Month 1 – Python Basics and Data Manipulation

Week 1: Introduction to Python Programming

This week I got introduced to Python basics. I learned about variables and different data types like integers, floats, strings, and booleans. I also practiced how to take user input and print outputs.

I understood how operators work in Python (arithmetic, comparison, and logical). Then I studied conditional statements like if-else and also loops (for and while) which help in repeating tasks.

For practice, I created small programs like a temperature converter and a simple calculator. My project for the week was an Average Temperature Calculator which takes 7 days' input and gives the weekly average.

Overall, I feel more comfortable with the syntax of Python now and can write small scripts on my own.

Client Project: Average Temperature Calculator

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

Sample Code

```
# Average Temperature Calculator
# Take input for 7 days
temperatures = []
for i in range(7):
    temp = float(input(f"Enter temperature for day {i+1}: "))
    temperatures.append(temp)

# Calculate average
average_temp = sum(temperatures) / len(temperatures)

print("\nTemperatures entered:", temperatures)
print("Average Temperature of the week:", round(average_temp, 2))
```

Expected Output

```
Enter temperature for day 1: 30
Enter temperature for day 2: 32
Enter temperature for day 3: 28
Enter temperature for day 4: 31
Enter temperature for day 5: 29
Enter temperature for day 6: 33
Enter temperature for day 7: 30

Temperatures entered: [30.0, 32.0, 28.0, 31.0, 29.0, 33.0, 30.0]
Average Temperature of the week: 30.43
```

Week 2: Data Structures and Functions

This week focused on Python data structures. I learned about lists, tuples, dictionaries, and sets and how they are used to store and manage data. Lists are flexible, tuples are fixed, dictionaries use key-value pairs, and sets automatically remove duplicates.

I also studied functions in Python, including normal functions, lambda functions, and recursion. I got introduced to list comprehensions, which make writing loops shorter and cleaner.

For practice, I wrote programs to find the sum of squares and filter even numbers from a list. My project for this week was a Data Cleaning Script which removed duplicates from a dataset and filtered values less than 50.

I understood the importance of data structures in organizing data and functions in reusing code.

Client Project: Data Cleaning Script

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

Sample Code

```
# Data Cleaning Script
data = [45, 67, 45, 89, 23, 67, 90, 12, 100, 23]

# Remove duplicates using set
unique_data = list(set(data))

# Filter numbers greater than or equal to 50
filtered_data = [x for x in unique_data if x >= 50]
```

```
print("Original Data:", data)
print("After Removing Duplicates:", unique_data)
print("After Filtering (>=50):", filtered_data)
```

Expected Output

```
Original Data: [45, 67, 45, 89, 23, 67, 90, 12, 100, 23]
After Removing Duplicates: [100, 67, 12, 45, 23, 90, 89]
After Filtering (>=50): [100, 67, 90, 89]
```

Week 3: NumPy and Pandas for Data Manipulation

This week was about libraries that make data handling easier. I learned about NumPy, which is mainly used for arrays and numerical operations. I practiced creating arrays, slicing them, and performing aggregate functions.

Next, I studied Pandas, which is very useful for working with tabular data. I worked with Series and DataFrames, learned indexing, and performed basic operations. I also learned about handling missing data using dropna() and fillna(). Grouping and aggregation using groupby() was also covered.

My project was about cleaning a dataset with missing values and then calculating the average marks. This made me realize how often data comes with missing values and how Pandas simplifies cleaning tasks.

Overall, I feel more confident in working with real datasets now.

Client Project: Clean Missing Values & Calculate Average

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

Sample Code

```
import pandas as pd


# Sample dataset with missing values
data = {
    "Name": ["Anu", "Rahul", "Meera", "John", "Sara"],
    "Marks": [85, None, 78, None, 92]
}


df = pd.DataFrame(data)
```

```
print("Original Data:\n", df)

# Remove rows with missing values
cleaned_df = df.dropna()

# Calculate average
average_marks = cleaned_df["Marks"].mean()

print("\nAfter Cleaning:\n", cleaned_df)
print("\nAverage Marks:", round(average_marks, 2))
```

Expected Output

```
Original Data:
        Name   Marks
0       Anu    85.0
1       Rahul  NaN
2       Meera  78.0
3       John   NaN
4       Sara   92.0

After Cleaning:
        Name   Marks
0       Anu    85.0
2       Meera  78.0
4       Sara   92.0

Average Marks: 85.0
```

Week 4: Data Visualization with Matplotlib and Seaborn

This week I focused on data visualization. I learned about Matplotlib for basic plots like line graphs, bar charts, scatter plots, and histograms. I also customized plots with titles, labels, and legends.

Then I moved to Seaborn, which is built on top of Matplotlib and provides advanced plots. I practiced histograms, boxplots, pairplots, and heatmaps. Seaborn makes the plots look more professional and easier to create.

For my project, I created a Visualization Dashboard where I used scatter plots, histograms, and a heatmap to analyze student marks.

This week showed me the importance of visualizing data to find patterns and insights. I enjoyed this part a lot because it made data more meaningful and easy to understand.

Client Project: Visualization Dashboard

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

Sample Code

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Sample dataset
data = {
    "Name": ["Anu", "Rahul", "Meera", "John", "Sara", "Akhil", "Priya"],
    "Marks": [85, 67, 78, 90, 92, 55, 73],
    "Subject": ["Math", "Math", "Science", "Science", "Math", "Science", "Math"]
}

df = pd.DataFrame(data)

# Scatter plot
plt.figure(figsize=(8,5))
sns.scatterplot(x="Name", y="Marks", hue="Subject", data=df)
plt.title("Student Marks by Subject")
plt.show()

# Histogram
plt.figure(figsize=(8,5))
sns.histplot(df["Marks"], bins=5, kde=True)
```

```
plt.title("Distribution of Marks")
plt.show()

# Heatmap
plt.figure(figsize=(6,4))
sns.heatmap(df.corr(), annot=True, cmap="coolwarm")
plt.title("Correlation Heatmap")
plt.show()
```

Expected Plots

- Scatter plot (marks vs. students, color-coded by subject)
- Histogram (distribution of marks)
- Heatmap (correlation of marks with other numerical fields)