Decorators in Flask

Estimated time needed: **10** minutes

Objectives

After reading this page, you will be able to:

1. Understand what are decorators
2. Understand the two kind of decorators you come across in a python application.
3. When and how to use the decorators.

What are decorators

Decorators help in annotating the methods and tell what a particular method is meant for. These are different from comments. This is used by the interpreter while running the code.

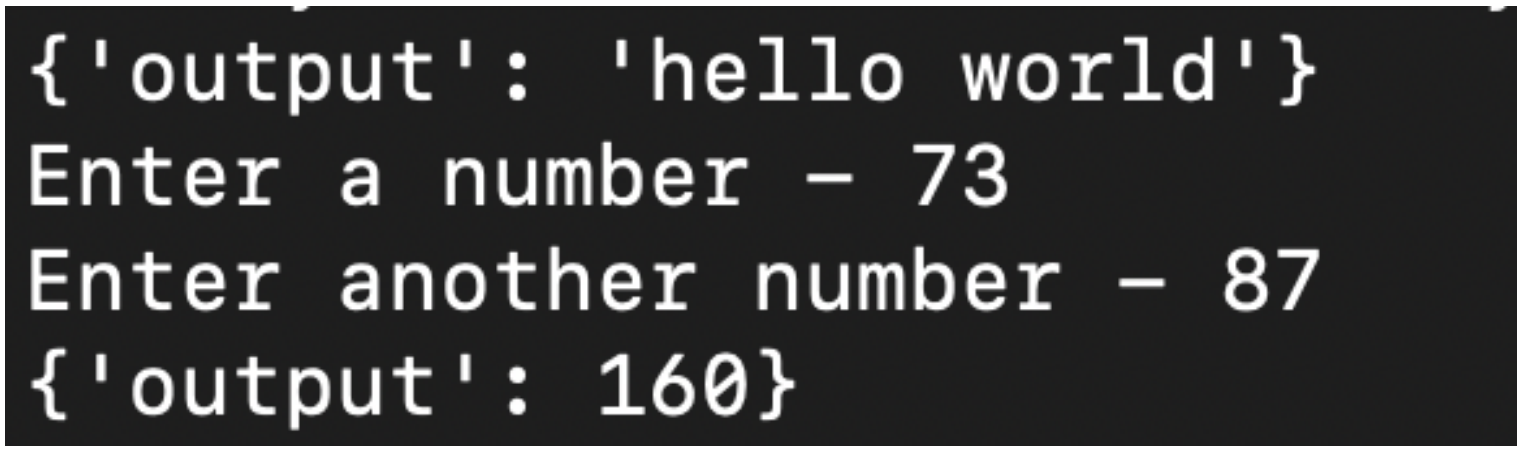
Method decorators

Let's say we have a python code where we want all the output to be in JSON format. It doesn't make sense to include code for these in each of the methods as it makes the lines of code redundant. In such cases, we can handle this with a decorator.

```
def jsonify_decorator(function):
    def modifyOutput():
        return {"output":function()}
    return modifyOutput

@jsonify_decorator
def hello():
    return 'hello world'


@jsonify_decorator
def add():
    num1 = input("Enter a number - ")
    num2 = input("Enter another number - ")
    return int(num1)+int(num2)

print(hello())
print(add())
```

The method decorator is also referred to as the wrapper, which wraps the output of the function, that it decorates. In the above code sample, `jsonify-decorator` is the decorator method. We have added this decorator to `hello()` and `add()`. The output of these method calls will now be wrapped and decorated with the jsonify_decorator.

The output of invoking the above python code will be:



As you can see the method call has been wrapped with the decorator. The code has a free will to choose the name of the decorator. Here the name chosen is `jsonify_decorator`.

Route Decorators

You may have observed when you visit any domain you have the root and then have other end-points you can access. See the examples below.

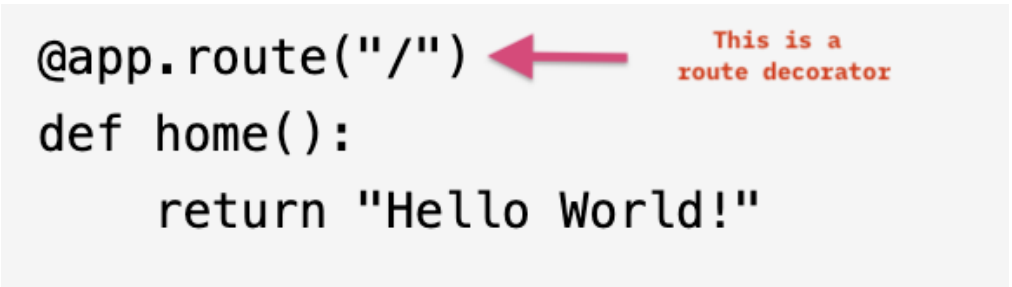
`https://mydomain.com/`

`https://mydomain.com/about`

`https://mydomain.com/register`

We will look at creating these endpoints when we will create a web application using the flask module in the labs that will follow.

But to define these endpoints in Python we use what we call **Route Decorators**.



@app.route("/") is a Python decorator that Flask provides to assign URLs in our app to functions easily. You can easily tell that the decorator is telling our @app that whenever a user visits our application's domain, in our case, execute the `home()` function.

We can handle multiple routes with a single function by just stacking additional route decorators above the method which should be invoked when the route is called. The following is a valid example of serving the same "Hello World!" message for 3 separate routes:

```
@app.route("/")
@app.route("/home")
@app.route("/index")
def home():
    return "Hello World!"
```

The route decorators also take `method` as a second parameter, which can be set to the type of HTTP methods, the route will support. We will learn about this in the later sections.

The route decorator can also be more specific. For example, to get the details of a user whose userId is U0001, you may go to `http://mydomain.com/userdetails/U0001`. It doesn't make sense to define a different route for each user you may be dealing with. In such cases, we define the route like this.

```
@app.route("/userdetails/<userid>")
def getUserDetails(userid):
    return "User Details for  "+userid
```

Congratulations! You have just learned about decorators in python. Go ahead and use it in your code.

Authors

Lavanya

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2021-05-28	1.0	Lavanya	Created the document for reading