When should I use @classmethod and when def method(self)?

Asked 10 years, 8 months ago Modified 9 months ago Viewed 50k times

128

While integrating a Django app I have not used before, I found two different ways to define functions inside the class. The author seems to use them both distinctively and intentionally. The first one is the one that I myself use a lot:

```
class Dummy(object):

    def some_function(self, *args, **kwargs):
        # do something here
        # self is the class instance
```

The other one is the one I never use, mostly because I do not understand when and what to use it for:

```
class Dummy(object):

    @classmethod
    def some_function(cls, *args, **kwargs):
        # do something here
        # cls refers to what?
```

The `classmethod` decorator in the python documentation says:

> A class method receives the class as the implicit first argument, just like an instance method receives the instance.

So I guess `cls` refers to `Dummy` itself (the `class`, not the instance). I do not exactly understand why this exists, because I could always do this:

```
type(self).do_something_with_the_class
```

Is this just for the sake of clarity, or did I miss the most important part: spooky and fascinating things that couldn't be done without it?

python class-method method-dispatch

Share Follow

edited May 3, 2022 at 20:02
TylerH
20.4k 62 75 96

asked May 14, 2012 at 15:54
marue
5,488 7 36 65

3 Answers

Sorted by:

Highest score (default)

90

Your guess is correct - you understand *how* `classmethod` s work.

The why is that these methods can be called both on an instance OR on the class (in both cases, the class object will be passed as the first argument):

```
class Dummy(object):

    @classmethod
    def some_function(cls,*args,**kwargs):
        print cls

#both of these will have exactly the same effect
Dummy.some_function()
Dummy().some_function()
```

On the use of these on instances: There are at least two main uses for calling a classmethod on an instance:

1. `self.some_function()` will call the version of `some_function` on the actual type of `self`, rather than the class in which that call happens to appear (and won't need attention if the class is renamed); and
2. In cases where `some_function` is necessary to implement some protocol, but is useful to call on the class object alone.

The difference with `staticmethod`: There is another way of defining methods that don't access instance data, called `staticmethod`. That creates a method which does not receive an implicit first argument at all; accordingly it won't be passed any information about the instance or class on which it was called.

```
In [6]: class Foo(object): some_static = staticmethod(lambda x: x+1)

In [7]: Foo.some_static(1)
Out[7]: 2

In [8]: Foo().some_static(1)
Out[8]: 2

In [9]: class Bar(Foo): some_static = staticmethod(lambda x: x*2)

In [10]: Bar.some_static(1)
Out[10]: 2

In [11]: Bar().some_static(1)
Out[11]: 2
```

The main use I've found for it is to adapt an existing function (which doesn't expect to receive a `self`) to be a method on a class (or object).

Share Follow

edited May 14, 2012 at 17:11

answered May 14, 2012 at 15:58
Marcin
47.8k 17 127 199

2 Nice one: I like the answer being split up into how - why. As far as i got it now @classmethod allows to access the function without the need for an instance. This is exactly what i was looking for, thank you. – marue May 14, 2012 at 16:04

1 @marcin True duck typing does give it some other dimension. It was more about not writing things like `self.foo()` when it should be `Bar.foo()`. – Voo May 14, 2012 at 16:16

5 @Voo Actually, `self.foo()` is preferable, because `self` may be an instance of a subclass which implements its own `foo`. – Marcin May 14, 2012 at 16:17

1 @Marcin I think this depends on what you want to achieve. But i get your point. – marue May 14, 2012 at 16:19

3 You should use a different lambda for `Bar`, as `1+1 == 2 == 1*2`, so it's impossible to tell from the results shown that `Bar().static_method` is actually called. – George V. Reilly Jan 6, 2017 at 6:28

1

One of the most common uses of `classmethod` in Python is factories, which are one of the most efficient methods to build an object. Because `classmethod`s, like `staticmethod`s, do not need the construction of a class instance. (But then if we use `staticmethod`, we would have to hardcode the instance class name in the function)

This blog does a great job of explaining it: https://iscinumpy.gitlab.io/post/factory-classmethods-in-python/

Share Follow edited Jan 23, 2022 at 23:23 answered Jan 23, 2022 at 8:38

aerin
19k 28 97 133

Why hardcoding the class name is a problem? – lcjury Oct 23, 2022 at 22:33

-4

If you add decorator @classmethod, That means you are going to make that method as static method of java or C++. (static method is a **general term** I guess **;)**) Python also has @staticmethod. and difference between classmethod and staticmethod is whether you can access to class or static variable using argument or classname itself.

```
class TestMethod(object):
    cls_var = 1
    @classmethod
    def class_method(cls):
        cls.cls_var += 1
        print cls.cls_var

    @staticmethod
    def static_method():
        TestMethod.cls_var += 1
        print TestMethod.cls_var
#call each method from class itself.
TestMethod.class_method()
TestMethod.static_method()

#construct instances
testMethodInst1 = TestMethod()
testMethodInst2 = TestMethod()
```

```
#call each method from instances
testMethodInst1.class_method()
testMethodInst2.static_method()
```

all those classes increase cls.cls_var by 1 and print it.

And every classes using same name on same scope or instances constructed with these class is going to share those methods. There's only one TestMethod.cls_var and also there's only one TestMethod.class_method() , TestMethod.static_method()

And important question. why these method would be needed.

classmethod or staticmethod is useful when you make that class as a factory or when you have to initialize your class only once. like open file once, and using feed method to read the file line by line.

Share Follow

edited May 14, 2012 at 17:07

answered May 14, 2012 at 15:59

Ryan Kim
258 1 6

2 (a) Static methods are something else; (b) classmethods are not shared by every class. – Marcin May 14, 2012 at 16:03

@Marcin thanks for letting me know my unclear definitions and all that. – Ryan Kim May 14, 2012 at 17:08