Q-1: Rectangle Class

1. Write a Rectangle class in Python language, allowing you to build a rectangle with length and width attributes.
2. Create a Perimeter() method to calculate the perimeter of the rectangle and a Area() method to calculate the area of the rectangle.
3. Create a method display() that display the length, width, perimeter and area of an object created using an instantiation on rectangle class.

Eg. After making above classes and methods, on executing below code:-

```
my_rectangle = Rectangle(3 , 4)
my_rectangle.display()
```

Output:

```
The length of rectangle is:  3
The width of rectangle is:  4
The perimeter of rectangle is:  14
The area of rectangle is:  12
```

```
# write your code here

class Rectangle:

    def __init__(self,l,b):
        self.__length = l
        self.__width = b

    def __perimeter(self):
        return 2*(self.__length + self.__width)

    def __area(self):
        return self.__length * self.__width

    def display(self):
        print('length',self.__length)
        print('width',self.__width)
        print('perimeter',self.__perimeter())
        print('area',self.__area())

obj = Rectangle(4,5)
obj.display()
```

```
length 4
width 5
perimeter 18
area 20
```

Q-2: Bank Class

1. Create a Python class called `BankAccount` which represents a bank account, having as attributes: `accountNumber` (numeric type), `name` (name of the account owner as string type), `balance`.
2. Create a constructor with parameters: `accountNumber, name, balance`.
3. Create a `Deposit()` method which manages the deposit actions.
4. Create a `Withdrawal()` method which manages withdrawals actions.
5. Create an `bankFees()` method to apply the bank fees with a percentage of 5% of the balance account.
6. Create a `display()` method to display account details. Give the complete code for the BankAccount class.

Eg. After making above classes and methods, on executing below code:-

```
newAccount = BankAccount(2178514584, "Mandy" , 2800)

newAccount.Withdrawal(700)
```

```
newAccount.Deposit(1000)

newAccount.display()
```

Output:

```
Account Number :  2178514584
Account Name :  Mandy
Account Balance :  3100 ₹
```

```
# write your code here

class Bank:

    def __init__(self,name,acc_no,balance):
        self.__name = name
        self.__acc_no = acc_no
        self.__balance = balance

    def display(self):
        print('Account Number :',self.__acc_no)
        print('Account Name :',self.__name)
        print('Account Balance :',self.__balance,'₹')

    def deposit(self,amount):
        self.__balance = self.__balance + amount

    def withdrawl(self,amount):
        if amount>self.__balance:
            print('Insufficient funds')
        else:
            self.__balance = self.__balance - amount
            reduction = self.__bankFees()
            self.__balance = self.__balance - reduction

    def __bankFees(self):
        return 0.05*self.__balance

cust = Bank('nitish',1000000,500)
cust.deposit(500)
cust.withdrawl(100)
cust.display()
```

```
Account Number : 1000000
Account Name : nitish
Account Balance : 855.0 ₹
```

Q-3:Computation class

1. Create a `Computation` class with a default constructor (without parameters) allowing to perform various calculations on integers numbers.
2. Create a method called `Factorial()` which allows to calculate the factorial of an integer n. Integer n as parameter for this method
3. Create a method called `naturalSum()` allowing to calculate the sum of the first n integers 1 + 2 + 3 + .. + n. Integer n as parameter for this method.
4. Create a method called `testPrime()` in the Calculation class to test the primality of a given integer n, n is Prime or Not? Integer n as parameter for this method.
5. Create a method called `testPrims()` allowing to test if two numbers are prime between them. Two integers are prime to one another if they have only `1` as their common divisor. Eg. 4 and 9 are prime to each other.
6. Create a `tableMult()` method which creates and displays the multiplication table of a given integer. Then create an `allTablesMult()` method to display all the integer multiplication tables 1, 2, 3, ..., 9.
7. Create a static `listDiv()` method that gets all the divisors of a given integer on new list called Ldiv. Create another `listDivPrim()` method that gets all the prime divisors of a given integer.

```
# write your code here
#Write your code here

class Computation:
    def __init__ (self):
        pass
# --- Factorial ------------
    def factorial(self, n):
        j = 1
        for i in range (1, n + 1):
            j = j * i
        return j

# --- Sum of the first n numbers ----
    def naturalSum(self, n):
        j = 1
        for i in range (1, n + 1):
            j = j + i
        return j

# --- Primality test of a number ------------
    def testPrime(self, n):
        j = 0
        for i in range (1, n + 1):
            if (n% i == 0):
                j = j + 1
        if (j == 2):
            return True
        else:
            return False

# --- Primality test of two integers ------------
    def testPrims(self, n, m):

        # initialize the number of commons divisors
        commonDiv = 0
        for i in range (1, n + 1):
            if (n% i == 0 and m% i == 0):
                commonDiv = commonDiv + 1
        if commonDiv == 1:
            print ("The numbers", n, "and", m, "are co-primes")
        else:
            print ("The numbers", n, "and", m, "are not co-primes")

#---Multiplication table-------------
    def tableMult(self, k):
        for i in range (1,10):
            print (i, "x", k, "=", i * k)

# --- All multiplication tables of the numbers 1, 2, .., 9
    def allTables(self):
        for k in range (1,10):
            print ("\nthe multiplication table of:", k, "is:")
            for i in range (1,10):
                print (i, "x", k, "=", i * k)

# ----- list of divisors of an integer
    def listDivisor(self, n):
        # initialization of the list of divisors
        lDiv = []
        for i in range (1, n + 1):
            if (n% i == 0):
                lDiv.append (i)
        return lDiv

# ------ list of prime divisors of an integer ----------------
    def listPrimeDivisor(self, n):
        # initialization of the list of divisors
        lDiv = []
        for i in range (1, n + 1):
            if (n% i == 0 and self.testPrime(i)):
                lDiv.append(i)
        return lDiv

# Instantiation example
Comput= Computation ()
Comput.testPrims(13, 7)
```

```
print ("List of divisors of 18:", Comput.listDivisor(18))
print ("List of prime divisors of 18:", Comput.listPrimeDivisor(18))
Comput.allTables()
```

Q-4: Build flashcard using class in Python.

Build a flashcard using class in python. A flashcard is a card having information on both sides, which can be used as an aid in memoization. Flashcards usually have a question on one side and an answer on the other.

Example 1:

Approach:

- Create a class named FlashCard.
- Initialize dictionary fruits using **init**() method. Here you have to define fruit name as key and it's color as value. E.g., {"Banana": "yellow", "Strawberries": "pink"}
- Now randomly choose a pair from fruits by using *random* module and store the key in variable *fruit* and *value* in variable color.
- Now prompt the user to answer the color of the randomly chosen fruit.
- If correct print correct else print wrong.

Output:

```
welcome to fruit quiz
What is the color of Strawberries
pink
Correct answer
Enter 0, if you want to play again: 0
What is the color of watermelon
green
Correct answer
Enter 0, if you want to play again: 1
```

```
# write your code here
import random

class FlashCard:

    def __init__(self):
        self.__fruits = {
            'apple':'red',
            'banana':'yellow',
            'watermelon':'green',
            'strawberry':'pink',
            'guava':'green'
        }

    def quiz(self):

        while True:

import random
d = {
            'apple':'red',
            'banana':'yellow',
            'watermelon':'green',
            'strawberry':'pink',
            'guava':'green'
        }


print(fruit)
print(color)
```

```
     guava
     green
```

```
print( Welcome to the fruit quiz )
```

Q-5: Problem 5 based on OOP Python.

TechWorld, a technology training center, wants to allocate courses for instructors. An instructor is identified by name, technology skills, experience and average feedback. An instructor is allocated a course, if he/she satisfies the below two conditions:

- eligibility criteria:
  - if experience is more than 3 years, average feedback should be 4.5 or more
  - [illegible]