Python's @classmethod and @staticmethod Explained

*By Scott Robinson (https://twitter.com/ScottWRobinson) •
6 Comments (/pythons-classmethod-and-staticmethod-explained/#disqus_thread)*

Python is a unique language in that it is fairly easy to learn, given its straight-forward syntax, yet still extremely powerful. There are a lot more features under the hood than you might realize. While I could be referring to quite a few different things with this statement, in this case I'm talking about the decorators (https://wiki.python.org/moin/PythonDecorators) `@classmethod` and `@staticmethod`. For many of your projects, you probably didn't need or encounter these features, but you may find that they come in handy quite a bit more than you'd expect. It's not as obvious how to create Python static methods, which is where these two decorators come in.

In this article I'll be telling you what each of these decorators do, their differences, and some examples of each.

The @classmethod Decorator

This decorator exists so you can create class methods that are passed the actual class object within the function call, much like `self` is passed to any other ordinary instance method in a class.

In those instance methods, the `self` argument is the class instance object itself, which can then be used to act on instance data. `@classmethod` methods also have a mandatory first argument, but this argument isn't a class instance, it's actually the uninstantiated class itself. So, while a typical class method might look like this:

```
class Student(object):

    def __init__(self, first_name, last_name):
        self.first_name = first_name
        self.last_name = last_name

scott = Student('Scott',  'Robinson')
```

A similar `@classmethod` method would be used like this instead:

```
class Student(object):

    @classmethod
    def from_string(cls, name_str):
        first_name, last_name = map(str, name_str.split(' '))
        student = cls(first_name, last_name)
        return student

scott = Student.from_string('Scott Robinson')
```

This follows the static factory pattern (https://stackoverflow.com/a/929273/2684304) very well, encapsulating the parsing logic inside of the method itself.

The above example is a very simple one, but you can imagine more complicated examples that make this more attractive. Imagine if a `Student` object could be serialized in to many different formats. You could use this same strategy to parse them all:

```
class Student(object):

    @classmethod
    def from_string(cls, name_str):
        first_name, last_name = map(str, name_str.split(' '))
        student = cls(first_name, last_name)
        return student

    @classmethod
    def from_json(cls, json_obj):
        # parse json...
        return student

    @classmethod
    def from_pickle(cls, pickle_file):
        # load pickle file...
        return student
```

The decorator becomes even more useful when you realize its usefulness in sub-classes. Since the class object is given to you within the method, you can still use the same `@classmethod` for sub-classes as well.

The @staticmethod Decorator

The `@staticmethod` decorator is similar to `@classmethod` in that it can be called from an uninstantiated class object, although in this case there is no `cls` parameter passed to its method. So an example might look like this:

```
class Student(object):

    @staticmethod
    def is_full_name(name_str):
        names = name_str.split(' ')
        return len(names) > 1

Student.is_full_name('Scott Robinson')   # True
Student.is_full_name('Scott')            # False
```

Since no `self` object is passed either, that means we also don't have access to any instance data, and thus this method can not be called on an instantiated object either.

These types of methods aren't typically meant to create/instantiate objects, but they may contain some type of logic pertaining to the class itself, like a helper or utility method.

@classmethod vs @staticmethod

The most obvious thing between these decorators is their ability to create static methods within a class. These types of methods can be called on uninstantiated class objects, much like classes using the `static` keyword in Java.

There is really only one difference between these two method decorators, but it's a major one. You probably noticed in the sections above that `@classmethod` methods have a `cls` parameter sent to their methods, while `@staticmethod` methods do not.

This `cls` parameter is the class object we talked about, which allows `@classmethod` methods to easily instantiate the class, regardless of any inheritance going on. The lack of this `cls` parameter in `@staticmethod` methods make them true static methods in the traditional sense. They're main purpose is to contain logic pertaining to the class, but that logic should not have any need for specific class instance data.

A Longer Example

Now let's see another example where we use both types together in the same class:

```
# static.py

class ClassGrades:

    def __init__(self, grades):
        self.grades = grades

    @classmethod
    def from_csv(cls, grade_csv_str):
        grades = map(int, grade_csv_str.split(', '))
        cls.validate(grades)
        return cls(grades)


    @staticmethod
    def validate(grades):
        for g in grades:
            if g < 0 or g > 100:
                raise Exception()

try:
    # Try out some valid grades
    class_grades_valid = ClassGrades.from_csv('90, 80, 85, 94, 70')
    print 'Got grades:', class_grades_valid.grades

    # Should fail with invalid grades
    class_grades_invalid = ClassGrades.from_csv('92, -15, 99, 101, 77, 65, 100')
    print class_grades_invalid.grades
except:
    print 'Invalid!'
```

```
$ python static.py
Got grades: [90, 80, 85, 94, 70]
Invalid!
```

Notice how the static methods can even work together with `from_csv` calling `validate` using the `cls` object. Running the code above should print out an array of valid grades, and then fail on the second attempt, thus printing out "Invalid!".

Conclusion

In this article you saw how both the `@classmethod` and `@staticmethod` decorators work in Python, some examples of each in action, and how they differ from each other. Hopefully now you can apply them to your own projects and use them to continue to improve the quality and organization of your own code.

Have you ever used these decorators before, and if so, how? Let us know in the *comments!*

---

python (/tag/python/), explained (/tag/explained/)

(https://twitter.com/share?text=Python's%20%40classmethod%20and%20%40staticmethod%20Explained&url=https://stackabuse.com/pythons-classmethod-and-staticmethod-explained/)

(https://www.facebook.com/sharer/sharer.php?u=https://stackabuse.com/pythons-classmethod-and-staticmethod-explained/)

(https://www.linkedin.com/shareArticle?mini=true%26url=https://stackabuse.com/pythons-classmethod-and-staticmethod-explained/%26source=https://stackabuse.com)

(/author/scott/)

About Scott Robinson (/author/scott/)

Twitter (https://twitter.com/ScottWRobinson)

‹ Previous Post (/using-global-variables-in-node-js/)

Next Post › (/get-query-strings-and-parameters-in-express-js/)

Ad

Follow Us

Twitter (https://twitter.com/StackAbuse)

Facebook (https://www.facebook.com/stackabuse)

RSS (https://stackabuse.com/rss/)

Newsletter

Skylight *1 day ago* (https://hireremote.io/remote-job/2596-software-engineer-at-skylight)

react-js (https://hireremote.io/remote-react-js-jobs) python (https://hireremote.io/remote-python-jobs) ruby-on-rails (https://hireremote.io/remote-ruby-on-rails-jobs) healthcare (https://hireremote.io/remote-healthcare-jobs)

Senior Software Engineer
Aloe Care Health *1 day ago* (https://hireremote.io/remote-job/2599-senior-software-engineer-at-aloe-care-health)

aws (https://hireremote.io/remote-aws-jobs) gcp (https://hireremote.io/remote-gcp-jobs) jvm (https://hireremote.io/remote-jvm-jobs) java (https://hireremote.io/remote-java-jobs)

Voter Protection & Insights Analyst
The Democratic Party *2 days ago* (https://hireremote.io/remote-job/2587-voter-protection-and-insights-analyst-at-the-democratic-party)

python (https://hireremote.io/remote-python-jobs)

More jobs (https://hireremote.io)

Jobs via HireRemote.io (https://hireremote.io)

Prepping for an interview?

(https://stackabu.se/daily-coding-problem)

- Improve your skills by solving one coding problem every day
- Get the solutions the next morning via email
- Practice on **actual problems** asked by top companies, like:

Google facebook amazon.com Microsoft

Daily Coding Problem (https://stackabu.se/daily-coding-problem)

Ad

Recent Posts

Handling CORS with Node.js (/handling-cors-with-node-js/)

Command Line Arguments in Java - Accessing and Mapping to Data Types (/java-command-line-arguments-accessing-and-mapping-data-types/)

Deep Learning in Keras - Building a Deep Learning

Tags

ai (/tag/ai/)

algorithms (/tag/algorithms/)

amqp (/tag/amqp/)

angular (/tag/angular/)

announcements (/tag/announcements/)

apache (/tag/apache/)

apache commons (/tag/apache-commons/)

Follow Us

Twitter (https://twitter.com/StackAbuse)
Facebook (https://www.facebook.com/stackabuse)
RSS (https://stackabuse.com/rss/)

Model (/untitled/)

api (/tag/api/)

arduino (/tag/arduino/)

artificial intelligence (/tag/artificial-intelligence/)