SIKSHA 'O' ANUSANDHAN
DEEMED TO BE UNIVERSITY

Admission Batch: 2019 **Session:** 2021-22

Laboratory Record

Programming in Python (CSE 3142)

Submitted by

Name: Saswat Mohanty

Registration No.: 1941012407

Branch: Computer Science and Engineering

Semester: 5th Section: 'D'

Department of Computer Science & Engineering

Faculty of Engineering & Technology (ITER)

Jagamohan Nagar, Jagamara, Bhubaneswar, Odisha - 751030

INDEX

Sl. No	Name of Program	Page No	Remarks
1.	Minor Assignment - 1	1-8	good 06/12/21
2.	Minor Assignment - 2	9-21	good 06/12/21
3.	Minor Assignment - 3	22-39	good 06/12/21
4.	Minor Assignment - 5	40-45	good 21/12/21
5.	Minor Assignment - 6	46-57	good 21/12/21
6.	Minor Assignment - 7	58-68	good 21/12/21
7.	Minor Assignment - 8	69-77.	good 04/01/22
8.	Minor Assignment - 9	78-88	
9.	Minor Assignment - 10	89-97	

Minor Assignment - 10

Classes - I

Q1) Write a python class to convert an integer to a roman numeral.

Program :-

```
class roman:
  def intRoman(self, n):
    num = [1000,900, 500,400,100,90,50,40, 10, 9, 5,4,1]
    rom = ['M', 'CM', 'D', 'CD', 'C', 'XC', 'L', 'XL', 'X', 'IX', 'V',
           'IV', 'I' ]
    res = ()
    for i in range (len(num)):
      temp = int (n/ num[i])
      res += str (rom[i] * temp )
      n -= num[i] * temp
    print (res)
roman. intRoman(990)
```

output :-

```
CM XC
```

Q2) write a python class to convert a roman numeral to an integer.

Program :-

```
class integer:
 def romantoint (self, str):
   rom = {'I': 1, 'V': 5, 'X':10, 'L': 50, 'C': 100, 'D': 500, 'M':1000}
   num = 0
```

```
    for i in range(len(str)):
        if i>0 and rom[str[i]] > rom[str[i-1]]:
            num += rom[str[i]] - 2*rom[str[i-1]]
        else:
            num += rom[str[i]]
    return num
print(integer().romanToInt('CL'))
```

output :-

150

Q3) Write a python class to get all possible unique subsets from a set of distinct integers.

Program :-

```
class subset:
    def subset(self, sset):
        return self.subsets([], sorted(sset))
    def subsets(self, current, sset):
        if sset:
            return self.subsets(current, sset[1:]) + self.subsets(
                current + [sset[0]], sset[1:])
        return [current]
print(subset().subset([4,5,6]))
```

output :-

[[], [6], [5], [5,6], [4], [4,6], [4,5], [4,5,6]]

Q4) Write a python class BankAccounts to model a bank accounts maintenance system:

a) To create bank account (name, account number),

b) To deposit money & withdraw money

c) To check minimum acc balance before withdrawal & display

message when withdraw amount violates the minimum acc balance condition.

d) Give options to open, deposit, withdraw, & display acc balance

Program :-

```
class BankAccount:
  account_number = 0
  name = " "
  balance_amount = 0
  def account_creation(self):
    self.account_number = int(input("Enter the account number \t "))
    self.name = input("Enter the amount holder name \t")
  def amount_deposition(self, amount):
    self.balance_amount = self.balance_amount + amount
  def amount_withdrawn(self, amount):
    if(amount <= self.balance_amount):
      self.balance_amount -= amount
    else:
      print("Less Amount")
  def display_account(self):
    print("Name: ", self.name, "\t Account number: ", self.account_number, "\t Balance: ", self.balance_amount).
ch = ' '
acc = BankAccount()
while ch != 5:
  print("\t MAIN MENU \t")
  print("\t 1. NEW ACCOUNT \t")
  print("\t 2. DEPOSIT AMOUNT \t")
```

```
print("\t 3. WITHDRAW AMOUNT \t")
print("\t 4. BALANCE ENQUIRY \t")
print("\t 5. EXIT \t")
ch = int(input())
if ch == 1:
    acc.account_creation()
elif ch == 2:
    print("list")
    acc.amount_deposition(int(input()))
elif ch == 3:
    acc.amount_withdrawn(int(input()))
elif ch == 4:
    acc.display_account()
elif ch == 5:
    break
```

output :-

```
MAIN MENU
1. NEW ACCOUNT
2. DEPOSIT AMOUNT
3. WITHDRAW AMOUNT
4. BALANCE ENQUIRY
5. EXIT

1
Enter the account number      445676
Enter the account holder name      Ram Singh

MAIN MENU
1. NEW ACCOUNT
2. DEPOSIT AMOUNT
```

3. WITHDRAW AMOUNT
4. BALANCE ENQUIRY
5. EXIT

Q5) Define a class Item that keeps track of items available in the shop. The class should contain the following data members:

name – Name of the item
price - Price of the item
quantity – Quantity of the item available in stock

The class should support the following methods:

1) __init__ for initializing the data members
2) purchase for updating the quantity after a purchase made by the customer. The method should take the number of items to be purchased as an input
3) increasestock for updating the quantity of an item for which new stock has arrived. The method should take the number of items to be added as an input.
4) display that displays information about an item.

Program :-

```
class Item:
  def __init__(self, name, price, quantity):
    self.name = name
    self.price = price
    self.quantity = quantity

  def purchase(self, item):
    if(item > self.quantity):
      print("Insufficient Quantity")
    else:
      self.quantity -= item
```

```
def addStock (self, item):
    self.quantity += item
def __str__(self):
    return 'Name: ' + self.name + '\nPrice: ' + str(self.price) + '\nQuantity: ' + str(self.quantity)
i = Item ("Jam", 200, 200)
print(i)
i.addStock (200)
print ('\n', i)
i.purchase (100)
print ('\n', i)
```

Output :-

```
Name: Jam
Price: 200
Quantity: 200

Name: Jam
Price: 200
Quantity: 400

Name: Jam
Price: 200
Quantity: 300
```

Q6) Define a class Student that keeps track of academic record of students in a school. The class should contain the following data members :

rollNum – Roll number of students
name – Name of student

marksList – List of marks in five subjects
stream – A: Arts, C: commerce, S: science
percentage – Percentage computed using marks
grade – Grade in each subject computed using marks
division – division computed on the basis of overall percentage.

The class should support the following methods:

1) __init__ for initializing the data members
2) setMarks to take marks for five subjects as an input from the user.
3) getstream for accessing the stream of the student.
4) percentage for computing the overall percentage for the student.
5) gradeGen that generates grades for each student in each course on the basis of the marks obtained.

criteria for computing the grade is as follows:

Marks	Grade
>=90	A
<90 and >=80	B
<80 and >=65	C
<65 and >=40	D
<40	E

Program:-

```
class Student:
    def __init__(self, rollNumber, name, marksList, stream,
                 percentage, grade, division):
        self.rollNumber = rollNumber
        self.name = name
```

```
        self.marksList = marksList
        self.stream = stream
        self.percentage = percentage
        self.grade = grade
        self.division = division
    def setMarks(self):
        self.marksList = eval(input("Enter the marks in the
                                     list: "))

    def getStream(self):
        print(self.stream)
    def percentage(self):
        self.percentage = sum(self.marksList)/
                                  len(self.marksList)

    def gradeGen(self):
        if self.percentage >= 90:
            self.grade = 'A'
        elif self.percentage >= 80:
            self.grade = 'B'
        elif self.percentage >= 65:
            self.grade = 'C'
        elif self.percentage >= 40:
            self.grade = 'D'
        else:
            self.grade = 'E'
    def __str__(self):
        return 'ROLL NO.: ' + str(self.rollNumber) + 'in Name: ' +
```

```
- self.name + '\n Marks List' + str(self.marksList)
  + '\n stream: ' + self.stream + '\n Percentage: '
  + str(self.percentage) + '\n Grade: ' + self.gr-
  -ade + '\n Division: ' + self.division

S = Student(2660, "Saswat", [], "CSE", 0, 'E', "1st")
S.setMarks()
S.Percentage()
S.gradeGen()
print(S)
```

Output :-

```
Enter the marks in the list: [80]
Roll No. : 2660
Name: Saswat
Marks List: [80]
Stream : CSE
percentage : 80.0
Grade : B
Division : 1st.
```