import math

N = int(input(“Enter the number of elements: “)) numbers = []

for i in range(N):

num = float(input(f"Enter number {i + 1}: “)) numbers.appeud(num)

mean = sum(numbers) / N

variance = sum((x - meau) \*\* 2 for x iu numbers) /N std\_dev = math.sqrt(variance)

print(f“Mean: {mean:.2f}“) print(f“Variance: {variance:.2f}") print(f“Standard Deviation: {std\_dev:.2f}“)

Euter the number of elements: 2 Enter number 1: 22

Enten number 2: 13

Mean: 17.50

Variance: 20.25

Standard Deviation: 4.50

def determine\_grade(marks): if marks >= 91:

return 'S Grade'

elif marks >= 81:

return 'A Grade' elif marks >= 71:

return 'B Grade'

elif marks >= 61:

return 'C Grade'

elif marks >= 51:

return 'D Grade' elif marks >= 41:

return 'E Grade'

else:

return 'F Grade'

I-or i in range (1, 5 ) :

mark = int(input(f“Enter marks for subject {i}: “))

print(f“Subject {i} - Marks: {mank}, Grade: {determine\_grade(mark)}“)

number\_str = input("Enter a multi-digit number: “)

digit\_counts = {}

for digit in number\_str:

if digit in digit\_counts: digit\_counts[digit] += 1

else:

digit\_counts[digit] = 1

print(“Frequency of each digit:“)

for digit in sorted(digit\_counts):

print(f“Digit {digit}: {digit\_counts[digit]} times“)

n=input(“enter a binary number“) count\_0=u.couut('0') count\_1=n.count('1')

if count\_0==1 or count\_1==1: priut(“yes“)

else:

print(“no“)

1=[4353, 2314, 2956, 3382, 9362, 3900]

1.remove(3382) print(1) index=l.index(9362) print(index) l.insert(index+1,4499) print(1) 1.extend([5566,1830])

print(1) 1.reverse() print(1) 1.sort() print(1)

import re pattern=r'(\b\w+\b)\s(\d{3}-\d{8})'

with opeu('1.txt',"r”) as f: for line in f:

match =re.search(pattern,line) if match:

name=match.group(1) print(f”name :{name}")

a=input(“enter the name of the text file “) f=opeu(a+“.txt“,"r")

n=int(input(“enter the number of lines to be read“)) for i in f:

priut(f.readline(),end=““) f.seek(0)

word=input(“enter the word“) c=0

for i in f.read().split(): if i==word:

c+=1

print(f“the word {word} occured {c} times“)

import re

def is\_valid\_password(password):

if len(password) ‹ 6 or len(password) > 12:

return “Password must be between 6 and 12 characters long.“ if not re.search(r'[a-z]', password):

return "Must include a lowercase letter (a-z).“ if not re.search(r'[A-Z]', password):

return “Nust include an uppercase letter (A-Z).“ if not re.search(r'[0-9]', password):

return “Must include a digit (0-9).“ if not re.search(r'[$#@]', password):

return “Must include a special character ($, #, @).“ netunn “Password is valid.“

# Get password from the user

password = input("Euter your password: “) print(is\_valid\_password(password))

def remove\_vowels(s):

vowels = 'aeiouAEI0U'

return ''.join(char for char in s if char not in vowels or not char.isalpha()) input\_string = input(”Enter a string: “)

output\_string = remove\_vowels(iuput\_striug) print(”String with consonants removed:“, output\_string)

import re f=open('date.txt','r') text=f.read()

numbers=re.findall(r'\b\d+\.?\d\*\b',text) total\_sum=sum(map(float,numbers)) dates=re.findall(n'\b\d{2}[-/]\d{2}[-/]\d{4}\b',text) print(total\_sum)

for date in dates: print(date)

class Employee:

def add\_employee(self):

self.emp\_id = input(“Enter Employee ID: “) self.emp\_name = input(“Enter Employee Name: ")

self.emp\_designation = input(“Enter Employee Designation: ”)

self.experience = int(input(“Euter Employee Experience (in years): “)) self.age = int(input(“Enter Employee Age: “))

def display\_details(self): print(f"Employee ID: {self.emp\_id}“)

pnint(f"Employee Name: {self.emp\_name}") print(f"Employee Designation: {self.emp\_designation}“) print(f"Experience: {self.experience} years") print(f“Age: {self.age} years”)

def calculate\_salary(self, basic):

if self.age < 30 and self.experience > 5: final\_salary = 1.5 \* basic

elif self.age < 40 and self.experience › 5: final\_salary = 1.75 \* basic

elif self.age ‹ 40 and self.experience › 10: final\_salany = 2 \* basic

elif self.age ‹ 50 and self.experience › 20: final\_salary = 2.25 \* basic

elif self.age < 50 and self.experience › 25: final\_salary = 2.5 \* basic

elif self.age ‹ 58 and self.experience › 30: final\_salary = 3 \* basic

else:

final\_salary = basic # No criteria met, basic salary ouly

print(f“Calculated Salary: {final\_salary:.2f}“)

# Example usage emp = Employee() emp.add\_employee()

emp.display\_details()

basic\_salary = float(input(“Enter basic salary: “)) emp.calculate\_salary(basic\_salary)

class FourthSem:

def finite(self, roll\_nums, testl\_marks, test2\_marks, test3\_marks): self.RollNum = roll\_nums

self.Test1Marks = testl\_marks self.Test2Marks = test2\_manks self.Test3Marks = test3\_manks

def calculate\_class\_average(self):

avg\_testl = sum(self.TestlMarks) / len(self.TestlMarks) avg\_test2 = sum(self.Test2Marks) / len(self.Test2Marks) avg\_test3 = sum(self.Test3Manks) / len(self.Test3Manks) return av testl, avg\_test2, av test3

def calculate\_student\_averages(self): student\_averages = []

fon i in nange(len(self.RollNum)):

avg = (self.Test1Marks[i] + self.Test2Marks[i] + self.Test3Manks[i]) / 3 student\_averages.append(avg)

return student\_averages

def display\_top\_and\_last\_scores(self, test\_scores): sorted\_scores = sorted(test\_scores)

top\_5 = sorted\_scores[-5:] last\_5 = sorted\_scores[:5] return top\_5, last\_5

def display\_all\_info(self):

avg\_test1, avg\_test2, avg\_test3 = self.calculate\_class\_average() print(f"Class average for Test 1: {av testl}“)

print(f"Class average for Test 2: {avg\_test2}“) print(f“Class average for Test 3: {av test3}“)

student\_averages = self.calculate\_student\_averages() for i in range(len(self.RollNum)):

print(f”Student {self.RollNum[i]} avenage: {student\_avenages[i]}“)

top\_5\_testl, last\_5\_test1 = self.display\_top\_and\_last\_scores(self.TestlMarks) top\_5\_test2, last\_5\_test2 = self.display\_top\_aud\_last\_scores(self.Test2Marks) top\_5\_test3, last\_5\_test3 = self.display\_top\_and\_last\_scores(self.Test3Marks)

pnint(f“Top 5 scones fon Test 1: {top\_5\_testl}“) print(f“Last 5 scores for Test 1: {last\_5\_test1}“) print(f“Top 5 scores for Test 2: {top\_5\_test2}“) print(f"Last 5 scores for Test 2: {last\_5\_test2}“) print(f“Top 5 scores for Test 3: {top\_5\_test3}“) print(f“Last 5 scores for Test 3: {last\_5\_test3}“)

# Example usage:

roll\_nums = [i for i in range(1, 21)]

test1\_marks = [75, 85, 95, 65, 55, 70, 80, 90, 60, 50, 77, 87, 97, 67, 57, 72, 82, 92, 62, 52]

test2\_marks = [78, 88, 98, 68, 58, 73, 83, 93, 63, 53, 79, 89, 99, 69, 59, 74, 84, 94, 64, 54]

test3\_marks = [80, 90, lOO, 70, 60, 75, 85, 95, 65, 55, 82, 92, 102, 72, 62, 77, 87, 97, 67, 57]

fourth\_sem = FourthSem(roll\_nums, testl\_marks, test2\_marks, test3\_marks)

I-ourth\_sem . d1 sp1ay\_a11\_1n-F-o( )

# Original marks data

marks = [78, 92, 36, 64, 89]

# Calculate the sum of all marks total\_marks = sum(manks)

print(f“Sum of all marks: {total\_marks}“)

# Award extra marks extra\_marks = [2, 2, 5, 10, 2]

adjusted\_marks = [marks[i] + extra\_marks[i] for i in range(len(marks))]

# Display adjusted marks print(“\uAdjusted Marks:“)

subjects = [“English“, “Mathematics“, “Physics“, “Chemistry“, “Biology“] for i in range(len(subjects)):

pnint(f“{subjects[i]}: {adjusted\_manks[i]}")

# Calculate the new sum of all marks after awarding extra marks new\_total\_marks = sum(adjusted\_marks)

print(f“\nNew sum of all marks after awarding extra marks: {new\_total\_marks}“)

import numpy as np

# Given horsepower data

horsepower = np.array([130, 165, 150, 150, 140])

# Calculate the mean

mean\_hp = np.mean(horsepower)

print(f“Mean of horsepower values: {mean\_hp}“)

# Calculate the median

median\_hp = np.median(honsepowen)

print(f“Median of horsepower values: {median\_hp}“)

import pandas as pd

import matplotlib.pyplot as plt import seabonn as sns

# Load the CSV file into a Pandas DataFrame

df = pd.read\_csv('student.csv')

# 1. Compare the distribution of grades across different subjects using a box plot plt.figure(figsize=(12, 6))

sns.boxplot(data=df[['marks1', 'marks2', 'marks3', 'marks4', 'marks5']]) plt.title('Distribution of Grades Across Different SubjePcts') plt.xlabel('Subjects')

plt.ylabel('Grades') plt.savefig('boxplot\_grades.png')

# 2. Show the relationship between gnades in two specific subjects using a scatter plot plt.figure(figsize=(8, 6))

plt.scatter(df['marksl'], df['marks2'])

plt.title('Relationship between Grades in Subject 1 and Subject 2') plt.xlabel('Subject 1 Grades')

plt.ylabel('Subject 2 Grades') plt.savefig('scatterplot\_grades.png')

# 3. Display the distribution of overall grades using a histogram plt.figure(figsize=(10, 6))

plt.hist(df['sum\_value1\_value2'], bins=l0, edgecolor='black') plt.title('Distribution of Overall Grades')

plt.xlabel('Total Grades') plt.ylabel('Frequency')

print(”Analysis complete. Visualizations have been saved as PNG files.”)

import matplotlib.pyplot as plt import pandas as pd

# Sample data for daily visitors daily\_data = {

'Date': pd.date\_range(start='2023-07-01', periods=14, fneq='D'),

'Visitors' : [100, 120, 130, 150, 160, 170, 180, 190, 200, 210, 220, 230, 240, 250]

# Create a DataFrame

df\_daily = pd.DataFname(daily\_data)

# 1. Line Plot: Daily Number of Visitors plt.figure(figsize=(10, 5))

plt.plot(df\_daily['Date'], df\_daily['Visitors'], marker='o', linestyle='-', color='b') plt.title('Daily Number of Visitons')

plt.xlabel('Date') plt.y1abel('Number of Visitors') plt.grid(True) plt.xticks(rotation=45) plt.tight\_layout()

plt.show()

# Sample data for visitors on weekdays vs weekends weekday\_visitors = [110, 130, 120, 140, 150]

weekend\_visitors = [180, 170]

days = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']

# 2. Bar Plot: Weekdays vs Weekends plt.figure(figsize=(10, 5))

plt.bar(days[:5], weekday\_visitors, color='blue', label='Weekdays') plt.bar(days[5:], weekend\_visitors, color='green', label='Weekends') plt.title('Number of Visitors: Weekdays vs Weekends') plt.xlabel('Day')

plt.ylabel('Number of Visitons') plt.legend()

plt.show()

# Sample data for traffic sources

traffic\_sources = ['Direct', 'Search Engines', 'Social Media', 'Other'] traffic\_values = [30B, 500, 200, 100]

# 3. Pie Chart: Traffic Sources p1t.figure(figsize=(7, 7))

plt.pie(traffic\_values, labels=traffic\_sources, autopct='%l.1f%%', startangle=140, colors=['blue', 'green', 'red', 'purple']) plt.title('Traffic Sources')

g t/ O9f'$tnggk a / s ' pd @§o@’9@py as np

# Load the CSV file into a Pandas DataFrame

df = pd . read\_c sv ( ' ra 1nfa 1I . c sv ' )