Python Lab Experiment -8

Q1. Define a class Cab having following specifications:

1. Init method that initializes driver name, kms and rate/km.
2. Cab Class had a method rateperkm() that returns the running charges as kms*rate
3. There are 3 drivers (driver1, driver2 and driver3) who have their own rate (rate1, rate2 and rate3)
4. Create three objects of the class Cab (firstcab, secondcab and thirdcab) and use to get the name of each driver along with the charges.

Sample Inputs (kms, driver1,rate1.driver2,rate2,driver3,rate3)

```
kms=25
driver_1=Raju
rate_1=11
driver_2= Rahul
rate_2=13
driver_3=Viswas
rate_3=15
```

Sample Output

```
First Cab Driver: Raju
First Cab Payment: 275
Second Cab Driver: Rahul
Second Cab Payment: 325
Third Cab Driver: Viswas
Third Cab Payment: 375
```

Q2. Given two user inputs x and n, calculate the value of x^n.
For this expression evaluation design a class and an object to call the method to implement this.

Sample Input 1
(X, n)= 5, 2
Sample Output 1
25

Sample Input 2
(X, n)= 2, -2
Sample Output 2
0.25

Q3. Define a class cars that uses init method to initialize the name, model and speed of a car.

1. Cars has two methods accelerate() and brakes(), that takes the value speed from init method.

2. accelerate() returns the Cars.speed +70 whereas brakes() returns Cars.speed-20.

Sample Input_1
Name: Tata
Model: Punch
Speed: 100
Sample Output_1
When the car accelerates, speed is 170
Car brakes applied; speed is 80

Q4. WAP to create a class UPES with three attributes, namely School name, number of students and number of faculties. Add a method in the class to show these attributes. Create three objects of this class UPES and show their details.

Database:

School	Students	Faculty
SoCS	1000	150
Media	500	50
Law	450	25

Q5. We have two circles with given coordinates of their centers C1(x1, y1) and C2(x2, y2) and radius R1 and R2. Create a class with a method to check if the given circles

a) Inside the other
b) touch each other
c) Intersect each other
d) Do not overlap

Hint:
Distance between centres C1 and C2 is calculated as
distance = sqrt((x1 – x2)^2 + (y1 – y2)^2).

Test case 1:
Input :
3
4
14
18
5
8
Output :
Circle not touch to each other

Test case 2:
Input :
-10
8
14
-24
30
10
Output :
Circle touch to each other

Test case 3:
Input:
2
3
15
28
12
10
Output:
Circle not touch to each other

Test case 4:
Input:
0
0
1
1
5
2
Output:
Circle B is inside A

Q6. Write a Python code to check a given number is odd or even using class. For this, design a class namely "even_odd" and a method "check" and create an object to check the number using this function.

Sample Input
15
Sample Output
Number is even

Sample Input
10
Sample Output
Number is odd

Q7. A person has a list of words, where the words are written in small case letters. He wants to convert each word of that list into uppercase letters. Write a python program (a function) that converts small case word list to uppercase words list. For Example ['delhi', 'panjab'] will be input and output will be ['DELHI', 'PANJAB'].

Test Case 1:
Input:
['mohan', 'surendra', 'nitin']
Output:
['MOHAN', 'SURENDRA', 'NITIN']

Test Case 2:
Input:
['python', 'upes’]
Output:
['PYTHON', ‘UPES’]

Test Case 3:
Input:
['delhi', 'punjab']
Output:
['DELHI', 'PUNJAB']

Test Case4:
Input:
['Hello', 'Who', 'are', 'you']
Output:
['HELLO', 'WHO', 'ARE', 'YOU']

Q8. Counting Upper and Lower case and Space symbols: Design a python module that will count both upper, lower case symbols, and spaces in a given paragraph or sentences. Create a module named case_counting.py which has the funtion string_test for performing the count. Create an other program file main.py which import the case_counting module.

Hint: Use Dictionary to store count.

Test Case 1
Input:
Enter any sentence:
Pollution is very high in Delhi-NCR
Output:
Original String: Pollution is very high in Delhi-NCR
No. of Upper case characters: 5
No. of Lower case Characters: 24
No. of spaces: 5

Q9. Perfect number
Design and code a function viz., “perfect()” that determines if parameter number is a perfect number. Use this function in a program that determines and prints all the perfect numbers between 1 and N in a list.
[An integer number is said to be “perfect number” if its factors, including 1(but not the number itself), sum to the number. E.g., 6 is a perfect number because 6=1+2+3].

Test Case_1
Input:
Enter the value of N: 1000
Output:
Perfect numbers are: [6, 28, 496]

Test Case_2
Input:
Enter the value of N: 10000
Output:
Perfect numbers are: [6, 28, 496, 8128]

Test Case_3
Input:
Enter the value of N: 50
Output:
Perfect numbers are: [6, 28]

Q10. Practice Question: Student Database Management:

Develop a Python program to manage a student database. Implement functionalities to add new students, display all students' information, search for a student by their ID, and save the database to a file named "students.txt".

```
class Student:
  def __init__(self, id, name, age):
    self.id = id
```

```
        self.name = name
        self.age = age

    def __str__(self):
        return f"ID: {self.id}, Name: {self.name}, Age: {self.age}"

def save_students_to_file(students, filename):
    with open(filename, 'w') as file:
        for student in students:
            file.write(f"{student.id},{student.name},{student.age}\n")

def load_students_from_file(filename):
    students = []
    with open(filename, 'r') as file:
        for line in file:
            data = line.strip().split(',')
            student = Student(data[0], data[1], data[2])
            students.append(student)
    return students

# Usage example:
students = [Student("101", "Alice", 20), Student("102", "Bob", 21)]
save_students_to_file(students, "students.txt")
loaded_students = load_students_from_file("students.txt")
for student in loaded_students:
    print(student)
```

Q11. Practice Question: Employee Directory:

Implement a Python program to maintain an employee directory. Allow users to add new employees, update employee information, delete employees, display all employees' details, and save the directory to a file named "employees.txt".

```
class Employee:
    def __init__(self, id, name, position):
        self.id = id
        self.name = name
        self.position = position

    def __str__(self):
        return f"ID: {self.id}, Name: {self.name}, Position: {self.position}"

def save_employees_to_file(employees, filename):
    with open(filename, 'w') as file:
        for emp in employees:
            file.write(f"{emp.id},{emp.name},{emp.position}\n")

def load_employees_from_file(filename):
    employees = []
    with open(filename, 'r') as file:
        for line in file:
            data = line.strip().split(',')
            emp = Employee(data[0], data[1], data[2])
            employees.append(emp)
    return employees

# Usage example:
employees = [Employee("001", "Alice", "Manager"), Employee("002", "Bob", "Developer")]
save_employees_to_file(employees, "employees.txt")
loaded_employees = load_employees_from_file("employees.txt")
for emp in loaded_employees:
    print(emp)
```

Q12. Exception Handling:

1) Write a Python program that takes two numbers from the user and divides the first number by the second number. Handle the ZeroDivisionError exception if the second number is zero.

2) Develop a Python program that reads the contents of a file specified by the user. Handle the FileNotFoundError exception if the file does not exist.

3) Create a Python program that prompts the user to enter an integer. Handle the ValueError exception if the input is not an integer.

4) Write a Python program that prompts the user to enter the index of a list and then prints the element at that index. Handle the IndexError exception if the index is out of range.

5) Develop a Python program that defines a dictionary and prompts the user to enter a key to retrieve the corresponding value. Handle the KeyError exception if the key does not exist in the dictionary.

6) Design a Python program that writes user input to a file named "output.txt". Handle the IOError exception if there is an error while writing to the file.

7) Create a Python program that prompts the user to enter two numbers and then concatenates them as strings. Recognize TypeError exception if the inputs are not convertible to strings.

8) Write a Python program that prompts the user to enter a number between 1 and 10. Handle the ValueError exception if the input is not within the specified range.

9) Develop a Python program that performs arithmetic operations based on user input. Recognize the ArithmeticError exception for invalid operations.