



**MALAD KANDIVALI EDUCATION SOCIETY'S
NAGINDAS KHANDWALA COLLEGE OF COMMERCE,
ARTS & MANAGEMENT STUDIES & SHANTABEN NAGINDAS
KHANDWALA COLLEGE OF SCIENCE
MALAD [W], MUMBAI – 64
(AUTONOMOUS)**

**(Reaccredited 'A' Grade by NAAC)
(AFFILIATED TO UNIVERSITY OF MUMBAI)
(ISO 9001:2015)**

CERTIFICATE

Name: Ms. MENARIA KOMAL CHANDRAPRAKASH

Roll No: 42 Programmed: BSC IT Semester: II

This is certified to be a bonfire record of practical works done by the above student in the college laboratory for the course **OBJECT ORIENTED PROGRAMMING I** (Course Code:2022UISPR) for the partial fulfillment of Second Semester of BSC IT during the academic year 2020-2021.

The journal work is the original study work that has been duly approved in the year 2020-2021 by the undersigned.

**External Examiner
(Mrs. Niramaye Deshpande)**

Subject-In-Charge

Date of Examination: (College Stamp)

Name: MENARIA KOMAL CHANDRAPRAKASH

Roll No: 42

Sr. No.	DATE	TITLE	SIGN
1.	15/01/2021	Creating Class diagram with the class and their attributes and methods	
2.	22/01/2021	Defining and using a class.	
3.	29/01/2021	Defining methods with and without attributes in a class.	
4.	05/02/2021	Creating and using constructor and Destructor	
5.	12/02/2021	Using property getters and setters.	
6.	19/02/2021	Implementing various forms of Inheritance.	
7.	26/02/2021	Implementing Polymorphism by Overloading Overriding methods.	
8.	12/03/2021	Implementing concept of Operator Overloading.	
9.	26/03/2021	Implementing abstract classes and interfaces.	
10.	09/04/2021	Implementing concept of Composition.	

Name: KOMAL MENARIA
Class: FYBSCIT
Roll No: 42

PRACTICAL-1A

Aim: Draw class diagram for calculating area of Rectangle, Square, Circle and Ellipse and write a program in Python for the same.

Writeup:

Class: A class is a user-defined blueprint or prototype from which objects are created. Classes provide a means of bundling data and functionality together. Creating a new class creates a new type of object, allowing new instances of that type to be made.

Object: An Object is an instance of a Class. A class is like a blueprint while an instance is a copy of the class with actual values.

Class method: Class methods must have an extra first parameter in the method definition. We do not give a value for this parameter when we call the method, Python provides it.

Class attributes: belong to the class itself they will be shared by all the instances. Such attributes are defined in the class body parts usually at the top, for legibility.

Class diagram:

Square
+length of side
+CalculateArea() +CalculatePerimeter()

Rectangle
+width +height
+CalculateArea() +CalculatePerimeter()

Circle
+Radius

Name: KOMAL MENARIA

Class: FYBSCIT

Roll No: 42

+CalculateArea()
+CalculatePerimeter()

Ellipse

+Semi-major axis

+Semi-major axis

+CalculateArea()

+CalculatePerimeter()

1Rectangle:

Code:

```
practical1ofoop.py - C:/Users/HP/practical1ofoop.py (3.8.7)
File Edit Format Run Options Window Help

class Rectangle:
    def CalculateArea(self):
        print("Enter length:")
        self.l=float(input())
        print("Enter breadth:")
        self.b=float(input())
        area=self.l*self.b
        print("Area of rectangle is =%f"%(area))

    def CalculatePerimeter(self):
        perimeter=2*(self.l+self.b)
        print("Perimeter of rectangle is =%f"%(perimeter))

c=Rectangle()
x=c.CalculateArea()
y=c.CalculatePerimeter()
```

Output

Name: KOMAL MENARIA

Class: FYBSCIT

Roll No: 42

```
IDLE Shell 3.8.7
File Edit Shell Debug Options Window Help
Python 3.8.7 (tags/v3.8.7:6503f05, Dec 21 2020, 17:59:51) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/HP/practicallofoop.py =====
Enter length:
4
Enter breadth:
5
Area of rectangle is =20.000000
Perimeter of rectangle is =18.000000
>>> |
```

2 Square:

Code:

```
practical1square.py - C:/Users/HP/practical1square.py (3.8.7)
File Edit Format Run Options Window Help
class Square:
    def CalculateArea(self):
        print("Enter side:")
        self.s=float(input())
        area=self.s*self.s
        print("Area of Square is =%f"%(area))

    def CalculatePerimeter(self):
        perimeter=4*self.s
        print("Perimeter of Square is =%f"%(perimeter))

c=Square()
c.CalculateArea()
c.CalculatePerimeter()
```

Output:

Name: KOMAL MENARIA

Class: FYBSCIT

Roll No: 42

```
IDLE Shell 3.8.7
File Edit Shell Debug Options Window Help
Python 3.8.7 (tags/v3.8.7:6503f05, Dec 21 2020, 17:59:51) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/HP/practicalsquare.py =====
Enter side:
4
Area of Square is =16.000000
Perimeter of Square is =16.000000
>>>
```

3Circle

Code:

```
practical1circle.py - C:/Users/HP/practical1circle.py (3.8.7)
File Edit Format Run Options Window Help

class Circle:
    def CalculateArea(self):
        print("Enter radius:")
        self.r=float(input())
        area=3.14*self.r*self.r
        print("Area of Circle is =%f"%(area))

    def CalculatePerimeter(self):
        perimeter=2*3.14*self.r
        print("Perimeter of Circle is =%f"%(perimeter))

c=Circle()
c.CalculateArea()
c.CalculatePerimeter()
```

Output:

```
IDLE Shell 3.8.7
File Edit Shell Debug Options Window Help
Python 3.8.7 (tags/v3.8.7:6503f05, Dec 21 2020, 17:59:51) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/HP/practical1circle.py =====
Enter radius:
5
Area of Circle is =78.500000
Perimeter of Circle is =31.400000
>>> |
```

4Ellipse:

Code:

Name: KOMAL MENARIA

Class: FYBSCIT

Roll No: 42

```
practical1ellipse.py - C:/Users/HP/practical1ellipse.py (3.8.7)
File Edit Format Run Options Window Help
import math
class Ellipse:
    def CalculateArea(self):
        print("Enter axis 1:")
        self.a=float(input())
        print("Enter axis 2:")
        self.b=float(input())
        area=3.14*self.a*self.b
        print("Area of ellipse is =%f"%(area))

    def CalculatePerimeter(self):
        perimeter=2*3.14*math.sqrt((self.a*self.a+self.b*self.b)/(2))
        print("Perimeter of ellipse is =%f"%(perimeter))

c=Ellipse()
x=c.CalculateArea()
y=c.CalculatePerimeter()
```

Output:

```
IDLE Shell 3.8.7
File Edit Shell Debug Options Window Help
Python 3.8.7 (tags/v3.8.7:6503f05, Dec 21 2020, 17:59:51) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/HP/practical1ellipse.py =====
Enter axis 1:
7
Enter axis 2:
8
Area of ellipse is =175.840000
Perimeter of ellipse is =47.204551
>>> |
```

PRACTICAL-1B

AIM:- Draw class diagram for performing all arithmetic operations and write a program in python to define arithmetic which will define arithmetic class and required methods for doing all arithmetic operations.

Class diagram:-

Arithmetic
+a
+b

Name: KOMAL MENARIA

Class: FYBSCIT

Roll No: 42

**+accept()
+add()
+subtract()
+division()
+modulus()**

CODE:

```
Arithmetic.py - C:/Users/HP/Arithmetic.py (3.8.7)
File Edit Format Run Options Window Help

class Arithmetic:
    def accept(self):
        print("Enter two numbers")
        self.a=int(input())
        self.b=int(input())

    def sum(self):
        x=self.a+self.b
        print("sum is",x)
    def sub(self):
        y=self.a-self.b
        print("Subtraction is",y)
    def multi(self):
        z=self.a*self.b
        print("Multiplication is",z)
    def div(self):
        p=self.a/self.b
        print("Division is",p)
    def join(self):
        q=str(self.a)+str(self.b)
        print("Concatination is",q)

al=Arithmetic()
al.accept()
al.sum()
al.sub()
al.multi()
al.div()
al.join()
```

OUTPUT:

```
IDLE Shell 3.8.7
File Edit Shell Debug Options Window Help
Python 3.8.7 (tags/v3.8.7:6503f05, Dec 21 2020, 17:59:51) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/HP/Arithmetic.py =====
Enter two numbers
7
8
sum is 15
Subtraction is -1
Multiplication is 56
Division is 0.875
Concatination is 78
>>> |
```


Name: KOMAL MENARIA
Class: FYBSCIT
Roll No: 42

PRACTICAL 2A

Aim : Draw a class diagram and define class in python for creating Bank account and showing basic transaction such as Deposit and withdraw of money on specific account and show updated balance along with customer details.

Writeup:

A class in Python can be defined using the class keyword. As per the syntax above, a class is defined using the class keyword followed by the class name and : operator after the class name, which allows you to continue in the next indented line to define class members.

To create Bankaccount class with deposit, withdraw function to perform some simple bank operations like deposit and withdrawal of money. This step is followed by declaring that balance is 0 using self next, we use a display function to display the remaining balance in the account.

Class diagram :

bank
+name +ac_no +init_bal
+display() +deposit() +withdraw()

Name: KOMAL MENARIA

Class: FYBSCIT

Roll No: 42

CODE:

```
practical 2A oop.py - C:/Users/HP/practical 2A oop.py (3.8.8)
File Edit Format Run Options Window Help

class bank:
    cust_name=str(input("Enter your name:"))
    acc_no=int(input("Enter your account number:"))
    bal=int(input("Enter your balance:"))

    def deposit(self):
        dep_amt=int(input("Enter the amount to be deposit:"))
        self.bal=self.bal+dep_amt
        print("Current balance is",self.bal)

    def withdraw(self):
        with_amt=int(input("Enter the amount to be withdraw:"))
        self.bal=self.bal-with_amt
        print("Your current balance is",self.bal)
c=bank()
a=int(input("Enter 1 is you want to deposit some amount or 2 if you want to withdraw:"))
if(a==1):
    c.deposit()
elif(a==2):
    c.withdraw()
```

OUTPUT:

```
IDLE Shell 3.8.8
File Edit Shell Debug Options Window Help

Python 3.8.8 (tags/v3.8.8:024d805, Feb 19 2021, 13:18:16) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/HP/practical 2A oop.py =====
Enter your name:komal menaria
Enter your account number:401209
Enter your balance:10000
Enter 1 is you want to deposit some amount or 2 if you want to withdraw:2
Enter the amount to be withdraw:3000
Your current balance is 7000
>>>
```

Practical 2B

Aim: Draw a class digram and define class in python to accept student information such as roll no,name,class and marks of five subjects.Calculate total and percentage and display the complete information

Writeup:

Name: KOMAL MENARIA

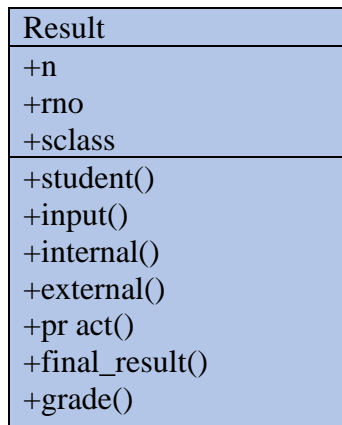
Class: FYBSCIT

Roll No: 42

Accept – This method takes details from the user like name, roll number, and marks for two different subjects.

Display – This method displays the details of every student

Class diagram :



Code:

Name: KOMAL MENARIA

Class: FYBSCIT

Roll No: 42

practical 2b oop.py - C:/Users/HP/practical 2b oop.py (3.8.8)

File Edit Format Run Options Window Help

```
class marks:
    student_name=str(input("Enter your name:"))
    student_rno=int(input("Enter your roll no:"))
    sclass=input("Enter your class:")

    def subjects(self):
        self.dbms=int(input("Enter marks in DBMS:"))
        self.dm=int(input("Enter marks in DM:"))
        self.pp=int(input("Enter marks in python programming:"))
        self.oop=int(input("Enter marks in object oriented programming:"))
        self.wp=int(input("Enter marks in web programming:"))

    def internal_marks(self):
        print("Enter your marks out of 40 for internals:")
        self.subjects()
        self.int_marks=self.dbms+self.dm+self.pp+self.oop+self.wp

    def theory_marks(self):
        print("Enter your marks out of 60 for theory:")
        self.subjects()
        self.th_marks=self.dbms+self.dm+self.pp+self.oop+self.wp

    def practicals(self):
        print("Enter your marks out of 50 for practicals:")
        self.subjects()
        self.pr_marks=self.dbms+self.dm+self.pp+self.oop+self.wp
        self.marks=self.int_marks+self.th_marks+self.pr_marks
        self.total=(40*5)+(60*5)+(50*5)
        self.percent=(self.marks/self.total)*100
        self.percentage=round(self.percent)

    def grade(self):
        if self.percentage>90:
            print("You pass!")
            print("Your percentage is:",self.percentage)
            print("O Grade")
        elif self.percentage>75:
            print("You pass!")
            print("Your percentage is:",self.percentage)
            print("A+ Grade")
        elif self.percentage>60:
            print("You pass!")
            print("Your percentage is:",self.percentage)
            print("A Grade")
        elif self.percentage>50:
            print("You pass!")
            print("Your percentage is:",self.percentage)
            print("B+ Grade")
        elif self.percentage>45:
            print("You pass!")
            print("Your percentage is:",self.percentage)
            print("B Grade")
        else:
            print("You Fail!")
            print("Your percentage is:",self.percentage)

    def display(self):
        print("Your name is:",self.student_name)
        print("Your roll no is:",self.student_rno)
        print("Your class is:",self.sclass)
        print("Your total marks:",self.total)
        print("Your percentage:",self.percentage)
```

practical 2b oop.py - C:/Users/HP/practical 2b oop.py (3.8.8)

File Edit Format Run Options Window Help

```
print("You pass!")
print("Your percentage is:",self.percentage)
print("O Grade")
elif self.percentage>75:
    print("You pass!")
    print("Your percentage is:",self.percentage)
    print("A+ Grade")
elif self.percentage>60:
    print("You pass!")
    print("Your percentage is:",self.percentage)
    print("A Grade")
elif self.percentage>50:
    print("You pass!")
    print("Your percentage is:",self.percentage)
    print("B+ Grade")
elif self.percentage>45:
    print("You pass!")
    print("Your percentage is:",self.percentage)
    print("B Grade")
else:
    print("You Fail!")
    print("Your percentage is:",self.percentage)

def display(self):
    print("Your name is:",self.student_name)
    print("Your roll no is:",self.student_rno)
    print("Your class is:",self.sclass)
    print("Your total marks:",self.total)
    print("Your percentage:",self.percentage)

c=marks()
c.internal_marks()
c.theory_marks()
c.practicals()
c.display()
c.grade()
```

Output:

Name: KOMAL MENARIA

Class: FYBSCIT

Roll No: 42

```
IDLE Shell 3.8.8
File Edit Shell Debug Options Window Help
Python 3.8.8 (tags/v3.8.8:024d805, Feb 19 2021, 13:18:16) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/HP/practical 2b oop.py =====
Enter your name:KOMAL MENARIA
Enter your roll no:42
Enter your class:FYBSC IT
Enter your marks out of 40 for internals:
Enter marks in DBMS:34
Enter marks in DM:33
Enter marks in python programming:40
Enter marks in object oriented programming:24
Enter marks in web programming:33
Enter your marks out of 60 for theory:
Enter marks in DBMS:40
Enter marks in DM:55
Enter marks in python programming:56
Enter marks in object oriented programming:55
Enter marks in web programming:45
Enter your marks out of 50 for practicals:
Enter marks in DBMS:44
Enter marks in DM:34
Enter marks in python programming:44
Enter marks in object oriented programming:33
Enter marks in web programming:23
Your name is: KOMAL MENARIA
Your roll no is: 42
Your class is: FYBSC IT
Your total marks: 750
Your percentage: 79
You pass!
Your percentage is: 79
A+ Grade
>>>
```

PRACTICAL 3A

Aim: Draw class diagram and define class Book to accept name of book, author, price and number of copies to be purchased using method without parameter. Calculator total price and display all details.

Writeup:

Accept – This method takes details from the user name of book, author, price and number of copies to be purchased for Calculation.

Display – This method displays the details of book with total price

Class diagram:

Book
+name

Name: KOMAL MENARIA

Class: FYBSCIT

Roll No: 42

+author +price +copies
+accept() +total_price() +details()

Code:

```
practical3oop.py - C:/Users/HP/practical3oop.py (3.8.8)
File Edit Format Run Options Window Help

class Book:
    def accept(self):
        self.name=str(input("Enter name of the Book: "))
        self.author=str(input("Enter author of the Book: "))
        self.price=float(input("Enter price of the Book: "))
        self.n_copies=int(input("Enter number of copies: "))
    def total_price(self):
        t=self.price*self.n_copies
        print("Total Price: ",t)
    def details(self):
        print("Purchase details".center(70,"-"))
        print("Name of book:",self.name)
        print("Author of book:",self.author)
        print("Price of book:",self.price)
        print("Number of copies:",self.n_copies)

c= Book()
c.accept()
c.details()
c.total_price()
```

Output:

```
IDLE Shell 3.8.8
File Edit Shell Debug Options Window Help
Python 3.8.8 (tags/v3.8.8:024d805, Feb 19 2021, 13:18:16) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/HP/practical3oop.py =====
Enter name of the Book: The Kalam
Enter author of the Book: dr prasad ray
Enter price of the Book: 200
Enter number of copies: 5
-----Purchase details-----
Name of book: The Kalam
Author of book: dr prasad ray
Price of book: 200.0
Number of copies: 5
Total Price: 1000.0
>>> |
```

PRACTICAL 3B

Aim: Draw class diagram and define class Time to accept time in hours and minutes in two different instances. Pass these instances as an argument to method to add time in hour and minutes and display total time.

Writeup:

Name: KOMAL MENARIA

Class: FYBSCIT

Roll No: 42

Accept : This method takes details from the user time to accept time in hours and minutes in two different instances.

Sum: Add time in hour and minutes

Display –:This method displays the detail with total time

Class diagram:

Final_time
+h +m
+accept() +sum() +display()

Code:

```
practical3b oop.py - C:/Users/HP/practical3b oop.py (3.8.8)
File Edit Format Run Options Window Help

class Time:
    h=0;
    m=0;
    def accept(self):
        print("Enter time in hour and mins");
        self.h=int(input())
        self.m=int(input())
    def display(self):
        print(self.h,"Hours and",self.m,"minites")
    def sum(self,t1,t2):
        self.m=t1.m+t2.m
        self.h=self.m/60
        self.m=self.m%60
        self.h=self.h+t1.h+t2.h
t1_obj=Time()
t1_obj.accept()
t2_obj=Time()
t2_obj.accept()
t3=Time()
t3.sum(t1_obj,t2_obj)
print("t1_obj=")
t1_obj.display()
print("t2_obj=")
t2_obj.display()
print("t3=")
t3.display()
```

Output:

Name: KOMAL MENARIA

Class: FYBSCIT

Roll No: 42

```
IDLE Shell 3.8.8
File Edit Shell Debug Options Window Help
Python 3.8.8 (tags/v3.8.8:024d805, Feb 19 2021, 13:18:16) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/HP/practical3b oop.py =====
Enter time in hour and mins
6
40
Enter time in hour and mins
4
50
t1_obj=
6 Hours and 40 minites
t2_obj=
4 Hours and 50 minites
t3=
11.5 Hours and 30 minites
>>> |
```

PRACTICAL 4A

Aim: creating and using constructor and destructor

Writeup:

Python Destructors - Destroying the Object. Just like a constructor is used to create and initialize an object, a destructor is used to destroy the object and perform the final clean up.

The Compiler calls the Constructor whenever an object is created. Constructors initialize values to object members after storage is allocated to the object. Whereas, Destructor on the other hand is used to destroy the class object.

a) Draw class diagram and write a program to calculate the area of the triangle using constructor and destructor.

Class diagram:

Rect
+length +breadth
+area() + __del__()

Code:

Name: KOMAL MENARIA

Class: FYBSCIT

Roll No: 42

```
practical3c dbms.py - C:/Users/HP/practical3c dbms.py (3.8.8)
File Edit Format Run Options Window Help

class rect:
    def __init__(self,length,breadth):
        print("I'm initialising the rectangle class")
        self.length=length
        self.breadth=breathdth
    def area(self):
        self.area=self.length*self.breadth
        print("Area is",self.area)
    def __del__(self):
        print("Instance destroyed")
c=rect(20,30)
c.area()
cl=rect(20,40)
cl.area()
cl.__del__()

```

Output:

```
IDLE Shell 3.8.8
File Edit Shell Debug Options Window Help
Python 3.8.8 (tags/v3.8.8:024d805, Feb 19 2021, 13:18:16) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/HP/practical3c dbms.py =====
I'm initialising the rectangle class
Area is 600
I'm initialising the rectangle class
Area is 800
Instance destroyed
>>> |

```

PRACTICAL 4B

Aim: Define class employee to accept company name address employee ID, employee name, qualification of employee destination employee and basic salary. calculate net salary using the following information for allowance and deduction as travelling allowance is 80% of basic salary, DA is 70% of basic salary, HRA is 25% of basic salary and employee provident fund (EPF) IS 10% of basic salary calculate net salary using the formula

NET SALARY = BASIC SALARY+ TRAVELLING ALLOWANCE+ DA+ HRA- EPF

Initialize employee object using constructor and destroy it once it is of no use using destructor.

CLASS diagram:

Employee
+compname
+comp_add

Name: KOMAL MENARIA

Class: FYBSCIT

Roll No: 42

+emp_id

+qual

+des

+sal

+net_sal()

Code:

```
practical 4b oop.py - C:/Users/HP/practical 4b oop.py (3.8.8)
File Edit Format Run Options Window Help

class Employee:
    def __init__(self):
        self.comp_name=str(input("Enter company name:"))
        self.comp_add=str(input("Enter your company address:"))
        self.emp_id=int(input("Enter your emp id:"))
        self.name=str(input("Enter your name:"))
        self.qual=str(input("Enter your qualification:"))
        self.des=str(input("Enter your designation:"))
        self.basic_sal=int(input("Enter your basic salary:"))

    def calc(self):
        TA=18/100*self.basic_sal
        DA=70/100*self.basic_sal
        HRA=25/100*self.basic_sal
        EPF=10/100*self.basic_sal
        net_sal=self.basic_sal+TA+DA+HRA+EPF
        print("Net salary is:",net_sal)
    def __del__(self):
        print("Instance destroyed")

c=Employee()
c.calc()
c.__del__()
```

Output:

```
IDLE Shell 3.8.8
File Edit Shell Debug Options Window Help

Python 3.8.8 (tags/v3.8.8:024d805, Feb 19 2021, 13:18:16) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/HP/practical 4b oop.py =====
Enter company name:komal cafe coffee
Enter your company address:radha nagar, tulinj road,nallasopara ,mumbai
Enter your emp id:401209
Enter your name:komal menaria
Enter your qualification:FY BSC IT
Enter your designation: CA
Enter your basic salary:20000
Net salary is: 44600.0
Instance destroyed
>>> |
```

PRACTICAL 5A

Name: KOMAL MENARIA

Class: FYBSCIT

Roll No: 42

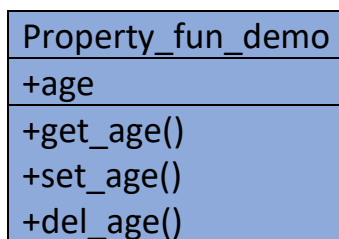
Aim: Using property getter and setter.

a) Draw class diagram and write a program to implement getter and setters property.

Writeup:

Setter will set the value of a by checking the conditions we have mentioned in the method. Another way to use the property is... Pass all the getter and setter methods to the property and assign it to the variable which you have to use as a class attribute. Make the setter and getter methods as private to hide them.

CLASS DIAGRAM:



CODE:

```
practical 5a oop.py - C:/Users/HP/practical 5a oop.py (3.8.8)
File Edit Format Run Options Window Help
# Python program showing use of property() function.
class property_fun_demo:
    # def __init__(self):
    #     self.__age = 0

    # function to get values of _age.
    def get_age(self):
        print("Getter method called. ")
        return self.__age

    #function to set values of _age.
    def set_age(self,a):
        print("Setter method called. ")
        self.__age = a

    # function to delete _age attribute.
    def del_age(self):
        del self.__age

    age = property(get_age,set_age,del_age)

mark = property_fun_demo()
mark.age = 10
print(mark.age)
```

OUTPUT:

Name: KOMAL MENARIA

Class: FYBSCIT

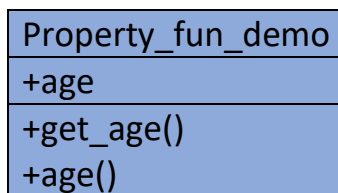
Roll No: 42

```
IDLE Shell 3.8.8
File Edit Shell Debug Options Window Help
Python 3.8.8 (tags/v3.8.8:024d805, Feb 19 2021, 13:18:16) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/HP/practical 5a oop.py =====
Setter method called.
Getter method called.
10
>>>
```

PRACTICAL -5B

Aim: Write a program to demonstrate the use of getter and setter method using property decorator.

Class diagram:



CODE:

```
practical 5b oop.py - C:\Users\HP\practical 5b oop.py (3.8.8)
File Edit Format Run Options Window Help
# to demonstrate the use of getter and setter method using property decorator.
class property_decorator_demo:
    def __init__(self):
        self._age = 0

    @property
    def get_age(self):
        print("Getter method called. ")
        return self._age

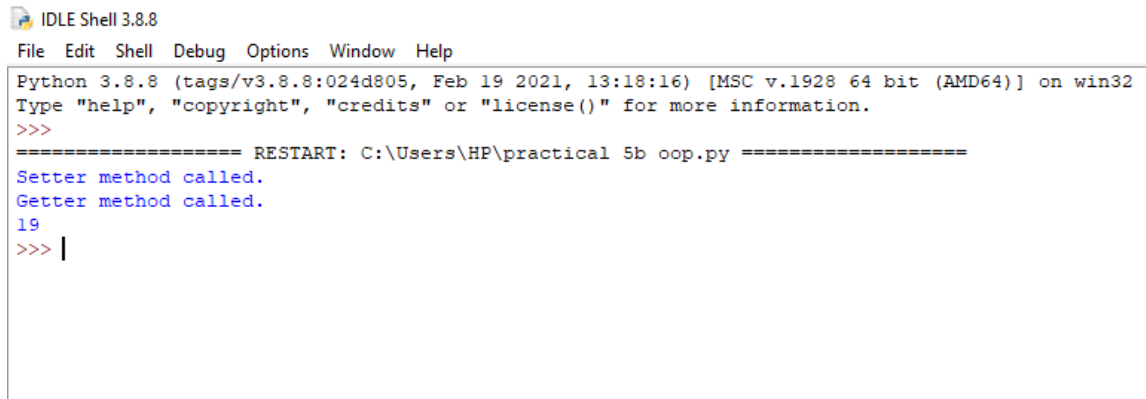
    @get_age.setter
    def age(self,a):
        if(a < 18):
            raise ValueError("Sorry your age is below eligibility criteria.")
        print("Setter method called.")
        self._age =a

# Creating a object of class.
mark = property_decorator_demo()

#Calling method through object.
mark.age = 19
print(mark.age)
|
```

Name: KOMAL MENARIA
Class: FYBSCIT
Roll No: 42

OUTPUT:



```
IDLE Shell 3.8.8
File Edit Shell Debug Options Window Help
Python 3.8.8 (tags/v3.8.8:024d805, Feb 19 2021, 13:18:16) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\HP\practical 5b oop.py =====
Setter method called.
Getter method called.
19
>>> |
```

Practical: 6 A

Aim: Draw class diagram and write a program to implement single inheritance

Writeup:

In single inheritance, a class is allowed to inherit from only one class. i.e. one sub class is inherited by one base class only.

Hybrid Inheritance:

Hybrid Inheritance is implemented by combining more than one type of inheritance.

Multiple inheritance:

When a class can be derived from more than one base class this type of inheritance is called multiple inheritance.

Multilevel inheritance:

In this type of inheritance, a derived class is created from another derived class.

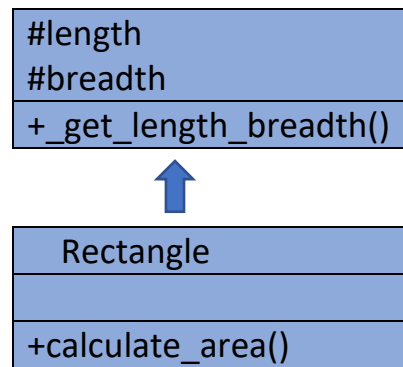
CLASS DIAGRAM:



Name: KOMAL MENARIA

Class: FYBSCIT

Roll No: 42



CODE:

```
practical 6a oop.py - C:/Users/HP/practical 6a oop.py (3.8.8)
File Edit Format Run Options Window Help

class shape:
    def _get_Length_Breadth(self):
        self._length = int(input("Enter Length : "))
        self._breadth = int(input("Enter breadth :"))

class Rectangle(shape):
    def calculate_area(self):
        print("Area of rectangle is",self._length*self._breadth)

obj =Rectangle()
obj._get_Length_Breadth()
obj.calculate_area()
```

OUTPUT:

```
IDLE Shell 3.8.8
File Edit Shell Debug Options Window Help

Python 3.8.8 (tags/v3.8.8:024d805, Feb 19 2021, 13:18:16) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/HP/practical 6a oop.py =====
Enter Length : 12
Enter breadth :12
Area of rectangle is 144
>>> |
```

Practical: 6 B

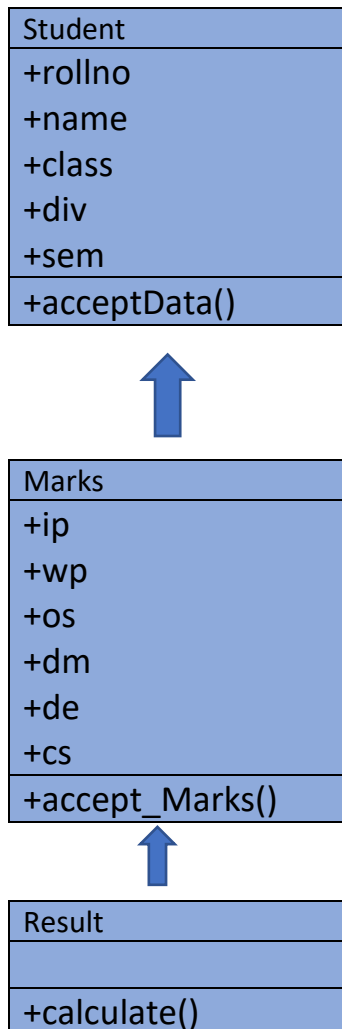
Name: KOMAL MENARIA

Class: FYBSCIT

Roll No: 42

Aim: Draw class diagram and write a program to implement multi-level inheritance and make a class diagram for it.

CLASS DIAGRAM:



CODE:

Name: KOMAL MENARIA

Class: FYBSCIT

Roll No: 42

practical 6b oop.py - C:\Users\HP\practical 6b oop.py (3.8.8)

File Edit Format Run Options Window Help

```
class student:

    def acceptData(self):
        self.rollno = int(input("Roll No : "))
        self.name = str(input("Name : "))
        self.class_ = str(input("class : "))
        self.div = str(input("Division : "))
        self.sem = input("semester : ")

class Marks(student):

    def accept_marks(self):
        self.dm = int(input("Discrete Mathematics : "))
        self.pp = int(input("Python Programming : "))
        self.oops = int(input("Object oriented programming : "))
        self.dbms = int(input("Database management system : "))
        self.wp = int(input("web programming : "))
        self.it = int(input("IT tools : "))

class Result(Marks):

    def calculate(self):
        self.total = (self.dm+self.pp+self.oops+self.dbms+self.wp+self.it)
        self.avg = self.total/6
        self.percentage = (self.total * 100)/600
        print("\nTotal Marks : ",self.total)
        print("Average Marks : ",self.avg)
        print("Percentage : ",self.percentage)

c=Result()
c.acceptData()
c.accept_marks()
c.calculate()
```

OUTPUT:

IDLE Shell 3.8.8

File Edit Shell Debug Options Window Help

Python 3.8.8 (tags/v3.8.8:024d805, Feb 19 2021, 13:18:16) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>>

===== RESTART: C:\Users\HP\practical 6b oop.py =====

Roll No : 42

Name : komal menaria

class : fybscit

Division : a

semester : 1

Discrete Mathematics : 22

Python Programming : 22

object oriented programming : 66

Database management system : 77

web programming : 88

IT tools : 33

Total Marks : 308

Average Marks : 51.333333333333336

Percentage : 51.333333333333336

>>> |

Name: KOMAL MENARIA

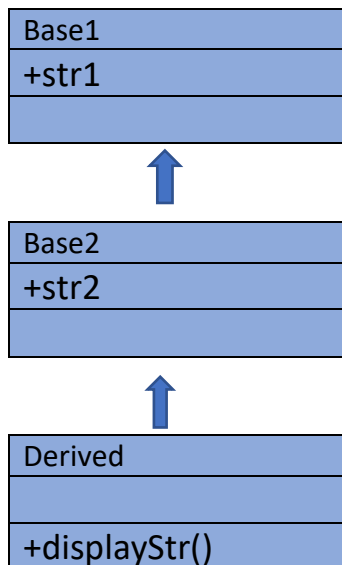
Class: FYBSCIT

Roll No: 42

PRACTICAL: 6 C

Aim: Draw class diagram and write a program to implement multi-level inheritance.

CLASS DIAGRAM:



CODE:

```
practical 6c oop.py - C:/Users/HP/practical 6c oop.py (3.8.8)
File Edit Format Run Options Window Help
#Program to demonstrate Multiple Inheritance

class Base1(object):

    def __init__(self):
        self.str1 = "Hello"
        print(self.str1)
        print("komal")

class Base2(object):

    def __init__(self):
        self.str2 = "Hi"
        print(self.str1)
        print("Devangi")

class Derived(Base1 , Base2):

    def __init__(self):
        Base1.__init__(self)
        Base2.__init__(self)
        print("Nice TO meet you")

    def displayStr(self):
        print(self.str1, self.str2)

c = Derived()
c.displayStr()
```

Name: KOMAL MENARIA

Class: FYBSCIT

Roll No: 42

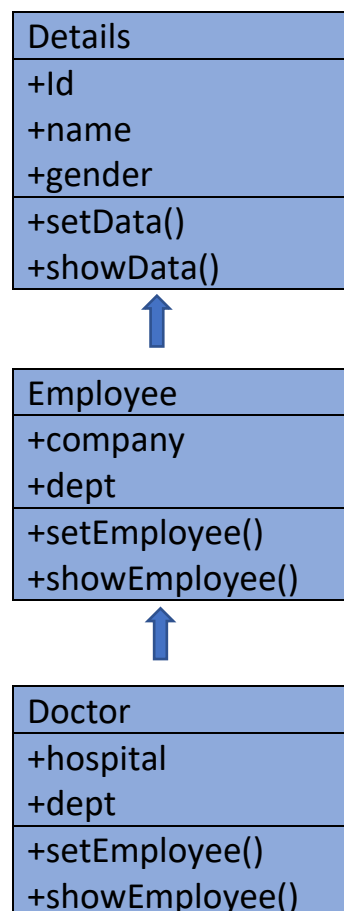
OUTPUT:

```
IDLE Shell 3.8.8
File Edit Shell Debug Options Window Help
Python 3.8.8 (tags/v3.8.8:024d805, Feb 19 2021, 13:18:16) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/HP/practical 6c oop.py =====
Hello
komal
Hello
Devangi
Nice TO meet you
Hello Hi
>>> |
```

PRACTICAL: 6 D

Aim: Draw class diagram and write a program to implement hierarchical inheritance.

CLASS DIAGRAM:



Name: KOMAL MENARIA

Class: FYBSCIT

Roll No: 42

CODE:

```
practical 6d oop.py - C:\Users\HP\practical 6d oop.py (3.8.8)
File Edit Format Run Options Window Help

class Details:

    def __init__(self):
        self.__id = "<No Id>"
        self.__name = "<No Name>"
        self.__gender = "<No Gender>"

    def setData(self, id, name, gender):
        self.__id = id
        self.__name = name
        self.__gender = gender

    def showData(self):
        print("Id : ", self.__id)
        print("Name : ", self.__name)
        print("Gender : ", self.__gender)

class Employee(Details):

    def __init__(self):
        self.__company = "<No Company>"
        self.__dept = "<No Dept>"

    def setEmployee(self, id, name, gender, comp, dept):
        self.setData(id, name, gender)
        self.__company = comp
        self.__dept = dept

    def showEmployee(self):
        self.showData()
        print("Company : ", self.__company)
        print("Department : ", self.__dept)

class Doctor(Details):

    def __init__(self):
        self.__hospital = "<No Hospital>"
        self.__dept = "<No Dept>"

    def setEmployee(self, id, name, gender, hos, dept):
        self.setData(id, name, gender)

        self.__hospital = hos
        self.__dept = dept

    def showEmployee(self):
        self.showData()
        print("Hospital : ", self.__hospital)
        print("Department : ", self.__dept)

def main():
    print("Employee Object")
    e = Employee()
    e.setEmployee(1, "Komal Menaria", "Female", "gmr", "excavation")
    e.showEmployee()
    print("\nDoctor object")
    d = Doctor()
    d.setEmployee(1, "Devangi Parmar", "Female", "aiims", "eyes")
    d.showEmployee()

if __name__ == "__main__":
    main()
```

OUTPUT:

Name: KOMAL MENARIA

Class: FYBSCIT

Roll No: 42

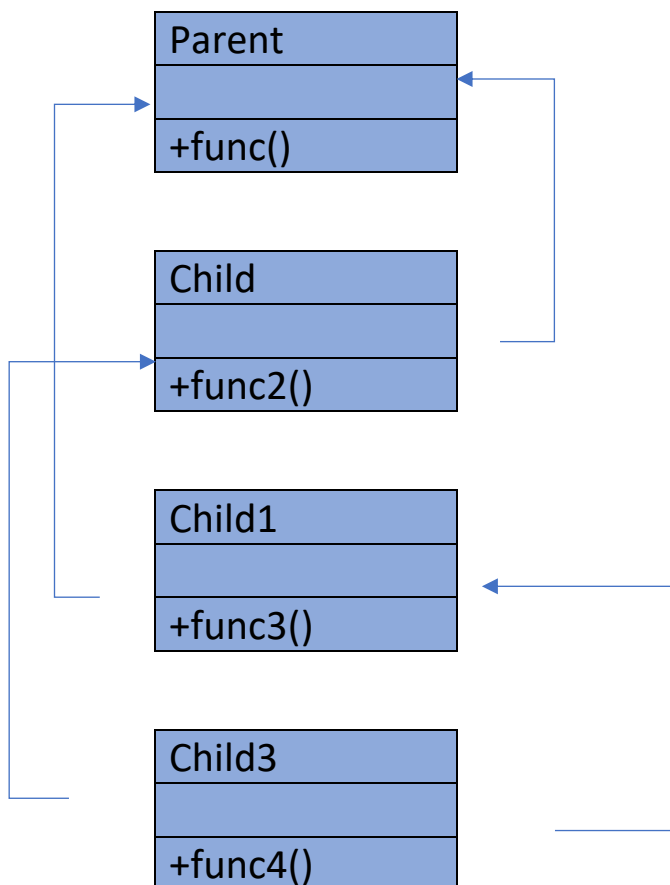
```
IDLE Shell 3.8.8
File Edit Shell Debug Options Window Help
Python 3.8.8 (tags/v3.8.8:024d805, Feb 19 2021, 13:18:16) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\HP\practical 6d oop.py =====
Employee Object
Id : 1
Name : Komal Menaria
Gender : Female
Company : gmr
Department : excavation

Doctor object
Id : 1
Name : Devangi Parmar
Gender : Female
Hospital : <No Hospital>
Department : eyes
>>>
```

PRACTICAL: 6 E

Aim: Draw a program to implement hybrid inheritance and make a class diagram for it.

CLASS DIAGRAM:




Name: KOMAL MENARIA

Class: FYBSCIT

Roll No: 42


CODE:

 practical 6e oop.py - C:/Users/HP/practical 6e oop.py (3.8.8)

File Edit Format Run Options Window Help

```
class Parent:
    def func1(self):
        print("this is function one")
class child(Parent):
    def func2(self):
        print("this is function 2")
class child1(Parent):
    def func3(self):
        print("this is function 3")
class child3(child , child1):
    def func4(self):
        print("this is function 4")
ob = child3()
ob.func1()
ob.func2()
ob.func3()
ob.func4()
|
```

OUTPUT:

 IDLE Shell 3.8.8

File Edit Shell Debug Options Window Help

Python 3.8.8 (tags/v3.8.8:024d805, Feb 19 2021, 13:18:16) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>>

===== RESTART: C:/Users/HP/practical 6e oop.py =====

this is function one

this is function 2

this is function 3

this is function 4

>>> |

PRACTICAL-7A

Aim: Write a program to implement method overloading

Writeup:

Polymorphism: Polymorphism means the ability to take various forms. In python, Polymorphism allows us to define methods in the child class with the same name as defined in their parent class.

Overloading: It is the ability of a function or an operator to behave in different ways based on the parameters that are passed to the function or the operands that the operator acts on.

Method overloading in python: In python, we can create a method that can be called in different ways so, we can have a method that has zero, one or more number of parameters. Depending on the method definition, we can call it with zero, one or more argument.

Method Overriding: while child class can access the parent class methods, it can also provide a new implementation to the parent class methods, which is called method overriding.

CODE:

Name: KOMAL MENARIA

Class: FYBSCIT

Roll No: 42

practical 7 A oop.py - C:/Users/HP/practical 7 A oop.py (3.8.8)

File Edit Format Run Options Window Help

#PROGRAM for method overloading

```
class Human:
    def sayHello(self, name= None):
        if name is not None:
            print("Hello" + name)
        else:
            print("Hello")
```

#create instance

obj = Human()

#call the method

obj.sayHello()

|

Output:

IDLE Shell 3.8.8

File Edit Shell Debug Options Window Help

Python 3.8.8 (tags/v3.8.8:024d805, Feb 19 2021, 13:18:16) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>>

===== RESTART: C:/Users/HP/practical 7 A oop.py =====

Hello

>>> |

PRACTICAL-7B

Aim: Write a program to implement method overloading

CODE:

practical 7 B oop.py - C:/Users/HP/practical 7 B oop.py (3.8.8)

File Edit Format Run Options Window Help

#method overloading

```
class employee:
    def add(self, salary, incentive):
        print('Total salary in base class= ', salary+incentive)
```

class department(employee):

temp = 'I m member of dept class'

def add(self, salary , incentive):

print(self.temp)

print('Total salary in derived class= ', salary+incentive)

dept = department()

dept.add(45000,5000)

|

Name: KOMAL MENARIA

Class: FYBSCIT

Roll No: 42

OUTPUT:

```
IDLE Shell 3.8.8
File Edit Shell Debug Options Window Help
Python 3.8.8 (tags/v3.8.8:024d805, Feb 19 2021, 13:18:16) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/HP/practical 7 B oop.py =====
l m member of dept class
Total salary in derived class= 50000
>>> |
```

PRACTICAL-7C

Aim: Write a program to implement method overloading using super

CODE:

```
practical 7 C oop.py - C:/Users/HP/practical 7 C oop.py (3.8.8)
File Edit Format Run Options Window Help
#method overloading using(super)
class Employee:
    def add(self, salary,incentive):
        print('Total salary in base class= ', salary+incentive)

class Department(Employee):
    temp = 'l m member of dept class'
    def add(self, salary , incentive):
        print(self.temp)
        print('Total salary in derived class=', salary+incentive)
        super(Department,self).add(salary,incentive)

dept = Department()
dept.add(45000,5000)
```


Name: KOMAL MENARIA

Class: FYBSCIT

Roll No: 42

```
IDLE Shell 3.8.8
File Edit Shell Debug Options Window Help
Python 3.8.8 (tags/v3.8.8:024d805, Feb 19 2021, 13:18:16) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/HP/practical 7 C oop.py =====
1 m member of dept class
Total salary in derived class= 50000
Total salary in base class= 50000
>>> |
```

PRACTICAL-7D

Aim: Write a program to implement method overloading using multiple class

CODE:

```
practical 7 D oop.py - C:/Users/HP/practical 7 D oop.py (3.8.8)
File Edit Format Run Options Window Help
import math

class Polygons:
    def number_of_sides(self):
        return 0
    def area(self):
        return 0
    def perimeter(self):
        return 0

class Triangle(Polygons):
    def number_of_sides(self):
        return 3
    def area(self):
        base=10
        height=12
        return 1 / 2 * base * height

    def perimeter(self):
        a=10
        b=10
        c=12
        if a + b > c:
            return a+b+c
        else:
            return "Invalid input make sure a+b>c"

class Rhombus(Polygons):
    def number_of_sides(self):
        return 4
    def area(self):
        p=4
        q=5
        return p*q/2
    def perimeter(self):
        a=5
        return 4*a
```

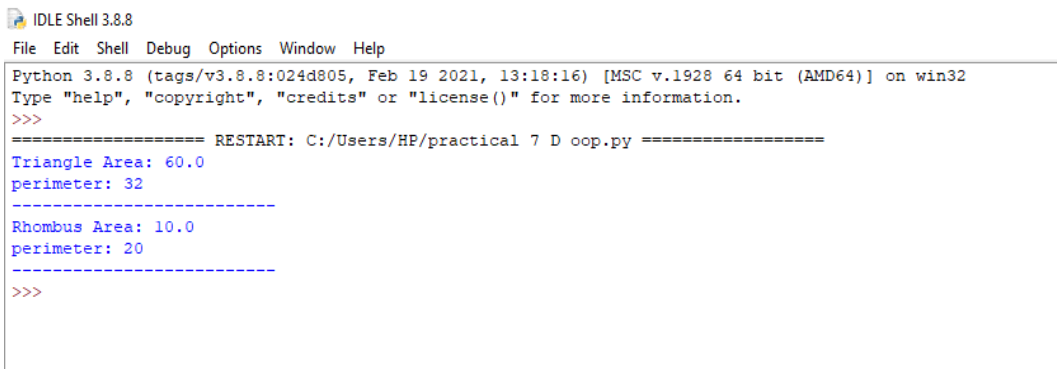
Name: KOMAL MENARIA

Class: FYBSCIT

Roll No: 42

```
tri = Triangle()
print("Triangle Area:",tri.area())
print("perimeter:",tri.perimeter())
print("-----")

rho = Rhombus()
print("Rhombus Area:",rho.area())
print("perimeter:",rho.perimeter())
print("-----")
```



```
IDLE Shell 3.8.8
File Edit Shell Debug Options Window Help
Python 3.8.8 (tags/v3.8.8:024d805, Feb 19 2021, 13:18:16) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/HP/practical 7 D oop.py =====
Triangle Area: 60.0
perimeter: 32
-----
Rhombus Area: 10.0
perimeter: 20
-----
>>>
```

PRACTICAL 8A

Aim: Write a program to demonstrate the use of different types of built in methods used in operator overloading.

Writeup:

Operator Overloading means giving extended meaning beyond their predefined operational meaning. For example operator + is used to add two integers as well as join two strings and merge two lists. It is achievable because '+' operator is overloaded by int class and str class. You might have noticed that the same built-in operator or function shows different behavior for objects of different classes, this is called Operator Overloading.

Name: KOMAL MENARIA

Class: FYBSCIT

Roll No: 42

practical 8A OOP.py - C:\Users\HP\practical 8A OOP.py (3.8.8)

```
File Edit Format Run Options Window Help
print("x.__mul__(y)=",x.__mul__(y))

print("\nx / y=",x/y)
print("x.__truediv__(y)=",x.__truediv__(y))

print("\nx ** y=",x**y)
print("x.__pow__(y)=",x.__pow__(y))

print("\nx % y=",x%y)
print("x.__mod__(y)=",x.__mod__(y))

print("\nx == y=",x==y)
print("x.__eq__(y)=",x.__eq__(y))

print("\nx != y=",x!=y)
print("x.__ne__(y)=",x.__ne__(y))

print("\nx >= y=",x>=y)
print("x.__ge__(y)=",x.__ge__(y))

print("\nx <= y=",x<=y)
print("x.__le__(y)=",x.__le__(y))
print("-----")

str1 = "special method"
print("\nstr1=",str1)

print("\n'ods' in str1=", "ods" in str1)
print("str1.__contains__('ods')=",str1.__contains__("ods"))

print("\n len(str1) =",len(str1))
print("str1.__len__() =",str1.__len__())

list1=[5,3,20]
print("\nlist1 =",list1)

print("\nlist1[1] =",list1[1])
print("list1.__getitem__(1)=",list1.__getitem__(1))

print("str(list1)=",str(list1))
```

Output:

IDLE Shell 3.8.8

```
File Edit Shell Debug Options Window Help
Python 3.8.8 (tags/v3.8.8:024d805, Feb 19 2021, 13:18:16) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\HP\practical 8A OOP.py =====
x = 5 y= 3

x + y= 8
x.__add__(y)= 8

x * y= 15
x.__mul__(y)= 15

x / y= 1.6666666666666667
x.__truediv__(y)= 1.6666666666666667

x ** y= 125
x.__pow__(y)= 125

x % y= 2
x.__mod__(y)= 2

x == y= False
x.__eq__(y)= False

x != y= True
x.__ne__(y)= True

x >= y= True
x.__ge__(y)= True

x <= y= False
x.__le__(y)= False
-----

str1= special method

'ods' in str1= False
str1.__contains__('ods')= False

len(str1) = 14
str1.__len__() = 14
```

Name: KOMAL MENARIA

Class: FYBSCIT

Roll No: 42

```
list1 = [5, 3, 20]

list[1] = 3
list1.__getitem__(1)= 3
str(list1)= [5, 3, 20]
>>> |
```

PRACTICAL 8B

Aim: Write a program to overload Binary Plus operator.

CODE:

assgnment2(1).py - C:/Users/HP/assgnment2(1).py (3.8.8)

File Edit Format Run Options Window Help

```
# Python Program illustrate how
# to overload an binary + operator
class A:
    def __init__(self, a):
        self.a = a

    # adding two objects
    def __add__(self, o):
        return self.a + o.a
ob1 = A(40)
ob2 = A(2)
ob3 = A("FY")
ob4 = A("IT")

print(ob1 + ob2)
print(ob3 + ob4)
|
```

Output:

IDLE Shell 3.8.8

File Edit Shell Debug Options Window Help

Python 3.8.8 (tags/v3.8.8:024d805, Feb 19 2021, 13:18:16) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>>

===== RESTART: C:/Users/HP/assgnment2(1).py =====

42

FYIT

>>> |

Name: KOMAL MENARIA

Class: FYBSCIT

Roll No: 42

PRACTICAL 8C

Aim: Write a program to overload binary plus to add two complex numbers.

CODE:

```
assignment2(1).py - C:/Users/HP/assignment2(1).py (3.8.8)
File Edit Format Run Options Window Help

# Python Program to perform addition
# of two complex numbers using binary
# + operator overloading.

class complex:
    def __init__(self, a, b):
        self.a = a
        self.b = b

    # adding two objects
    def __add__(self, other):
        return self.a + other.a, self.b + other.b

    def __str__(self):
        return self.a, self.b

Ob1 = complex(5, 8)
Ob2 = complex(2, 3)
Ob3 = Ob1 + Ob2
print(Ob3)
```

PRACTICAL 8D

Aim: Write a program to overload comparison operator

CODE:

```
assignment2(1).py - C:/Users/HP/assignment2(1).py (3.8.8)
File Edit Format Run Options Window Help

# Python program to overload
# a comparison operators

class A:
    def __init__(self, a):
        self.a = a
    def __gt__(self, other):
        if(self.a>other.a):
            return True
        else:
            return False

ob1 = A(2)
ob2 = A(3)
if(ob1>ob2):
```

Name: KOMAL MENARIA

Class: FYBSCIT

Roll No: 42

OUTPUT:

```
IDLE Shell 3.8.8
File Edit Shell Debug Options Window Help
Python 3.8.8 (tags/v3.8.8:024d805, Feb 19 2021, 13:18:16) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/HP/assignment2(1).py =====
object2 is greater than object1
>>>
```

PRACTICAL 8E

5. Write a program to overload equality and less than operator.

CODE:

```
assignment2(1).py - C:/Users/HP/assignment2(1).py (3.8.8)
File Edit Format Run Options Window Help
# Python program to overload equality
# and less than operators

class A:
    def __init__(self, a):
        self.a = a
    def __lt__(self, other):
        if(self.a<other.a):
            return "object1 is less than object2"
        else:
            return "object2 is less than object1"
    def __eq__(self, other):
        if(self.a == other.a):
            return "Both are equal"
        else:
            return "Not equal"

ob1 = A(2)
ob2 = A(3)
print(ob1 < ob2)

ob3 = A(4)
ob4 = A(4)
print(ob1 == ob2)
```

Name: KOMAL MENARIA

Class: FYBSCIT

Roll No: 42

OUTPUT:

```
IDLE Shell 3.8.8
File Edit Shell Debug Options Window Help
Python 3.8.8 (tags/v3.8.8:024d805, Feb 19 2021, 13:18:16) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/HP/assignment2(1).py =====
object1 is lessthan object2
Not equal
>>>
```

PRACTICAL 9

Aim: Write a program to calculate no of sides, area and perimeter of shades like rectangle, square, circle and ellipse by implementing the concept of abstract classes and make a class diagram for it.

Writeup:

An abstract class can be considered as a blueprint for other classes. It allows you to create a set of methods that must be created within any child classes built from the abstract class. A class which contains one or more abstract methods is called an abstract class. An abstract method is a method that has a declaration but does not have an implementation. While we are designing large functional units we use an abstract class. When we want to provide a common interface for different implementations of a component, we use an abstract class.

Name: KOMAL MENARIA

Class: FYBSCIT

Roll No: 42

Class diagram:

Shape {Abstract}
+no_of_sides() +calculate_area() +calculates_perimeter()

Ellipse
+semiMajorAxis +semiMinorAxis
+calculate_area() +calculate_perimeter()

Rectangle
+length +breadth
+calculate_area() +calculate_perimeter()

Square
+side
+calculate_area() +calculate_perimeter()

Code:

Name: KOMAL MENARIA

Class: FYBSCIT

Roll No: 42

practical 9 oop.py - C:/Users/HP/practical 9 oop.py (3.8.8)

File Edit Format Run Options Window Help

```
from abc import ABC
import math

class Shape(ABC):

    def no_of_sides(self):
        pass

    def calculate_area(self):
        pass

    def calculate_perimeter(self):
        pass

class Rectangle(Shape):
    def no_of_sides(self):
        self.length = int(input("Enter length : "))
        self.breadth = int(input("Enter Breadth : "))

    def calculate_area(self):
        print("Area of Rectangle : ",self.length*self.breadth,"\n")

    def calculate_perimeter(self):
        print("Perimeter of rectangle : ",2*(self.length+self.breadth),"\n")

class Square(Shape):

    def no_of_sides(self):
        self.side = int(input("Enter side : "))

    def calculate_area(self):
        print("Area of square : ",self.side**2,"\n")

    def calculate_perimeter(self):
        print("Perimeter of square : ", 4 * self.side,"\n")

class Circle(Shape):

    def no_of_sides(self):
        self.radius = int(input("Radius : "))
```

Output:

IDLE Shell 3.8.8

File Edit Shell Debug Options Window Help

Python 3.8.8 (tags/v3.8.8:024d805, Feb 19 2021, 13:18:16) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>>

===== RESTART: C:/Users/HP/practical 9 oop.py =====

Enter length : 435
Enter Breadth : 45
Area of Rectangle : 19575

Perimeter of rectangle : 960

Enter side : 245
Area of square : 60025

Perimeter of square : 980

Radius : 24
Area of circle : 150.85714285714286
Perimeter of circle : 150.85714285714286

Enter Semimajor Axis : 42
Enter Semiminor Axis : 52
Area of Ellipse : 6864.0
Perimeter of Ellipse : 5134.710436579741
>>>

Name: KOMAL MENARIA
Class: FYBSCIT
Roll No: 42

PRACTICAL 10 A

Aim: Implementing concept of Composition.

Writeup:

Composition is a concept that models a has a relationship. It enables creating complex types by combining objects of other types. This means that a class Composite can contain an object of another class Component. This relationship means that a Composite has a Component.

CLASS DIAGRAM:

Salary
+pay
+get_total()

Employee
+pay
+bonus
+annual_sal()

CODE:

Name: KOMAL MENARIA

Class: FYBSCIT

Roll No: 42

practical 10 A.py - C:/Users/HP/practical 10 A.py (3.8.8)

File Edit Format Run Options Window Help

```
#Program demonstrating the concept corpostion

#Creating container class

class Salary:

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

    def get_total(self):
        return (self.pay*12)

class Employee:

    def __init__(self,pay,bonus):
        self.pay = pay
        self.bonus = bonus

    def annual_sal(self):
        return 'Total:'+str(self.pay.get_total()+self.bonus)

#Creating objects
obj_sal = Salary(70000)
obj_exp = Employee(obj_sal,30000)
print(obj_exp.annual_sal())
```

Output:

IDLE Shell 3.8.8

File Edit Shell Debug Options Window Help

Python 3.8.8 (tags/v3.8.8:024d805, Feb 19 2021, 13:18:16) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>>

===== RESTART: C:/Users/HP/practical 10 A.py =====

Total:870000

>>>

Name: KOMAL MENARIA

Class: FYBSCIT

Roll No: 42

PRACTICAL 10 B

Aim: Implementing concept of Composition.

CODE:

```
practical 10 B.py - C:/Users/HP/practical 10 B.py (3.8.8)
File Edit Format Run Options Window Help

class Component:
    def __init__(self):
        print("Component class object created....")

    def m1(self):
        print("Component class m1() method executed.....")

class Composite:
    def __init__(self):
        self.obj1 = Component()
        print("Composite class object also created....")
    def m2(self):
        print("Composite class m2() method executed.....")
        self.obj1.m1()

obj2 = Composite()
obj2.m2()
```

OUTPUT:

```
IDLE Shell 3.8.8
File Edit Shell Debug Options Window Help

Python 3.8.8 (tags/v3.8.8:024d805, Feb 19 2021, 13:18:16) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/HP/practical 10 B.py =====
Component class object created....
Composite class object also created....
Composite class m2() method executed.....
Component class m1() method executed.....
>>> |
```