

NAME: MAYOORA.K

REG.NO.: 192124120

COURSE CODE/NAME: DSA0406/Fundamentals of Data Science

1. Scenario: You are working on a project that involves analyzing student performance data for a class of 32 students. The data is stored in a NumPy array named `student_scores`, where each row represents a student and each column represents a different subject. The subjects are arranged in the following order: Math, Science, English, and History. Your task is to calculate the average score for each subject and identify the subject with the highest average score.

Question: How would you use NumPy arrays to calculate the average score for each subject and determine the subject with the highest average score? Assume 4x4 matrix that stores marks of each student in given order.

CODE:

```
import numpy as np

student_scores = np.array([
    [80, 90, 85, 70],
    [75, 88, 78, 85],
    [92, 81, 86, 78],
    [88, 79, 90, 82]
])

subject_averages = np.mean(student_scores, axis=0)
max_index = np.argmax(subject_averages)
subjects = ["Math", "Science", "English", "History"]
highest_avg_subject = subjects[max_index]

print("Average scores per subject:", subject_averages)
print("Subject with highest average score:", highest_avg_subject)
```

OUTPUT:

```
File Edit Shell Debug Options Window Help
Python 3.11.4 (tags/v3.11.4:d2340ef, Jun 7 2023, 05:45:37) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\Mayu\OneDrive\Documents\Desktop\FOD\Lab Question 1.py
Average scores per subject: [83.75 84.5 84.75 78.75]
Subject with highest average score: English
>>>
```

2. Scenario: You are a data analyst working for a company that sells products online. You have been tasked with analyzing the sales data for the past month. The data is stored in a NumPy array.

Question: How would you find the average price of all the products sold in the past month?

Assume 3x3 matrix with each row representing the sales for a different product

CODE:

```
import numpy as np

sales_data = np.array([
    [10, 20, 200],
    [15, 25, 375],
    [20, 30, 600]
])

average_price = np.mean(sales_data[:, 1])

print("Average price of all products sold:", average_price)
```

OUTPUT:

```
e Edit Shell Debug Options Window Help
Python 3.11.4 (tags/v3.11.4:d2340ef, Jun 7 2023, 05:45:37) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>
= RESTART: C:\Users\Mayu\OneDrive\Documents\Desktop\FOD\Lab Question 2.py
Average price of all products sold: 25.0
>>
```

3. Scenario: You are working on a project that involves analyzing a dataset containing information about houses in a neighborhood. The dataset is stored in a CSV file, and you have imported it into a NumPy array named `house_data`. Each row of the array represents a house, and the columns contain various features such as the number of bedrooms, square footage, and sale price.

Question: Using NumPy arrays and operations, how would you find the average sale price of houses with more than four bedrooms in the neighborhood?

CODE:

```
import numpy as np

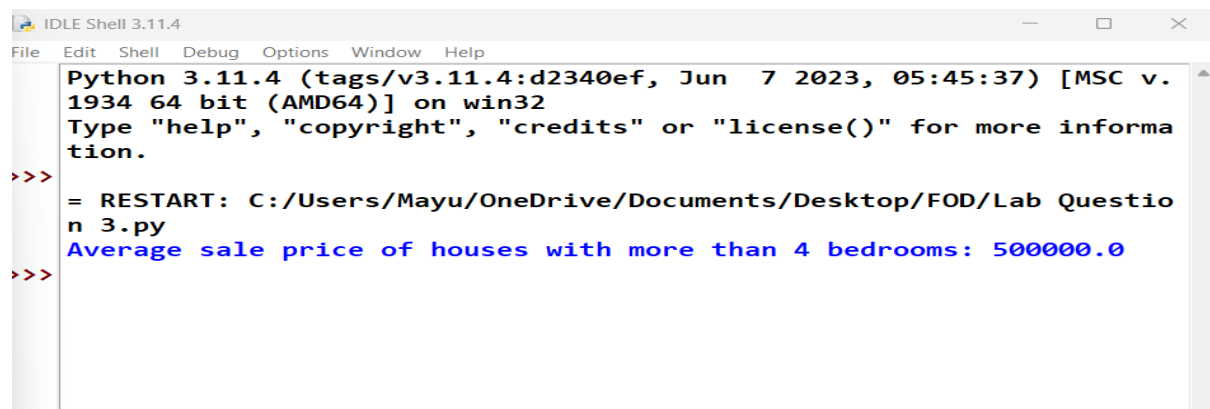
house_data = np.array([
    [3, 1500, 250000],
    [5, 2500, 400000],
    [4, 2000, 350000],
    [6, 3000, 500000],
    [3, 1800, 275000],
    [7, 3500, 600000]
])

houses_with_more_than_4_bedrooms = house_data[house_data[:, 0] > 4]

average_sale_price = np.mean(houses_with_more_than_4_bedrooms[:, 2])

print("Average sale price of houses with more than 4 bedrooms:", average_sale_price)
```

OUTPUT:



```
IDLE Shell 3.11.4
File Edit Shell Debug Options Window Help
Python 3.11.4 (tags/v3.11.4:d2340ef, Jun 7 2023, 05:45:37) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Mayu/OneDrive/Documents/Desktop/FOD/Lab Question 3.py
Average sale price of houses with more than 4 bedrooms: 500000.0
>>>
```

4. Scenario: You are working on a project that involves analyzing the sales performance of a company over the past four quarters. The quarterly sales data is stored in a NumPy array named `sales_data`, where each element represents the sales amount for a specific quarter. Your task is to calculate the total sales for the year and determine the percentage increase in sales from the first quarter to the fourth quarter.

Question: Using NumPy arrays and arithmetic operations calculate the total sales for the year and determine the percentage increase in sales from the first quarter to the fourth quarter?

CODE:

```
import numpy as np

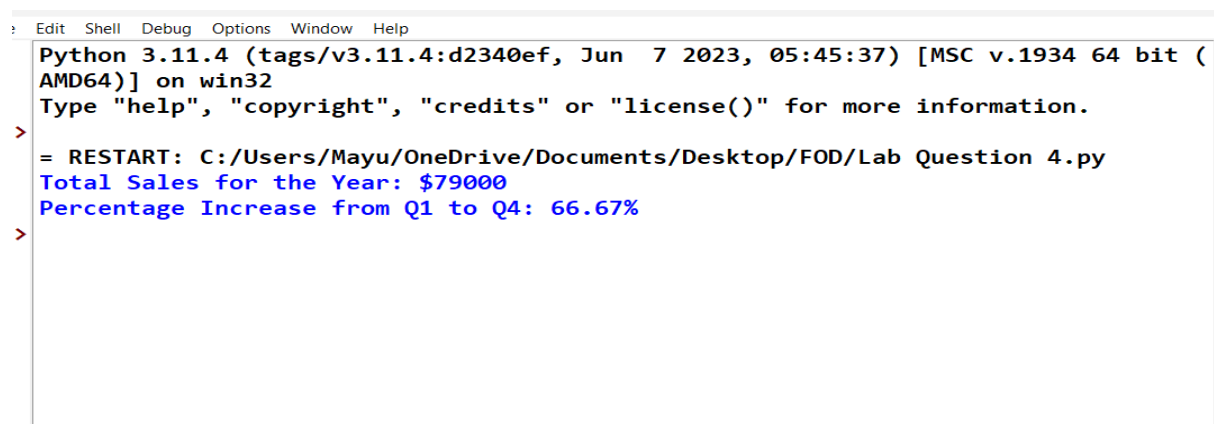
sales_data = np.array([15000, 18000, 21000, 25000])

total_sales = np.sum(sales_data)

percentage_increase = ((sales_data[3] - sales_data[0]) / sales_data[0]) * 100

print(f"Total Sales for the Year: ${total_sales}")

print(f"Percentage Increase from Q1 to Q4: {percentage_increase:.2f}%")
```

OUTPUT:

```
Python 3.11.4 (tags/v3.11.4:d2340ef, Jun 7 2023, 05:45:37) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>
= RESTART: C:/Users/Mayu/OneDrive/Documents/Desktop/FOD/Lab Question 4.py
Total Sales for the Year: $79000
Percentage Increase from Q1 to Q4: 66.67%
>
```

5. Scenario: You are a data analyst working for a car manufacturing company. As part of your analysis, you have a dataset containing information about the fuel efficiency of different car models. The dataset is stored in a NumPy array named `fuel_efficiency`, where each element represents the fuel efficiency (in miles per gallon) of a specific car model. Your task is to calculate the average fuel efficiency and determine the percentage improvement in fuel efficiency between two car models.

Question: How would you use NumPy arrays and arithmetic operations to calculate the average fuel efficiency and determine the percentage improvement in fuel efficiency between two car models?

CODE:

```
import numpy as np

fuel_efficiency = np.array([22, 25, 30, 35, 40])

average_efficiency = np.mean(fuel_efficiency)

percentage_improvement = ((fuel_efficiency[4] - fuel_efficiency[0]) / fuel_efficiency[0]) * 100

print(f"Average Fuel Efficiency: {average_efficiency:.2f} MPG")
```

```
print(f"Percentage Improvement between Model 1 and Model 5:  
{percentage_improvement:.2f}%")
```

OUTPUT:

```
File Edit Shell Debug Options Window Help
Python 3.11.4 (tags/v3.11.4:d2340ef, Jun 7 2023, 05:45:37) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Mayu/OneDrive/Documents/Desktop/FOD/Lab Question 5.py
Average Fuel Efficiency: 30.40 MPG
Percentage Improvement between Model 1 and Model 5: 81.82%
>>>
```

6. Scenario: You are a cashier at a grocery store and need to calculate the total cost of a customer's purchase, including applicable discounts and taxes. You have the item prices and quantities in separate lists, and the discount and tax rates are given as percentages. Your task is to calculate the total cost for the customer.

Question: Use arithmetic operations to calculate the total cost of a customer's purchase, including discounts and taxes, given the item prices, quantities, discount rate, and tax rate?

CODE:

```
item_prices = [10.0, 5.0, 2.5]
quantities = [2, 4, 3]
discount_rate = 10
tax_rate = 8
subtotals = [p * q for p, q in zip(item_prices, quantities)]
total_before_discount = sum(subtotals)
discount_amount = total_before_discount * (discount_rate / 100)
total_after_discount = total_before_discount - discount_amount
tax_amount = total_after_discount * (tax_rate / 100)
final_total = total_after_discount + tax_amount
print(f"Final Total: ${final_total:.2f}")
```

OUTPUT:

```

Python 3.11.4 (tags/v3.11.4:d2340ef, Jun 7 2023, 05:45:37) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>
= RESTART: C:/Users/Mayu/OneDrive/Documents/Desktop/FOD/Lab Question 6.py
Final Total: $46.17
>

```

7. Scenario: You are working as a data analyst for an e-commerce company. You have been given a dataset containing information about customer orders, stored in a Pandas DataFrame named `order_data`. The DataFrame has columns for customer ID, order date, product name, and order quantity. Your task is to analyze the data and answer specific questions about the orders.

Question: Using Pandas DataFrame operations, how would you find the following information from the `order_data` DataFrame:

1. The total number of orders made by each customer.
2. The average order quantity for each product.
3. The earliest and latest order dates in the dataset.

CODE:

```

import pandas as pd

order_data = pd.DataFrame({
    'customer_id': [101, 102, 101, 103, 102],
    'order_date': pd.to_datetime(['2023-01-10', '2023-01-15', '2023-02-01', '2023-01-20',
    '2023-03-05']),
    'product_name': ['Widget', 'Gadget', 'Widget', 'Widget', 'Gadget'],
    'order_quantity': [3, 5, 2, 4, 1]
})

orders_per_customer = order_data['customer_id'].value_counts()

avg_quantity_per_product = order_data.groupby('product_name')['order_quantity'].mean()

earliest_order = order_data['order_date'].min()

latest_order = order_data['order_date'].max()

print("1.Orders per customer:")

print(orders_per_customer)

print("\n2.Average quantity per product:")

print(avg_quantity_per_product)

```

```
print(f"\n3.Earliest order date: {earliest_order}")
```

```
print(f"Latest order date: {latest_order}")
```

OUTPUT:

```
Python 3.11.4 (tags/v3.11.4:0234061, Jun 7 2023, 05:42:37) [AMD64 v1.1554 04 010  
AMD64] on win32  
Type "help", "copyright", "credits" or "license()" for more information.  
>>  
= RESTART: C:/Users/Mayu/OneDrive/Documents/Desktop/FOD/Lab QQuestion 7.py  
1.Orders per customer:  
customer_id  
101    2  
102    2  
103    1  
Name: count, dtype: int64  
  
2.Average quantity per product:  
product_name  
Gadget    3.0  
Widget    3.0  
Name: order_quantity, dtype: float64  
  
3.Earliest order date: 2023-01-10 00:00:00  
Latest order date: 2023-03-05 00:00:00  
>>
```

8. Scenario: You are a data scientist working for a company that sells products online. You have been tasked with analyzing the sales data for the past month. The data is stored in a Pandas data frame.

Question: How would you find the top 5 products that have been sold the most in the past month?

CODE:

```
import pandas as pd  
  
sales_data = pd.DataFrame({  
    'product_name': ['A', 'B', 'A', 'C', 'B', 'D', 'E', 'A', 'C', 'D'],  
    'quantity_sold': [5, 3, 2, 7, 1, 4, 6, 3, 2, 5]  
})  
  
top_products =  
sales_data.groupby('product_name')['quantity_sold'].sum().sort_values(ascending=False).head(5)  
  
print("Top 5 products sold the most:")  
  
print(top_products)
```

OUTPUT:

```
Python 3.11.4 (tags/v3.11.4:d2340ef, Jun 7 2023, 05:45:37) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>
= RESTART: C:/Users/Mayu/OneDrive/Documents/Desktop/FOD/Lab Question 8.py
Top 5 products sold the most:
product_name
A    10
C     9
D     9
E     6
B     4
Name: quantity_sold, dtype: int64
>
```

9. Scenario: You work for a real estate agency and have been given a dataset containing information about properties for sale. The dataset is stored in a Pandas DataFrame named `property_data`. The DataFrame has columns for property ID, location, number of bedrooms, area in square feet, and listing price. Your task is to analyze the data and answer specific questions about the properties.

Question: Using Pandas DataFrame operations, how would you find the following information from the `property_data` DataFrame:

1. The average listing price of properties in each location.
2. The number of properties with more than four bedrooms.
3. The property with the largest area.

CODE:

```
import pandas as pd

property_data = pd.DataFrame({
    'property_id': [1, 2, 3, 4, 5],
    'location': ['Downtown', 'Suburb', 'Downtown', 'Suburb', 'Rural'],
    'bedrooms': [3, 5, 4, 6, 2],
    'area_sqft': [1500, 2500, 1800, 3000, 1200],
    'listing_price': [300000, 450000, 350000, 500000, 200000]
})

avg_price_by_location = property_data.groupby('location')['listing_price'].mean()
num_properties_gt_4_bedrooms = property_data[property_data['bedrooms'] > 4].shape[0]
largest_area_property = property_data.loc[property_data['area_sqft'].idxmax()]
```



```

print("1.Average listing price by location:")

print(avg_price_by_location)

print(f"\n2.Number of properties with more than 4 bedrooms:
{num_properties_gt_4_bedrooms}")

print("\n3.Property with the largest area:")

print(largest_area_property)

```

OUTPUT:

```

Python 3.11.4 (tags/v3.11.4:d2340ef, Jun 7 2023, 05:45:37) [MSC v.1934 64 bit
AMD64] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>
= RESTART: C:/Users/Mayu/OneDrive/Documents/Desktop/FOD/Lab Question 9.py
1.Average listing price by location:
location
Downtown      325000.0
Rural          200000.0
Suburb         475000.0
Name: listing_price, dtype: float64

2.Number of properties with more than 4 bedrooms: 2

3.Property with the largest area:
property_id    4
location       Suburb
bedrooms       6
area_sqft      3000
listing_price  500000
Name: 3, dtype: object
>

```

10. Scenario: You are working on a data visualization project and need to create basic plots using Matplotlib. You have a dataset containing the monthly sales data for a company, including the month and corresponding sales values. Your task is to develop a Python program that generates line plots and bar plots to visualize the sales data.

Question:

1. How would you develop a Python program to create a line plot of the monthly sales data?
- 2: How would you develop a Python program to create a bar plot of the monthly sales data?

CODE:

```

import matplotlib.pyplot as plt

months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun']

sales = [1000, 1200, 1100, 1500, 1700, 1600]

```

```
plt.figure(figsize=(10, 4))  
plt.subplot(1, 2, 1)  
plt.plot(months, sales, marker='o', linestyle='-', color='b')  
plt.title('Monthly Sales (Line Plot)')  
plt.xlabel('Month')  
plt.ylabel('Sales')  
plt.subplot(1, 2, 2)  
plt.bar(months, sales, color='pink')  
plt.title('Monthly Sales (Bar Plot)')  
plt.xlabel('Month')  
plt.ylabel('Sales')  
plt.tight_layout()  
plt.show()
```

OUTPUT:

