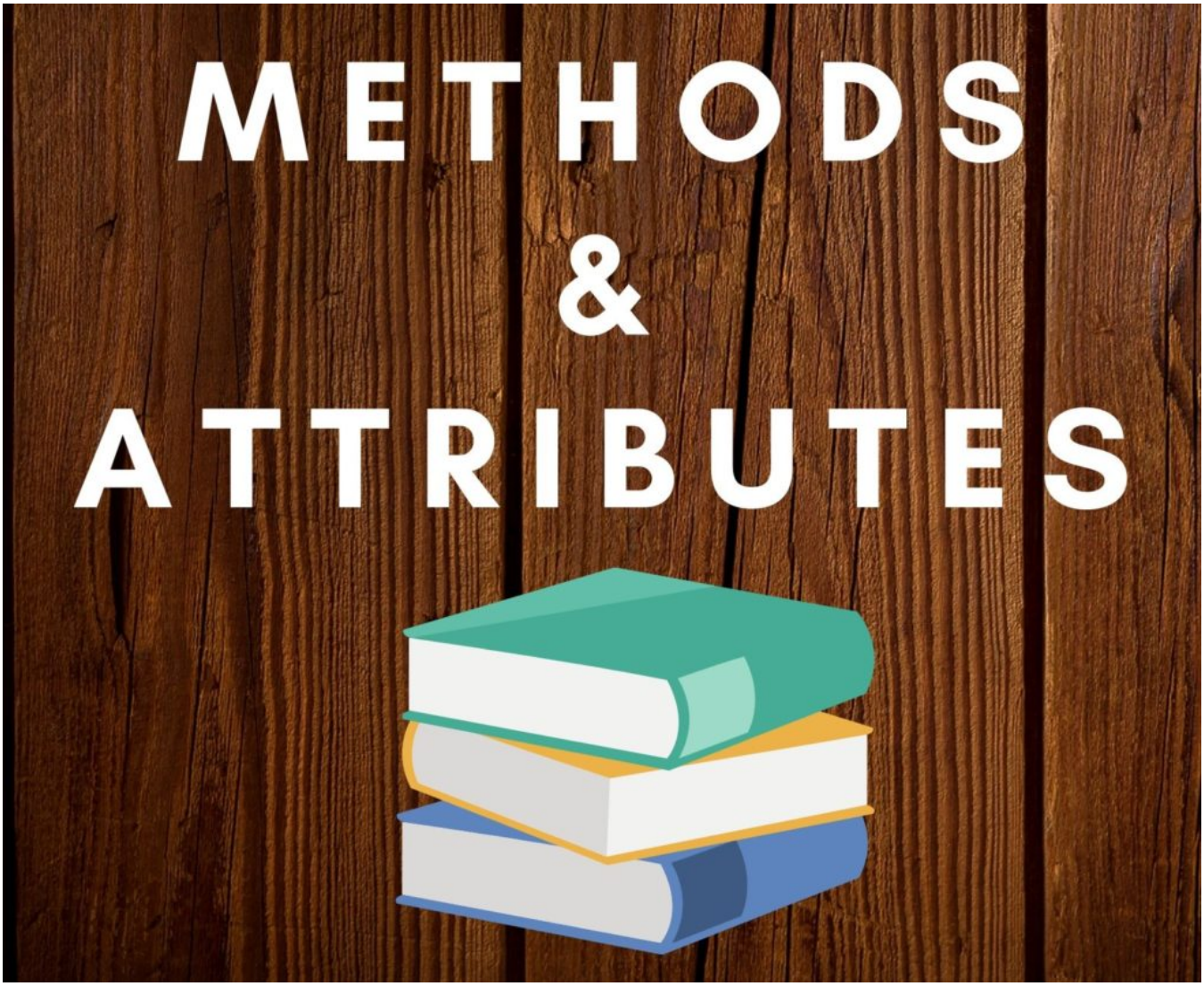
Method and Attributes in Python-Complete Tutorial

Python / March 23, 2020



This **Method and Attributes in Python-Complete Tutorial** is made for you to give complete knowledge of the **Method and Attributes in Python**. At the end of this tutorial, you will have full knowledge of the **Method and Attributes in Python**.

Hello, & Welcome! to the **Method and Attributes in Python-Complete Tutorial**.

Part 14- Method and Attributes in Python-Complete Tutorial

In this article of Python Tutorial, you will learn following-

1. Instance Methods and Attributes in Python.
2. Class level Methods and Class level Attributes in Python.
3. Difference between Instance Attributes and Class level Attributes.
4. Difference between Instance Methods and Class level Methods.
5. Static Method.

Instance Methods and Attributes in Python.

The methods and attributes which we create in the class are known as the Instance method and Instance attributes.

Instance methods and Instance attributes are used only with the instance of a class or object of a class.

Let's see in the example what is instance method and attributes in python-

```
Class Car:
   def change_name(self,new_name):
       self.name=new_name
```

Here "**change_name**" is an instance method and "**new_name**" is an instance attribute.

Class level Methods and Class level Attributes in Python.

Class level Attributes-

When we create any attribute inside the class, not inside any function of a class, it is known as class-level attributes.

Confused?...

Don't worry! I will explain to you with the help of an example-

```
class Planet
    shape="round"
```

Here, the "shape" attribute is a class-level attribute, because it is created inside the class not under any function of a class.

In the previous example "new_name" is not a class-level attribute because it is inside the function of a class.

I hope now you understand class level attributes 🙂

Class level Methods-

Class level Methods are different from normal methods. In order to create class level method, you need to write **decorator @classmethod** and inside the method, you need to write **"cls"** unlike in normal method in python, we write self.

Let's understand with the help of an example-

```
class Planet
    shape="round"
    @classmethod     #decorator
    def common(cls)
        //method body
```

Here, "common" is a class-level method, and inside a class-level method, you need to pass "cls".

Difference between Instance Attributes and Class level Attributes.

Class Level attributes are common to the class, and it is accessible by class name as well as with instance or objects.

But,

Instance attributes are not accessible with a class name.

Let's understand with the help of an example-

```
Class Planet
    shape="round"
print(Planet.shape)     #Planet is a class name

OUTPUT-
round
```

Here, we call the class-level attribute "shape" in a print statement with a class name. It didn't give any error, because class-level attributes are accessible with a class name.

Similarly, a class-level attribute is accessible with instances or objects of a class.

Let's have a look at the example-

```
Class Planet
    shape="round"
naboo=Planet()
print(naboo.shape)     #naboo is object of a class

OUTPUT-
round
```

Here, "naboo" is an object of a class and we are accessing class-level attributes with the object.

But, if you want to access any **instance attribute** with class name, it will give an error.

Let's see in the example-

```
Class Car:
   def change_name(self,new_name):
       self.name=new_name
print(Car.new_name)       #Car is a class name

OUTPUT-
Error
```

Here, "new_name" is an instance attribute and we are calling it with a class name, which we can't. That's why an error is coming.

In Short- The main difference between class-level attributes and instance attribute is that you can call a class-level attribute with a class name but you can't call instance attributes with a class name.

Difference between Instance Methods and Class level Methods.

As you learn in the difference of class-level attributes and instance attributes, the same with methods.

Instance methods are the methods that are created for instance of a class. It shows individuals property and it is accessible only with an instance of a class. which means you can call Instance methods only with object of a class.

But,

Class level methods are common for all and it is accessible with both a class name and object of a class.

Let's see how you can access class-level methods-

```
class Planet
    shape="round"
    @classmethod     #decorator
    def common(cls)
       return f'All planets are {cls.shape}'
```

So, here in **class level method**, you don't need to pass "self", you can use "cls". With the help of "cls", you can call class attributes like- **cls.shape**.

You can access this class-level method with the class name as well as to object of the class.

Let's see in the example-

```
class Planet
    shape="round"
    @classmethod     #decorator
    def common(cls)
        return f'All planets are {cls.shape}'
print(Planet.common())

OUTPUT-
All planets are round.
```

Here, we are calling the **class-level method** "common" with a class name **Planet**.

You can also call class-level methods with the object of the class. Let's see in the example-

```
class Planet
    shape="round"
    @classmethod     #decorator
    def common(cls)
        return f'All planets are {cls.shape}'
naboo=Planet()
```

```
print(naboo.common())    #naboo is a object of a class

OUTPUT-
All planets are round.
```

Here, naboo is an object of a class and we are calling a class-level method with an object.

But, you can't access the instance method with the class name.

Static Method.

The static method has access to only its own attributes. It doesn't have access to class-level attributes or methods not to instance attributes or methods. Which means Static methods can't use **Class-level(methods & attributes) and Instance(methods & attributes.**

A static method can only use its own attributes or variable.

Let's see how to create a static method-

```
@staticmethod
def spin(speed='2000 mileperhour')
return f'Planet spins at {speed}'
```

Here, "speed" is a static method attribute, therefore static method can only use this attribute.

NOTE- The Static method is accessible with a class name as well as with instance or object. This means you can call a static method with the class name and with the object name too.

Static method neither takes "self" nor "cls".

For example-

```
class Planet
    shape="round"
```

```
    @classmethod     #decorator
    def common(cls)
        return f'All planets are {cls.shape}'
    @staticmethod
    def spin(speed='2000 mileperhour')
        return f'Planet spins at {speed}'
print(Planet.spin())    #Planet is a class name

OUTPUT-
Planet spins at 2000 mileperhour
```

Here, we are calling a static method “spin” with a class name. Similarly, you can call a static method with object of a class.

Let’s see-

```
class Planet
    shape="round"
    @classmethod     #decorator
    def common(cls)
        return f'All planets are {cls.shape}'
    @staticmethod
    def spin(speed='2000 mileperhour')
        return f'Planet spins at {speed}'
naboo=Planet()
print(naboo.spin())    #naboo is a object of a class

OUTPUT-
Planet spins at 2000 mileperhour
```

Here, we are calling a static method with the object of a class “naboo”.

That’s all for the **Method and Attributes in Python**. I hope now you have a better understanding of the **Method and Attributes in Python.**

Congratulations! You successfully learned the **Method and Attributes** in Python.

In the next tutorial, we will start learning **Modules and Packages in Python**.

Till then, Enjoy Learning Python!

All the Best!

Are you ML Beginner and confused, from where to start ML, then read my BLOG – How do I learn Machine Learning?

If you are looking for Machine Learning Algorithms, then read my Blog – Top 5 Machine Learning Algorithm.

If you are wondering about Machine Learning, read this Blog- What is Machine Learning?

Thank YOU!

Though of the Day...

> *"Live as if you were to die tomorrow. Learn as if you were to live forever."*
>
> *– Mahatma Gandhi*

← Previous Post Next Post →

1 thought on “Method and Attributes in Python-Complete Tutorial”

Pingback: Classes in Python-Complete Tutorial for Everyone(2020)

Leave a Comment

Your email address will not be published. Required fields are marked *

Type here..

Name*

Email*

Website

Save my name, email, and website in this browser for the next time I comment.

Post Comment »

Follow Us on Twitter-

Follow us on Pinterest-

Our Facebook Group-