



[Home](#) » [Python](#) » [Python Object-Oriented Programming \(OOP\)](#) » Python Static Method

Explained With Examples

# Python Static Method Explained With Examples

Updated on: October 21, 2021 | + [2 Comments](#)

In [Object-oriented programming](#), at the class level, we use class methods and static methods.

- **[Class methods](#)**: Used to access or modify the state of the class. if we use only [class variables](#), we should declare such methods as a class method.
- **Static methods**: A static method is a general utility method that performs a task in isolation. Inside this method, we don't use instance or class variable because this static method doesn't take any parameters like `self` and `cls`.

**Also, read** [Python Class method vs Static method vs Instance method](#).

**After reading this article, you'll learn:**

---

- 
- How to create and use the static methods in Python
  - Create staticmethod using the `@staticmethod` decorator and `staticmethod()` function

## Table of contents

- [What is Static Methods in Python](#)
- [Define Static Method in Python](#)
  - [Example: Create Static Method Using @staticmethod Decorator](#)
- [Advantages of a Static Method](#)
- [The staticmethod\(\) function](#)
- [Call Static Method from Another Method](#)

# What is Static Methods in Python

**A static method is a general utility method that performs a task in isolation.**  
Static methods in Python are similar to those found in Java or C++.

A static method is bound to the [class](#) and not the object of the class. Therefore, we can call it using the class name.

A static method doesn't have access to the class and [instance variables](#) because it does not receive an implicit first argument like `self` and `cls`. Therefore **it cannot modify the state of the object or class.**

The class method can be called using `ClassName.method_name()` as well as by using an object of the class.

```
class Employee:
    @staticmethod
    def sample(x):
        print('Inside static method', x)

# call static method
```

```
Employee.sample(10)
```

```
# can be called using object
```

```
emp = Employee()
```

```
emp.sample(10)
```



Run

## Define Static Method in Python

Any method we create in a class will automatically be created as an [instance method](#). We must explicitly tell Python that it is a static method using the `@staticmethod` **decorator** or `staticmethod()` **function**.

Static methods are defined inside a class, and it is pretty similar to defining a regular [function](#). To declare a static method, use this idiom:

```
class C:
    @staticmethod
    def f(arg1, arg2, ...): ...
```

**Example: Create Static Method Using `@staticmethod` Decorator**

To make a method a static method, add `@staticmethod` decorator before the method definition.

The `@staticmethod` decorator is a built-in function decorator in Python to declare a method as a static method. It is an expression that gets evaluated after our function is defined.

In this example, we will create a static method `gather_requirement()` that accepts

the project name and returns all requirements to complete under this project.

Static methods are a special case of methods. Sometimes, you'll write code that belongs to a class, but that doesn't use the object itself at all. It is a utility method and doesn't need an object ( `self` parameter) to complete its operation. So we declare it as a static method. Also, we can call it from another method of a class.

```
class Employee(object):

    def __init__(self, name, salary, project_name):
        self.name = name
        self.salary = salary
        self.project_name = project_name

    @staticmethod
    def gather_requirement(project_name):
        if project_name == 'ABC Project':
            requirement = ['task_1', 'task_2', 'task_3']
        else:
            requirement = ['task_1']
```



```
        return requirement

# instance method
def work(self):
    # call static method from instance method
    requirement = self.gather_requirement(self.project_name)
    for task in requirement:
        print('Completed', task)

emp = Employee('Kelly', 12000, 'ABC Project')
emp.work()
```



Run

## Output:

```
Completed task_1
Completed task_2
Completed task_3
```

# Advantages of a Static Method

Here, the static method has the following advantages

- **Consume Less memory:** Instance methods are object too, and creating them has a cost. Having a static method avoids that. Let's assume you have ten employee objects and if you create `gather_requirement()` as a [instance method](#) then Python have to create a ten copies of this method (seperate for each object) which will consume more memeory. On the other hand static method has only one copy per class.

```
kelly = Employee('Kelly', 12000, 'ABC Project')
jessa = Employee('Jessa', 7000, 'XYZ Project')

# false
# because seperate copy of instance method is created for each object
print(kelly.work is jessa.work)

# True
# because only one copy is created
# kelly and jess objects share the same methods
print(kelly.gather_requirement is jessa.gather_requirement)

# True
print(kelly.gather_requirement is Employee.gather_requirement)
```



- **To Write Utility functions:** Static methods have limited use because they don't have access to the attributes of an object ([instance variables](#)) and class attributes ([class variables](#)). However, they can be helpful in utility such as conversion from one type to another. The parameters provided are enough to operate.
- **Readability:** Seeing the `@staticmethod` at the top of the method, we know that the method does not depend on the object's state or the class state.

## The `staticmethod()` function

Some code might use the old method of defining a static method, using `staticmethod()` as a function rather than a decorator.

You should only use `staticmethod()` function to define static method if you have to

support older versions of Python (2.2 and 2.3). Otherwise, it is recommended to use the `@staticmethod` decorator.

### Syntax:

```
staticmethod(function)
```

- `function`: It is the name of the method you want to convert as a static method.
- It returns the converted static method.

### Example:

```
class Employee:
    def sample(x):
        print('Inside static method', x)

# convert to static method
Employee.sample = staticmethod(Employee.sample)
# call static method
Employee.sample(10)
```



The `staticmethod()` approach is helpful when you need a reference to a function from a class body and you want to avoid the automatic transformation to the instance method.

# Call Static Method from Another Method

Let's see how to call a static method from another static method of the same class. Here we will class a static method from a class method.

```
class Test :
    @staticmethod
    def static_method_1():
        print('static method 1')

    @staticmethod
    def static_method_2() :
        Test.static_method_1()

    @classmethod
    def class_method_1(cls) :
        cls.static_method_2()
```

```
# call class method  
Test.class_method_1()
```



 Run

### Output:

```
static method 1
```

**Reference:** [Static method documentation](#)



Filed Under: [Python](#), [Python Object-Oriented Programming \(OOP\)](#)

**Did you find this page helpful?** Let others know about it. **Sharing helps me** continue to create free Python resources.

[Tweet](#)[F share](#)[in share](#)[P Pin](#)

## About Vishal



Founder of [PYnative.com](#) I am a Python developer and I love to write articles to help developers. Follow me on [Twitter](#). All the best for your future Python endeavors!

## Related Tutorial Topics:

[Python](#) [Python Object-Oriented Programming \(OOP\)](#)

## Python Exercises and Quizzes

Free coding exercises and quizzes cover Python basics, data structure, data analytics, and more.

**15+ Topic-specific Exercises and Quizzes**

Each Exercise contains 10 questions

Each Quiz contains 12-15 MCQ

[Exercises](#)[Quizzes](#)

## Posted In

[Python](#) [Python Object-Oriented Programming \(OOP\)](#)

[Tweet](#)

[F share](#)

[in share](#)

[P Pin](#)

## Python OOP

- ➔ [Python OOP](#)
- ➔ [Classes and Objects in Python](#)
- ➔ [Constructors in Python](#)

- ➡ Python Destructors
- ➡ Encapsulation in Python
- ➡ Polymorphism in Python
- ➡ Inheritance in Python
- ➡ Python Instance Variables
- ➡ Python Instance Methods
- ➡ Python Class Variables
- ➡ Python Class Method
- ➡ Python Static Method
- ➡ Python Class Method vs. Static Method vs. Instance Method
- 🔗 Python OOP exercise

## All Python Topics

[Python Basics](#) [Python Exercises](#) [Python Quizzes](#) [Python File Handling](#) [Python OOP](#)  
[Python Date and Time](#) [Python Random](#) [Python Regex](#) [Python Pandas](#)  
[Python Databases](#) [Python MySQL](#) [Python PostgreSQL](#) [Python SQLite](#) [Python JSON](#)

## About PYnative

**PYnative.com** is for Python lovers. Here, You can get Tutorials, Exercises, and Quizzes to practice and **improve your Python skills**.

## Explore Python

- Learn Python
- Python Basics
- Python Databases
- Python Exercises
- Python Quizzes
- Online Python Code Editor
- Python Tricks

## Follow Us

## Legal Stuff

To get New Python Tutorials, Exercises, and Quizzes

- [Twitter](#)
- [Facebook](#)
- [Sitemap](#)

- [About Us](#)
- [Contact Us](#)

We use cookies to improve your experience. While using PYNative, you agree to have read and accepted our [Terms Of Use](#), [Cookie Policy](#), and [Privacy Policy](#).

Copyright © 2018-2021 · [pynative.com]

---

AN ELITE CAFEMEDIA TECH PUBLISHER