	SHRI SANT GAJANAN MAHARAJ COLLEGE OF ENGG.	**LABORATORY MANUAL**
	PRACTICAL EXPERIMENT INSTRUCTION SHEET	
	EXPERIMENT TITLE: Write python program in which an class is define, then create object of that class and call simple print function define in class.	
EXPERIMENT NO. : **SSGMCE/WI/IT/01/3IT09/01**	ISSUE NO. : 00	ISSUE DATE : 30.07.2023
REV. DATE :	REV. NO. :	DEPTT. : INFORMATION TECHNOLOGY
LABORATORY : 3IT09 COMPUTER SKILL LAB – I	SEMESTER : III	PAGE: 1 OF 2

1.0) AIM: Write python program in which an class is define, then create object of that class and call simple print function define in class.

2.0) SCOPE: The scope of this Python program is to introduce students to the concept of object-oriented programming (OOP) by demonstrating the creation of a class, instantiation of an object, and calling a method within the class. It serves as a foundational example of OOP principles in Python.

3.0) FACILITIES/ APPARATUS:

1. Python development environment (e.g., IDLE)
2. Input mechanism (keyboard)
3. Computer with Python installed

4.0) THEORY:

Program Description:

1. Class Definition: The program begins by defining a class named MyClass. Inside the class, we have two components:
    - Constructor Method (__init__): This method is called when an object of the class is created. In our example, the constructor doesn't initialize any specific attributes.
    - Print Message Method (print_message): This method is defined within the class and prints the message "Hello from MyClass!" when called.
2. Object Creation: After defining the class, the program proceeds to create an object of the MyClass class by invoking the class constructor. This creates an instance of the class and assigns it to the variable my_object.
3. Method Invocation: The program then calls the print_message() method on the my_object instance. This is achieved using dot notation (my_object.print_message()), and it triggers the execution of the print_message method.

Example:

Suppose we have the following code:

Program

```
my_object = MyClass()
my_object.print_message()
```

The output will be:

```
Hello from MyClass!
```

Program

```
# Define a class named MyClass
class MyClass:
    # Constructor method to initialize the object
    def __init__(self):
        pass  # No need to initialize anything in this example

    # Define a simple print function within the class
    def print_message(self):
        print("Hello from MyClass!")

# Create an object of the MyClass class
my_object = MyClass()

# Call the print_message function of the object
my_object.print_message()
```

Output

```
Hello from MyClass!
```

In this program:

- We define a class named MyClass. Inside the class:
- We have a constructor method __init__(self) which is called when an object of the class is created. In this example, the constructor doesn't do anything specific.
- We define a simple function print_message(self) within the class. This function prints the message "Hello from MyClass!" when called.
- We create an object of the MyClass class by calling MyClass(). This creates an instance of the class and assigns it to the variable my_object.
- We call the print_message() function on the my_object instance. This invokes the function, and it prints the message to the console.

This program demonstrates how to define a class, create an object of that class, and call a function within the class using the object. It provides a basic example of object-oriented programming (OOP) in Python.

4.2) Program Execution:

1. Line 4-7: We define a class named MyClass. Inside the class:
   - The constructor method (__init__) is defined. It takes self as a parameter, which represents the instance of the class. In this example, the constructor does nothing specific.

- The print_message method is defined, taking self as a parameter. This method prints the message "Hello from MyClass!" when called.

2. Line 10: An object of the MyClass class is created by calling MyClass(). This is known as instantiation. The created object is assigned to the variable my_object.

3. Line 13: We call the print_message() method on the my_object instance using dot notation (my_object.print_message()). This invokes the method, and the message is printed to the console.

5.0) Conclusion:

In conclusion, this program provides an essential introduction to object-oriented programming (OOP) in Python. It demonstrates the fundamental concepts of defining classes, creating objects, and calling methods within those classes. Students can learn the importance of encapsulation and code organization through classes and objects. Understanding OOP is crucial for building complex, modular, and maintainable Python programs, making it a valuable skill for aspiring programmers.