PAGE 21

**10 MARKS Write a python program to show use of multiple exception handing.**

def divide(x, y):

try:

result = x / y

except ZeroDivisionError:

print("Division by zero!")

except TypeError:

print("Invalid type(s) for division")

else:

print("Result is", result)

divide(2, 1)

divide(2, 0)

divide("2", "1")

**20 MARKS Write a python function to check whether a number is perfect or not.**

n = int(input("Enter any number: "))

sum1 = 0

for i in range(1, n):

if(n % i == 0):

sum1 = sum1 + i

if (sum1 == n):

print("The number is a Perfect number!")

else:

print("The number is not a Perfect number!")

**20 MARKS // OR QUESTION // Write a python program to display only those words from the text file which contains three**

**characters in it.**

# Open the file for reading

with open('textfile.txt', 'r') as file:

# Iterate over each line in the file

for line in file:

# Split the line into a list of words

words = line.split()

# Iterate over each word in the line

for word in words:

# Check if the word has 3 characters

if len(word) == 3:

# Print the word

print(word)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

PAGE 22

**10 MARKS Write a python program to show how to use else clause with try and except clauses.**

def divide(x, y):

try:

result = x / y

except ZeroDivisionError:

print("Division by zero!")

else:

print("Result is", result)

divide(2, 1)

divide(2, 0)

divide(3, 2)

**20 MARKS Write a python program to count and display even and odd numbers of a List**

# Initialize the list of numbers

numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# Initialize the counts of even and odd numbers

even\_count = 0

odd\_count = 0

# Iterate over the numbers in the list

for number in numbers:

# Check if the number is even

if number % 2 == 0:

# If it is, increment the count of even numbers

even\_count += 1

else:

# If it's not, increment the count of odd numbers

odd\_count += 1

# Print the counts

print("There are", even\_count, "even numbers in the list.")

print("There are", odd\_count, "odd numbers in the list.")

**20 MARKS // OR QUESTION // Write a python program to find sum of items of a Dictionary.**

# Initialize the dictionary

my\_dict = {'a': 1, 'b': 2, 'c': 3, 'd': 4}

# Initialize the sum

sum = 0

# Iterate over the items in the dictionary

for key, value in my\_dict.items():

# Add the value to the sum

sum += value

# Print the sum

print("The sum of the items in the dictionary is", sum)

PAGE 23

**10 MARKS . Write a python program to show use of finally clause of exception handling**

def divide(x, y):

try:

result = x / y

except ZeroDivisionError:

print("Division by zero!")

else:

print("Result is", result)

finally:

print("Executing finally clause")

divide(2, 1)

divide(2, 0)

divide(3, 2)

**20 MARKS Write a program that returns a list that contains only the elements that are common between the lists (without duplicates). Make sure your program works on two lists of different sizes.**

def common\_elements(list1, list2):

# Convert the lists to sets and find the intersection

common\_set = set(list1) & set(list2)

# Convert the set back to a list and return it

return list(common\_set)

# Test the function with two lists of different sizes

list1 = [1, 2, 3, 4, 5]

list2 = [3, 4, 5, 6, 7, 8]

print(common\_elements(list1, list2))

**20 MARKS // OR QUESTION // Write a python program which accepts file name and word to be searched in file. Display line numbers which contain given word and total occurrences of it**.

import sys

def find\_word(filename, word):

# Initialize the line number and the count

line\_number = 0

count = 0

# Open the file

with open(filename, 'r') as file:

# Iterate over each line in the file

for line in file:

# Increment the line number

line\_number += 1

# Split the line into a list of words

words = line.split()

# Iterate over each word in the line

for w in words:

# If the word matches the search word, increment the count

if w == word:

count += 1

# Print the line number

print("Line", line\_number, ":", line.strip())

# Print the total count

print("Total occurrences:", count)

# Get the file name and the search word from the command line

filename = sys.argv[1]

word = sys.argv[2]

# Call the find\_word function

find\_word(filename, word)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

PAGE 25

**10 MARKS Write a python program to show use of assert keyword.**

def is\_even(n):

return n % 2 == 0

assert is\_even(2) == True

assert is\_even(3) == False

assert is\_even(0) == True

print("All assertions pass.")

**20 MARKS Write a python program to perform following task. [20 marks]**

**a. Calculate the factorial of given number.**

**b. Reverse the given number**

def factorial(n):

if n < 0:

raise ValueError("Factorial is not defined for negative numbers")

elif n == 0:

return 1

else:

return n \* factorial(n-1)

def reverse(n):

return int(str(n)[::-1])

n = 15

fact = factorial(n)

rev = reverse(n)

print("Factorial of", n, "is", fact)

print("Reverse of", n, "is", rev)

20 MARKS

**OR QUESTION Write a python program which takes file name as input and print the lines after making**

**only first character of each word in the sentence capitalized.**

import os

import re

def capitalize\_first(line):

return re.sub(r"\b[a-z]", lambda match: match.group(0).upper(), line)

filename = input("Enter a file name: ")

if not os.path.exists(filename):

print("Error: File not found.")

else:

with open(filename) as f:

lines = f.readlines()

capitalized\_lines = [capitalize\_first(line) for line in lines]

print("\n".join(capitalized\_lines))

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

PAGE 26

**10 MARKS Write a python program to create a lambda function that adds 15 to a given number passed in as an argument.**

add\_15 = lambda x: x + 15

result = add\_15(10)

print(result) # 25

**20 MARKS Write a Python program to reverse the content of a file and store it in another file.**

def reverse\_file(in\_filename, out\_filename):

with open(in\_filename) as in\_file:

content = in\_file.read()

with open(out\_filename, 'w') as out\_file:

out\_file.write(content[::-1])

reverse\_file('input.txt', 'output.txt')

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

PAGE 27

**10 MARKS Write a python program to show how to handle multiple exceptions.**

try:

# Code that might raise an exception goes here

pass

except Exception1:

# Code to handle Exception1 goes here

pass

except Exception2:

# Code to handle Exception2 goes here

pass

except Exception3:

# Code to handle Exception3 goes here

pass

**20 MARKS . Write a python program to display tables from m to n. [20 marks]**

Example Input: m=3, n=7

Output: 3\*1=3 4\*1=4 …… 7\*1=7

3\*2=6 4\*2=8 …… 7\*2=14

.

.

.

3\*10=30 4\*10=40 …… 7\*10=70

def display\_tables(m, n):

for i in range(m, n+1):

for j in range(1, 11):

print(f"{i}\*{j}={i\*j}", end=' ')

print()

display\_tables(3, 7)

20 MARKS

**OR QUESTION Write a python program to accept directory name and print names of all files whose extension is ‘.txt’ in the given directory.**

import os

def print\_txt\_files(directory):

for filename in os.listdir(directory):

if filename.endswith('.txt'):

print(filename)

directory = input("Enter a directory name: ")

print\_txt\_files(directory)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

PAGE 28

**10 MARKS Write a python program to calculate the cube of all numbers from 1 to n**

def calculate\_cubes(n):

# Initialize an empty list to store the cubes

cubes = []

# Iterate through the range of numbers from 1 to n

for i in range(1, n+1):

# Calculate the cube of the current number

cube = i\*\*3

# Add the cube to the list

cubes.append(cube)

return cubes

# Test the function with n = 5

print(calculate\_cubes(5))

**20 MARKS Write a python program to display all prime numbers within given range.**

def display\_prime\_numbers(lower, upper):

# Iterate through the range of numbers from lower to upper

for num in range(lower, upper+1):

# Assume that the number is prime

prime = True

# Check if the number is prime

for i in range(2, num):

if (num % i) == 0:

prime = False

break

# If the number is prime, print it

if prime:

print(num)

# Test the function with lower = 10 and upper = 20

display\_prime\_numbers(10, 20)

20 MARKS

**OR QUESTION Write a python program to list only files from a directory and print files count.**

import os

def list\_files(directory):

# Get a list of all files and directories in the specified directory

files\_and\_dirs = os.listdir(directory)

# Initialize a counter for the number of files

file\_count = 0

# Iterate through the list

for item in files\_and\_dirs:

# Check if the item is a file (not a directory)

if os.path.isfile(os.path.join(directory, item)):

# If it's a file, print the filename and increment the counter

print(item)

file\_count += 1

# Print the total number of files

print(f'Number of files: {file\_count}')

# Test the function with the current directory

list\_files('.')

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

PAGE 29

**10 MARKS Write a python program to find square of given number using list comprehension**

def find\_square(num):

# Use list comprehension to calculate the square of the number

square = [i\*\*2 for i in [num]]

# Return the square

return square[0]

# Test the function with num = 5

print(find\_square(8))

**20 MARKS Write a python program which will find all such numbers which are divisible by 3 and not by 7 within given range m to n**

def find\_numbers(m, n):

# Initialize an empty list to store the numbers

numbers = []

# Iterate through the range of numbers from m to n

for i in range(m, n+1):

# If the number is divisible by 3 and not by 7, add it to the list

if (i % 3 == 0) and (i % 7 != 0):

numbers.append(i)

return numbers

# Test the function with m = 10 and n = 20

print(find\_numbers(10, 20))

20 MARKS

**OR QUESTION Write a python program to reverse each word of file and also count total lines.**

def reverse\_words\_in\_file(filename):

# Initialize a counter for the number of lines

line\_count = 0

# Open the file in read mode

with open(filename, 'r') as file:

# Read each line of the file

for line in file:

# Split the line into words

words = line.split()

# Reverse each word

reversed\_words = [word[::-1] for word in words]

# Join the reversed words into a single string

reversed\_line = ' '.join(reversed\_words)

# Print the reversed line

print(reversed\_line)

# Increment the line counter

line\_count += 1

# Print the total number of lines

print(f'Number of lines: {line\_count}')

# Test the function with a sample file

reverse\_words\_in\_file('sample.txt')

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

PAGE 30

**10 MARKS Write a python program which create a lambda function that multiplies argument x with argument y and print the result.**

# Define the lambda function

multiply = lambda x, y: x \* y

# Test the lambda function

result = multiply(5, 6)

print(result)

**20 MARKS Write a python program to display all files in directory and subdirectories**

import os

def list\_files(directory):

# Iterate through the files and directories in the specified directory

for item in os.listdir(directory):

# Construct the full path of the item

item\_path = os.path.join(directory, item)

# If the item is a file, print its name

if os.path.isfile(item\_path):

print(item)

# If the item is a directory, recursively call the function to list its files

elif os.path.isdir(item\_path):

list\_files(item\_path)

# Test the function with the current directory

list\_files('.')

20 MARKS

**OR QUESTION Write a python program to delete repeated lines from a file.**

def delete\_duplicate\_lines(filename):

# Open the file in read mode

with open(filename, 'r') as file:

# Read the lines of the file into a list

lines = file.readlines()

# Remove duplicates from the list of lines

lines = list(set(lines))

# Open the file in write mode

with open(filename, 'w') as file:

# Write the unique lines back to the file

for line in lines:

file.write(line)

# Test the function with a sample file

delete\_duplicate\_lines('sample.txt')

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

PAGE 31

**10 MARKS Write a python program to find the repeated items of a tuple.**

def find\_repeated(tup):

# Convert the tuple to a set to remove duplicates

tup\_set = set(tup)

# Initialize an empty list to store the repeated items

repeated = []

# Iterate through the set of unique items

for item in tup\_set:

# If the item appears more than once in the tuple, add it to the list of repeated items

if tup.count(item) > 1:

repeated.append(item)

# Return the list of repeated items

return repeated

# Test the function with a sample tuple

print(find\_repeated((1, 2, 3, 2, 3, 4, 3)))

**20 MARKS Write a python program with user defined function which accept long string containing multiple words and it return same string with the words in backwards order.**

def reverse\_words(string):

# Split the string into a list of words

words = string.split()

# Reverse the list of words

words.reverse()

# Join the reversed list of words into a single string

reversed\_string = ' '.join(words)

# Return the reversed string

return reversed\_string

# Test the function with a sample string

print(reverse\_words('This is a sample string'))

20 MARKS

**OR QUESTION Define a class Employee having members – id, name, department, salary. Create a subclass called ―Manager with member bonus. Define methods accept and display in both the classes. Create n objects of the Manager class and display the details of the manager**

**having the maximum total salary (salary + bonus).**

class Employee:

def \_\_init\_\_(self, id, name, department, salary):

self.id = id

self.name = name

self.department = department

self.salary = salary

def accept(self):

self.id = input('Enter employee ID: ')

self.name = input('Enter employee name: ')

self.department = input('Enter employee department: ')

self.salary = input('Enter employee salary: ')

def display(self):

print(f'ID: {self.id}')

print(f'Name: {self.name}')

print(f'Department: {self.department}')

print(f'Salary: {self.salary}')

class Manager(Employee):

def \_\_init\_\_(self, id, name, department, salary, bonus):

super().\_\_init\_\_(id, name, department, salary)

self.bonus = bonus

def accept(self):

super().accept()

self.bonus = input('Enter manager bonus: ')

def display(self):

super().display()

print(f'Bonus: {self.bonus}')

# Create a list of Manager objects

managers = [Manager(1, 'John', 'Marketing', 50000, 10000),

Manager(2, 'Jane', 'Sales', 60000, 15000),

Manager(3, 'Bob', 'HR', 45000, 7000)]

# Find the manager with the highest total salary (salary + bonus)

max\_salary = 0

max\_manager = None

for manager in managers:

total\_salary = manager.salary + manager.bonus

if total\_salary > max\_salary:

max\_salary = total\_salary

max\_manager = manager

# Display the details of the manager with the highest total salary

print('Manager with highest total salary:')

max\_manager.display()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

PAGE 32

**10 MARKS Write a python program to find the length of a set. (Don’t use built in function len)**

def set\_length(s):

# Initialize a counter

count = 0

# Iterate through the set

for \_ in s:

# Increment the counter

count += 1

# Return the counter

return count

# Test the function with a sample set

print(set\_length({1, 2, 3, 4, 5}))

**20 MARKS Write a python program that accepts a sentence and calculate the number of uppercase letters and lowercase letters**

def count\_letters(sentence):

# Initialize counters for upper and lower case letters

upper\_count = 0

lower\_count = 0

# Iterate through the characters in the sentence

for c in sentence:

# If the character is uppercase, increment the upper case counter

if c.isupper():

upper\_count += 1

# If the character is lowercase, increment the lower case counter

elif c.islower():

lower\_count += 1

# Return the counts

return upper\_count, lower\_count

# Test the function with a sample sentence

print(count\_letters('This is a Sample Sentence'))

20 MARKS

**OR QUESTION Define a class named Rectangle which can be constructed by a length and width. The Rectangle class has amethod which can compute the area and perimeter. Display area and perimeter. Also delete the object.**

class Rectangle:

def \_\_init\_\_(self, length, width):

self.length = length

self.width = width

def area(self):

return self.length \* self.width

def perimeter(self):

return 2 \* (self.length + self.width)

def \_\_del\_\_(self):

print('Rectangle object deleted')

# Create a rectangle object

rect = Rectangle(5, 6)

# Compute and display the area and perimeter

print(f'Area: {rect.area()}')

print(f'Perimeter: {rect.perimeter()}')

# Delete the rectangle object

del rect

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

PAGE 33

**20 MARKS Write a python program that accepts a sentence and calculate the number of letters and digits in it.**

def count\_letters\_digits(sentence):

# Initialize counters for letters and digits

letter\_count = 0

digit\_count = 0

# Iterate through the characters in the sentence

for c in sentence:

# If the character is a letter, increment the letter counter

if c.isalpha():

letter\_count += 1

# If the character is a digit, increment the digit counter

elif c.isdigit():

digit\_count += 1

# Return the counts

return letter\_count, digit\_count

# Test the function with a sample sentence

print(count\_letters\_digits('This is a Sample Sentence with 123 Numbers'))

**20 MARKS**

**OR QUESTION Write a python program to create a class Circle and compute the area and the circumference of the Circle. (Use parameterized constructor).**

import math

class Circle:

def \_\_init\_\_(self, radius):

self.radius = radius

def area(self):

return math.pi \* self.radius\*\*2

def circumference(self):

return 2 \* math.pi \* self.radius

# Create a circle object with radius 5

circle = Circle(5)

# Compute and display the area and circumference of the circle

print(f'Area: {circle.area()}')

print(f'Circumference: {circle.circumference()}')

PAGE 34

**10 MARKS Write a program which checks whether given element exists within a tuple.**

def element\_exists(tup, element):

# Iterate through the elements in the tuple

for e in tup:

# If the element is found, return True

if e == element:

return True

# If the element is not found, return False

return False

# Test the function with a sample tuple and element

print(element\_exists((1, 2, 3, 4, 5), 3))

print(element\_exists((1, 2, 3, 4, 5), 3))

**20 MARKS Write a Python program to find the greatest common divisor (gcd) of two integers.**

def gcd(a, b):

# If one of the numbers is 0, return the other number

if a == 0:

return b

if b == 0:

return a

# If both numbers are the same, return the number

if a == b:

return a

# If one of the numbers is even, divide it by 2 and check the gcd again

if a % 2 == 0:

return gcd(a // 2, b)

if b % 2 == 0:

return gcd(a, b // 2)

# If both numbers are odd, use the Euclidean algorithm

if a > b:

return gcd((a - b) // 2, b)

return gcd((b - a) // 2, a)

# Test the function with two sample integers

print(gcd(12, 16))

print(gcd(60, 48))

20 MARKS

**OR QUESTION Define a class Student having members – rollno, name, age, gender. Create a subclass called ―Test with member marks of 3 subjects. Create three objects of the Test class and display all the details of the student with percentage.**

class Student:

def \_\_init\_\_(self, rollno, name, age, gender):

self.rollno = rollno

self.name = name

self.age = age

self.gender = gender

class Test(Student):

def \_\_init\_\_(self, rollno, name, age, gender, marks):

super().\_\_init\_\_(rollno, name, age, gender)

self.marks = marks

def percentage(self):

return sum(self.marks) / len(self.marks)

# Create three test objects

test1 = Test(1, 'Alice', 20, 'Female', [80, 85, 90])

test2 = Test(2, 'Bob', 21, 'Male', [75, 80, 85])

test3 = Test(3, 'Charlie', 22, 'Male', [70, 75, 80])

# Display the details and percentage of each student

print(f'Roll No.: {test1.rollno}')

print(f'Name: {test1.name}')

print(f'Age: {test1.age}')

print(f'Gender: {test1.gender}')

print(f'Percentage: {test1.percentage()}')

print()

print(f'Roll No.: {test2.rollno}')

print(f'Name: {test2.name}')

print(f'Age: {test2.age}')

print(f'Gender: {test2.gender}')

print(f'Percentage: {test2.percentage()}')

print()

print(f'Roll No.: {test3.rollno}')

print(f'Name: {test3.name}')

print(f'Age: {test3.age}')

print(f'Gender: {test3.gender}')

print(f'Percentage: {test3.percentage()}')

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

PAGE 35

**10 MARKS Write a python program to find the repeated items of a tuple.**

def find\_repeated(tup):

# Initialize an empty list to store the repeated items

repeated = []

# Iterate through the items in the tuple

for item in tup:

# If the item appears more than once in the tuple, add it to the list

if tup.count(item) > 1:

repeated.append(item)

# Return the list of repeated items

return repeated

# Test the function with a sample tuple

print(find\_repeated((1, 2, 3, 4, 2, 3)))

**20 MARKS Write a python program to accept string and remove the characters which have odd index values of a given string using user defined function.**

def remove\_odd\_index(string):

# Initialize an empty string

result = ''

# Iterate through the characters in the string

for i, c in enumerate(string):

# If the index is even, add the character to the result

if i % 2 == 0:

result += c

# Return the result

return result

# Test the function with a sample string

print(remove\_odd\_index('abcdefghijklmnopqrstuvwxyz'))

20 MARKS

**OR QUESTION Define a class Person having members – name, address. Create a subclass called ―Employee with member staffed, salary. Create ‘n’ objects of the Employee class and display all the details of the Employee.**

class Person:

def \_\_init\_\_(self, name, address):

self.name = name

self.address = address

class Employee(Person):

def \_\_init\_\_(self, name, address, staffed, salary):

super().\_\_init\_\_(name, address)

self.staffed = staffed

self.salary = salary

# Create n objects of the Employee class

n = 3

employees = [Employee('Alice', '123 Main St', True, 50000),

Employee('Bob', '456 Main St', True, 60000),

Employee('Charlie', '789 Main St', False, 0)]

# Display the details of the employees

for employee in employees:

print(f'Name: {employee.name}')

print(f'Address: {employee.address}')

print(f'Staffed: {employee.staffed}')

print(f'Salary: {employee.salary}')

print()

PAGE 36

**10 MARKS Write a python program that prints out all the elements of the list that are less than 25**

# Define a sample list

lst = [10, 20, 30, 40, 50, 60, 70, 80, 90]

# Iterate through the elements in the list

for element in lst:

# If the element is less than 25, print it

if element < 25:

print(element)

**20 MARKS Create a class named Circle constructed by a radius and two methods which will compute the area and the perimeter of a circle.**

import math

class Circle:

def \_\_init\_\_(self, radius):

self.radius = radius

def area(self):

return math.pi \* self.radius\*\*2

def perimeter(self):

return 2 \* math.pi \* self.radius

# Create a circle object with radius 5

circle = Circle(5)

# Compute and display the area and perimeter of the circle

print(f'Area: {circle.area()}')

print(f'Perimeter: {circle.perimeter()}')

20 MARKS

**OR QUESTION For given a .txt file that has a list of a bunch of names, count how many of each name there are in the file and print count**.

# Open the file in read mode

with open('names.txt', 'r') as f:

# Read the contents of the file into a list

names = f.readlines()

# Initialize a dictionary to store the counts

counts = {}

# Iterate through the names in the list

for name in names:

# Strip the whitespace from the name

name = name.strip()

# If the name is not in the dictionary, add it and set the count to 1

if name not in counts:

counts[name] = 1

# If the name is already in the dictionary, increment the count

else:

counts[name] += 1

# Print the counts

for name, count in counts.items():

print(f'{name}: {count}')

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

PAGE 37

**10 MARKS Write a python program which reverse given string and displays both original and reversed string. (Don’t use built-in function)**

# Define a sample string

string = 'abcdefghijklmnopqrstuvwxyz'

# Initialize an empty string to store the reversed string

reversed\_string = ''

# Iterate through the characters in the string in reverse order

for i in range(len(string)-1, -1, -1):

# Add the character to the reversed string

reversed\_string += string[i]

# Print the original and reversed strings

print(f'Original string: {string}')

print(f'Reversed string: {reversed\_string}')

**20 MARKS Write a python program to implement binary search to search the given element using function.**

def binary\_search(lst, element):

# Get the indices of the first and last elements in the list

first = 0

last = len(lst) - 1

# Initialize a flag to indicate whether the element has been found

found = False

# While the element has not been found and the first index is less than or equal to the last index

while not found and first <= last:

# Calculate the midpoint of the list

midpoint = (first + last) // 2

# If the element is at the midpoint, set the found flag to True

if lst[midpoint] == element:

found = True

# If the element is less than the midpoint, search the left half of the list

elif element < lst[midpoint]:

last = midpoint - 1

# If the element is greater than the midpoint, search the right half of the list

else:

first = midpoint + 1

# Return the found flag

return found

# Test the binary search function with a sample list and element

print(binary\_search([1, 2, 3, 4, 5], 3))

print(binary\_search([1, 2, 3, 4, 5], 6))

20 MARKS

**OR QUESTION Write a python program that returns a list that contains only the elements that are common between the lists (without duplicates). Make sure your program works on two lists of different sizes.**

# Define two sample lists

list1 = [1, 2, 3, 4, 5]

list2 = [4, 5, 6, 7, 8, 9]

# Convert both lists to sets to remove duplicates

set1 = set(list1)

set2 = set(list2)

# Find the intersection of the sets

intersection = set1 & set2

# Convert the intersection set to a list and print it

result = list(intersection)

print(result)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

PAGE 38

**10 MARKS Write a python program to count the number of characters in a string without using any built-in function.**

# Define a sample string

string = 'Hello, world!'

# Initialize a counter to 0

count = 0

# Iterate through the characters in the string

for char in string:

# Increment the counter

count += 1

# Print the count

print(count)

**20 MARKS Define a class Person having members – name, address. Create a subclass called ―Employee with member staffid, salary. Create ‘n’ objects of the Employee class and display all the details of highest salaried employee**

# Define the Person class

class Person:

def \_\_init\_\_(self, name, address):

self.name = name

self.address = address

# Define the Employee subclass

class Employee(Person):

def \_\_init\_\_(self, name, address, staffid, salary):

super().\_\_init\_\_(name, address)

self.staffid = staffid

self.salary = salary

# Create a list of Employee objects

employees = [

Employee('John', '123 Main St', 1, 50000),

Employee('Jane', '456 Main St', 2, 60000),

Employee('Bob', '789 Main St', 3, 40000)

]

# Initialize variables to store the highest salary and the corresponding employee

highest\_salary = 0

highest\_salaried\_employee = None

# Iterate through the employees

for employee in employees:

# If the employee's salary is higher than the current highest salary, update the highest salary and employee

if employee.salary > highest\_salary:

highest\_salary = employee.salary

highest\_salaried\_employee = employee

# Display the details of the highest salaried employee

print(f'Name: {highest\_salaried\_employee.name}')

print(f'Address: {highest\_salaried\_employee.address}')

print(f'Staff ID: {highest\_salaried\_employee.staffid}')

print(f'Salary: {highest\_salaried\_employee.salary}')

20 MARKS

**OR QUESTION Write a python program to check if a given key already exists in a dictionary. If key exists replace with another key/value pair.**

# Define a sample dictionary

dictionary = {

'key1': 'value1',

'key2': 'value2',

'key3': 'value3'

}

# Define the key and value to add or replace

key = 'key2'

value = 'new value'

# If the key is in the dictionary, update the value

if key in dictionary:

dictionary[key] = value

# If the key is not in the dictionary, add the key/value pair

else:

dictionary[key] = value

# Print the updated dictionary

print(dictionary)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

PAGE 39

**10 MARKS Write a python program to get a single string from two given strings and swap the first**

**two characters of each string.**

**Sample String: 'abc', 'xyz'**

**Expected Output: xycabz**

# Define the two sample strings

string1 = 'abc'

string2 = 'xyz'

# Swap the first two characters of each string

swapped1 = string2[:2] + string1[2:]

swapped2 = string1[:2] + string2[2:]

# Concatenate the swapped strings

result = swapped1 + swapped2

# Print the result

print(result)

**20 MARKS Define a class Person having members – name, address. Create a subclass called ―Employee with members staffed, salary. Create ‘n’ objects of the Employee class and display all the details of the Employee.**

# Define the Person class

class Person:

def \_\_init\_\_(self, name, address):

self.name = name

self.address = address

# Define the Employee subclass

class Employee(Person):

def \_\_init\_\_(self, name, address, staffid, salary):

super().\_\_init\_\_(name, address)

self.staffid = staffid

self.salary = salary

# Create a list of Employee objects

employees = [

Employee('John', '123 Main St', 1, 50000),

Employee('Jane', '456 Main St', 2, 60000),

Employee('Bob', '789 Main St', 3, 40000)

]

# Iterate through the employees

for employee in employees:

# Display the employee's details

print(f'Name: {employee.name}')

print(f'Address: {employee.address}')

print(f'Staff ID: {employee.staffid}')

print(f'Salary: {employee.salary}')

print()

20 MARKS

**OR QUESTION . Write a python program to create a tuple of n numbers and print maximum, minimum, and sum of elements in a tuple. (Don’t use built-in functions)**

# Define a sample tuple of numbers

tuple = (1, 2, 3, 4, 5)

# Initialize variables to store the maximum, minimum, and sum

maximum = tuple[0]

minimum = tuple[0]

sum = 0

# Iterate through the elements in the tuple

for element in tuple:

# Update the maximum and minimum if necessary

if element > maximum:

maximum = element

if element < minimum:

minimum = element

# Add the element to the sum

sum += element

# Print the results

print(f'Maximum: {maximum}')

print(f'Minimum: {minimum}')

print(f'Sum: {sum}')

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

PAGE 40

**10 MARKS Write a python program to accept and convert string in uppercase or vice versa.**

# Define a sample string

string = 'Hello, World!'

# Convert the string to uppercase

uppercase = string.upper()

# Convert the string to lowercase

lowercase = string.lower()

# Print the results

print(f'Original: {string}')

print(f'Uppercase: {uppercase}')

print(f'Lowercase: {lowercase}')

**20 MARKS Write a python program to create a class Calculator with basic calculator operations**

**(addition,subtraction,division,multiplication,remainder).**

# Define the Calculator class

class Calculator:

def addition(self, x, y):

return x + y

def subtraction(self, x, y):

return x - y

def division(self, x, y):

return x / y

def multiplication(self, x, y):

return x \* y

def remainder(self, x, y):

return x % y

# Create a Calculator object

calculator = Calculator()

# Perform some calculations

result1 = calculator.addition(5, 10)

result2 = calculator.subtraction(10, 5)

result3 = calculator.division(10, 5)

result4 = calculator.multiplication(5, 10)

result5 = calculator.remainder(10, 3)

# Print the results

print(f'5 + 10 = {result1}')

print(f'10 - 5 = {result2}')

print(f'10 / 5 = {result3}')

print(f'5 \* 10 = {result4}')

print(f'10 % 3 = {result5}')

**20 MARKS**

**OR QUESTION Write a python program to perform operations on sets which includes union of two sets,**

**an intersection of sets, set difference and a symmetric difference.**

# Define two sample sets

set1 = {1, 2, 3, 4, 5}

set2 = {4, 5, 6, 7, 8}

# Perform set operations

union = set1 | set2 # Union of the sets

intersection = set1 & set2 # Intersection of the sets

difference = set1 - set2 # Difference between the sets

symmetric\_difference = set1 ^ set2 # Symmetric difference between the sets

# Print the results

print(f'Set 1: {set1}')

print(f'Set 2: {set2}')

print(f'Union: {union}')

print(f'Intersection: {intersection}')

print(f'Difference: {difference}')

print(f'Symmetric Difference: {symmetric\_difference}')

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*