Python Lab file

Made by :- SACHIN RAO

```
# 1.Write a program illustrating class definition and accessing class members.
class First :
    class_members = "this is a class member"
    def __init__(self,constructer):
        self.cons = constructer
    def display(self):
        return f"this is {self.cons}  in display fucntion"
first = First("default constructer called")
print(first.display())
# in  below line class memebers are aCESSED is
print(First.class_members)
```

```
# 2.	Write a program to implement default constructor, parameterized constructor, and destructor.

class Addition:
    fnum = 0
    snum = 0
    finalanswer = 0


    # parameterized constructor
    def __init__(self, f, s):
        self.fnum = f

        self.snum = s


    def display(self):

        print("First number = " + str(self.fnum))

        print("Second number = " + str(self.snum))

        print("Addition of two numbers = " + str(self.finalanswer))


    def calculate(self):

        self.finalanswer = self.fnum + self.snum





# creating object of the class

# this will invoke parameterized constructor

obj = Addition(1000, 2000)


# perform Addition

obj.calculate()


# display result

obj.display()

# Python program to illustrate destructor

class Employee:


# Initializing

def __init__(self):

print('Employee created.')
```

```
# Deleting (Calling destructor)
def __del__(self):
print('Destructor called, Employee deleted.')

obj = Employee()
del obj
```

#3. Create a Python class named Rectangle constructed by a length and width

```
class Rectangle1:
   area1 = 0
   def __int__(self , length , width):
      self.l = length
      self.w = width
   def area (self):
      return self.l * self.w



rec = Rectangle1(10,7)
print("area of ractangle is " , rec.area)
```

'''4. Create a class called Numbers, which has a single class attribute called MULTIPLIER, and a constructor which takes the parameters x and y (these should all be numbers).

a. Write an instance method called add which returns the sum of the attributes x and y.

b. Write a class method called multiply, which takes a single number parameter a and returns the product of a and MULTIPLIER.

c.	Write a static method called subtract, which takes two number objects, b and c, and returns b - c.

d.	Write a method called value which returns a tuple containing the values of x and y.

'''

```
class Numbers :
    MULTIPLIER = 1
    def __init__(self , x, y):
        self.x = x
        self.y = y

    def sum(self):
        return self.x + self.y
    @classmethod
    def multiplier(cls,a):
        cls.a =a
        return cls.MULTIPLIER * cls.a
    @staticmethod
    def subtract (b , c) :
        return b-c
    def value (self) :
        return ( self.x, self.y)

c = int(input("enter parameter x"))
w = int(input("enter parameter y "))
num = Numbers(c , w)
print("sum is" , num.sum())
print("multiplication is " , num.multiplier(10))
print("subtraction is ", num.subtract(20 , 10))

print(f"tuple of x and y is {num.value()}")
```

```
"""5.     Create a class named as Student to store the name and marks in three subjects.

Use List to store the marks.

a.        Write an instance method called compute to compute total marks and average marks of a student.

b.        Write a method called display to display student information.

"""


class Student :

    totalmarks = 0

    avgmarks = 0

    c=0

    name =  input("enter name of student")

    marks = [int(x) for x in input("enter marks of subjects").split(",")]


    def compute(self):

        for i in self.marks:

            self.c+=1

            self.totalmarks += i

            self.avgmarks =  self.totalmarks/self.c

    def display(self):

        print("avg marks are", self.avgmarks)

        print("total marks are", self.totalmarks)

obj = Student()
```

```
obj.compute()

obj.display()
```

```
'''7.Create a Python class named Circle constructed by a radius.

     Use a class variable to define the value of constant PI.

a.          Write two methods to be named as area and circum to compute the area and the perimeter of a
circle respectively

    by using class variable PI.

b.          Write a method called display to print area and perimeter.

'''

class Circle :

    pi = 3.14

    def __init__(self , radius):

        self.radius = radius

    def area (self) :

        self.circle_area = self.pi*self.radius*self.radius

    def crcum (self) :

        self.circle_perimeter = 2*self.pi*self.radius

    def display(self):

        print(f"Perimeter of Circle is {self.circle_perimeter}")

        print(f"Area of circle is {self.circle_area:.2f}")

obj = Circle(3)

obj.area()

obj.crcum()

obj.display()
```

```
'''8.	Create a class called String that stores a string and all its status details
	such as number of uppercase letters, lowercase letters, vowels ,consonants and space in instance variables.
a.	Write methods named as count_uppercase, count_lowercase, count_vowels, count_consonants
   and count_space to count corresponding values.
b.	Write a method called display to print string along with all the values computed by methods in (a).
'''
class String :
    string= input("enter a string")
    uppercase =0
    lowercase =0
    vowels = 0
    consonants =0
    space = 0
    def count_uppercase(self):
        for u in  self.string :
            if u.isupper() == 'True' :
                self.uppercase +=1

        return self.uppercase
    def count_lowercase(self):
        for l in self.string:
            if l.islower() == 'True':
                self.lowercase += 1
    def count_vowels (self):
        for ch in self.string:
            if (ch == 'a' or ch == 'e' or ch == 'i' or ch == 'o' or ch == 'u' or ch == 'A' or ch == 'E' or ch == 'I' or ch
== 'O' or ch == 'U'):
                self.vowels += 1

    def count_consonants(self):
        for ch in self.string:
            if (ch != 'a' or ch != 'e' or ch != 'i' or ch != 'o' or ch != 'u' or ch != 'A' or ch != 'E' or ch != 'I' or ch !=
'O' or ch != 'U'):
                self.consonants +=1



    def count_space(self):
        for ch in self.string:
            if ch == " " :
                self.space +=1

    def display(self):
        print("uppercase are " , self.uppercase)
        print("lowercase are ", self.lowercase)
        print("vowels are ", self.vowels)
        print("space are ", self.space)
        print("consonants are ", self.consonants)
obj = String()
obj.count_uppercase()
obj.count_lowercase()
obj.count_vowels()
obj.count_consonants()
obj.count_space()
obj.display()

--------------------------------------------------------------------------------------------------------------------------------------------


'''
#9.     Write a program that has a class called Fraction with attributes numerator and denominator.
a.      Write a method called getdata to enter the values of the attributes.
b.      Write a method show to print the fraction in simplified form.

'''
class Fraction :

  numerator = 0
  denominator = 0
  def getdata(self) :
    self.numerator = int(input("Enter Numerator"))
    self.denominator = int(input("Enter denomenator"))
  def show(self):
    a= self.numerator
    b = self.denominator
    """Calculate the Greatest Common Divisor of a and b.

      Unless b==0, the result will have the same sign as b (so that when
      b is divided by it, the result comes out positive).
      """
    while b:
            a, b = b, a % b

        q= (self.numerator) // a
        r = (self.denominator)  % a
        return f"Simplest form is {q}/{r}"
obj = Fraction()
obj.getdata()
print(obj.show)
```

#10. Write a program that has a class Numbers with a list as an instance variable.

a. Write a method called insert_element that takes values from user.

b. Write a class method called find_max to find and print largest value in the list.

```
class Numbers:
    list = []
    max = 0

    def insert_element(self):
        list = [int(x) for x in input("enter elements seperated by ,").split(",")]

    def find_max(self):
        self.max = self.list[0]
        for i in self.list:
            if i > self.max:
                self.max = i
        return f"largest value is {max}"
obj = Numbers()
obj.insert_element()
print(obj.find_max())
```

```
'''11.    Write a program that has a class Point with attributes x and y.
a.        Write a method called midpoint that returns a midpoint of a line joining two points.
b.        Write a method called length that returns the length of a line joining two points.
'''
class Point :
    x= 0
    y =0
    length1 =0
    def __init__(self , x1 ,y1 , x2,y2):
        self.x1 = x1
        self.y1= y1
        self.x2= x2
        self.y2= y2
    def midpoint(self) :
        self.x =  (((self.x1) + (self.x2)) / 2)
        self.y = (((self.y1) + (self.y2)) / 2)
        return f"midpoint coordinate  is {self.x} , {self.y}"

    def length(self):
        length1 = ((((self.x1-self.x2)**2)+((self.y1-self.y2)**2))**0.5)
        return f"length of line is {length1}"


obj = Point(1,3,4,5)
obj.midpoint()
obj.length()
```

```
"""
12.	Create a class called Complex.
Write a menu driven program to read, display, add and subtract two complex numbers by creating corresponding instance methods.
"""



class complex:
    sum1 = 0
    sum2 = 0
    sub1 = 0
    sub2 = 0

    def input_data(self):
        self.real_part1 = int(input("enter first no.real part"))
        self.cmplxpart1 = int(input("enter first complex  part"))
        self.real_part2 = int(input("enter sedcond no.real part"))
        self.cmplxpart2 = int(input("enter second complex  part"))

    def add(self):
        self.sum1 = self.real_part1 + self.real_part2
        self.sum2 = self.cmplxpart1 + self.cmplxpart2
        return f"summation  is {self.sum1}+{self.sum2} i"

    def subtract(self):
        self.sub1 = self.real_part1 - self.real_part2
        self.sub2 = self.cmplxpart1 - self.cmplxpart2
        return f"summation  is {self.sub1}+({self.sub2}i)"



choice = int(input("enter 1 for addition and 2 for subtraction"))
if choice == 1:
    obj = complex()

    obj.input_data()
    print(obj.add())

elif choice == 2:
    obj = complex()

    obj.input_data()
    print(obj.subtract())

else:
    print("wrong input")
```

```
'''
13.     Write a Program to illustrate the use of __str__(), __repr__(), __new__,
__doc__, __dict__, __name__ and __bases__ methods.
'''


class User:

    def __new__(cls, *args):
        print("calling __new__ magic method.")
        inst = object.__new__(cls)
        return inst

    def __init__(self, *args):
        print("calling __init__ magic method now")
        self.name = args[0]
        self.email = args[1]
        self.role = args[2]

    def __repr__(self):
        d = dict(name=self.name, email=self.email, role=self.role)
        return d

    def __str__(self):
        return 'name={} email={} role={}'.format(self.name, self.email, self.role)
obj = User()

class Employee(User):
    'Common base class for all employees'
    empCount = 0

    def __init__(self, name, salary):
        self.name = name
        self.salary = salary
        Employee.empCount += 1

    def displayCount(self):
        print("Total Employee %d" % Employee.empCount)

    def displayEmployee(self):
        print("Name : ", self.name, ", Salary: ", self.salary)

emp1 = Employee("Zara", 2000)
emp2 = Employee("Manni", 5000)
print("Employee.__doc__:", Employee.__doc__)
print("Employee.__name__:", Employee.__name__)
print("Employee.__module__:", Employee.__module__)
print("Employee.__bases__:", Employee.__bases__)
print("Employee.__dict__:", Employee.__dict__)
```

```
'''14.	Create a BankAccount class. Your class should support the following methods:
a.	__init__(self, account_no)
b.	deposit (self, account_no, amount)
c.	withdraw (self, account_no, amount)
d.	get_balance (self, account_no)'''
class BankAccount :
    def __init__(self , account_no):
        self.account_no = account_no

    def deposit(self, account_no, amount)


        self.account_no = account_no
        self.amount += amount

    def withdraw(self, account_no, amountw)
        self.account_no = account_no
        self.amount -= amountw
    def get_balance (self, account_no):


        print("account balance is " , self.amount)
obj = BankAccount(1234)
obj.deposit (1234,455)
obj.withdraw(1234,89)
obj.get_balance(1234)
```

'''

15. Write a program to illustrate the use of following built-in methods:

a. hasattr(obj,attr)

b. getattr(object, attribute_name [, default])

c. setattr(object, name, value)

d. delattr(class_name, name)

'''

```
class student:
    school="the baptist convent school"
    def _init_(self):
        self.name=" "
        self.marks=0
s1=student()
hasattr(s1,"name")
getattr(student,'school','no attribute exist')
setattr(student,'school','plato public school')
getattr(student,'school','no attribute exist')
delattr(student,'school')
```
