**ASSIGNMENT 1 SOLUTIONS**

**QUESTION 1 :**

# Step 1 : Define the list

numbers = [15, 8, 22, 7, 31, 4, 17]

# Step 2 : Print all even numbers from the list

print("Even numbers in the list:")

for num in numbers:

if num % 2 == 0:

print(num)

# Step 3 : Create a new list with squares of odd numbers

odd\_squares = []

for num in numbers:

if num % 2 != 0:

odd\_squares.append(num \*\* 2)

# Step 4 : Print the new list

print("List of squares of odd numbers:", odd\_squares)

output :

Even numbers in the list:

8

22

4

List of squares of odd numbers: [225, 49, 961, 289]

**QUESTION 2 :**

# Given sentence

sentence = "The book was interesting because the book covered many topics and the topics discussed in the book were engaging"

# Convert the sentence to lowercase and split into words

words = sentence.lower().split()

# Create an empty dictionary to store word counts

word\_count = {}

# Loop through each word in the list of words

for word in words:

# Increase the count for the word by 1, or add it with count 1 if not present

word\_count[word] = word\_count.get(word, 0) + 1

# Print the dictionary containing word counts

print(word\_count)

output :

{

'the': 4,

'book': 3,

'was': 1,

'interesting': 1,

'because': 1,

'covered': 1,

'many': 1,

'topics': 2,

'and': 1,

'discussed': 1,

'in': 1,

'were': 1,

'engaging': 1

}

**QUESTION 3 :**

def is\_prime(n):

if n <= 1:

return False

for i in range(2, int(n \*\* 0.5) + 1):

if n % i == 0:

return False

return True

nums = [2, 4, 5, 10, 13, 17, 20, 23]

prime\_nums = [num for num in nums if is\_prime(num)]

print("Prime numbers:", prime\_nums)

output :

Prime numbers: [2, 5, 13, 17, 23]

**QUESTION 4 :**

import numpy as np

# Step 1: Create 2D array with 10 students and 5 subjects

scores = np.random.randint(50, 101, size=(10, 5)) # 50 to 100 inclusive

# Step 2: Compute average score per student (row-wise)

average\_scores = np.mean(scores, axis=1)

# Step 3: Compute maximum score per subject (column-wise)

max\_scores = np.max(scores, axis=0)

# Step 4: Add the average scores as a new column

final\_array = np.column\_stack((scores, average\_scores))

# Step 5: Print all results

print("Original Scores (10 students × 5 subjects):")

print(scores)

print("\nAverage score per student:")

print(average\_scores)

print("\nMaximum score per subject:")

print(max\_scores)

print("\nFinal array with average added:")

print(final\_array)

**QUESTION 5 :**

# Suppose final\_scores is the data with student scores and averages

# The header names for the columns

header = "Subject1,Subject2,Subject3,Subject4,Subject5,Average"

# Save data to a CSV file with commas and the header at the top

np.savetxt("student\_scores.csv", final\_scores, delimiter=",", header=header, comments='', fmt='%.2f')

print("File saved as student\_scores.csv")

**QUESTION 6 :**

# Create DataFrame with 8 sample rows

data = {

"product": ["pen", "pencil", "notebook", "eraser", "marker", "scale", "sharpener", "glue"],

"Price": [1.5, 0.5, 2.0, 0.75, 1.8, 1.2, 0.9, 1.0],

"quantity": [10, 25, 15, 30, 12, 18, 20, 22]

}

df = pd.DataFrame(data)

print(df)

**1.** # Add TotalValue column

df['TotalValue'] = df['Price'] \* df['quantity']

print(df)

**2.** # Sort the DataFrame by TotalValue in descending order

df = df.sort\_values(by='TotalValue', ascending=False)

print(df)

**3.**# Group by Price and calculate average quantity

avg\_quantity = df.groupby('Price')['quantity'].mean()

print(avg\_quantity)

**QUESTION 7 :**

pandas as pd

# Load data

df = pd.read\_csv("train.csv")

# 1. Missing values

print(df.isnull().sum())

# 2. Passengers under 18

print(df[df['Age'] < 18][['Name', 'Age']])

# 3. Female survivors

print(df[(df['Sex'] == 'female') & (df['Survived'] == 1)][['Name', 'Sex', 'Survived']])

# 4. Average fare by class

print(df.groupby('Pclass')['Fare'].mean())

**#QUESTION 8 :**

pandas as pd

1. columns = ['Age', 'Sex', 'ChestPainType', 'RestingBP', 'Cholesterol',

'FastingBS', 'RestingECG', 'MaxHR', 'ExerciseAngina', 'Oldpeak',

'ST\_Slope', 'NumVesselsFluoro', 'Thalassemia', 'HeartDisease']

df = pd.read\_csv('processed.cleveland.data', names=columns)

2.import numpy as np

df.replace('?', np.nan, inplace=True)

df['NumVesselsFluoro'] = pd.to\_numeric(df['NumVesselsFluoro'], errors='coerce')

df['Thalassemia'] = pd.to\_numeric(df['Thalassemia'], errors='coerce')

df['Cholesterol'] = pd.to\_numeric(df['Cholesterol'], errors='coerce')

3.

a) print(df.isnull().sum())

b) df['NumVesselsFluoro'].fillna(df['NumVesselsFluoro'].median(), inplace=True)

df['Thalassemia'].fillna(df['Thalassemia'].mode()[0], inplace=True)

c)df.dropna(thresh=len(df.columns) - 2, inplace=True)

d) grouped = df.groupby('HeartDisease')['Cholesterol'].mean()

print(grouped)

output:

* People **with** heart disease (HeartDisease = 1)
* People **without** heart disease (HeartDisease = 0)

**Question 9 :**

matplotlib.pyplot as plt

# List of student names

students = ['Alice', 'Bob', 'Charlie', 'David', 'Eva']

# Their marks in each subject

math = [85, 78, 92, 88, 76]

science = [90, 82, 89, 94, 75]

english = [78, 85, 88, 80, 82]

history = [84, 80, 91, 79, 77]

# 🎯 First Task: Show Math marks using bars

plt.bar(students, math) # Bar chart

plt.title('Math Marks') # Chart title

plt.xlabel('Students') # X-axis label

plt.ylabel('Marks') # Y-axis label

plt.show() # Show the chart

# 🎯 Second Task: Show all subject marks using lines

plt.plot(students, math, label='Math') # Line for Math

plt.plot(students, science, label='Science') # Line for Science

plt.plot(students, english, label='English') # Line for English

plt.plot(students, history, label='History') # Line for History

# Add chart title and labels

plt.title('Marks in All Subjects')

plt.xlabel('Students')

plt.ylabel('Marks')

plt.legend() # Show which line is for which subject

plt.show()

**QUESTION 10 :**

# This function will find BMI and say if you're underweight, normal, etc.

def check\_bmi(weight, height):

bmi = weight / (height \* height) # BMI = weight ÷ (height × height)

# Check the BMI and return the result

if bmi < 18.5:

return "BMI is " + str(round(bmi, 2)) + " → Underweight"

elif bmi < 25:

return "BMI is " + str(round(bmi, 2)) + " → Normal"

elif bmi < 30:

return "BMI is " + str(round(bmi, 2)) + " → Overweight"

else:

return "BMI is " + str(round(bmi, 2)) + " → Obese"

# Try the function with different values

print(check\_bmi(60, 1.7)) # 60 kg, 1.7 m tall

print(check\_bmi(80, 1.7)) # 80 kg, 1.7 m tall

print(check\_bmi(45, 1.6)) # 45 kg, 1.6 m tall