functools.wraps

There is a little known tool that I wanted to cover in this section. It is called **wraps** and it too is a part of the functools module. You can use wraps as a decorator to fix docstrings and names of decorated functions. Why does this matter? This sounds like a weird edge case at first, but if you're writing an API or any code that someone other than yourself will be using, then this could be important. The reason being that when you use Python's introspection to figure out someone else's code, a decorated function will return the wrong information. Let's look at a simple example that I have dubbed decorum.py:

```
# decorum.py


def another_function(func):
    """
    A function that accepts another function
    """

    def wrapper():
        """
        A wrapping function
        """
        val = "The result of %s is %s" % (func(),
                                          eval(func())
                                          )

        return val
    return wrapper


@another_function
def a_function():
    """A pretty useless function"""
    return "1+1"


if __name__ == "__main__":
    print(a_function.__name__)
    print(a_function.__doc__)
```

In this code, we decorate the function called a_function with another_function. You can check a_function's name and docstring by printing them out using the function's __name__ and __doc__ properties. If you run this example, you'll get the following for output:

```
wrapper

A wrapping function
```

That's not right! If you run this program in IDLE or the interpreter, it becomes even more obvious how this can get really confusing, really quickly.

```
import decorum

help(decorum)
#Help on module decorum:

#NAME
#    decorum -

#FILE
#    /home/mike/decorum.py

#FUNCTIONS
#    a_function = wrapper()
#        A wrapping function

#    another_function(func)
#        A function that accepts another function

help(decorum.a_function)
#Help on function other_func in module decorum:

#wrapper()
#    A wrapping function
```

Basically what is happening here is that the decorator is changing the decorated function's name and docstring to its own.