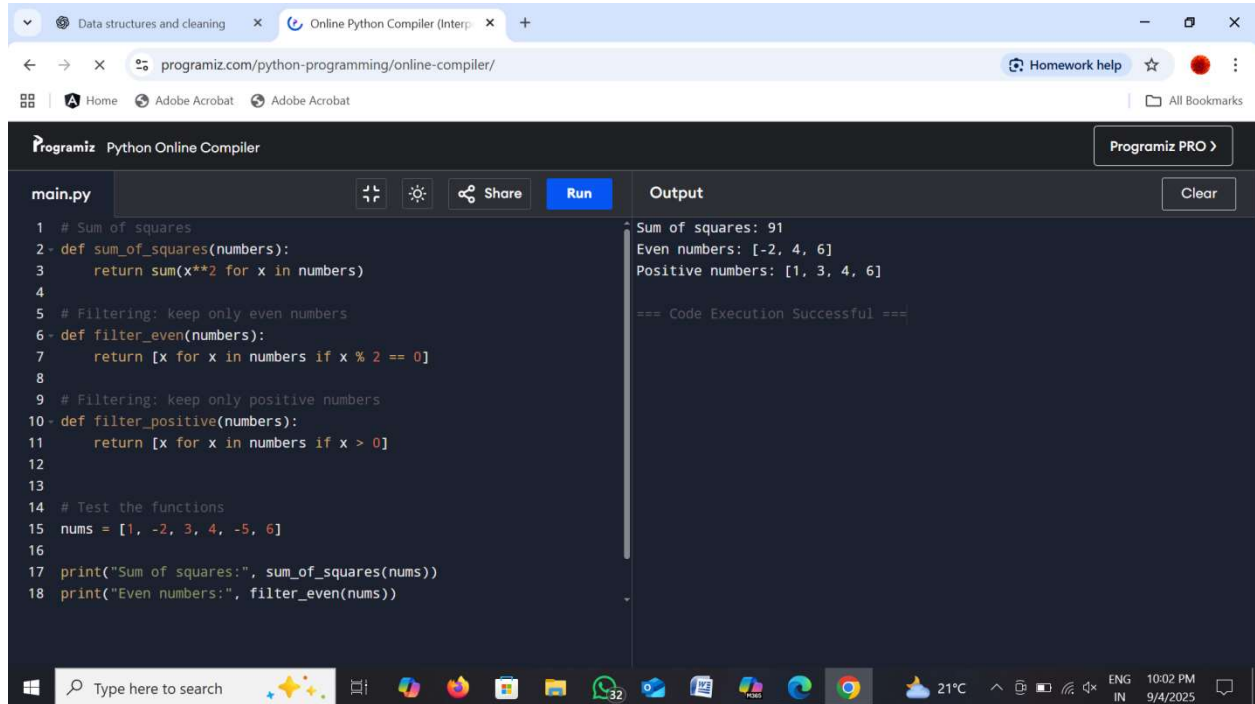
```
1 # Average Temperature Calculator
2 temperatures = []
3
4 for i in range(7):
5     temp = float(input(f"Enter temperature for day {i+1}: "))
6     temperatures.append(temp)
7
8 average_temp = sum(temperatures) / len(temperatures)
9 print("Temperatures:", temperatures)
10 print("Average Temperature of the Week:", average_temp)
11
```

Output:

```
Enter temperature for day 1: 27
Enter temperature for day 2: 25
Enter temperature for day 3: 26
Enter temperature for day 4: 25
Enter temperature for day 5: 26
Enter temperature for day 6: 28
Enter temperature for day 7: 27
Temperatures: [27.0, 25.0, 26.0, 25.0, 26.0, 28.0, 27.0]
Average Temperature of the Week: 26.285714285714285
=== Code Execution Successful ===
```

## Week 2: Data Structures and Functions

Hands-On: Work with data structures and write functions for data transformations (e.g., sum of squares, filtering).



The screenshot shows a web browser window with the URL `programiz.com/python-programming/online-compiler/`. The browser tabs include "Data structures and cleaning" and "Online Python Compiler (Interp...". The browser's address bar shows the URL, and the bookmarks bar includes "Home", "Adobe Acrobat", and "All Bookmarks".

The Programiz Python Online Compiler interface is displayed. The left pane shows the code in `main.py`:

```
1 # Sum of squares
2 def sum_of_squares(numbers):
3     return sum(x**2 for x in numbers)
4
5 # Filtering: keep only even numbers
6 def filter_even(numbers):
7     return [x for x in numbers if x % 2 == 0]
8
9 # Filtering: keep only positive numbers
10 def filter_positive(numbers):
11     return [x for x in numbers if x > 0]
12
13
14 # Test the functions
15 nums = [1, -2, 3, 4, -5, 6]
16
17 print("Sum of squares:", sum_of_squares(nums))
18 print("Even numbers:", filter_even(nums))
```

The right pane shows the output:

```
Sum of squares: 91
Even numbers: [-2, 4, 6]
Positive numbers: [1, 3, 4, 6]

=== Code Execution Successful ===
```

The bottom of the image shows a Windows taskbar with a search bar, task view button, and several application icons. The system tray on the right shows the temperature as 21°C, the time as 10:02 PM, and the date as 9/4/2025.

Client Project: Write a script for data cleaning (e.g., remove duplicates, filter data).

The screenshot shows the Programiz Python Online Compiler interface. The left pane contains a Python script named `main.py` that uses pandas to clean a dataset. The right pane shows the output of the script, which includes the original data, the cleaned data (with duplicates removed and filtered), and a success message.

**main.py**

```
1 import pandas as pd
2 # Sample dataset (you can replace with CSV or Excel file)
3 data = {
4     "Name": ["Alice", "Bob", "Charlie", "Alice", "David", "Bob"],
5     "Age": [25, 30, 35, 25, 40, 30],
6     "City": ["NY", "LA", "SF", "NY", "LA", "LA"]
7 }
8 df = pd.DataFrame(data)
9 print("Original Data:")
10 print(df)
11 # Step 1: Remove duplicates
12 df_clean = df.drop_duplicates()
13 # Step 2: Filter (e.g., only Age > 30)
14 df_clean = df_clean[df_clean["Age"] > 30]
15 print("\nCleaned Data:")
16 print(df_clean)
```

**Output**

Original Data:

	Name	Age	City
0	Alice	25	NY
1	Bob	30	LA
2	Charlie	35	SF
3	Alice	25	NY
4	David	40	LA
5	Bob	30	LA

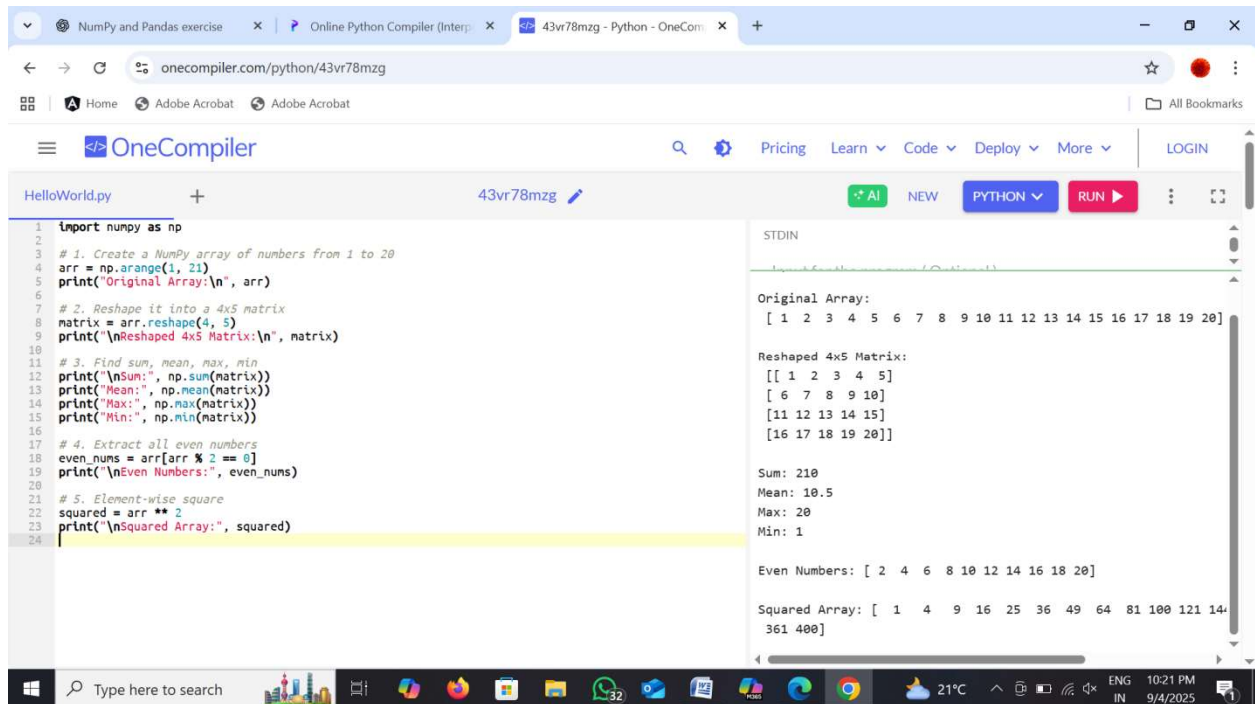
Cleaned Data:

	Name	Age	City
2	Charlie	35	SF
4	David	40	LA

=== Code Execution Successful ===

## Week 3: NumPy and Pandas for Data

Hands-On: Perform operations with NumPy and manipulate datasets with Pandas.



The screenshot shows the OneCompiler Python IDE with a file named 'HelloWorld.py'. The code performs several NumPy operations: creating an array from 1 to 20, reshaping it into a 4x5 matrix, calculating sum, mean, max, and min, extracting even numbers, and squaring each element. The output on the right shows the original array, the reshaped matrix, the calculated statistics, the even numbers, and the squared array.

```
1 import numpy as np
2
3 # 1. Create a NumPy array of numbers from 1 to 20
4 arr = np.arange(1, 21)
5 print("Original Array:\n", arr)
6
7 # 2. Reshape it into a 4x5 matrix
8 matrix = arr.reshape(4, 5)
9 print("\nReshaped 4x5 Matrix:\n", matrix)
10
11 # 3. Find sum, mean, max, min
12 print("\nSum:", np.sum(matrix))
13 print("Mean:", np.mean(matrix))
14 print("Max:", np.max(matrix))
15 print("Min:", np.min(matrix))
16
17 # 4. Extract all even numbers
18 even_nums = arr[arr % 2 == 0]
19 print("\nEven Numbers:", even_nums)
20
21 # 5. Element-wise square
22 squared = arr ** 2
23 print("\nSquared Array:", squared)
24
```

Original Array:  
[ 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20]

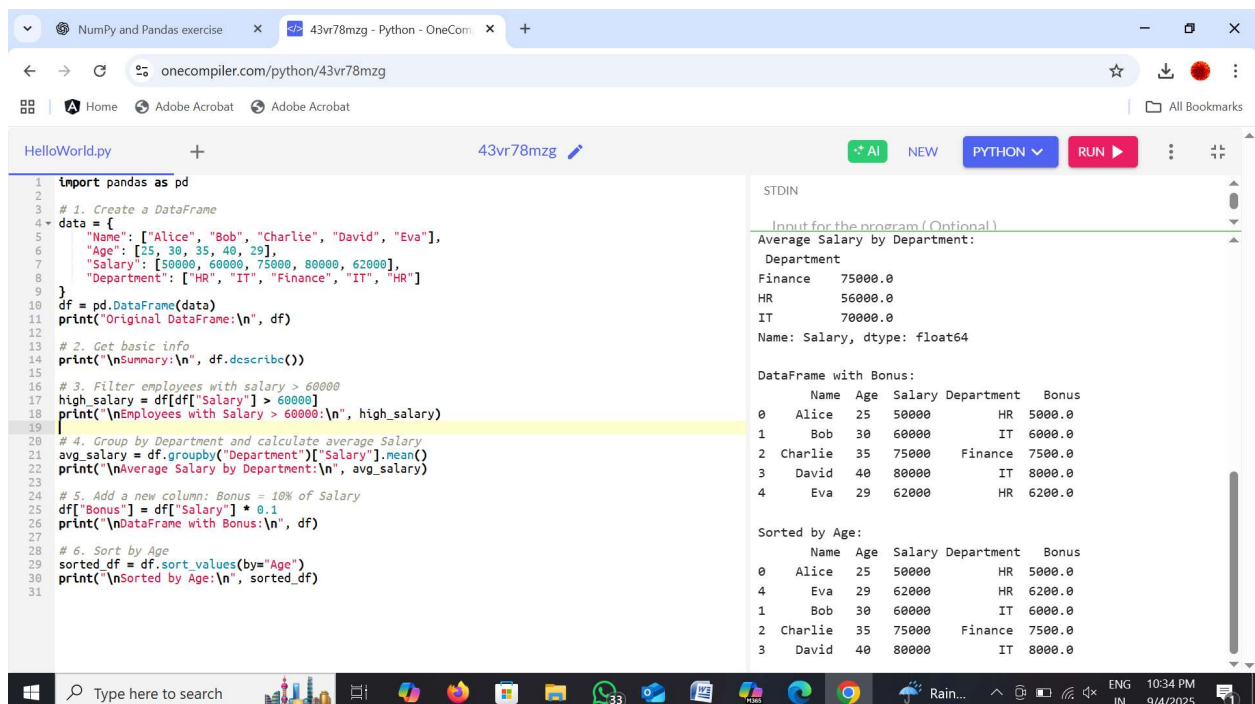
Reshaped 4x5 Matrix:  
[[ 1 2 3 4 5]  
[ 6 7 8 9 10]  
[11 12 13 14 15]  
[16 17 18 19 20]]

Sum: 210  
Mean: 10.5  
Max: 20  
Min: 1

Even Numbers: [ 2 4 6 8 10 12 14 16 18 20]

Squared Array: [ 1 4 9 16 25 36 49 64 81 100 121 144 169 196 225 256 289 324 361 400]

Client Project: Clean and aggregate a dataset (e.g., remove missing values, calculate averages).



The screenshot shows the OneCompiler Python IDE with a file named 'HelloWorld.py'. The code performs several Pandas operations: creating a DataFrame from employee data, getting basic info, filtering employees with salary > 60000, grouping by department to calculate average salary, adding a new column 'Bonus' (10% of salary), and sorting by age. The output on the right shows the average salary by department, the DataFrame with bonus, and the sorted DataFrame.

```
1 import pandas as pd
2
3 # 1. Create a DataFrame
4 data = {
5     "Name": ["Alice", "Bob", "Charlie", "David", "Eva"],
6     "Age": [25, 30, 35, 40, 29],
7     "Salary": [50000, 60000, 75000, 80000, 62000],
8     "Department": ["HR", "IT", "Finance", "IT", "HR"]
9 }
10 df = pd.DataFrame(data)
11 print("Original DataFrame:\n", df)
12
13 # 2. Get basic info
14 print("\nSummary:\n", df.describe())
15
16 # 3. Filter employees with salary > 60000
17 high_salary = df[df["Salary"] > 60000]
18 print("\nEmployees with Salary > 60000:\n", high_salary)
19
20 # 4. Group by Department and calculate average salary
21 avg_salary = df.groupby("Department")["Salary"].mean()
22 print("\nAverage Salary by Department:\n", avg_salary)
23
24 # 5. Add a new column: Bonus = 10% of Salary
25 df["Bonus"] = df["Salary"] * 0.1
26 print("\nDataFrame with Bonus:\n", df)
27
28 # 6. Sort by Age
29 sorted_df = df.sort_values(by="Age")
30 print("\nSorted by Age:\n", sorted_df)
31
```

Average Salary by Department:

Department	Average Salary
Finance	75000.0
HR	56000.0
IT	70000.0

DataFrame with Bonus:

	Name	Age	Salary	Department	Bonus
0	Alice	25	50000	HR	5000.0
1	Bob	30	60000	IT	6000.0
2	Charlie	35	75000	Finance	7500.0
3	David	40	80000	IT	8000.0
4	Eva	29	62000	HR	6200.0

Sorted by Age:

	Name	Age	Salary	Department	Bonus
0	Alice	25	50000	HR	5000.0
4	Eva	29	62000	HR	6200.0
1	Bob	30	60000	IT	6000.0
2	Charlie	35	75000	Finance	7500.0
3	David	40	80000	IT	8000.0

Output:

Original DataFrame:

	Name	Age	Salary	Department
0	Alice	25	50000	HR
1	Bob	30	60000	IT
2	Charlie	35	75000	Finance
3	David	40	80000	IT
4	Eva	29	62000	HR

Summary:

	Age	Salary
count	5.000000	5.000000
mean	31.800000	65400.000000
std	5.80517	12074.767078
min	25.000000	50000.000000
25%	29.000000	60000.000000
50%	30.000000	62000.000000
75%	35.000000	75000.000000
max	40.000000	80000.000000

Employees with Salary > 60000:

	Name	Age	Salary	Department
2	Charlie	35	75000	Finance
3	David	40	80000	IT
4	Eva	29	62000	HR

Average Salary by Department:

Department	
Finance	75000.0
HR	56000.0
IT	70000.0

Name: Salary, dtype: float64

DataFrame with Bonus:

	Name	Age	Salary	Department	Bonus
0	Alice	25	50000	HR	5000.0
1	Bob	30	60000	IT	6000.0
2	Charlie	35	75000	Finance	7500.0
3	David	40	80000	IT	8000.0
4	Eva	29	62000	HR	6200.0

Sorted by Age:

	Name	Age	Salary	Department	Bonus
0	Alice	25	50000	HR	5000.0
4	Eva	29	62000	HR	6200.0
1	Bob	30	60000	IT	6000.0
2	Charlie	35	75000	Finance	7500.0
3	David	40	80000	IT	8000.0

## Week 4: Data Visualization with Matplotlib and Seaborn

**Hands-On: Create visualizations for dataset analysis.**

```
import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# Sample dataset

data = {

    "Age": [25, 30, 35, 40, 29, 50, 45, 33],

    "Salary": [50000, 60000, 75000, 80000, 62000, 95000, 85000, 70000],

    "Department": ["HR", "IT", "Finance", "IT", "HR", "Finance", "IT", "Finance"]

}

df = pd.DataFrame(data)

# 1. Histogram of Age

plt.figure(figsize=(6,4))

plt.hist(df["Age"], bins=5, edgecolor="black")

plt.title("Distribution of Age")

plt.xlabel("Age")

plt.ylabel("Count")

plt.show()

# 2. Scatter Plot (Age vs Salary)

plt.figure(figsize=(6,4))
```

```
plt.scatter(df["Age"], df["Salary"], c="blue")
```

```
plt.title("Age vs Salary")
```

```
plt.xlabel("Age")
```

```
plt.ylabel("Salary")
```

```
plt.show()
```

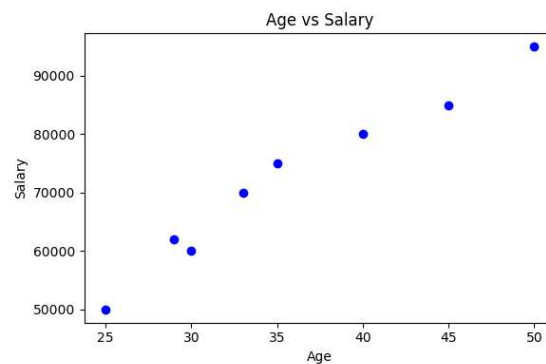
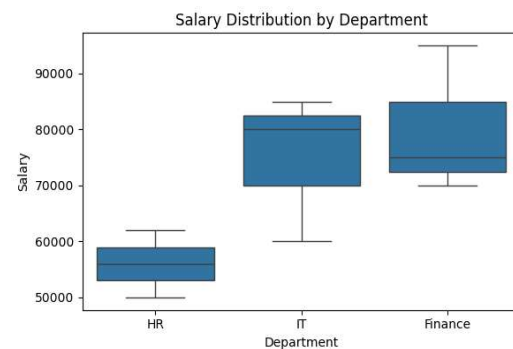
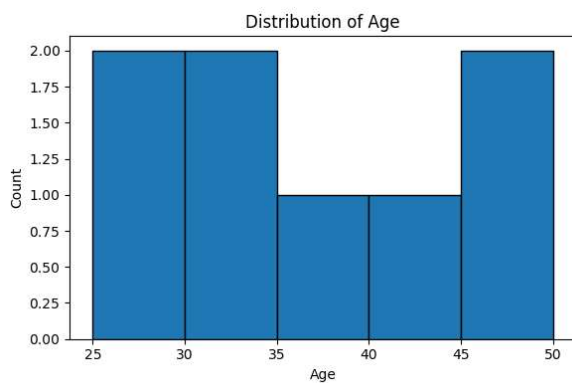
### # 3. Boxplot of Salary by Department

```
plt.figure(figsize=(6,4))
```

```
sns.boxplot(x="Department", y="Salary", data=df)
```

```
plt.title("Salary Distribution by Department")
```

```
plt.show()
```





**Client Project: Create a dashboard for visualizing relationships between features in a dataset (e.g., scatter plots, histograms).**

```
import pandas as pd

import plotly.express as px

import plotly.io as pio


# Sample dataset

data = {

    "Age": [25, 30, 35, 40, 29, 50, 45, 33],

    "Salary": [50000, 60000, 75000, 80000, 62000, 95000, 85000, 70000],

    "Department": ["HR", "IT", "Finance", "IT", "HR", "Finance", "IT", "Finance"]

}

df = pd.DataFrame(data)


# Scatter plot (Age vs Salary with Department color)

fig1 = px.scatter(df, x="Age", y="Salary", color="Department", size="Salary",

                  title="Age vs Salary by Department")

fig1.show()


# Histogram of Salary

fig2 = px.histogram(df, x="Salary", nbins=6, color="Department",

                    title="Salary Distribution")

fig2.show()
```

## # Boxplot Salary by Department

```
fig3 = px.box(df, x="Department", y="Salary", color="Department",  
              title="Salary Distribution by Department")
```

```
fig3.show()
```

