Object Oriented Programming
(Python l)
Lab 5

ASST.LEC Fatima Mohammed & Adnan Habeeb

Python Classes and Objects

Python is an object oriented programming language.
Almost everything in Python is an object, with its properties and methods.
A Class is like an object constructor, or a "blueprint" for creating objects.

Create a Class

To create a class, use the keyword class followed by an indented block with the class contents :

```
class class_name :
        statement 1……
        statement 2……
        function1……
            ……
        function2……
            ……
```

Ex

```
class MyClass:
  y = 10

p1 = MyClass()
print("y=",p1.y)
```

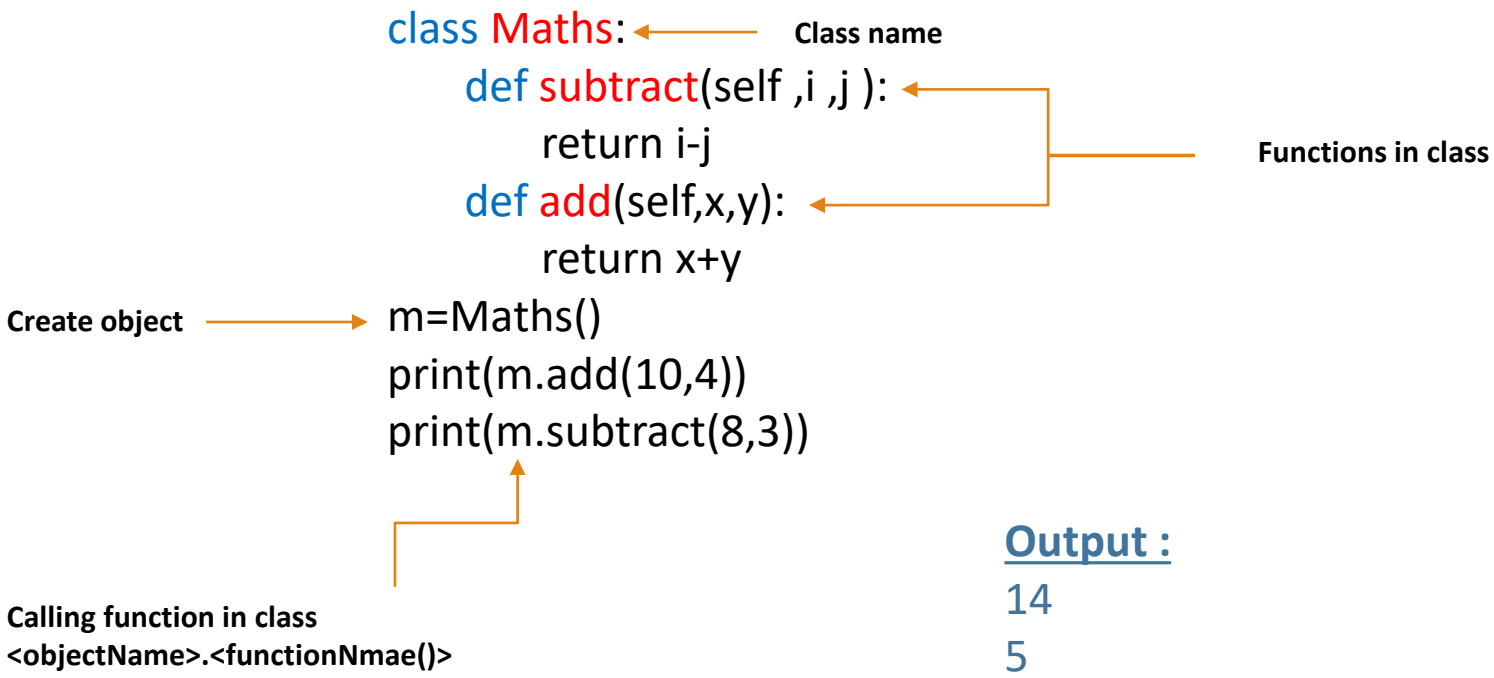
Class name

Object data attributes

Output :

y=10

Python Classes and Objects



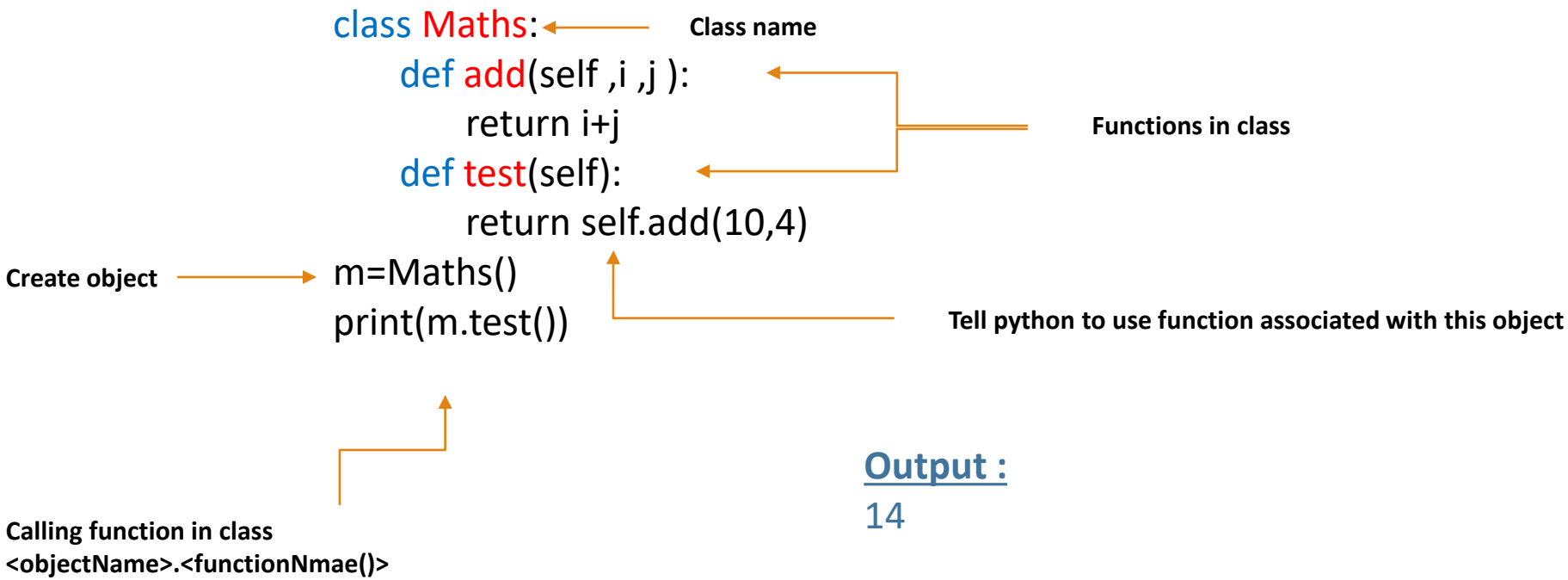
Output :

14

5

Note: Argument self refers to the obeject itself

Python Classes and Objects



Output :

14

Note: Argument self refers to the obeject itself

Python Classes and Objects

The Constructor or __init__() Function

All classes have a function called __init__(), which is always executed when the class is being initiated.
Use the __init__() function to assign values to object properties, or other operations that are necessary to do when the object is being created

```
class Person:
  def __init__(self, name, age):
    self.z = name
    self.y = age

p1 = Person("John", 36)

print(p1.z)
print(p1.y)
```

Use the __init__(self , name , age) function to assign values for name and age

Object data attributes

Create object

Use the __init__(self , name , age) function to assign values for name and age

Output :

John
34

Note: when referring to functions or attributes from within a class, we must always prefix the functions or attributes name with self.

Note: The __init__() function is called automatically every time the class is being used to create a new object.

Python Classes and Objects

The __str__() Function

The __str__() function controls what should be returned when the class object is represented as a string.

WITHOUT the __str__() function :

```
class Person:
  def __init__(self, name, age):
    self.z = name
    self.y = age

p1 = Person("John", 36)

print(p1)
```

Use the __init__(self , name , age) function to assign values for name and age

Object data attributes

Create object

Use the __init__(self , name , age) function to assign values for name and age

Output :

<__main__.Person object at 0x0000029CEEC330A0>

Python Classes and Objects

The __str__() Function

The __str__() function controls what should be returned when the class object is represented as a string.

The string representation of an object WITH the __str__() function:

```
class Person:
  def __init__(self, name, age):
    self.z = name
    self.y = age
  def __str__(self):
    return f"{self.z}({self.y})"

p1 = Person("John", 36)
print(p1)
```

Use the __init__(self , name , age) function to assign values for name and age

Object data attributes

Use the __str__()function to representation of object

Create object

Use the __init__(self , name , age) function to assign values for name and age

Output :

John(36)

Ex

```
class Employee:
   empcount=0
   def __init__(self, name ,salary):
      self.z=name
      self.y=salary
      Employee.empcount+=1
   def displaycount(self):
      print("Total Employee :", Employee.empcount)
   def displayemp(self):
      print("Name: “ , self.z," Salary: $“ , self.y)

emp1=Employee("Ali“ , 750000)
emp2=Employee("Mohammed“ , 800000)

emp1.displayemp()
emp2.displayemp()

print("Total Employee :“ , Employee.empcount)
emp1.displaycount()
emp2.displaycount()
```

Class name

Use the __init__(self , name , salary) function to assign values for name and salary

The variable empcount is a class variable whose value is shared among all instances of a this class

Accessing empcount from inside the class

Creating Instance Objects

Accessing Attributes

Accessing empcount from outside the class

Accessing Attributes

Output :

Name: Ali Salary: $ 750000
Name: Mohammed Salary: $ 800000
Total Employee : 2
Total Employee : 2
Total Employee : 2

Thank you