Python Practical (208) – Assignment-6

'''

Q.1. Write a Menu Driven (Menu: PUSH, POP, PEEP, PRINT, EXIT) program in Python to implement stack operations on a stack of integers using a class consisting of attributes Stack (a List consisting of elements of the Stack) and TOP, and methods PUSH, POP, PEEP and PRINT.

'''

```
class Stack :
    def __init__(self):
        self.lst = []

    def is_empty(self):
        return self.lst == []

    def push(self,val):
        self.lst.append(val)

    def pop(self):
        return self.lst.pop()

    def peep(self):
        print(self.lst[len(self.lst) - 1])

    def print(self):
        print(self.lst)



s1 = Stack()
while True:
    print("-------Operation-------")
    print("1 Push")
    print("2 Pop")
    print("3 Peep")
    print("4 Print")
```

```
print("5 Exit")
op = input("Select Option : ")
try:
   op = int(op)
except:
   print("Please enter Numbers only ")
   continue

if(op == 1):
   s1.push(int(input("Enter Value : ")));
elif(op == 2):
   if(s1.is_empty()):
      print("Stack is EMpty...")
   else:
      s1.pop()
elif(op == 3):
   if(s1.is_empty()):
      print("Stack is EMpty...")
   else:
      s1.peep()
elif(op == 4):
   if(s1.is_empty()):
      print("Stack is EMpty...")
   else:
      s1.print()
elif(op == 5):
   print("Thank You using STack Program")
   break
else:
   print("Please Enter Valid Selection....")
```

'''

Q.2. Write a Menu Driven (Menu: INSERT, DELETE, PRINT, EXIT) program in Python to implement Simple Queue Operations on a simple queue of integers using a class consisting of attributes SQueue (a List consisting of elements of the Simple Queue), Front and Rear and methods INSERT, DELETE and PRINT.

'''

```
class Queue:
    def __init__(self):
        self.lst = []

    def insert(self,val):
        self.lst.append(val)

    def is_empty(self):
        if(len(self.lst) == 0):
            return True
        else:
            return False

    def delete(self):
        del(self.lst[0])

    def print(self):
        print(self.lst)


q1 = Queue()

while True:
    print("----Menu-----")
    print("1 Insert")
    print("2 Delete")
    print("3 Print")
    print("4 Exit")
```

```
op = input("Select Menu : ")
try:
    op = int(op)
except:
    print("Please enter numbers only ")
    continue

if(op == 1):
    q1.insert(int(input("ENter Value : ")))

elif(op == 2):
    if(q1.is_empty() == False):
        q1.delete()
    else:
        print("Queue is EMpty...")

elif(op == 3):
    if(q1.is_empty() == False):
        q1.print()
    else:
        print("Queue is EMpty...")

elif(op == 4):
    print("Thank You for using Queue Program....")
    break

else:
    print("Please enter valid Selection....")
```

'''

Q.3. Write a Python program to do the following:

□ Define a class myDate consisting of attributes day, month and year and following methods:

o addDays(myDate, int) where the 1st argument is a myDate class type and 2nd argument is an integer type with default value as 0. The method should add/subtract int days (depending on the integer i.e. add if positive and subtract if negative) to/from the myDate and return new date of myDate type.

o formatDate(myDate, formatString) where the 1st argument is a myDate class type and 2nd argument is a format string of string type. The method will return the date in the given format. Consider only the following format strings in this program. 'dd-mm-yyyy', 'mm-dd-yyyy' and 'yyyy-mm-dd'.

□ Implement the above

'''

```
import datetime as dt

class myDate:
    def __init__(self,day,month,year):
        self.date = dt.datetime(year,month,day)

    def addDays(self,d):
        self.date = self.date + dt.timedelta(days=d)
        return self

    def formatDate(self,fs):
        if(fs == "dd-mm-yyyy"):
            return self.date.strftime("%d-%m-%Y")
        elif(fs == "mm-dd-yyyy"):
            return self.date.strftime("%m-%d-%Y")
        elif(fs == "yyyy-mm-dd"):
            return self.date.strftime("%Y-%m-%d")
        else:
            return self.date.strftime("%d-%m-%Y")
```

```
d = int(input("Date Day : "))
m = int(input("Date Month : "))
y = int(input("Date Year : "))

md = myDate(d,m,y)

while True:
    print("----Menu------")
    print("1 addDays")
    print("2 formatDate")
    print("3 Exit")

    op = int(input("ENter operation : "))

    if(op == 1):
        md.addDays(int(input("ENter Days : ")))
    elif(op == 2):
        print("----Menu for FormatDate------")
        print("1 dd-mm-yyyy")
        print("2 mm-dd-yyyy")
        print("3 yyyy-mm-dd")

        fdop = int(input("Select FormatDate type : "))
        if(fdop == 1):
            print("Date : ",md.formatDate("dd-mm-yyyy"))
        elif(fdop == 2):
            print("Date : ",md.formatDate("mm-dd-yyyy"))
        elif(fdop == 3):
            print("Date : ",md.formatDate("yyyy-mm-dd"))
        else:
            print("Date : ",md.formatDate("dd-mm-yyyy"))

    elif(op == 3):
        print("Thank You for using myDate Class....")
        break
```

```
else:
    print("Please enter valid Choice ....")
```

'''

Q.4. Write a Python program to overload:

□ the '+' operator for the string as under:

The 2 strings should be merged in such a way that the result will contain characters one by one first from the 1st string and then from the 2nd string as shown in the example given below.

Str1='VNSGU'

Str2='SURAT'

Then Str1+Str2 = 'VSNUSRGAUT' (i.e. here characters in Italics are from Str1 and rest from Str2)

□ the <, <=, >, >= and == operators for the strings as under:

Compare sum of ACII values of all the characters in both the strings and then compare the results. Return True if the sum for the 1st string is more than that for the 2nd string for the '>' operator and False otherwise. Similarly do for other operators.

'''

```
class String1:
    def __init__(self,word):
        self.word = word



    def __add__(self,other):
        l1 = len(self.word)
        l2 = len(other.word)
        ans = ""
        i=0
        j=0
        while(i<l1)and(j<l2):
            ans += self.word[i]
            ans += other.word[j]
            i+=1
            j+=1
        while(i<l1):
            ans += self.word[i]
            i+=1
        while(j<l2):
```

```
        ans += other.word[j]
        j+=1

    return ans

def __lt__(self,other):
    if self.acount < other.acount:
        return True
    else:
        return False

def __gt__(self,other):
    if self.acount > other.acount:
        return True
    else:
        return False

def __le__(self,other):
    if self.acount <= other.acount:
        return True
    else:
        return False

def __ge__(self,other):
    if self.acount >= other.acount:
        return True
    else:
        return False

def _eq__(self,other):
    if self.acount == other.acount:
        return True
    else:
        return False
```

```
s1 = String1("VNSGU")
s2 = String1("Surat")

s1.acount = 0
for i in s1.word:
   s1.acount += ord(i)



s2.acount = 0
for i in s2.word:
   s2.acount += ord(i)

print("Sum : ",s1 + s2)
print("Greater Than : ",s1 > s2)
print("Less Than : ",s1 < s2)
print("Greater Than Or Equal To : ",s1 >= s2)
print("Less Than Or Equal To : ",s1 <= s2)
print("Equal To : ",s1 == s2)
```

'''

Q.5. Write a Python program for the following:

□ Define a class accountHolder consisting of attributes accNo, accName, accEmail. It consists a method dispDetails to display the details in appropriate format.

□ Inherit two classes viz. depositAccount (accountBalance) and loanAccount (loanAmount, EMI, loanBalance). The depositAccount contains methods debitAmt(amt)-which debits amt amount from the accountBalance, creditAmt(amt)-which credits the amt amount to the accountBalance and dispTrans()-which displays the whole transaction in appropriate format showing initial balance, debit/credit amount and final amount.

'''

```
class accountHolder:
    def __init__(self,accNo,accName,accEmail):
        self.accNo = accNo
        self.accName = accName
        self.accEmail = accEmail

    def dispDetails(self):
        print("AccNo : ",self.accNo)
        print("AccName : ",self.accName)
        print("AccEmail : ",self.accEmail)



class depositAccount(accountHolder):
    def __init__(self,accBalance,accNo,accName,accEmail):
        accountHolder.__init__(self,accNo,accName,accEmail)
        #super().__init__(accNo,accName,accEmail)
        self.accBalance = accBalance

    def debitAmt(self,amt):
        self.accBalance = self.accBalance - amt
        self.dispTrans(amt,"debit")

    def creditAmt(self,amt):
        self.accBalance = self.accBalance + amt
```

```
        self.dispTrans(amt,"credit")

    def dispTrans(self,amt,transType):
        self.dispDetails()
        if(transType == "debit"):
            print("Initial Balance : ",self.accBalance+amt)
            print("Debit Amount : ",amt)
            print("FInal Balance : ",self.accBalance)
        elif(transType == "credit"):
            print("Initial Balance : ",self.accBalance-amt)
            print("Credit Amount : ",amt)
            print("FInal Balance : ",self.accBalance)



class loanAccount(accountHolder):
    def __init__(self,loanAmount,EMI,loanBalance,accNo,accName,accEmail):
        accountHolder.__init__(self,accNo,accName,accEmail)
        self.loanAmount = loanAmount
        self.EMI = EMI
        self.loanBalance = loanBalance

    def depositAmt(self,amt):
        self.loanBalance = self.loanBalance - amt
        self.dispTrans(amt)

    def dispTrans(self,amt):
        self.dispDetails()
        print("Initial Loan Amount : ",self.loanAmount)
        print("Initial EMI : ",self.EMI)
        print("Initial Loan Balance : ",self.loanBalance + amt)
        print("Deposit Amount in Loan : ",amt)
        print("Final Loan Balance : ",self.loanBalance)



accBalance = 10000
```

```
accNo = "111111111111"
accName = "Chandani SIngh"
accEMail = "chandani@gmail.com"
da = depositAccount(accBalance,accNo,accName,accEMail)

loanAmount = 10000
EMI = 1000
loanBalance = 10000
la = loanAccount(loanAmount,EMI,loanBalance,accNo,accName,accEMail)

while True:
    print("----Menu for Account-----")
    print("1 Deposit Account")
    print("2 Loan Account")
    print("3. Exit")
    op = int(input("ENter Option : "))
    if(op == 1):
        while True :
            print("----Menu Deposit Account-----")
            print("1 Debit Amt")
            print("2 Credit Amt")
            print("3 Exit")
            opd = int(input("Select Option : "))
            if(opd == 1):
                amt = int(input("Amount : "))
                da.debitAmt(amt)
            elif(opd == 2):
                amt = int(input("Amount : "))
                da.creditAmt(amt)
            elif(opd == 3):
                break
            else:
                print("Please select Valid Choice....")
    elif(op == 2):
        while True :
            print("----Menu Loan Account-----")
```

```
        print("1 Deposit Amt")
        print("2 Exit")
        opd = int(input("Select Option : "))
        if(opd == 1):
            amt = int(input("Amount : "))
            la.depositAmt(amt)
        elif(opd == 2):
            break
        else:
            print("Please select Valid Choice....")
elif(op == 3):
    break
```

'''

Q.6. Write a Python program to demonstrate multi-level, multiple inheritance and MRO.

'''

```
class Department:
    def __init__(self,dname,address):
        self.dname = dname
        self.address = address
    def dispDept(self):
        print("Department Name : ",self.dname)
        print("Address : ",self.address)

class Course1(Department):
    def __init__(self,c1name,c1year,dname,address):
        Department.__init__(self,dname,address)
        self.c1name = c1name
        self.c1year = c1year
    def dispCourse1(self):
        print("Course Name : ",self.c1name)
        print("Course Year : ",self.c1year)

class Course2(Department):
    def __init__(self,c2name,c2year,dname,address):
        Department.__init__(self,dname,address)
        self.c2name = c2name
        self.c2year = c2year
    def dispCourse2(self):
        print("Course Name : ",self.c2name)
        print("Course Year : ",self.c2year)

class Exam(Course1,Course2):
    def __init__(self,examdate,c1name,c1year,c2name,c2year,dname,address):
        Course1.__init__(self,c1name,c1year,dname,address)
        Course2.__init__(self,c2name,c2year,dname,address)
        self.examdate = examdate
    def dispExam(self):
```

```
        self.dispDept()
        self.dispCourse1()
        self.dispCourse2()
        print("Exam Date : ",self.examdate)


e = Exam("16-06-2022","MCA","2","PGDCA","1","DCS-VNSGU","Surat,Gujarat,India")
e.dispExam()
```

'''

Q.7. Write a Python program to demonstrate polymorphism using appropriate example.

'''

```
print("\nExamples of user-defined polymorphic functions : \n")

def add(x, y, z = 0):
    return x + y+z

# Driver code
print(add(2, 3))
print(add(2, 3, 4))




print("\nPolymorphism with class methods: \n")



class India():
    def capital(self):
        print("New Delhi is the capital of India.")

    def language(self):
        print("Hindi is the most widely spoken language of India.")

    def type(self):
        print("India is a developing country.")

class USA():
    def capital(self):
        print("Washington, D.C. is the capital of USA.")

    def language(self):
        print("English is the primary language of USA.")
```

```
    def type(self):
        print("USA is a developed country.")

obj_ind = India()
obj_usa = USA()
for country in (obj_ind, obj_usa):
    country.capital()
    country.language()
    country.type()



print("\nPolymorphism with Inheritance: \n")


class Bird:
  def intro(self):
    print("There are many types of birds.")

  def flight(self):
    print("Most of the birds can fly but some cannot.")

class sparrow(Bird):
  def flight(self):
    print("Sparrows can fly.")

class ostrich(Bird):
  def flight(self):
    print("Ostriches cannot fly.")

obj_bird = Bird()
obj_spr = sparrow()
obj_ost = ostrich()

obj_bird.intro()
obj_bird.flight()
```

```
obj_spr.intro()
obj_spr.flight()

obj_ost.intro()
obj_ost.flight()
```