**Assignment: 8**

**Experiments**

1. **Write a class named Rectangle. Take length and breadth as attributes and area as methods. Create a subclass Square with the only attribute as side and override the area method. Create Rectangle and Square object statically and dynamically and make use of area method.**

**Coding:**

class Rectangle:  
 # Constructor  
 def \_\_init\_\_(self, length, breadth):  
 self.length = length  
 self.breadth = breadth  
  
 # Area of Rectangle  
  
 def area(self):  
 return self.length \* self.breadth  
  
 # SubClass  
  
  
class Square(Rectangle):  
 # Constructor  
 def \_\_init\_\_(self, side):  
 self.length = side  
 self.breadth = side  
  
 # Objects of Rectangle  
  
  
a = int(input("Enter the length of Rectangle: "))  
b = int(input("Enter the breadth of Rectangle: "))  
obj1 = Rectangle(a, b)  
  
# Area of Rectangle  
print("Area of Rectangle:", obj1.area())  
  
# Object of Square  
c = int(input("Enter the side of Square: "))  
obj2 = Square(c)  
  
# Area of Square  
print("Area of Square:", obj2.area())

**Output:**

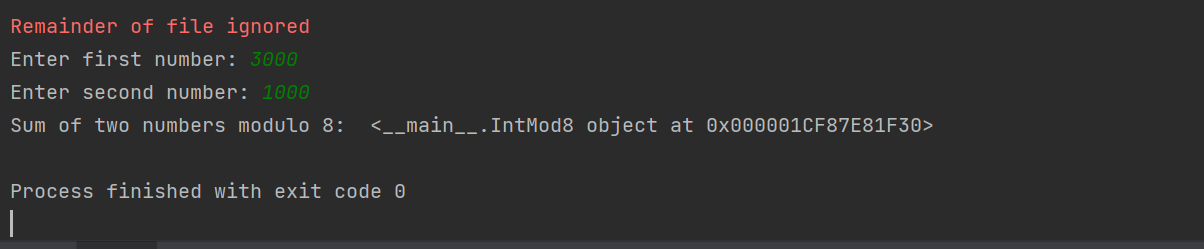


1. **Overload the + (addition) operator to restrict the addition of two integers to modulo 8.**

**Coding:**

class IntMod8:  
 def \_\_init\_\_(self,value):  
 self.value=value  
  
 def \_\_add\_\_(self,other):  
 return IntMod8(self.value%8 + other.value%8)  
  
x = IntMod8(int(input("Enter first number: ")))  
y = IntMod8(int(input("Enter second number: ")))  
  
print("Sum of two numbers modulo 8: ",x+y)

**Output:**



1. **Implement a Stack class using a list type. Provide push() and pop() operations. Demonstrate the usage.**

**Coding:**

class Stack:  
 def \_\_init\_\_(self):  
 self.stack = []  
  
 def push(self, val):  
 self.stack.append(val)  
  
 def pop(self):  
 return self.stack.pop()  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 s = Stack()  
 # Taking input from user  
 n = int(input("Enter the number of elements:"))  
 print("Enter the elements:")  
 for i in range(n):  
 s.push(input())  
  
 print("Stack elements are:")  
 while s.stack:  
 print(s.pop())

**Output:**

