Name: Hemang Ranga

Roll no : 20BCS057

OOP_Assignment-2

Problem-1

```
import numpy as np
class Circle():
  def __init__(self, r):
    self.radius = r

  def perimeter(self):
    return 2*(np.pi)*(self.radius)

  def area(self):
    return (np.pi)*((self.radius)**2)

Acircle = Circle(5)
print(Acircle.perimeter())
print(Acircle.area())
```

```
 31.41592653589793
 78.53981633974483
```

Problem 2

```
class Faculty:

    def setData(self):
        self.Name = input("Enter the name of the Employee : ")
        self.Emp_ID = (input("Enter the employee ID : "))
        self.Branch = input("Enter the branch : ")
        self.Salary = int(input("Enter the salary : "))

    def getData(self):
      print(f"Name: ", {self.Name})
      print(f"Employee ID: ", {self.Emp_ID})
      print(f"Branch: ", {self.Branch})
      print(f"Salary: ", {self.Salary})

      #return self.Name, self.Emp_ID, self.Branch, self.Salary

Fac_list = []
for i in range(5):
    temp = Faculty()
    print(f"\nEnter the data for employee {i+1}  : ")
    temp.setData()
```

```
        temp.setData()
        Fac_list.append(temp)

    for i in range(5):
        print(f"\nData of employee {i+1}  : ")
        Fac_list[i].getData()
```

```
 Enter the data for employee 1  :
 Enter the name of the Employee : Kailash
 Enter the employee ID : k15
 Enter the branch : ECE
 Enter the salary : 50000

 Enter the data for employee 2  :
 Enter the name of the Employee : Divya
 Enter the employee ID : k8
 Enter the branch : CSE
 Enter the salary : 55000

 Enter the data for employee 3  :
 Enter the name of the Employee : Abhishek
 Enter the employee ID : k2
 Enter the branch : DSAI
 Enter the salary : 50000

 Enter the data for employee 4  :
 Enter the name of the Employee : Tarun
 Enter the employee ID : k24
 Enter the branch : ECE
 Enter the salary : 60000

 Enter the data for employee 5  :
 Enter the name of the Employee : Komal
 Enter the employee ID : k17
 Enter the branch : CSE
 Enter the salary : 45000

 Data of employee 1  :
 Name:  {'Kailash'}
 Employee ID:  {'k15'}
 Branch:  {'ECE'}
 Salary:  {50000}

 Data of employee 2  :
 Name:  {'Divya'}
 Employee ID:  {'k8'}
 Branch:  {'CSE'}
 Salary:  {55000}

 Data of employee 3  :
 Name:  {'Abhishek'}
 Employee ID:  {'k2'}
 Branch:  {'DSAI'}
 Salary:  {50000}

 Data of employee 4  :
 Name:  {'Tarun'}
 Employee ID:  {'k24'}
 Branch:  {'ECE'}
 Salary:  {60000}
```

```
 Data of employee 5  :
 Name:  {'Komal'}
 Employee ID:  {'k17'}
 Branch:  {'CSE'}
 Salary:  {45000}
```

Problem-3

```
class Account:
    def __init__(self, accNo, accHolder, amount):
        self.AccountNumber = accNo
        self.AccountHolder = accHolder
        self.AccountAmount = amount

    def deposit(self):
        depAmount = int(input("Enter the amount to deposit : "))
        self.AccountAmount += depAmount
        print(f"Amount deposited : {depAmount}")
        print(f"New Balance : {self.AccountAmount}")

    def withdraw(self):
        withAmount = int(input("Enter the amount to withdraw : "))
        if withAmount > self.AccountAmount:
            print("Your Account don't have sufficient ammount !!")
            return -1
        self.AccountAmount -= withAmount
        print(f"Amount withdraw : {withAmount}")
        print(f"New Balance : {self.AccountAmount}")
        return withAmount

    def checkBalance(self):
        print(f"Balance : {self.AccountAmount}")
        return self.AccountAmount

    def getDetails(self):
        print(f"Account Number : {self.AccountNumber}")
        print(f"Account Holder : {self.AccountHolder}")
        print(f"Balance : {self.AccountAmount}")
        return self.AccountNumber, self.AccountHolder, self.AccountAmount

N = Account(225, "Rajesh", 4300)
N.deposit()
withdraw = N.withdraw()
balance = N.checkBalance()
details = N.getDetails()
```

```
 Enter the amount to deposit : 50000
 Amount deposited : 50000
 New Balance : 54300
 Enter the amount to withdraw : 3800
 Amount withdraw : 3800
 New Balance : 50500
 Balance : 50500
 Account Number : 225
```

```
Account Holder : Rajesh
Balance : 50500
```

Problem-4

```
class student():
  def __init__(self,S_Name,S_USN,S_Marks):
    self.Student_Name = S_Name
    self.Student_USN = S_USN
    self.Student_Marks = S_Marks
  def Students_Name(self):
    print("Student Name: ", self.Student_Name)
  def Students_USN(self):
    print("Student USN: ", self.Student_USN)
  def Students_Marks(self):
    print("Student Marks: ",self.Student_Marks)
marks = []
marks.append(87)
marks.append(85)
marks.append(98)

s1 = student("Hemang", 35, marks)
s1.Students_Name()
s1.Students_USN()
s1.Students_Marks()
```

```
Student Name:  Hemang
Student USN:  35
Student Marks:  [87, 85, 98]
```

Problem-5

```
class Patient:
  def __init__(self, name, admission_date, symptoms : list, oxygen_level, discharge_date):
    self.name = name
    self.admission_date = admission_date
    self.symptoms = symptoms
    self.oxygen_level = oxygen_level
    self.discharge_date = discharge_date

class Data:
  def __init__(self):
    self.oxygen_support = []
    self.general_ward = []

  def Patient(self, patient):
    if (patient.oxygen_level < 90):
      self.oxygen_support.append(patient)
    else:
      self.general_ward.append(patient)

  def numberOxygen(self):
    print(f"Total number of patients needing oxygen support is {len(self.oxygen_support)}"
```

```
    def numberGeneral(self):
      print(f"Total number of patients in general ward is {len(self.general_ward)}")

    def totalPatients(self):
      print(f"Total number of patients is {len(self.general_ward)+len(self.oxygen_support)}"

    def patientInfo(self):
      print("Patients in oxygen support")
      for patient in self.oxygen_support:
        print("\n")
        print("Name:", patient.name)
        print("Admission Date:", patient.admission_date)
        print("symptoms:", patient.symptoms)
        print("oxygen level:", patient.oxygen_level)
        print("Discharge Date:", patient.discharge_date)
        print("\n")

      print("Patients in General ward")
      for patient in self.general_ward:
        print("Name:", patient.name)
        print("Admission Date:", patient.admission_date)
        print("symptoms:", patient.symptoms)
        print("oxygen level:", patient.oxygen_level)
        print("Discharge Date:", patient.discharge_date)
        print("\n")

patient1 = Patient("Suresh", "13/09/2021", ["Fever", "Coughing"], 79, "29/09/2021")
patient2 = Patient("Abdul", "15/09/2021", ["Headaches", "Fever"], 95, "1/10/2021")

db = Data()
db.Patient(patient1)
db.Patient(patient2)
db.patientInfo()
db.numberOxygen()
db.numberGeneral()
db.totalPatients()
```

```
    Patients in oxygen support


    Name: Suresh
    Admission Date: 13/09/2021
    symptoms: ['Fever', 'Coughing']
    oxygen level: 79
    Discharge Date: 29/09/2021


    Patients in General ward
    Name: Abdul
    Admission Date: 15/09/2021
    symptoms: ['Headaches', 'Fever']
    oxygen level: 95
    Discharge Date: 1/10/2021


    Total number of patients needing oxygen support is 1
```

```
Total number of patients in general ward is 1
Total number of patients is 2
```

Problem-6

```
class time():
  def __init__(self,hr=0,min=0,sec=0):
    self.hour = hr
    self.minute = min
    self.second = sec
  def dis(self):
    print("Time :",self.hour,":",self.minute,":",self.second)
t1 = time()
t1.dis()
t1.hour = 5
t1.minute = 43
t1.second = 25
t1.dis()
```

```
Time : 0 : 0 : 0
Time : 5 : 43 : 25
```

Problem-7

```
class Student:
    def __init__(self, n, a, r):
        self.name = n
        self.age = a
        self.rollNo = r

    @classmethod
    def compare(cls, s1, s2):
        if s1.age == s2.age:
            print("Age of the students are equal.")
        else:
            print("Age of the students are not equal.")

std1 = Student("Rahul", 21, 33)
std2 = Student("Rohit", 22, 74)

Student.compare(std1, std2)
```

```
Age of the students are not equal.
```

Problem-8

```
class Student:
    sem = 5
    institute = "MNIT Jaipur"

    def setData(self):
        self.name = input("Enter the name of the student : ")
```

```
        self.id = int(input("Enter the USN of student : "))

    def getInstanceData(self):
        print("Name : ", self.name)
        print("USN = ", self.id)

    @classmethod
    def getClassData(cls):
        print("Semester : ", cls.sem)
        print("Institute : ", cls.institute)

    @staticmethod
    def getExplanation():
        print("Class variables (sem, institute) are printed using ClassMethod and Instance


s1 = Student()
s1.setData()
s1.getInstanceData()
Student.getClassData()
Student.getExplanation()
```

```
Enter the name of the student : Kunal
Enter the USN of student : 60
Name :  Kunal
USN =  60
Semester :  5
Institute :  MNIT Jaipur
Class variables (sem, institute) are printed using ClassMethod and Instance variables
```

Problem-9

```
class Student_Uni:

  def __init__(self, Name, roll_number, semester, laptop_name, laptop_cpu, laptop_ram, lap
    self.name = Name
    self.roll_num = roll_number
    self.sem = semester
    self.laptop = self.Laptop(laptop_name, laptop_cpu, laptop_ram, laptop_hard_disk)


  class Laptop:
    def __init__(self, name, cpu, ram : int, hard_disk):
      self.name = name
      self.cpu = cpu
      self.ram = ram
      self.hard_disk = hard_disk

    def specifications(self):
      print("Laptop Name:", self.name)
      print("Laptop CPU:", self.cpu)
      print("Laptop RAM:", self.ram, "GB")
```

```
        print("Laptop Hard disk:", self.hard_disk)

    def disp(self):
      print("Name:", self.name)
      print("Roll Number:", self.roll_num)
      print("Sem:", self.sem)
      self.laptop.specifications()

x = Student_Uni("Hemang", 57 , 3, "DELL Inspiron 3593", "Intel(R) Core(TM) i5-1035G1 CPU @
x.disp()
```

```
Name: Hemang
Roll Number: 57
Sem: 3
Laptop Name: DELL Inspiron 3593
Laptop CPU: Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz
Laptop RAM: 8 GB
Laptop Hard disk: 256 GB SSD/1 TB HDD
```

✓ 0s completed at 10:06 PM