Variables and Routing in Flask

Learning Objectives

After this lesson, you will be able to:

- Display variables on a webpage.
- Create a route in Flask.

Multiple Routes

- Our website is cool, but it’s just one page.
- What about recipe pages? “About” pages?
- We need to use `routes`.

But first, we need to learn `variables`.

Variables? Again?

- Yes! Regular variables.

```
x = "this string"
```

- Difference: Here, we're in the Flask app.
- Very specific use cases:
  - Routes (We're learning now.)
  - Templates (We'll learn next.)
  - Requests (We'll learn later.)

Three Ways to Read in a Variable

Variables come from:

- Within our Flask app.
- From another Python file.
- From any other file.

Method 1: Set Variables in Our Flask App

These *aren't* set inside `def hello()`.

- What does that make them?

`hello_variables.py`

```
from flask import Flask


app = Flask(__name__)


my_job_title = "Python pro"


@app.route('/')
def hello():
    return "Hello, " + my_job_title


if __name__ == '__main__':
```

We Do: In-App Variables

- We can practice this: In your existing `my_website.py`, comment out the `return render_template("index.html")`.

Instead, have:

```
my_job_title = "Python pro"


@app.route('/')
def home():
    return "Hello, " + my_job_title
```

Method 2: Read Variables From a Python File

- You're never limited to just one `.py` file!
- New Python file: `mySecrets.py`

```
username = "Guy Fieri"
password = "flavortown"
```

How would we print that in our Flask app?

Any ideas?

We Can Import the File

Your normal Flask app:

```
from flask import Flask

import mySecrets ## You can import any file!


app = Flask(__name__)


## Call it like a module.

my_name = mySecrets.username

my_password = mySecrets.password


@app.route('/')

def hello():
```

Method 2: Use Cases

Why?

- You have secret info (tokens, passwords, etc.) — keep them locally!
- You have many Flask pages, so you make a “master file” to hold all variables.

Your Turn: Another `py` File

Now it's your turn!

- Make a file called `python_variables.py` in the same folder as `my_website.py`.
- Insert some variables into `python_variables.py` - perhaps some books you like.
- Import `python_variables` into your Flask app, `my_website.py`.
- Display the values from `python_variables` in your Flask app.

Method 3: Reading From a Non-Python File

Let's create a `.txt` file called `hi.txt` in the same folder where our app lives. We'll include some Shakespeare poetry.

```
So are you to my thoughts as food to life,

Or as sweet-seasoned showers are to the ground;
```

How do you think we get this into our Flask app?

With File Open

Then, we'll add a bit in our Flask app:

```python
import os # Note the new import — to be in the file system.


file_path = '.'
# Note the "with"! We don't need "close".
with open(os.path.join(file_path, 'hi.txt')) as f:
    the_text = f.read()


@app.route('/text')
def read_txt():
    return the_text
```

You Do: Add a `.txt` File

Now it's your turn!

- Make a file called `more_variables.txt` in the same folder as `my_website.py`.
- Write some information into `more_variables.txt` — perhaps what you'd like for breakfast tomorrow.
- `import os` so you can find the file.
- Use this code:

```
with open(os.path.join(file_path, 'more_variables.txt')) as f:
    the_text = f.read()
```

- Display the text from `more_variables` in your Flask app.

Knowledge Check

What are the three approaches to read in variables to a Flask app?

Part 2. Routing

What Is That `@app.route('/')`, Anyway?

We have:

- Listen to an endpoint (here, `/`).
- Do `def home()` if someone goes there.

```
@app.route('/') # When someone goes here...
def home(): # Do this.
  return render_template("index.html")
```

`http://127.0.0.1:5000/` `=> render_template("index.html")"`

What if we want to go to `http://127.0.0.1:5000/sayHi`?

Suddenly, a New Page!

- This is **routing**.
- New pages on our web app!

```
@app.route('/sayHi') # When someone goes here...
def hello(): # Do this.
    return "Hello, Mr. Fieri."
```

We Do: Add a Route

- In `my_website.py`, under `def home()`, add:

```
@app.route('/sayHi') # When someone goes her
def hello(): # Do this.
    return "Hello, Mr. Fieri."
```

- Reload the page! Go to `http://127.0.0.1:5000/sayHi`.

What Is a Route?

- The URL: `http://127.0.0.1:5000/sayHi`
- We *route* to different URLs:
  - `http://127.0.0.1:5000/sayHi`
  - `http://127.0.0.1:5000/Cats`
  - `http://127.0.0.1:5000/profile`
- `sayHi`, `Cats`, `/`, and `profile` are **endpoints** from our main app.
- We only need to add:

```
@app.route('/<endpoint>')  # When someone goes he
def function_name(): # Do this.
    return string
```

You Do: Adding a Route

- In `my_website.py`, add a new route to a `randnum` endpoint.
- In the function for this endpoint, display a string that's a random number.
  - *Hint:* Remember the `random` module? You can use `randint(1, 100)`.
  - *Hint:* You can turn an integer to a string with `str(number)`.
- Reload the page and go to your endpoint to try it out!

Variables in the Route

- You can pass a variable in the route itself.
- It’s a dynamic endpoint!
- You can use that variable in your function.

```
@app.route('/sayHi/<name>')

def hello(name):

    return "Hello, " + name + ", your coding skills impress me!"
```

`http://localhost:5000/sayHi/Hari` `=> Hello, Hari, your coding skills impress me!"`

Your Turn!

Try adding route in your Flask app to have:

- A `/timesfour/<number>` route that displays the product of an integer in the route multiplied by four.
- A `repeat` route that takes a string passed into the URL, then displays it four times in a row.

Final Code Status

Your code should look similar to this:

```
from flask import Flask, render_template
import mySecrets ## You can import any file!
import python_variables
import os # Note the new import — to be in the file system.
import random

app = Flask(__name__)

## Call it like a module.
my_name = mySecrets.username
my_password = mySecrets.password
```

Summary

We covered variables and routing in Flask:

- Variables can be made:
    - In the Flask app: Used like normal variables.
    - In a Python file: Imported like a module.
    - In another file: Used `file` to read it.
- Routing:
    - `@app.route(<endpoint>)` is how we make new pages in our app!