Lab Report 4 Date:2081/03/7

Experiment 1: Object-Oriented programming

Title: Programming to learn about Object-Oriented programming.

Objective:The objective of this lab-work is to represent real world systems more effectively, enhance code reusability, maintainability, and extensibility.

Theory: Object-Oriented Programming is a programming paradigm that is based on the concept of objects. It focuses on organizing software design around data, or objects, rather than functions and logic. In OOP, objects represent real-world entities and are instances which are blueprint of the object.

1. Define a class for a bank account and implement methods for deposit and withdrawal.

*class Bankaccount:*

*def \_\_init\_\_(self,ac\_holder,balance):*

*self.ac\_holder=ac\_holder*

*self.balance=balance*

*def deposit(self,amount):*

*if amount>0:*

*self.balance+=amount*

*print("The deposited balance is ",amount," and new balance ",self.balance)*

*else:*

*print("The deposited balance must be positive")*

*def withdraw(self,amount):*

*if amount>0:*

*if amount<=self.balance:*

*self.balance-=amount*

*print("Withdarwn amount is ",amount, " and new balance is ",self.balance)*

*else:*

*print("Insufficient balance")*

*else:*

*print("Withdrawal amount must be positive.")*

*def get\_balance(self):*

*return self.balance*

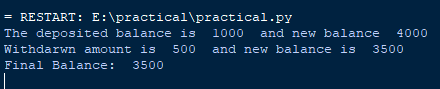
*account=Bankaccount("Alice",3000)*

*account.deposit(1000)*

*account.withdraw(500)*

*print("Final Balance: ",account.get\_balance())*

Output:



1. WAP to create a class for rectangle and calculate its perimeter and area.

*class Rectangle:*

*def \_\_init\_\_(self,length,breadth):*

*self.length=length*

*self.breadth=breadth*

*def area(self):*

*area=self.length\*self.breadth*

*print("The area of the rectangle is ",area)*

*def perimeter(self):*

*perimeter=2\*(self.length+self.breadth)*

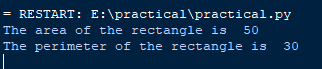
*print("The perimeter of the rectangle is ",perimeter)*

*r=Rectangle(10,5)*

*r.area()*

*r.perimeter()*

Output:



1. Create a class to represent complex number and perform addition and multiplication.

*class Complexnumber:*

*def \_\_init\_\_(self,real,imag):*

*self.real=real*

*self.imag=imag*

*def \_\_add\_\_(self,other):*

*real\_part=self.real+other.real*

*imag\_part=self.imag+other.imag*

*return Complexnumber(real\_part,imag\_part)*

*def \_\_mul\_\_(self,other):*

*real\_part=self.real\*other.real*

*imag\_part=self.imag\*other.imag*

*return Complexnumber(real\_part,imag\_part)*

*def \_\_repr\_\_(self):*

*return f"{self.real}+{self.imag}i" if self.imag >=0 else f"{self.real}+{self.imag}i"*

*complex1=Complexnumber(2,3)*

*complex2=Complexnumber(1,4)*

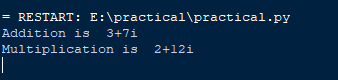
*result\_add=complex1+complex2*

*result\_mul=complex1\*complex2*

*print("Addition is ",result\_add)*

*print("Multiplication is ",result\_mul)*

Output:



1. Write a Python program to demonstrate method overloading.

*class Calculator:*

*def add(self, \*args):*

*total=0*

*for num in args:*

*total+=num*

*return total*

*calc=Calculator()*

*result1=calc.add(1,2)*

*result2=calc.add(5,10,15)*

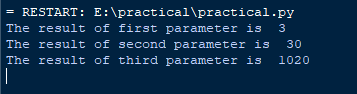
*result3=calc.add(100,220,300,400)*

*print("The result of first parameter is ",result1)*

*print("The result of second parameter is ",result2)*

*print("The result of third parameter is ",result3)*

Output:



1. Implement inheritance with a base class and derived class example.

*class Animal:*

*def \_\_init\_\_(self, name, age):*

*self.name = name*

*self.age = age*

*def speak(self):*

*return print(self.name,"makes sound")*

*class Dog(Animal):*

*def \_\_init\_\_(self, name, age, breed):*

*super().\_\_init\_\_(name, age)*

*self.breed = breed*

*def speak(self):*

*return print(self.name," barks")*

*class Cat(Animal):*

*def \_\_init\_\_(self, name, age, color):*

*super().\_\_init\_\_(name, age)*

*self.color = color*

*def speak(self):*

*return print(self.name," meow")*

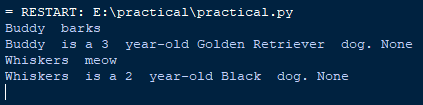
*dog = Dog("Buddy", 3, "Golden Retriever")*

*cat = Cat("Whiskers", 2, "Black")*

*print(dog.name ," is a",dog.age," year-old",dog.breed, " dog.", dog.speak())*

*print(cat.name ," is a",cat.age," year-old",cat.color, " dog.", cat.speak())*

Output:



Conclusion: In the above page we have done the objrct oriented programming with its output .