Python __init__

Summary: in this tutorial, you'll learn how to use the Python `__init__()` method to initialize object's attributes.

Introduction to the Python __init__() method

When you create a new object of a class (https://www.pythontutorial.net/python-oop/python-class/) , Python automatically calls the `__init__()` method to initialize the object's attributes (https://www.pythontutorial.net/python-oop/python-instance-variables/) .

Unlike regular methods (https://www.pythontutorial.net/python-oop/python-methods/) , the `__init__()` method has two underscores (__) on each side. Therefore, the `__init__()` is often called dunder init. The name comes abbreviation of the double underscores init.

The double underscores at both sides of the `__init__()` method indicate that Python will use the method internally. In other words, you should not explicitly call this method.

Since Python will automatically call the `__init__()` method immediately after creating a new object, you can use the `__init__()` method to initialize the object's attributes.

The following defines the `Person` class with the `__init__()` method:

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


if __name__ == '__main__':
    person = Person('John', 25)
    print(f"I'm {person.name}. I'm {person.age} years old.")
```

When you create an instance of the `Person` class, Python performs two things:

- First, create a new instance of the `Person` class by setting the object's namespace such as `__dict__` attribute to empty (`{}`).
- Second, call the `__init__` method to initialize the attributes of the newly created object.

Note that the `__init__` method doesn't create the object but only initializes the object's attributes. Hence, the `__init__()` is not a constructor.

If the `__init__` has parameters other than the `self`, you need to pass the corresponding arguments when creating a new object like the example above. Otherwise, you'll get an error.

The __init__ method with default parameters

The `__init__()` method's parameters can have default values. For example:

```
class Person:
    def __init__(self, name, age=22):
        self.name = name
        self.age = age


if __name__ == '__main__':
```

```
person = Person('John')
print(f"I'm {person.name}. I'm {person.age} years old.")
```

Output:

```
I'm John. I'm 22 years old.
```

In this example, the `age` parameter has a default value of `22`. Because we don't pass an argument to the `Person()`, the `age` uses the default value.

Summary

- Use the `__init__()` method to initialize the object's attributes.
- The `__init__()` doesn't create an object but is automatically called after the object is created.