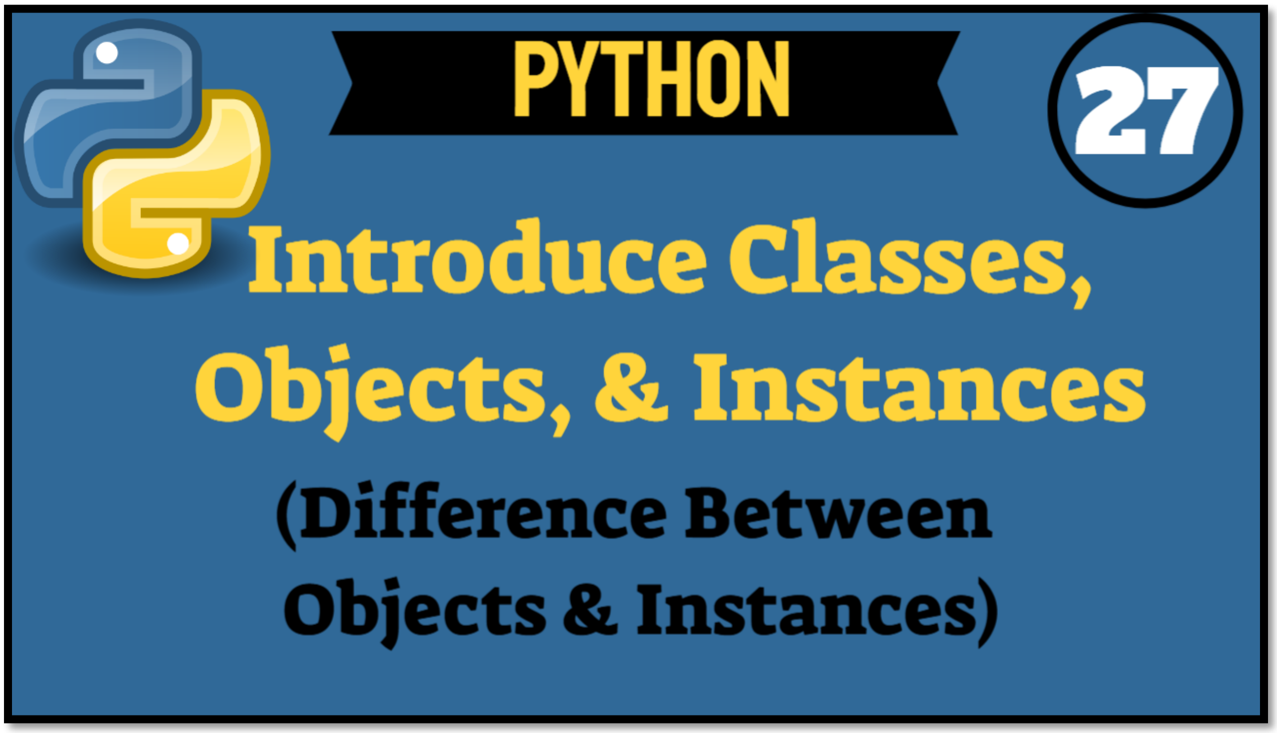
Rex Jones II

Introduce Classes, Objects & Instances


PYTHON
27
Introduce Classes, Objects, & Instances
(Difference Between Objects & Instances)

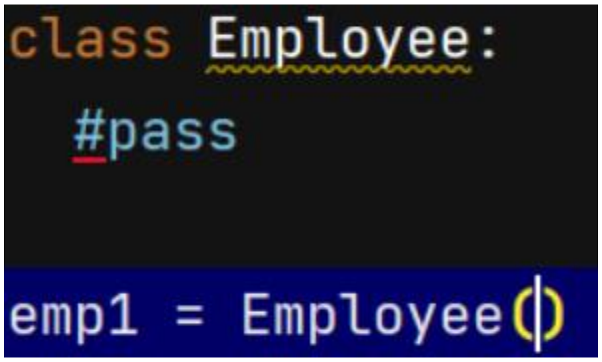
Python Video = https://youtu.be/SKuQ9rZl8Ho

Intro To Classes, Objects, & Instances

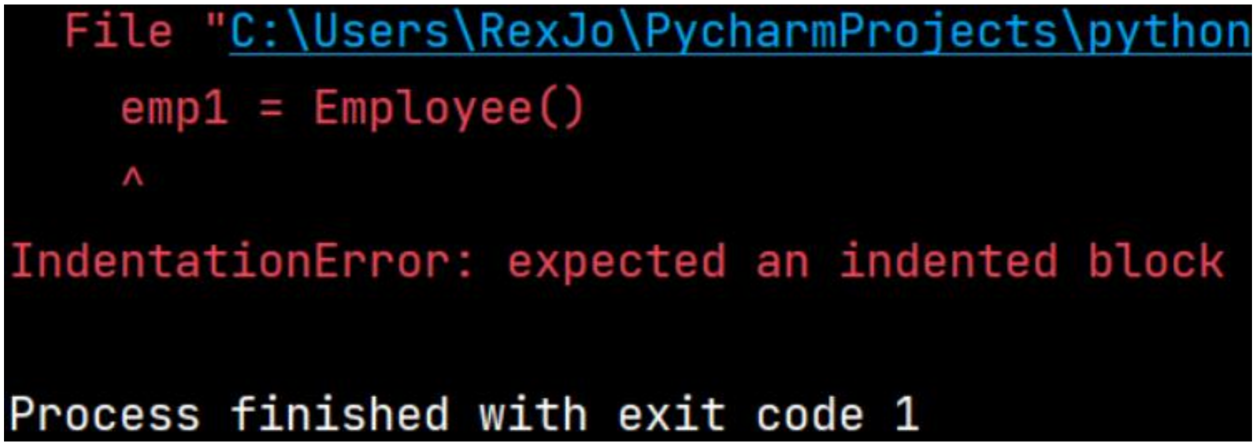
In this session, I will introduce Classes, Objects, and Instances. Classes and Objects help make our program more organized, powerful, and easy to reuse. We create a class to represent people, places, and things. A class represents a condition of the real-world. An object is based on a class and uses the class as a blueprint to create an instance. Therefore, each object is automatically supplied the same attributes and methods of a class. Creating an object from a class is called instantiation and we work with instances of a class.

To create a class, we write the keyword class. So far, in the previous sessions, we have covered some basic data types like int, string, float, and Boolean. Python also offer data structures like a list. However, not all data falls under these basic data types. As a result, Python allows us to create our own data type using a class. For example, a dog or a cat does not have a representation in one of these built-in data types. So to create our own data type. Let's say, we have an employee and to represent an employee. That means our class will be called Employee and I cannot forget the colon: Notice Employee starts with a capital letter. By convention, the name of a Python class begins with an uppercase letter.

If you had a class that you wanted to leave empty then we write pass. The purpose of pass is let Python know you want this class to be a placeholder for future code. It helps us to avoid getting an error. For example, if I comment #pass then write emp1 = Employee() and run.

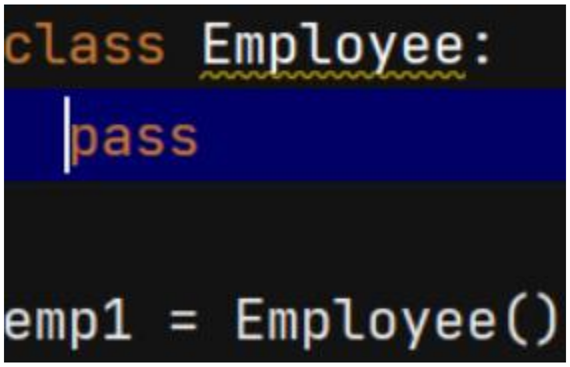


```
class Employee:
  #pass

emp1 = Employee()
```



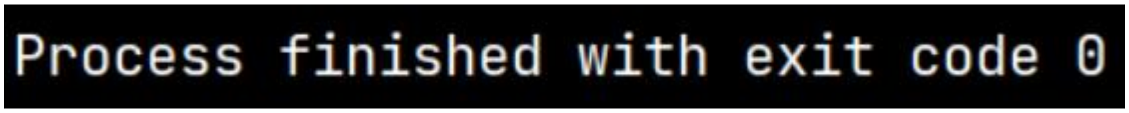
```
  File "C:\Users\RexJo\PycharmProjects\python
    emp1 = Employee()
    ^
IndentationError: expected an indented block

Process finished with exit code 1
```

We see an error. If I remove the comment then run again.



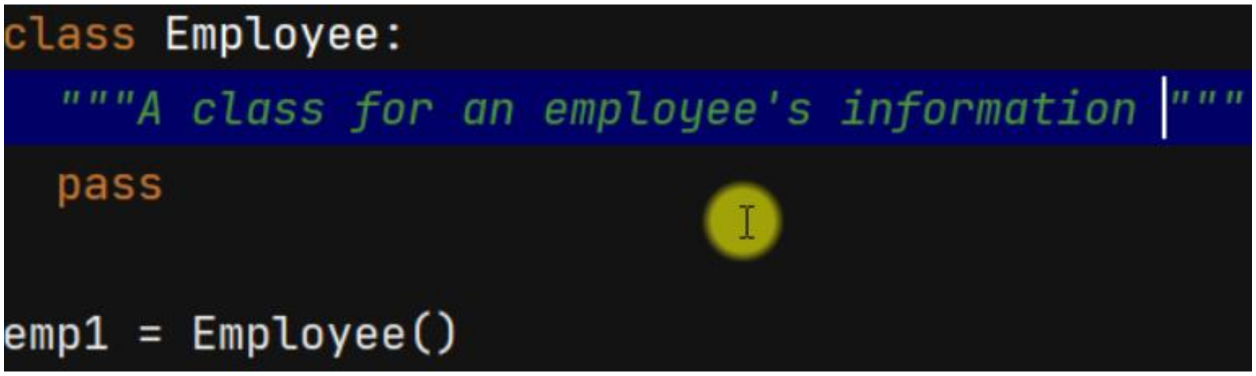
```
class Employee:
  pass

emp1 = Employee()
```

Now we do not see an error. Exit Code shows 0.



```
Process finished with exit code 0
```

We see a line under Employee. There's a line under Employee because I have not added a docstring. It's missing. The docstring is not required but it describes the class, I will add “”” A class for an employee's information ”””.

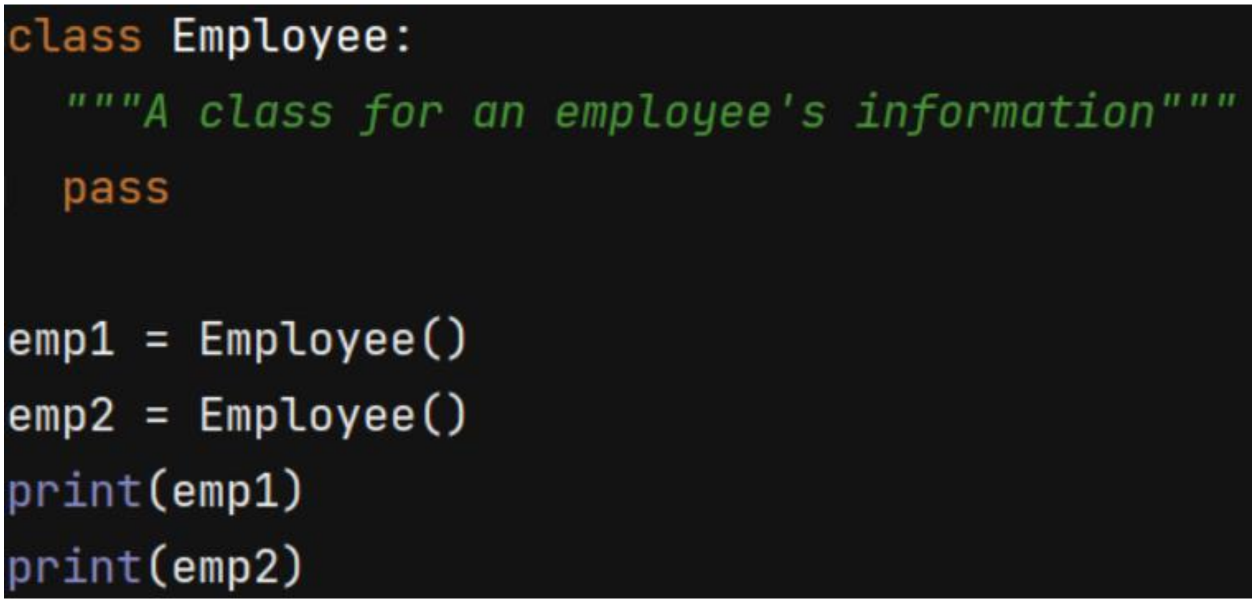


```
class Employee:
  """A class for an employee's information """
  pass


emp1 = Employee()
```

Sometimes, there's confusion around objects, classes, and instances. The Employee class is a template for the Employee() object. The Employee() object is assigned to the instance employee1. For example, name = 'Joe Doe'. Joe Doe is assigned to name. It's the same with objects. employee1 is not the object but refers to the object. employee1 is an instant variable and created when we instantiate the Employee() object.

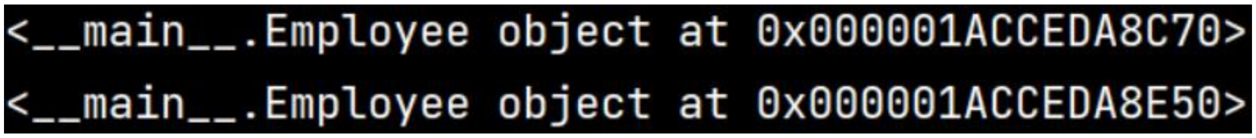
That's why we say employee 1 is an instance of the Employee class. You know how a company can have more than 1 employee. We can also create more than 1 instance of an employee by writing emp2 = Employee().

Now, we have 2 instances that refer to the Employee() object. At runtime, the Employee() object allocates memory for employee1 and allocate employee2. Let me show you when I print(emp1) and print(emp2).



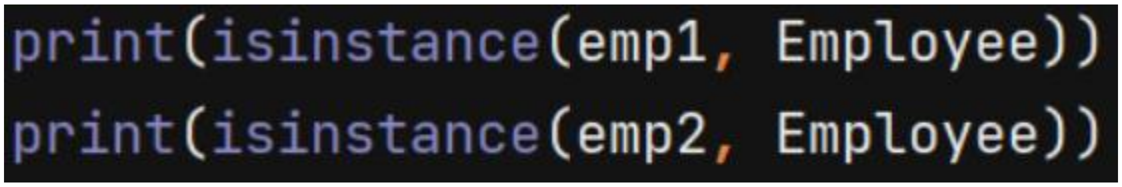
```
class Employee:
  """A class for an employee's information"""
  pass


emp1 = Employee()
emp2 = Employee()
print(emp1)
print(emp2)
```

Run and the console shows different memory locations.



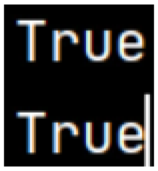
```
<__main__.Employee object at 0x000001ACCEDA8C70>
<__main__.Employee object at 0x000001ACCEDA8E50>
```

It shows different memory locations because the Employee object has memory for 2 instances. That means the Employee() object will return a reference to employee1 and also return a reference to employee2. To verify they are instances of the Employee class, we can write isInstance() then pass in 2 arguments: the Instance emp1, and the class Employee. Also, print(). I'm going to copy and paste then do the same for employee2. Now, when I run.



```
print(isinstance(emp1, Employee))
print(isinstance(emp2, Employee))
```

We see True 2 times.



```
True
True
```

True means they are instances of the Employee class. That's the difference between a class, object, and instance. To recap, the class is a blueprint for the object. Therefore, an object has a copy of all information from the class. That information is called attributes and methods. We only access the attributes and methods when creating an instance. In this example, the Employee() object encapsulates all of the information from the Employee class then allow employee1 and employee2 to access that information. Next, I will create information and show you how to access that information.

Contact Info

- ✓ Email Rex.Jones@Test4Success.org
- ✓ YouTube https://www.youtube.com/c/RexJonesII/videos
- ✓ Facebook https://facebook.com/JonesRexII
- ✓ Twitter https://twitter.com/RexJonesII
- ✓ GitHub https://github.com/RexJonesII/Free-Videos
- ✓ LinkedIn https://www.linkedin.com/in/rexjones34/