Day 1: Introduction to Python for Data Science

Objective:

- Learn **Python basics**, **NumPy**, and **Pandas** for Data Science.
- Perform hands-on exercises using **Google Colab**.
- Understand the real-world application of each concept.

1️⃣ Google Colab Setup

💼 **Scenario:**
You have joined a **data science team** that uses Google Colab for analysis. Your first task is to **set up the environment** and test a simple script.

1.1 Open Google Colab and Create a Notebook

- Go to Google Colab
- Click **File** → **New Notebook**
- Rename the notebook (e.g., `Day1_Python_for_DS`).

1.2 Print a Simple Statement

📌 **Activity:** Writing and executing a basic Python script.

```
print("Hello, Excited to start Data Science.")
```

🔹 **Explanation:** This prints a simple message to verify that our Python environment is working.

🔹 **Output:**

```
Hello, Excited to start Data Science.
```

2️⃣ Python Basics

💼 **Scenario:**
You're building a **customer profile system** for an e-commerce company. You need to store customer details, calculate discounts, and analyze data.

2.1 Variable Assignment (Storing Customer Data)

📌 **Activity:** Creating and storing customer information using Python variables.

```
customer_name = "John Doe"
customer_age = 28
customer_balance = 120.75

print("Customer:", customer_name)
print("Age:", customer_age)
print("Balance:", customer_balance)
```

🔹 **Explanation:**

- `customer_name` holds a **string** value.
- `customer_age` holds an **integer**.
- `customer_balance` holds a **floating-point** number.

🔹 **Output:**

```
Customer: John Doe
Age: 28
Balance: 120.75
```

2.2 Arithmetic Operations (Applying Discount)

📌 **Activity:** Performing basic mathematical operations to calculate a product discount.

```
product_price = 200
discount = product_price * 0.10  # 10% discount
final_price = product_price - discount
```

```
print("Final Price after Discount:", final_price)
```

🔹 **Explanation:**

- `product_price` stores the original price.
- `discount` calculates **10%** of the price.
- `final_price` computes the new price **after discount**.

🔹 **Output:**

```
Final Price after Discount: 180.0
```

3️⃣ NumPy Basics

💼 **Scenario:**
You are analyzing the **daily sales** of an online store over a week using NumPy.

3.1 Create NumPy Arrays (Sales Data)

📌 **Activity:** Storing **daily sales data** in a NumPy array.

```
import numpy as np

sales = np.array([150, 200, 250, 300, 400, 350, 500])  # Sales for
each day
print("Sales Data:", sales)
```

🔹 **Explanation:**

- The `np.array()` function creates an **array** that holds daily sales figures.

🔹 **Output:**

```
Sales Data: [150 200 250 300 400 350 500]
```

3.2 Statistical Analysis (Sales Performance)

📌 **Activity:** Calculating **mean, maximum, and minimum** sales.

```
print("Average Sales:", np.mean(sales))
print("Highest Sale:", np.max(sales))
print("Lowest Sale:", np.min(sales))
```

🔹 **Explanation:**

- `np.mean(sales)`: Calculates the **average** sales.
- `np.max(sales)`: Finds the **highest** sales figure.
- `np.min(sales)`: Identifies the **lowest** sales figure.

🔹 **Output:**

```
Average Sales: 307.14
Highest Sale: 500
Lowest Sale: 150
```

4️⃣ Pandas Basics

💼 **Scenario:**
Your company stores customer purchases in a **CSV file**, and you need to analyze it using Pandas.

4.1 Create a DataFrame (Customer Transactions)

📌 **Activity:** Creating a **table-like structure** using Pandas.

```
import pandas as pd

data = {
    "Customer": ["Alice", "Bob", "Charlie"],
    "Age": [25, 30, 35],
    "Amount Spent": [120, 200, 150]
}

df = pd.DataFrame(data)
print(df)
```

🔹 **Explanation:**

- `data` dictionary holds customer details.
- `pd.DataFrame(data)` converts the dictionary into a **structured table**.

🔹 **Output:**

```
  Customer  Age  Amount Spent
0    Alice   25           120
1      Bob   30           200
2  Charlie   35           150
```

4.2 Load and View a CSV File

📌 **Activity:** Uploading and reading a CSV file in Pandas.

```
from google.colab import files
uploaded = files.upload()  # Upload your CSV file

df = pd.read_csv("customer_data.csv")  # Replace with your file name
df.head()
```

🔹 **Explanation:**

- `files.upload()` opens a file picker to upload a CSV.
- `pd.read_csv()` loads the file into a Pandas DataFrame.

5️⃣ Data Manipulation with Pandas

💼 **Scenario:**
You need to **analyze** customer spending habits.

5.1 Filter High-Spending Customers

📌 **Activity:** Selecting customers who spent more than **$150**.

```
high_spenders = df[df["Amount Spent"] > 150]
print(high_spenders)
```

🔹 **Explanation:**

- `df[df["Amount Spent"] > 150]` filters rows where spending is over **$150**.

5.2 Sorting Customers by Spending

📌 **Activity:** Sorting customers from **highest to lowest spending**.

```
df_sorted = df.sort_values(by="Amount Spent", ascending=False)
print(df_sorted)
```

🔹 **Explanation:**

- `sort_values(by="Amount Spent", ascending=False)` arranges data from highest to lowest spender.

5.3 Add a New Column (Loyalty Points Calculation)

📌 **Activity:** Adding **Loyalty Points** based on spending.

```
df["Loyalty Points"] = df["Amount Spent"] // 10
print(df)
```

🔹 **Explanation:**

- `df["Loyalty Points"] = df["Amount Spent"] // 10` assigns **1 point per $10 spent**.

6️⃣ Saving Processed Data

💼 **Scenario:**
Your processed data must be **saved and shared** with the marketing team.

📌 **Activity:** Exporting cleaned data as a CSV file.

```
df.to_csv("cleaned_customer_data.csv", index=False)
files.download("cleaned_customer_data.csv")  # Download the file
```

🔹 **Explanation:**

- `to_csv()` saves the DataFrame as a **CSV file**.
- `files.download()` lets you **download** the file to your computer.

Summary

✔ Set up **Google Colab**
✔ Used **Python** for basic operations
✔ Analyzed sales data with **NumPy**
✔ Processed customer data using **Pandas**
✔ Filtered and saved insights