PRACTICAL – 5

PROGRAM -1

AIM- Program Based on Operator Overloading

CODE- OUTPUT:-

|  |  |
| --- | --- |
| |  | | --- | |  |   # This class represents a Jaguar car.  class Jaguar:    """    This class represents a Jaguar car.    """    def \_\_init\_\_(self,mileage):      """      Initialize the Jaguar car.      Args:        mileage: The mileage of the car.      """      self.mileage = mileage  # This class represents an Audi car.  class Audi:    """    This class represents an Audi car.    """    def \_\_init\_\_(self,mileage):      """      Initialize the Audi car.      Args:        mileage: The mileage of the car.      """      self.mileage = mileage    def \_\_add\_\_(self,other):      """      Add the mileage of two cars.      Args:        other: The other car.      Returns:        The combined mileage of the two cars.      """      return self.mileage + other.mileage  # Create a Jaguar car.  jaguar = Jaguar(12)  # Create an Audi car.  audi = Audi(10)  # Print the mileage of the Jaguar car.  print(jaguar.mileage)  # Print the mileage of the Audi car.  print(audi.mileage)  # Print the combined mileage of the Jaguar and Audi cars.  print(audi + jaguar) |

PROGRAM -2.1

AIM- Program Based on Method Overloading {Simple Parameter}

[TYPE ERROR]

CODE- OUTPUT:-

|  |  |
| --- | --- |
| |  | | --- | |  |   class Example:    def Add(self,a,b):      """This function adds two numbers"""      x=a+b      print(x)    def Add(self,a,b,c):      """This function adds three numbers"""      x=a+b+c      print(x)  E1=Example()  E1.Add(1,2)  E1.Add(1,2,3)  # Output = TypeError: Example.Add() missing 1 required positional argument: 'c' |

PROGRAM -2.2

AIM- Program Based on Method Overloading {Simple Parameter}

[WITHOUT TYPE ERROR]

CODE-

|  |
| --- |
| class Example:    def Add(self,a,b):      x=a+b      print(x)    def Add(self,a,b,c):      x=a+b+c      print(x)  E1=Example()  E1.Add(1,2,3)  # Output = 6 |

OUTPUT:-

|  |
| --- |
|  |

PROGRAM -3

AIM- Program Based Method Overloading {Default Parameter}

CODE-

|  |
| --- |
| print("HARSH D")  class Example:    """This class can add numbers together."""    def Add(self,x=0,y=0,z=0):      """Adds three numbers together and prints the result.      Keyword Arguments:      x: The first number      y: The second number      z: The third number      """      # Add the three numbers      sum = x+y+z        # Print the result      print("Sum = ",sum)  obj = Example()  obj.Add(10,20,30)  obj.Add(10)  obj.Add(10,20) |

OUTPUT:-

|  |
| --- |
|  |

PROGRAM -4

AIM- Program Based Method Overloading {Multidispatch Parameter}

CODE-

|  |
| --- |
| print("AHRSH D")  from multipledispatch import dispatch  # passing one parameter  @dispatch(int, int)  def sum(first, second):    """Add two integers together and print the results"""    result = first+second    print(result)  # passing two parameters  @dispatch(int, int, int)  def sum(first, second, third):    """Add three integers together and print the results"""    result = first + second + third    print(result)  # you can also pass data type of any value as per requirement  @dispatch(float, float, float)  def sum(first, second, third):    """Add three floats together and print the results"""    result = first + second + third    print(result)  # calling sum method with 2 arguments  sum(2, 3) # this will give output of 6  # calling sum method with 3 arguments but all int  sum(2, 3, 2) # this will give output of 12  # calling sum method with 3 arguments but all float  sum(2.2, 3.4, 2.3) # this will give output of 17.985999999999997 |

Output:-

|  |
| --- |
|  |

PROGRAM -5

AIM- Program Based Method Over - Ridding

CODE-

|  |
| --- |
| class Animal:                  # Parent class    """This class defines basic properties and behaviors of an animal."""    def walk(self):              #self is the object      """This method causes the animal to walk."""      print("Animal is Walking...")  # Method    def Communicate(self):        #Abstract Method      """This method causes the animal to communicate."""      print("Animal is Communicating...")   # Communicating is a method of Animal class  class Cat(Animal):    # Inheritance    """This class defines properties and behaviors of a cat."""    def walk(self):     # Polymorphism      """This method causes the cat to walk."""      print("Cat is Walking...")   #overrides the method of the parent class    def Eat(self):         #Method Overriding      """This method causes the cat to eat."""      print("Cat is Eating...") # Overriding  object = Cat()          # object is an instance of class Cat  print(object.walk()) # Cat is Walking...  print(object.Eat())  # This line prints out "Cat is Eating..." |

Output:-

|  |
| --- |
|  |