Set 7

Q1

Create a class customer with name, address as its member variables. In list of account holder’s information, find the account holder who live nearby (using pincodes). If a name of the person is given then, customers of the same bank lives nearby should be given as output.

In [ ]:

```
class Customer():

    def __init__(self, name, pincode):
        self.name = name
        self.pincode = pincode

    def reg(self):
        self.name=input("Enter name:")
        self.pincode=int(input("Enter pincode:"))

    def nearby(self, info):
        for i in info:
            if i.pincode == self.pincode and i.name != self.name:
                print("\n\nName: ", i.name, "\nPincode: ", i.pincode)


info = []
while True:
    print("1. add members \n2. find nearby users \n3. exit")
    x = int(input("Enter your choice: "))
    if x==1:
        a = Customer()
        a.reg()
        info.append(a)
    if x==2:
        name = input("Enter name: ")
        print("Near by users are: ")
        for i in info:
            if i.name == name:
                i.nearby(info)
            break
    if x==3:
        exit(0)
```

```
1. add members
2. find nearby users
3. exit
Enter your choice: 1
Enter name:Tanisha
Enter pincode:1234
1. add members
2. find nearby users
3. exit
Enter your choice: 1
Enter name:Yuvi
Enter pincode:1234
1. add members
2. find nearby users
3. exit
Enter your choice: 2
Enter name: Tanisha
Near by users are:


Name:  Yuvi
Pincode:  1234
1. add members
2. find nearby users
3. exit
Enter your choice: 3
1. add members
2. find nearby users
3. exit
```

Q2

Create a class that can calculate the IT the based on the loan and saving account transactions by the customer

- Take the total amount deposited through out the year in an array
- Deduct the expenditure on saving to the maximum of 1.5L.
- For the rest of the amount calculate 6% of IT till 10L
- 8% till 15L
- Above 15L make 10% as IT

In [3]:
```
class IT():

    def __init__(self, name, amt):
        self.name = name
        self.amt = amt

    def calc(self):
        if(self.amt <= 1000000):
            print("The IT amount for {} is Rs.{} only".format(self.name,
(self.amt*0.06)))
        elif(self.amt > 1000000 and self.amt <= 1500000):
            print("The IT amount for {} is Rs.{} only".format(self.name,
(self.amt*0.08)))
        else:
            print("The IT amount for {} is Rs.{} only".format(self.name,
(self.amt*0.1)))


b = IT("Ross", 900000)
c = IT("Chandler", 1600000)
d = IT("Monica", 1500000)
e = IT("Phoebe", 500000)
f = IT("Joey", 2000000)


b.calc()
c.calc()
d.calc()
e.calc()
f.calc()
```

```
The IT amount for Ross is Rs.54000.0 only
The IT amount for Chandler is Rs.160000.0 only
The IT amount for Monica is Rs.120000.0 only
The IT amount for Phoebe is Rs.30000.0 only
The IT amount for Joey is Rs.200000.0 only
```

Q3

Implement the following scenario using Inheritance. Create a class bank make appropriate variables, assign customer class as child of bank. Create classes Saving and Loan and Current account. Make all these classes as child of bank. Maintain transaction function in all these class to track the tractions

In [5]:
```
class Bank:
    def __init__(self,balance):
        self.balance=balance



class Customer(Bank):
    def __init__(self,balance):
        self.balance=balance



class Saving(Bank):
    def __init__(self,balance):
        self.balance=balance

    def deposit(self,deposit):
        self.balance+=deposit

    def withdraw(self,withdrawl):
        self.balance-=withdrawl



class Current(Bank):
    def __init__(self,balance):
        self.balance=balance

    def deposit(self,deposit):
        self.balance+=deposit

    def withdraw(self,withdrawl):
        self.balance-=withdrawl



class Loan(Bank):
    def __init__(self,balance):
        self.balance=balance

    def deposit(self,deposit):
        self.balance+=deposit

    def withdraw(self,withdrawl):
        self.balance-=withdrawl
```

In [6]:
```
cust1 = Saving(1000)
cust1.balance
```

Out[6]: 1000

In [7]:
```
cust1.deposit(200)
cust1.balance
```

Out[7]: 1200

Q4

In continuation with the above bank class. Now create scoreboard for the customers. Function name of score same in all three classes but scores the point according to the account type.

Saving account : Score 1 pt for each 2k credit and deduct .25 for each 2k debit, 1 point for consistent balance of 10k

Current account: Score 1 pt for each 2k credit 1 point for consistent balance of 10k, reduce 1 pt for overdue more than 25k

Loan amount: Give 1 point for each on time repay and reduce 0.5 points for penalty payment.

In [9]:
```
class Bank:

    balance=0
    withdrawl=0
    deposit=0

    def __init__(self,balance):
        self.balance=balance



class Customer(Bank):
    def __init__(self,balance):
        self.balance=balance



class Saving(Bank):
    score=0
    def __init__(self,balance):
        self.balance=balance
        self.score=0

    def score1(self,deposit):
        if(self.deposit>0):
            self.score+=deposit/2000
        if(self.balance>=10000):
            self.score+=1
        print(self.score)

    def score2(self,withdrawl):
        if(withdrawl>0):
            self.score-=withdrawl/8000
        if(self.balance>=10000):
            self.score+=1
        print(self.score)

    def deposit(self,deposit):
        self.deposit=deposit
        self.balance+=deposit
        self.score1(deposit)

    def withdraw(self,withdrawl):
        self.withdrawl=withdrawl
        self.balance-=withdrawl
        self.score2(withdrawl)



class Current(Bank):
    score=0
    def __init__(self,balance):
        self.balance=balance
        self.score=0

    def score1(self,deposit):
```

```
        if(self.deposit>0):
            self.score+=deposit/2000
        if(self.balance>=10000):
            self.score+=1
        print(self.score)

    def score2(self,withdrawl):
        if(self.balance<25000):
            self.score-=1
        if(self.balance>=10000):
            self.score+=1
        print(self.score)

    def deposit(self,deposit):
        self.deposit=deposit
        self.balance+=deposit
        self.score1(deposit)

    def withdraw(self,withdrawl):
        self.withdrawl=withdrawl
        self.balance-=withdrawl
        self.score2(withdrawl)



class Loan(Bank):
    score=0
    def __init__(self,loan_amount):
        self.loan_amount=loan_amount
        self.score=0

    def repay(self,deposit):
        self.deposit=deposit
        self.loan_amount-=deposit
        self.score1()

    def score1(self):
        if(self.loan_amount==0):
            self.score+=1
        else:
            self.score-=0.5
        print(self.score)
```

In [10]:
```
cust1 = Saving(20000)
cust1.withdraw(2000)
```

```
0.75
```

In [11]:
```
cust1.deposit(2000)
```

```
2.75
```

In [12]:
```
cust2 = Loan(20000)
cust2.repay(20000)
```

```
1
```

1. Allow the customers to rate the services of the bank, give (display)customer a choice of service which they want to rate, and allow them to rate. Display the total score given by the particular customer and as the whole. (As an advancement take the score along with date display a chart with sum of scores on each period for each service).

In [13]:

```
from datetime import datetime as d

li=[0,0,0,0]
choice=0

while(True):
    choice=int(input(('\nENTER THE NUMBER ACCORDING TO THE LIST TO GIVE R
EVIEWS->\n1 || DEPOSIT SERVICE\n2 || WITHDRAWL SERVICE\n3 || LOAN SANCTIO
N\n4 || LOAN REPAYMENT\nEXIT || 0\n')))
    if(choice==0):
        break
    rating=int(input('\nEnter the rating out of 10  '))
    li[choice-1]+=rating
    num1,num2=choice,rating
    date = d.now()
    if(num1==1):
        print('\n\nDEPOSIT SERVICE GOT THE RATING OF {}/10'.format(num2))
    if(num1==2):
        print('\n\nWITHDRAWL SERVICE GOT THE RATING OF {}/10'.format(num2
))
    if(num1==3):
        print('\n\nLOAN SANCTION SERVICE GOT THE RATING OF {}/10'.format(
num2))
    if(num1==4):
        print('\n\nLOAN REPAYMENT SERVICE GOT THE RATING OF {}/10'.format
(num2))
    print(date.strftime("DATED ON %Y-%m-%d"))

print('\n\n---------------------------------THE TOTAL SCOREBOARD--------
-----------------------------------')
print('DEPOSIT SERVICE  {}'.format(li[0]))
print('WITHDRAWL SERVICE  {}'.format(li[1]))
print('LOAN SANCTION SERVICE  {}'.format(li[2]))
print('LOAN REPAYMENT SERVICE  {}'.format(li[3]))
```

```
ENTER THE NUMBER ACCORDING TO THE LIST TO GIVE REVIEWS->
1 || DEPOSIT SERVICE
2 || WITHDRAWL SERVICE
3 || LOAN SANCTION
4 || LOAN REPAYMENT
EXIT || 0
1

Enter the rating out of 10  4


DEPOSIT SERVICE GOT THE RATING OF 4/10
DATED ON 2021-02-05

ENTER THE NUMBER ACCORDING TO THE LIST TO GIVE REVIEWS->
1 || DEPOSIT SERVICE
2 || WITHDRAWL SERVICE
3 || LOAN SANCTION
4 || LOAN REPAYMENT
EXIT || 0
0


----------------------------------THE TOTAL SCOREBOARD------------------
-------------------------
DEPOSIT SERVICE  4
WITHDRAWL SERVICE  0
LOAN SANCTION SERVICE  0
LOAN REPAYMENT SERVICE  0
```

In [ ]: