9.4.3 Set Up Flask and Create a Route

You need to get to work! You get out your laptop—it's time to install Flask and get acquainted with how to programmatically create a Flask route.

We need to set up Flask and then get started on creating our first Flask route. When creating a Flask application, we'll need to do a few things first. Here's our course of action:

1. Install Flask.
2. Create a new Python file.
3. Import the Flask dependency.
4. Create a new Flask app instance.
5. Create Flask routes.
6. Run a Flask app.

Install Flask

You can install Flask by running the following in the command line:

```
pip install flask
```

If you already have Flask installed, the output of your code will show that you have already installed many of the components, if not all of them. Running this command will ensure that you have the most up-to-date version of Flask.

We need to ensure that we have the right environment in order for Flask to run properly. To do this, we will need to make a change in VS Code. We just need to select the PythonData environment we created in previous modules. To do this, click on "Select Python Interpreter," then select the PythonData environment.

IMPORTANT

> Some users may need an additional package installed for this section. In your terminal, run the following code: `pip install psycopg2-binary` to add it to your coding environment.

Create a New Python File and Import the Flask Dependency

Create a new Python file called `app.py`. You should create this file in VS Code.

Once the Python file is created, we can import the dependency we need. This dependency will enable your code to access all that Flask has to offer.

To import the Flask dependency, add the following to your code:

```
from flask import Flask
```

Create a New Flask App Instance

We're now ready to create a new Flask app instance. "Instance" is a general term in programming to refer to a singular version of something. Add the following to your code to create a new Flask instance called `app`:

```
app = Flask(__name__)
```

IMPORTANT

> You probably noticed the `__name__` variable inside of the `Flask()` function. Let's pause for a second and identify what's going on here.
>
> This `__name__` variable denotes the name of the current function. You can use the `__name__` variable to determine if your code is being run from the command line or if it has been imported into another piece of code. Variables with underscores before and after them are called **magic methods** in Python.
>
> For more information, see this **article on magic methods (https://www.geeksforgeeks.org/dunder-magic-methods-python/)** .

Create Flask Routes

Our Flask app has been created—let's create our first route!

First, we need to define the starting point, also known as the **root.** To do this, we'll use the function `@app.route('/')`. Add this to your code now.

```
@app.route('/')
```

NOTE

> Notice the forward slash inside of the `app.route`? This denotes that we want to put our data at the root of our routes. The forward slash is commonly known as the highest level of hierarchy in any computer system.

Next, create a function called `hello_world()`. Whenever you make a route in Flask, you put the code you want in that specific route below `@app.route()`. Here's what it will look like:

```
@app.route('/')
def hello_world():
    return 'Hello world'
```

Great job! You have just created your first Flask route! Now that we have some code, let's keep running.

Run a Flask App

The process of running a Flask app is a bit different from how we've run Python files. To run the app, we're first going to need to use the command line to navigate to the folder where we've saved our code. You should save this code in the same folder you've used in the rest of this module.

Once you've ensured that your code is saved in the proper directory, then run the following command if you are on a Mac. This command sets the `FLASK_APP` environment variable to the name of our Flask file, `app.py`.

NOTE

> Environment variables are essentially dynamic variables in your computer. They are used to modify the way a certain aspect of the computer operates. For our FLASK_APP environment variable, we want to modify the path that will run our `app.py` file so that we can run our file.

```
export FLASK_APP=app.py
```

There won't be any output when you run this command, so don't worry if you don't see anything.

If you are on a Windows computer, you will need to do the same thing, but in a slightly different way. Start by opening up Anaconda Powershell. Once you've done that, enter this command.

```
set FLASK_APP=app.py
```

Now let's run our Flask app. To do this, type the following command in your command line and press Enter:

```
flask run
```

When you run this command, you'll notice a line that says "Running on" followed by an address. This should be your localhost address and a port number.

IMPORTANT

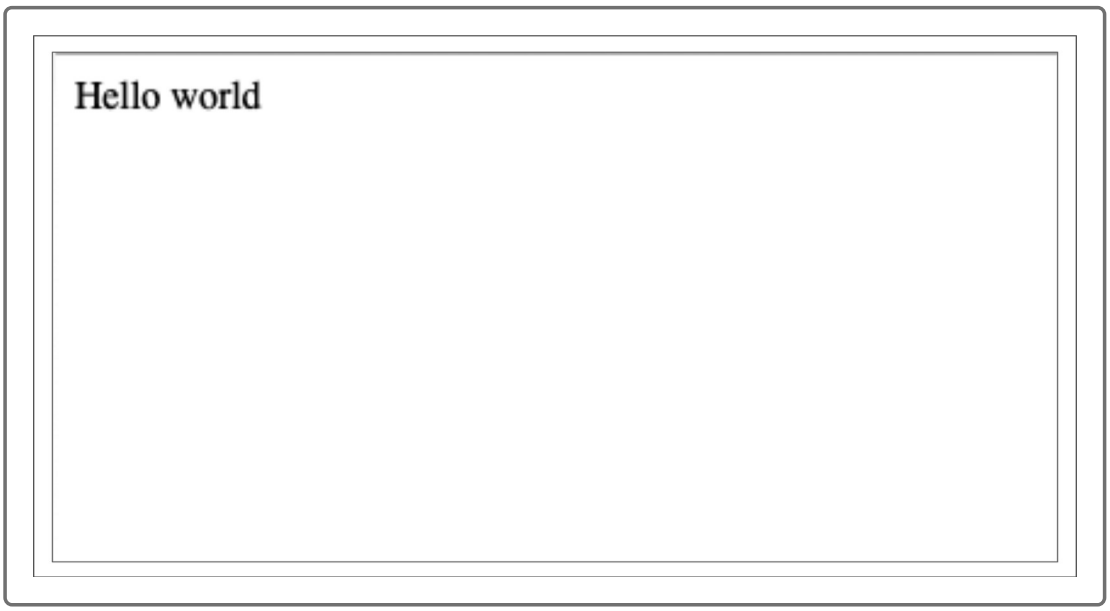
> A port number is essentially the endpoint of a given program or service.

> Any Flask application you create can have whatever port number you would like, but the most common is 5000.

Copy and paste your localhost address into your web browser. Generally, a localhost will look something like this, for context.

```
localhost:5000
```

This is what you should see:



You ran your first Flask app! You should feel ready to create more routes that incorporate our analysis.

For some extra practice, let's create another route in the following Skill Drill.

SKILL DRILL

Think of some simple code from which you could create a route. Then try to create a new route

implementing that logic.

ADD/COMMIT/PUSH

Great work on the Flask app. Now is a good time to add, commit, and push your updates to GitHub. Remember these steps:

1. `git add < FILE NAME>`
2. `git commit -m "< ADD COMMIT MESSAGE HERE>"`
3. `git push origin main`