Student Name: _____ **Roll No:** _____ **Section:** _____

Lab No. 09

Lab 09– Operator overloading and Case Study on Polymorphism and Abstract Classes.

Objectives:

1. Polymorphism
2. Operator overloading
3. Abstract Classes

1. Polymorphism

The word polymorphism means having many forms. In programming, polymorphism means same function name (but different signatures) being uses for different types. Regular expressions use two types of characters:

- Polymorphism means that different types respond to the same function.
- Polymorphism is very useful as it makes programming more intuitive and therefore easier.
- Polymorphism is a fancy word that just means the same function is defined on objects of different types.

2. Operator overloading in Python

1. Operators are used in Python to perform specific operations on the given operands. The operation that any particular operator will perform on any predefined data type is already defined in Python.
2. Each operator can be used in a different way for different types of operands. For example, + operator is used for adding two integers to give an integer as a result but when we use it with float operands, then the result is a float value and when + is used with string operands then it concatenates the two operands provided.
3. This different behavior of a single operator for different types of operands is called Operator Overloading. The use of + operator with different types of operands is shown below

Student Name: _____ Roll No: _____ Section: _____

Operator	Special Method	Description
+	__add__(self, object)	Addition
-	__sub__(self, object)	Subtraction
*	__mul__(self, object)	Multiplication

3. Can + Operator Add anything?

The answer is No, it cannot. Can you use the + operator to add two objects of a class. The + operator can add two integer values, two float values or can be used to concatenate two strings only because these behaviors have been defined in python.

So if you want to use the same operator to add two objects of some user defined class then you will have to defined that behavior yourself and inform python about that.

If you are still not clear, let's create a class and try to use the + operator to add two objects of that class

Example 1

```
class Complex:

    def __init__(self, r, i):

        self.real = r

        self.img = i


c1 = Complex(5,3)

c2 = Complex(2,4)

print("sum = ", c1+c2)
```

Student Name: _____ Roll No: _____ Section: _____

Output:

4. Overloading + operator

```
class Complex:
    # defining init method for class
    def __init__(self, r, i):
        self.real = r
        self.img = i

    # overloading the add operator using special function
    def __add__(self, sec):
        r = self.real + sec.real
        i = self.img + sec.img
        return complex(r,i)

    # string function to print object of Complex class
    def __str__(self):
        return str(self.real)+' + '+str(self.img)+'i'

c1 = Complex(5,3)
c2 = Complex(2,4)
print("sum = ",c1+c2)
```

Output:

Student Name: _____ Roll No: _____ Section: _____

5. Overloading - Operator

```
class Point:
    def __init__(self, x=0, y=0):
        self.x = x
        self.y = y

    def __str__(self):
        return "({0},{1})".format(self.x, self.y)

    def __sub__(self, other):
        x = self.x - other.x
        y = self.y - other.y
        return Point(x, y)
p1 = Point(5, 4)
p2 = Point(4, 2)

print(p1-p2)
```

Output:

Student Name: _____ **Roll No:** _____ **Section:** _____

Abstract Classes in Python

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.

```
Syntax
from abc import ABC
Class ClassName(ABC):
```

Case Study 1:

Create a class fruit Use polymorphism to show the functions of these such as type (), color (), taste () etc.

Case Study 2:

Create an ABC class of Vehicle and add some abstract method like model, color now add some classes in which you will implement abstract class like Car, motorbike

Student Name: _____ **Roll No:** _____ **Section:** _____

Programming Exercise (Python)

Task 1:

Create a class distance that take feet, inches and then perform an operator overloading (Overload + Operator).

Task 2:

Create a class time that take hour's minutes, and second and then perform an operator overloading (Overload - Operator).