Assignment: Hands-On with ML Essentials (NumPy, Pandas, Matplotlib)

This notebook contains 5 questions to build your skills with Python's core data science libraries. The goal is to practice common operations used in data analysis and machine learning.

Library Imports

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Question 1

Instructions: Use NumPy to perform a basic analysis of numerical data.

- **Task**: Create a 1D NumPy array representing daily sales figures and calculate basic statistics.
- **Hints**:
    - Use `np.array()` to create the array.
    - NumPy arrays have built-in methods like `.mean()`, `.max()`, and `.min()`.
    - The `.size` attribute gives the total number of elements.

```
import numpy as np

# Step 1: Create a 1D NumPy array (daily sales figures)
sales = np.array([120, 150, 100, 130, 170, 160, 140])

print("Daily Sales Data:", sales)

# Step 2: Calculate basic statistics
average_sales = sales.mean()   # Mean (average)
max_sales = sales.max()        # Maximum value
min_sales = sales.min()        # Minimum value
total_days = sales.size        # Number of days (elements)

# Step 3: Display the results
print("\nSales Analysis:")
print("Average sales:", average_sales)
print("Highest sales:", max_sales)
print("Lowest sales:", min_sales)
print("Total number of days:", total_days)
```

```
Daily Sales Data: [120 150 100 130 170 160 140]

Sales Analysis:
Average sales: 138.57142857142858
Highest sales: 170
Lowest sales: 100
Total number of days: 7
```

Question 2 : Creating a Student Roster

Instructions: Use the Pandas library to create a simple, structured table of student information.

- **Task**: Create a Pandas DataFrame from a Python dictionary.
- **Hints**:
    - A DataFrame can be created from a dictionary where keys become column names.
    - Use `pd.DataFrame()` to perform the conversion.
    - Use `.head()` to view the first few rows and `.shape` to see the dimensions.

```
import pandas as pd

# Step 1: Create a Python dictionary
student_data = {
    "Name": ["Alice", "Bob", "Charlie", "David", "Emma"],
    "Name": ["Alice", "Bob", "Charlie", "David", "Emma"],
    "Age": [20, 21, 19, 22, 20],
    "Grade": ["A", "B", "A", "C", "B"],
    "City": ["New York", "Los Angeles", "Chicago", "Houston", "Phoenix"]
}

# Step 2: Convert the dictionary into a DataFrame
df = pd.DataFrame(student_data)

# Step 3: Display the first few rows and DataFrame info
print("Student DataFrame:")
print(df.head())        # View the first few rows
print("\nDataFrame Shape (rows, columns):", df.shape)
```

```
Student DataFrame:
      Name  Age Grade         City
0    Alice   20     A     New York
1      Bob   21     B  Los Angeles
2  Charlie   19     A      Chicago
3    David   22     C      Houston
4     Emma   20     B      Phoenix

DataFrame Shape (rows, columns): (5, 4)
```

Question 3: Visualizing Monthly Profits

Instructions: Use Matplotlib to create a simple bar graph showing the profit for each month.

- **Task**: Create a bar graph from pre-defined lists of data.
- **Hints**:
  - Use `plt.bar()` to create the chart.
  - Use `plt.title()`, `plt.xlabel()`, and `plt.ylabel()` to add labels.

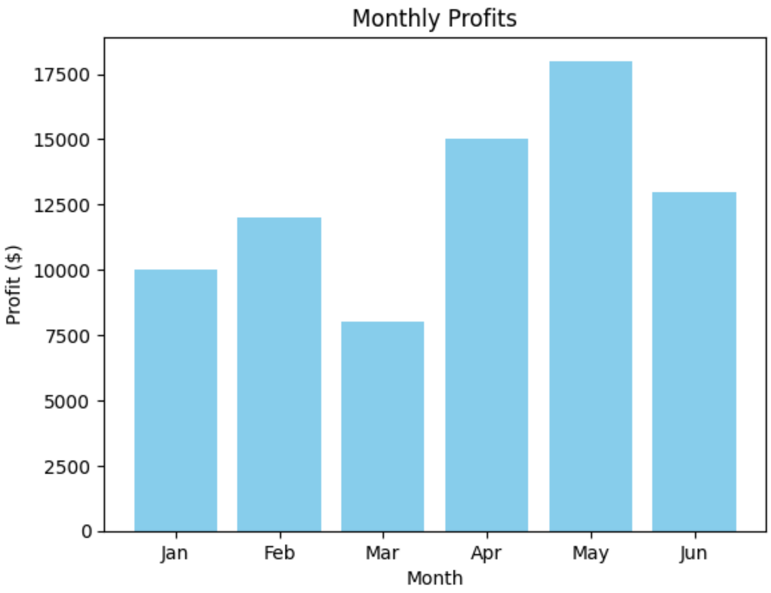
```
import matplotlib.pyplot as plt

# Step 1: Pre-defined data
months = ["Jan", "Feb", "Mar", "Apr", "May", "Jun"]
profits = [10000, 12000, 8000, 15000, 18000, 13000]

# Step 2: Create the bar graph
plt.bar(months, profits, color="skyblue")

# Step 3: Add labels and title
plt.title("Monthly Profits")
plt.xlabel("Month")
plt.ylabel("Profit ($)")

# Step 4: Display the graph
plt.show()
```



Question 4 : Manipulating a Weather Data Grid

Instructions: You are given a 2D NumPy array representing a grid of temperature readings. Manipulate this data using slicing and reshaping.

- **Task**: Slice a 2D NumPy array to extract specific parts of the grid, and then reshape it.
- **Hints**:
  - Slicing syntax is `array[row_start:row_end, col_start:col_end]`.
  - A single colon (`:`) selects all elements along that axis.

```
import numpy as np

# Step 1: Create a 2D NumPy array (temperature grid)
temperatures = np.array([
    [22, 24, 25, 23],
    [21, 23, 24, 22],
    [20, 22, 23, 21],
    [19, 21, 22, 20]
])

print("Original temperature grid:")
print(temperatures)

# Step 2: Slice the array
# Extract the top-left 2x2 part of the grid
top_left = temperatures[0:2, 0:2]
print("\nTop-left 2x2 grid:")
print(top_left)

# Step 3: Extract a specific row (e.g., 2nd row)
second_row = temperatures[1, :]
print("\nSecond row readings:")
print(second_row)

# Step 4: Reshape the entire grid into a 1D array
reshaped = temperatures.reshape(16)
print("\nReshaped 1D array:")
print(reshaped)

# Step 5: Reshape back to 2 rows and 8 columns
reshaped2D = temperatures.reshape(2, 8)
print("\nReshaped 2x8 grid:")
print(reshaped2D)
```

```
Original temperature grid:
[[22 24 25 23]
 [21 23 24 22]
 [20 22 23 21]
 [19 21 22 20]]

Top-left 2x2 grid:
[[22 24]
 [21 23]]

Second row readings:
[21 23 24 22]

Reshaped 1D array:
[22 24 25 23 21 23 24 22 20 22 23 21 19 21 22 20]

Reshaped 2x8 grid:
[[22 24 25 23 21 23 24 22]
 [20 22 23 21 19 21 22 20]]
```

Question 5 : Filtering an Inventory DataFrame

Instructions: Analyze a DataFrame of product inventory to find items that meet specific criteria.

- **Task**: Use logical operations to filter a Pandas DataFrame based on multiple conditions.
- **Hints**:
  - Use boolean indexing like `df[df['column'] < 10]`.
  - For multiple conditions, use `&` (AND) or `|` (OR) and wrap each condition in `()`.

```
import pandas as pd

# Step 1: Create a sample DataFrame
data = {
    "Product": ["Apples", "Bananas", "Cherries", "Dates", "Grapes"],
    "Quantity": [5, 25, 8, 12, 3],
    "Price": [2.5, 1.2, 3.0, 2.8, 4.0]
}

df = pd.DataFrame(data)
print("Original Inventory:")
print(df)

# Step 2: Filter the DataFrame
# Example 1: Find items with Quantity < 10
low_stock = df[df["Quantity"] < 10]
print("\nItems with Quantity < 10:")
print(low_stock)

# Example 2: Find items with Quantity < 10 AND Price > 2
low_and_expensive = df[(df["Quantity"] < 10) & (df["Price"] > 2)]
print("\nItems with Quantity < 10 AND Price > 2:")
print(low_and_expensive)

# Example 3: Find items with Quantity > 20 OR Price < 2
high_or_cheap = df[(df["Quantity"] > 20) | (df["Price"] < 2)]
print("\nItems with Quantity > 20 OR Price < 2:")
print(high_or_cheap)
```

```
Original Inventory:
    Product  Quantity  Price
0    Apples         5    2.5
1   Bananas        25    1.2
2  Cherries         8    3.0
3     Dates        12    2.8
4    Grapes         3    4.0

Items with Quantity < 10:
    Product  Quantity  Price
0    Apples         5    2.5
2  Cherries         8    3.0
4    Grapes         3    4.0

Items with Quantity < 10 AND Price > 2:
    Product  Quantity  Price
0    Apples         5    2.5
2  Cherries         8    3.0
4    Grapes         3    4.0

Items with Quantity > 20 OR Price < 2:
    Product  Quantity  Price
 1  Bananas        25    1.2
```