Programming Fundamentals

Lecture 10 – Introduction to Python Classes and Objects

Iresh Bandara

ROBERT GORDON UNIVERSITY ABERDEEN
TEF Gold
INFORMATICS INSTITUTE OF TECHNOLOGY

Learning Outcomes

- This lecture addresses LO1,LO2 for the module
- On completion of this lecture, students are expected to use
  - Define a class
  - Create and initialize objects
  - Call members and functions
  - Constructors and predefined methods
- However the understanding of OOP concepts and complex class-object scenarios will be expected only in the following semester

Agenda

- Intro to Python classes and object
- Class members and functions
- Calling members and functions
- Constructors
- Exercises

Introduction

- Main idea of this lecture is to introduce you the Python classes and objects
- OOP concept will be introduced in the following lecture
- Try to understand how to use classes and object concept just for the moment.
- It will be important for you in the next semester.

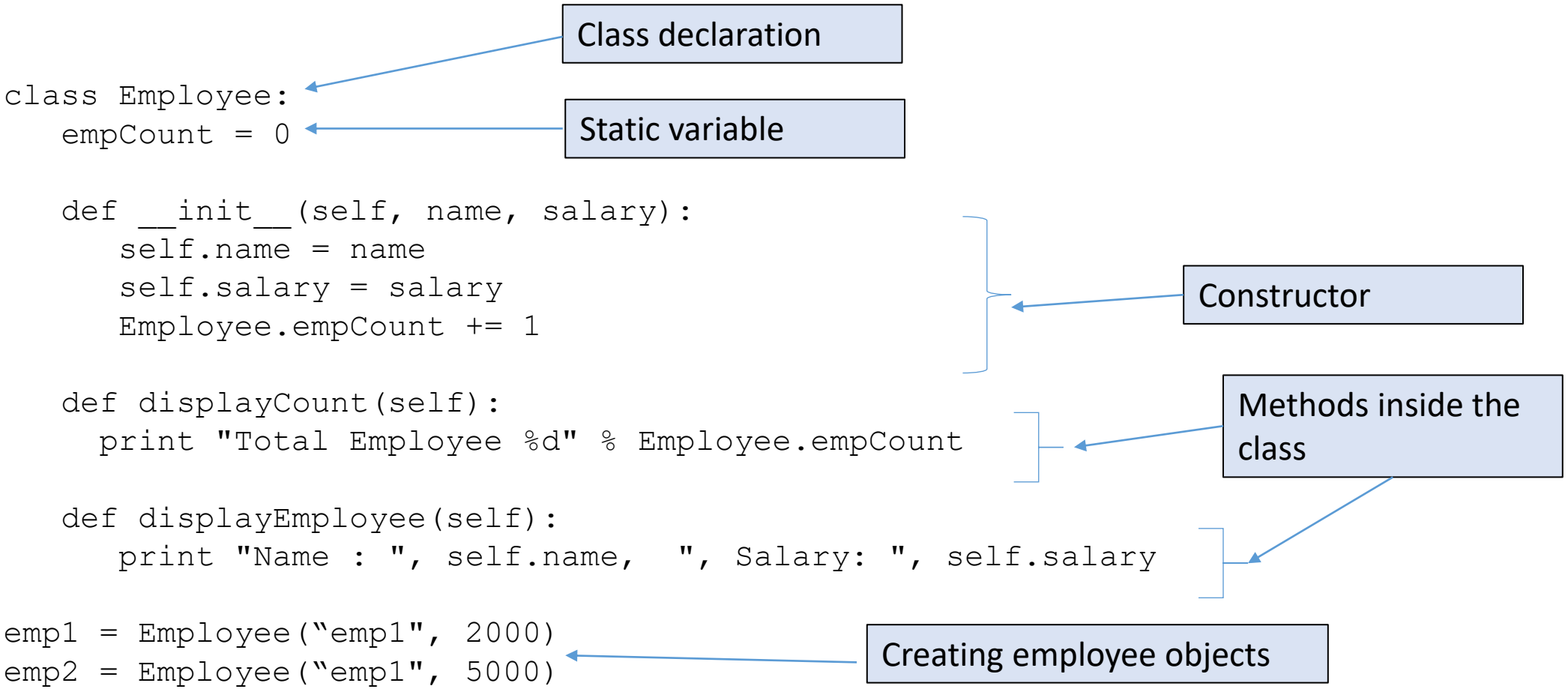
OOP, Defining a Class

- Python was built as a procedural language
  - Python does not consider as a Pure Object Oriented Language
  - Java probably does classes better than Python (gasp)

- Declaring a class:

```
class name:
    statements
```

OOP, Defining a Class



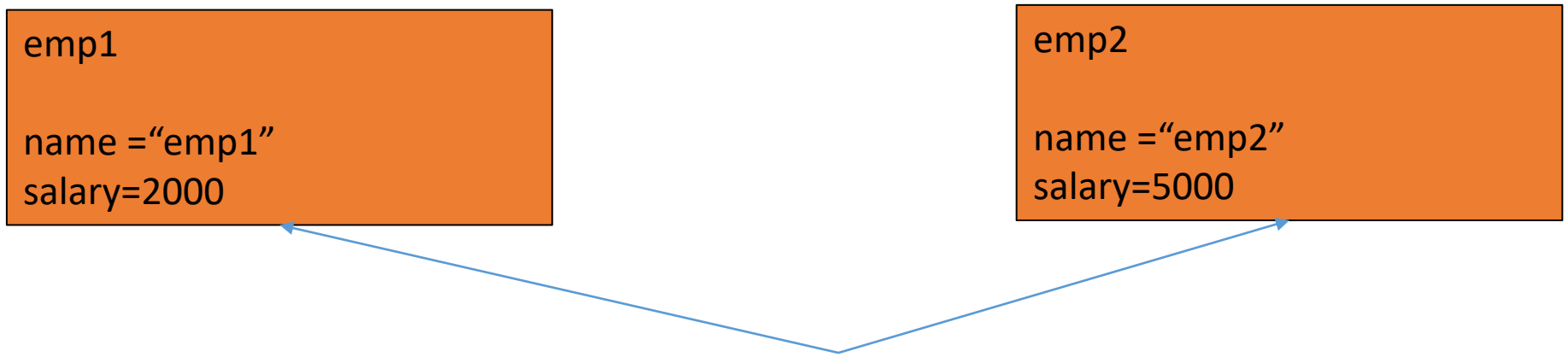

```
class Employee:
   empCount = 0

   def __init__(self, name, salary):
      self.name = name
      self.salary = salary
      Employee.empCount += 1

   def displayCount(self):
     print "Total Employee %d" % Employee.empCount

   def displayEmployee(self):
      print "Name : ", self.name,  ", Salary: ", self.salary

emp1 = Employee("emp1", 2000)
emp2 = Employee("emp1", 5000)
```

Class and Object

- **Class** – A user-defined prototype for an object that defines a set of attributes that characterize any object of the class. The attributes are data **members** (class variables and instance variables) and **methods**, accessed via dot notation.
- **Object** – A unique instance of a data structure that's defined by its class. An object comprises both data members (class variables and instance variables) and methods.

Creating an Instance of a Class

```
#creation of two objects using the above class
emp1 = Employee("emp1", 2000)
emp2 = Employee("emp2", 5000)
```



2 different objects. Note the state of emp1 and emp1 is different

Accessing attributes

```
emp1.displayEmployee()
emp2.displayEmployee()
print "Total Employee %d" % Employee.empCount
```

- Salary and name are two different instance variables inside the Employee class
- empCount is a class/static variable. It belongs to the class. Not for the objects
- Challenge : guess the output?

Calling Methods

- A client can call the methods of an object in two ways:
  - (the value of `self` can be an implicit or explicit parameter)

1) **object . method (parameters)**

or

2) **Class . method (object, parameters)**

- Example:

```
emp = Employee(3, -4)
emp1.displayEmployee() #option1
Employee.displayEmployee(emp1) #option2
```

Putting everything together

```
class Employee:
   empCount = 0
   def __init__(self, name, salary):
      self.name = name
      self.salary = salary
      Employee.empCount += 1

   def displayCount(self):
     print "Total Employee %d" % Employee.empCount

   def displayEmployee(self):
      print "Name : ", self.name,  ", Salary: ", self.salary

emp1 = Employee("emp1", 2000)
emp2 = Employee("emp2", 5000)
emp1.displayEmployee()
emp2.displayEmployee()
print "Total Employee %d" % Employee.empCount
```

Accessing attributes 2

- Add, remove, or modify attributes of classes and objects at any time

```
emp1.age = 7  # Add an 'age' attribute.
emp2.age = 8  # Modify 'age' attribute.
del emp1.age  # Delete 'age' attribut
```

- Inbuilt methods to access attributes.

```
hasattr(emp1, 'age')    # Returns true if 'age' attribute exists
getattr(emp1, 'age')    # Returns value of 'age' attribute
setattr(emp1, 'age', 8) # Set attribute 'age' at 8
delattr(empl, 'age')    # Delete attribute 'age'
```

- You can have custom getters and setters also to set and get member values in Employee class too

Constructors

```
def __init__(self, parameter, ..., parameter):
    statements
```

- a constructor is a special method with the name `__init__`
- This will be called when creating an object
- Example:

```
class Employee:
    def __init__(self, name, salary):
        self.name = name
        self.salary = salary
    ...
```

toString and __str__

```
def __str__(self):
    return string
```

- equivalent to Java's `toString` (converts object to a string)
- invoked automatically when `str` or `print` is called
- Add the following method to Employee class

```
def __str__(self):
    return "(" + str(self.name) + ", " + str(self.salary) + ")"
```

- After adding the method print the object outside the class scope

```
emp1 = Employee("emp1",2000)
print(emp1)
```

Next semester

- This is only an introduction to classes and objects in Python
- OOP concepts will be thoroughly discussed in the next semester
- But if you are interested here are the keywords
  - Objects and classes
  - Object oriented principles
  - Inheritance, Abstraction, Polymorphism, Encapsulation
  - Method overriding
  - Interfaces and abstract classes

Exercise

- Write a Python class named Circle constructed by a radius and two methods which will compute the area and the perimeter of a circle.

- Create a class called **Student** and inside the student you need to store following information
  - Name, age, hometown, more member you want to add
  - School : This should be again an object / not a string

  Hint: You will need an another class called **School** and after you set values for the school object, you can pass it to the Student.
  - Demonstrate both classes are working as expected.

Summary

- Declaring a class

```
class name:
    statements
```

- Static members are defined outside the methods, but inside the class
- __init__ is the constructor. Mainly used to initialize an object
- You can delete, modify and attributes using the object
- Mainly two ways for calling methods
- __str__ is a predefined method and use to print the object