Python Class

Summary: in this tutorial, you'll learn about Python classes and objects and how to define a new class.

Objects

An object is a container that contains **data** and **functionality**.

The **data** represents the object at a particular moment in time. Therefore, the data of an object is called the **state**. Python uses **attributes** to model the state of an object.

The functionality represents the **behaviors** of an object. Python uses functions to model the behaviors. When a function is associated with an object, it becomes a **method** of the object.

In other words, an object is a container that contains the **state** and **methods**.

Before creating objects, you define a class first. And from the class, you can create one or more objects. The objects of a class are also called **instances** of a class.

Define a class

To define a class in Python, you use the `class` keyword followed by the class name and a colon. The following example defines a `Person` class:

```
class Person:
    pass
```

By convention, you use capitalized names for classes in Python. If the class name contains multiple words, you use the `CamelCase` format, for example `SalesEmployee` .

Since the `Person` class is incomplete; you need to use the `pass` statement to indicate that you'll add more code to it later.

To create an instance of a class, you use the class name with parentheses like this:

```
person = Person()
```

When printing out the `person` object, you'll see its name and memory address:

```
class Person:
    pass


print(person)
```

Output:

```
<__main__.Person object at 0x000001C46D1C47F0>
```

To get an identity of an object, you use the id() function. For example:

```
print(id(person))
```

Output:

```
1943155787760
```

The id of an object is unique. In CPython, the id() returns the memory address of an object. The hex() function converts the integer returned by the id() function to a lowercase hexadecimal string prefixed with 0x:

```
print(hex(id(person)))
```

Output:

```
0x1c46d1c47f0
```

The person object is an instance of the `Person` class. The `isinstance()` function returns `True` if an object is an instance of a class:

```
print(isinstance(person, Person))  # True
```

A class is also an object in Python

Everything in Python is an object, including classes.

When you define the `Person` class, Python creates an object with the name `Person`. The `Person` object has attributes. For example, you can find its name using the `__name__` attribute:

```
print(Person.__name__)
```

Output:

```
Person
```

The `Person` object has the type of `type`:

```
print(type(Person))
```

Output:

```
<class 'type'>
```

The `Person` class also has a behavior. For example, it can create a new instance:

```
person = Person()
```

Summary

- An object is container that contains state and behavior.
- A class is a blueprint for creating objects.
- In Python, a class is also an object, which is an instance of the `type`.