

Date – 31/05/2021

Name – NIRAV MOJAGAR

College – GOVERNMENT ENGINEERING COLLEGE, MODASA

Branch – Information Technology (BE)

Sem -7th

## Task

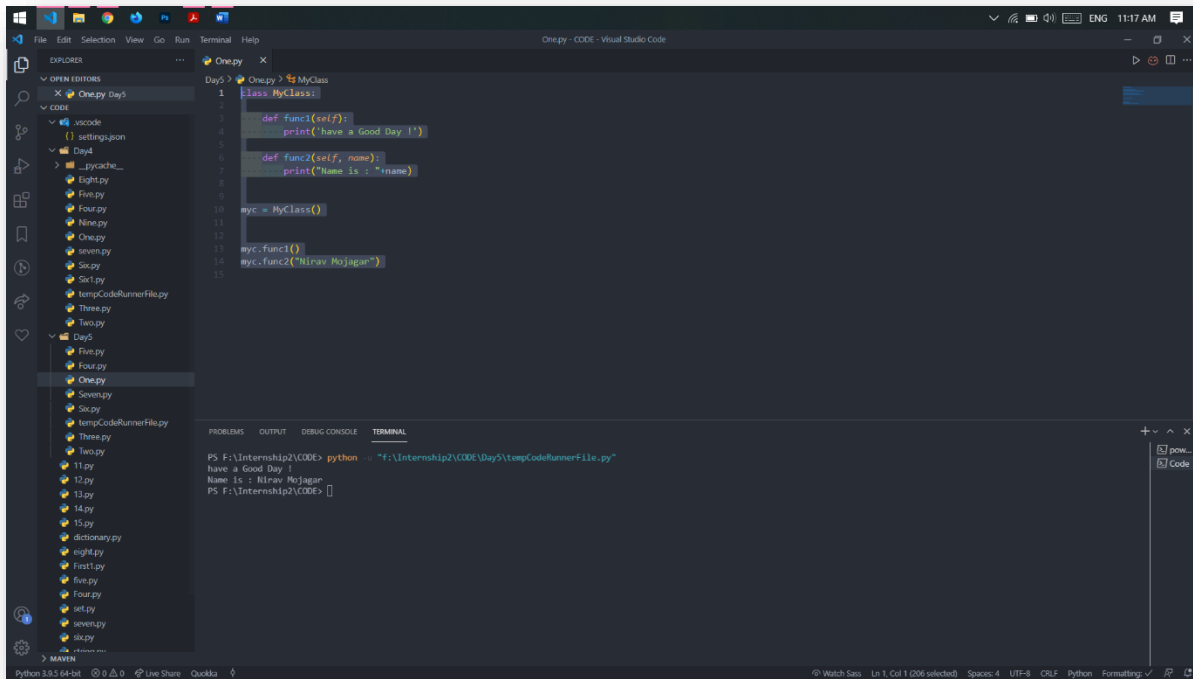
### Python OOP'S Concept –

#### 1. Code



```
1  class MyClass:
2
3      def func1(self):
4          print('have a Good Day !')
5
6      def func2(self, name):
7          print("Name is : "+name)
8
9
10 myc = MyClass()
11
12
13 myc.func1()
14 myc.func2("Nirav Mojagar")
15
```

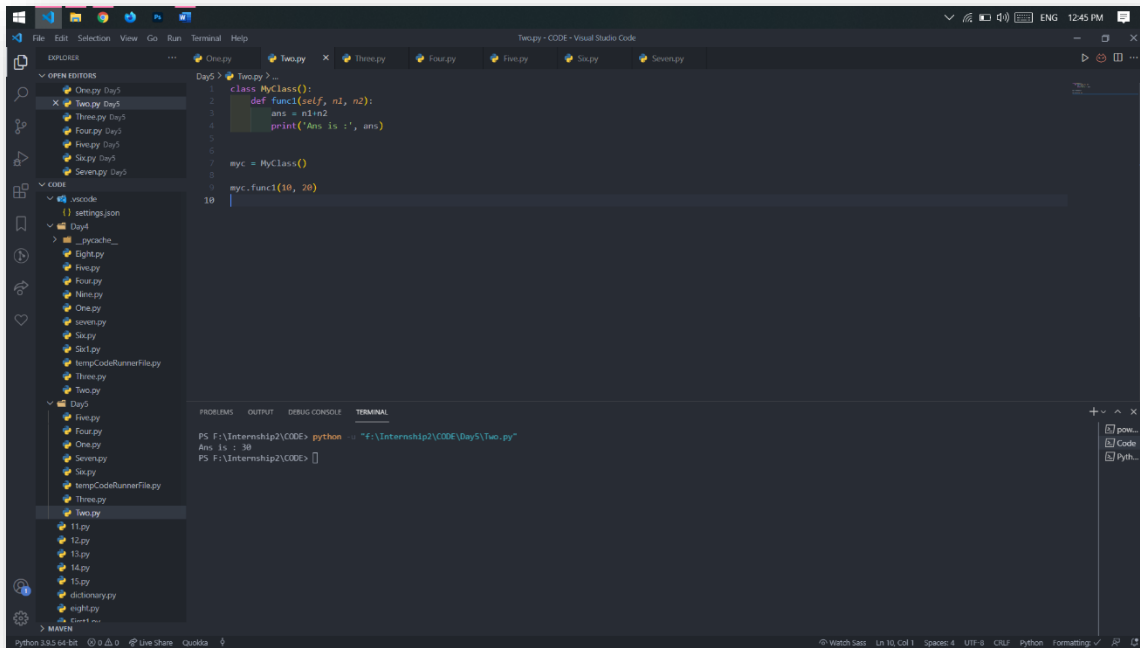
#### Output :-



## 2. Code

```
1 class MyClass():
2     def func1(self, n1, n2):
3         ans = n1+n2
4         print('Ans is :', ans)
5
6
7 myc = MyClass()
8
9 myc.func1(10, 20)
```

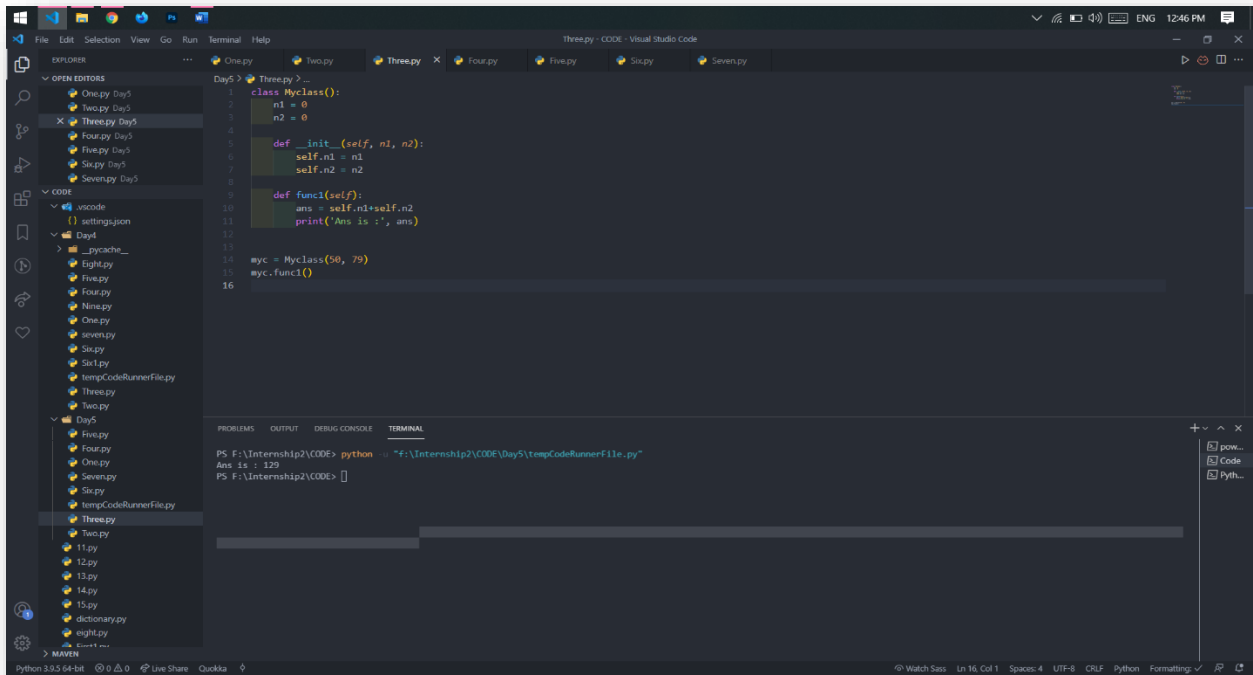
### Output :-



### 3. Code

```
1 class MyClass():
2     n1 = 0
3     n2 = 0
4
5     def __init__(self, n1, n2):
6         self.n1 = n1
7         self.n2 = n2
8
9     def func1(self):
10        ans = self.n1+self.n2
11        print('Ans is :', ans)
12
13
14 myc = MyClass(50, 79)
15 myc.func1()
```

## Output



## 4. Code

```

1 class Parent():
2     def __init__(self):
3         print("Calling parent constructor")
4
5     def parentMethod(self):
6         print("Calling parent method ")
7
8     # define class child
9
10
11 class Child(Parent):
12     def __init__(self):
13         print("Calling the Constructor")
14
15     def childMethod(self):
16         print("Calling Child method")
17
18
19 c = Child()
20 c.childMethod()
21 c.parentMethod()
  
```

Output :-

```

1 # Inheritance Example
2 class Parent():
3     def __init__(self):
4         print("Calling parent constructor")
5
6     def parentMethod(self):
7         print("Calling parent method ")
8
9     # define class child
10
11
12 class Child(Parent):
13     def __init__(self):
14         print("Calling the Constructor")
15
16     def childMethod(self):
17         print("Calling Child method")
18
19
20 c = Child()
21 c.childMethod()
22 c.parentMethod()
23

```

Terminal Output:

```

PS F:\Internship2\CODE> python "f:\Internship2\CODE\Day5\Four.py"
Calling the Constructor
Calling Child method
Calling parent method
PS F:\Internship2\CODE>

```

5. Code :-

```

1 class MyParentClass1():
2
3     def method_Parent1(self):
4         print("Parent1 method called")
5
6
7 class MyParentClass2():
8     def method_Parent2(self):
9         print("Parent2 method called")
10
11
12 class ChildClass(MyParentClass1, MyParentClass2):
13
14     def child_method(self):
15         print("Child method")
16
17
18 c = ChildClass()
19 c.method_Parent1()
20 c.method_Parent2()
21 c.child_method()

```

**Output :-**

The screenshot shows a Visual Studio Code editor with a Python file named 'Five.py'. The code defines two parent classes, 'MyParentClass1' and 'MyParentClass2', each with a method 'method\_Parent1' and 'method\_Parent2' respectively. A child class 'ChildClass' inherits from both parents and has a method 'child\_method'. The script creates an instance 'c' of 'ChildClass' and calls 'c.method\_Parent1()', 'c.method\_Parent2()', and 'c.child\_method()'. The terminal output shows the execution of these methods in sequence.

```

1 class MyParentClass1():
2     def method_Parent1(self):
3         print("Parent1 method called")
4
5 class MyParentClass2():
6     def method_Parent2(self):
7         print("Parent2 method called")
8
9 class ChildClass(MyParentClass1, MyParentClass2):
10     def child_method(self):
11         print("Child method")
12
13 c = ChildClass()
14 c.method_Parent1()
15 c.method_Parent2()
16 c.child_method()

```

Terminal Output:

```

PS F:\Internship2\CODE> python -i "F:\Internship2\CODE\Day5\Five.py"
Parent1 method called
Parent2 method called
Child method
PS F:\Internship2\CODE>

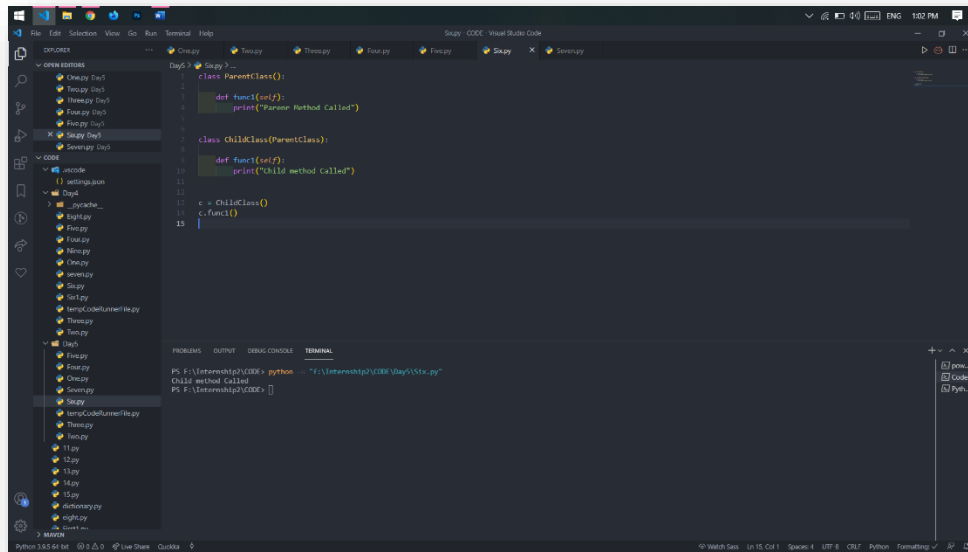
```

**6. Code**

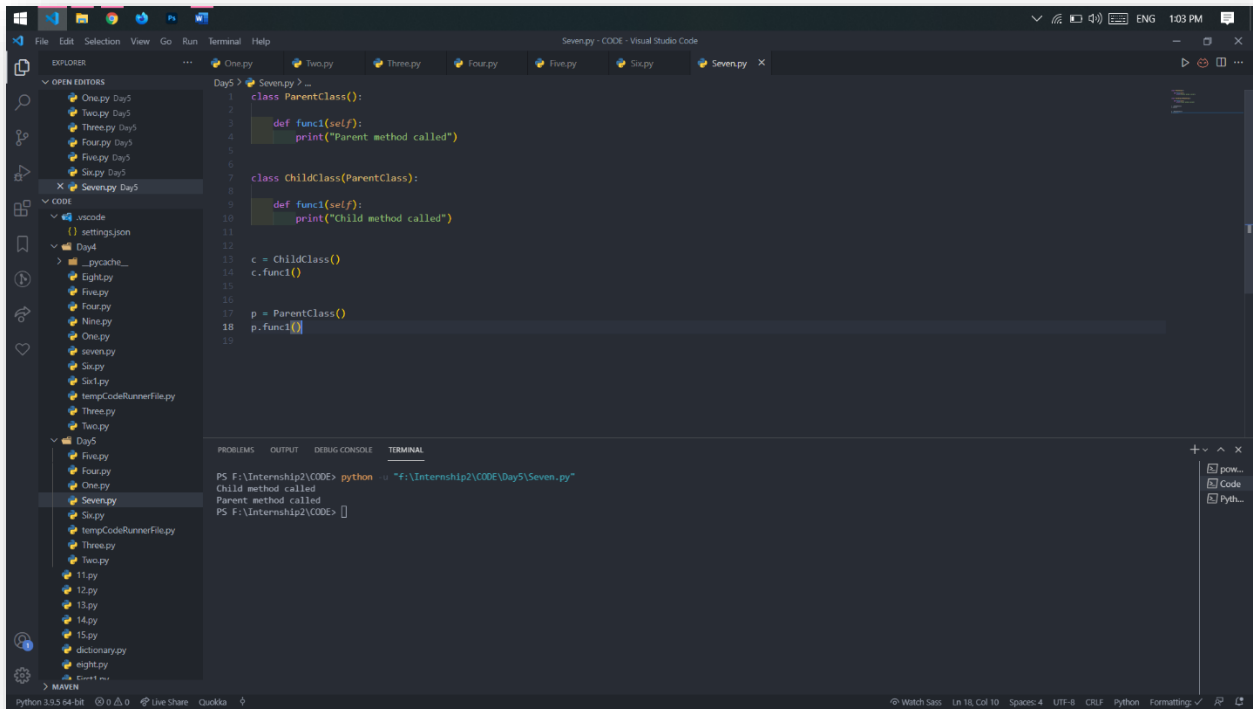
```

1 class ParentClass():
2
3     def func1(self):
4         print("Parent Method Called")
5
6 class ChildClass(ParentClass):
7
8     def func1(self):
9         print("Child method Called")
10
11
12
13 c = ChildClass()
14 c.func1()

```

Output :-7. Code :

```
1 class ParentClass():
2
3     def func1(self):
4         print("Parent method called")
5
6
7 class ChildClass(ParentClass):
8
9     def func1(self):
10        print("Child method called")
11
12
13 c = ChildClass()
14 c.func1()
15
16
17 p = ParentClass()
18 p.func1()
```

Output :-

The screenshot shows the Visual Studio Code editor with a file named `Seven.py` open. The code defines a `ParentClass` and a `ChildClass` that inherits from it. Both classes have a `func1` method that prints a message. The `ChildClass` also inherits the `func1` method from the `ParentClass`. The script creates an instance of `ChildClass` and calls `func1`, then creates an instance of `ParentClass` and calls `func1`.

```
1 class ParentClass():
2     def func1(self):
3         print("Parent method called")
4
5 class ChildClass(ParentClass):
6     def func1(self):
7         print("Child method called")
8
9 c = ChildClass()
10 c.func1()
11
12 p = ParentClass()
13 p.func1()
```

The terminal output shows the execution of the script, displaying the messages "Child method called" and "Parent method called" in sequence.

```
PS F:\Internship2\CODE> python -u "f:\Internship2\CODE\Day5\Seven.py"
Child method called
Parent method called
PS F:\Internship2\CODE>
```