# How to build a web application using Flask and store the codebase in GITHUB repository

## Introduction:

In each section, I will show pieces of code for you to follow along. All the code used in the tutorial is available in this GitHub Repository
https://github.com/KishoreNandhu/Harmon_international/tree/main/Recap_Session

## How Does a Flask App Work?

The code lets us run a basic web application that we can serve, as if it were a website.

```
from flask import Flask app = Flask(__name__)

@app.route("/")
def home():

return "Hello, World!"

if __name__ == "__main__": app.run(debug=True)
```

This piece of code is stored in our main.py.

**Line 1:** Here we are importing the Flask module and creating a Flask web server from the Flask module.

**Line 3: __name__ means this current file**. In this case, it will be main.py. This current file will represent my web application.

We are creating an instance of the Flask class and calling it app. Here we are creating a new web application.

**Line 5:** It represents the default page. For example, if I go to a website such as "google.com/" with nothing after the slash. Then this will be the default page of Google.

**Line 6–7**: When the user goes to my website and they go to the default page (nothing after the slash), then the function below will get activated.

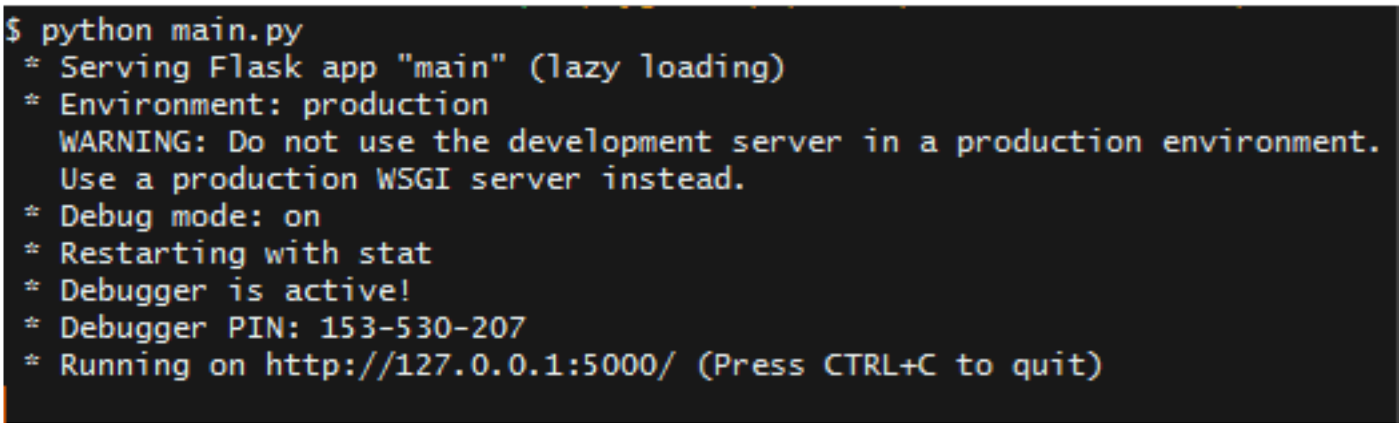
**Line 9:** When you run your Python script, Python assigns the name "__main__" to the script when executed.

If we import another script, the **if statement will prevent other scripts from running.** When we run main.py, it will change its name to __main__ and only then will that if statement activate.

**Line 10:** This will run the application. Having `debug=True` allows possible Python errors to appear on the web page. This will help us trace the errors.

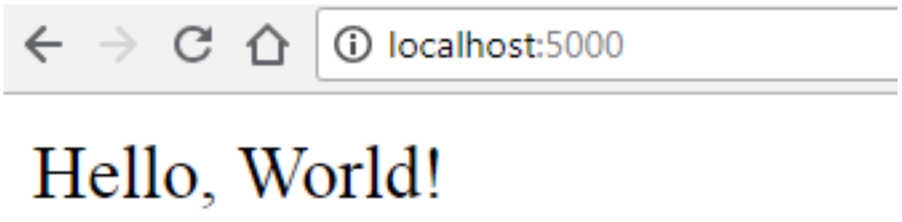
### Let's Try Running main.py

In your Terminal or Command Prompt go to the folder that contains your main.py. Then do **`py main.py`** or **`python main.py`**. In your terminal or command prompt you should see this output.



```
$ python main.py
 * Serving Flask app "main" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 153-530-207
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

The important part is where it says `Running on http://127.0.0.1:5000/`. Go to that address and you should see the following:



localhost:5000

Hello, World!

website with Flask!

## More Fun with Flask:

Earlier you saw what happened when we ran main.py with one route which was app.route("/").

Let's add more routes so you can see the difference.

```
from flask import Flask
```

Congrats! You made a

```
app = Flask(__name__)

@app.route("/")
def home():

return "Hello, World!"

@app.route("/salvador") def salvador():

return "Hello, Salvador"

if __name__ == "__main__": app.run(debug=True)
```

In **lines 9–11**. we added a new route, this time to **/salvador.**
Now run the main.py again and go to http://localhost:5000/salvador.

So far we have been returning text. Let's make our website look nicer by adding HTML and CSS.

### HTML, CSS

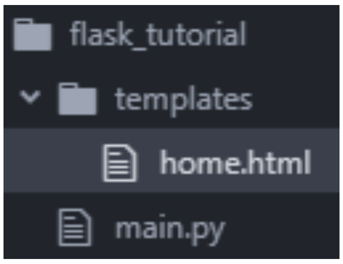
## HTML and Templates in Flask:

First create a new HTML file. I called mine **home.html.** Here is some code to get you started.

```
<!DOCTYPE html>
<html lang="en" dir="ltr">

<head>
<meta charset="utf-8"> <title>Flask Tutorial</title>

  </head>
  <body>

<h1> My First Try Using Flask </h1>

    <p> Flask is Fun </p>
  </body>

</html>
```

*Important Point To Remember*

The Flask Framework looks for HTML files in a folder called **templates.** You **need to create a templates** folder and put all your HTML files in there.



folder

Remember to always keep the main.py outside of your templates

Now we need to change our main.py so that we can view the HTML file we created.

```
from flask import Flask, render_template app = Flask(__name__)

@app.route("/")
def home():

return render_template("home.html")

@app.route("/salvador") def salvador():

return "Hello, Salvador"

if __name__ == "__main__": app.run(debug=True)

  We made two new changes
```

**Line 1:** We imported `render_template()` method from the flask framework. `render_template()` looks for a template (HTML file) in the templates folder. Then it will render the template for which you ask. Learn more about render_templates() function.

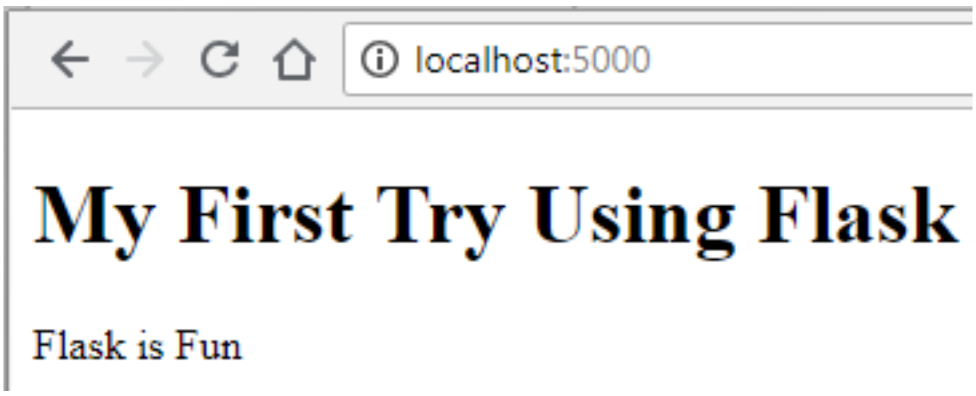
**Line 7:** We change the return so that now it returns `render_template("home.html")`. This will let us view our HTML file.

Now visit your localhost and see the changes: http://localhost:5000/.

## Let's add more pages

Let's create an **about.html** inside the **templates folder.**

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
```


localhost:5000

My First Try Using Flask

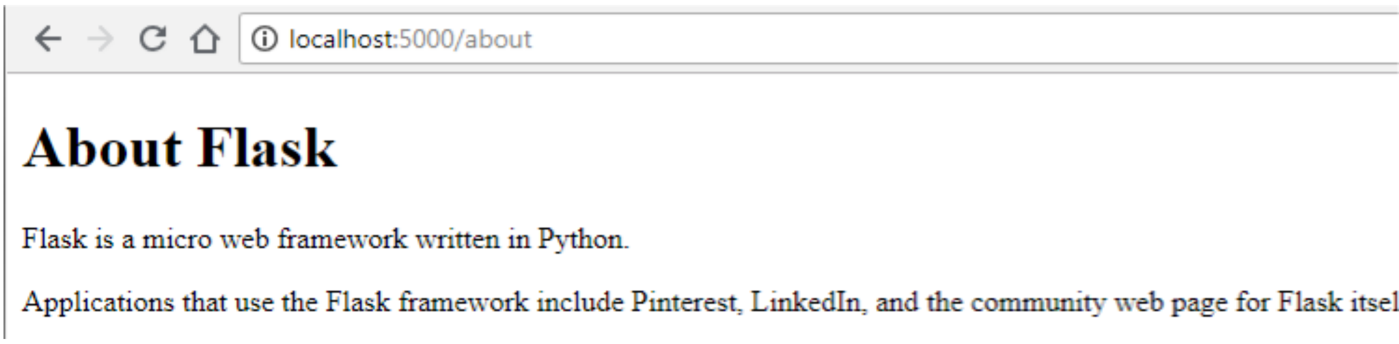
Flask is Fun

```
<head>
<meta charset="utf-8"> <title>About Flask</title>

  </head>
  <body>

<h1> About Flask </h1>
<p> Flask is a micro web framework written in Python.</p>
<p> Applications that use the Flask framework include Pinterest,

LinkedIn, and the community web page for Flask itself.</p> </body>

</html>
```

Let’s make a change similar to what we did before to our main.py.

```
from flask import Flask, render_template app = Flask(__name__)

@app.route("/")
def home():

return render_template("home.html")

@app.route("/about) def about():

return render_template("about.html")

if __name__ == "__main__": app.run(debug=True)
```

We made three new changes:
**Line 9:** Change the route to `"/about"`**.**
**Line 10:** Change the function so it is now `def about():`
**Line 11:** Change the return so that now it returns `render_template("about.html")`. Now see the changes: http://localhost:5000/about.


localhost:5000/about

About Flask

Flask is a micro web framework written in Python.

Applications that use the Flask framework include Pinterest, LinkedIn, and the community web page for Flask itsel

## Let’s Connect Both Pages with a Navigation

To connect both pages we can have a navigation menu on the top. We can use Flask to make the process of creating a navigation menu easier.

First, let’s create a **template.html.** This **template.html** will serve as a parent template. Our two child templates will inherit code from it.

```
<!DOCTYPE html>
<html lang="en" dir="ltr">

<head>
<meta charset="utf-8">
<title>Flask Parent Template</title>
<link rel="stylesheet" href="{{ url_for('static',

filename='css/template.css') }}"> </head>
<body>

<header>
<div class="container">

<h1 class="logo">First Web App</h1> <strong><nav>

<ul class="menu">
<li><a href="{{ url_for('home') }}">Home</a></li> <li><a href="{{
url_for('about') }}">About</a></li>

          </ul>
        </nav></strong>
      </div>
    </header>
    {% block content %}
    {% endblock %}
 </body>
</html>
```

**Line 13–14:** We use the function called `url_for()`**.** It accepts the name of the function as an argument. Right now we gave it the name of the function. More information on **url_for() function**.

The two lines with the curly brackets will be **replaced by the content of home.html and about.html.** This will depend on the URL in which the user is browsing.

These changes allow the child pages (home.html and about.html) to connect to the parent (template.html). This allows us to not have to **copy the code for the navigation menu in the about.html and home.html.**

Content of about.html:

```
<!DOCTYPE html>
<html lang="en" dir="ltr">

<head>
<meta charset="utf-8"> <title>About Flask</title>

  </head>
  <body>

{% extends "template.html" %} {% block content %}

<h1> About Flask </h1>
<p> Flask is a micro web framework written in Python.</p>
<p> Applications that use the Flask framework include Pinterest,

LinkedIn, and the community web page for Flask itself.</p>


    {% endblock %}
  </body>

</html>
```

Content of home.html:

```
<!DOCTYPE html>
<html lang="en" dir="ltr">

<head>
<meta charset="utf-8"> <title>Flask Tutorial</title>

  </head>
  <body>

{% extends "template.html" %} {% block content %}

<h1> My First Try Using Flask </h1> <p> Flask is Fun </p>

    {% endblock %}
  </body>

</html>
```

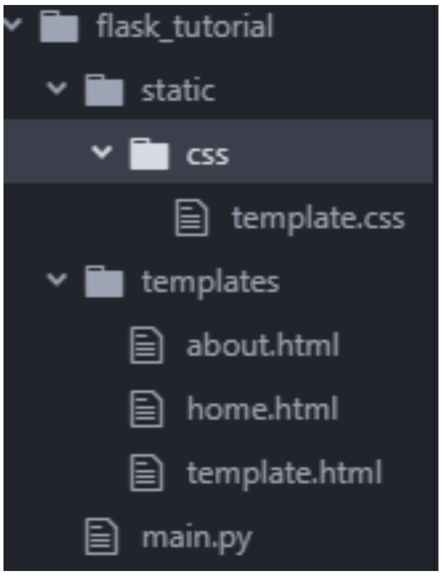
Let’s try adding some CSS.

## Adding CSS to Our Website

### An important note to remember

In the same way as we created a folder called **templates** to store all our HTML templates, we need a folder called **static**.

In **static**, we will store our CSS, JavaScript, images, and other necessary files. That is why it is important that you should create a **CSS folder to store your stylesheets.** After you do this, your project folder should look like this:



```
flask_tutorial
  static
    css
      template.css
  templates
    about.html
    home.html
    template.html
  main.py
```

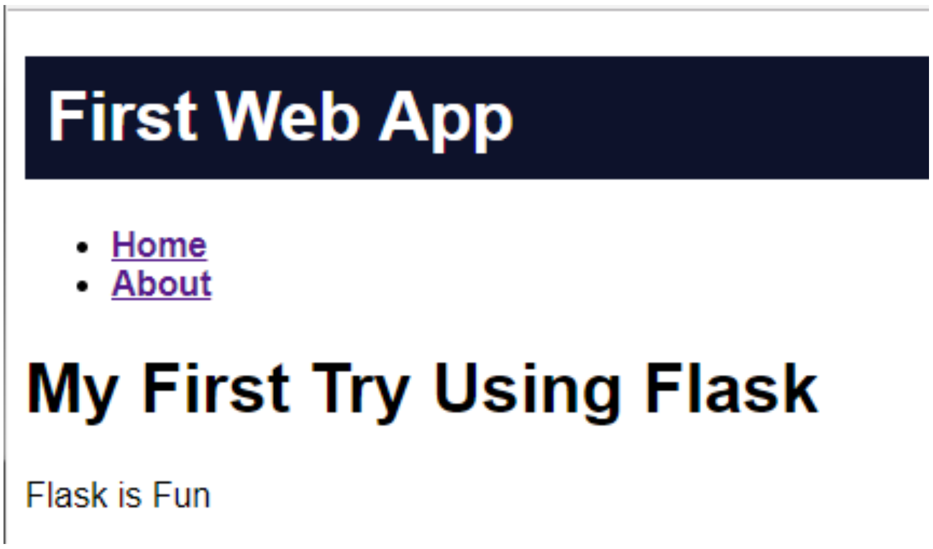
## Linking our CSS with our HTML file

Our template.html is the one that links all pages. We can insert the code here and it will be applicable to all child pages.

```
<!DOCTYPE html>
<html lang="en" dir="ltr">

<head>
<meta charset="utf-8">
<title>Flask Parent Template</title>

<link rel="stylesheet" href="{{ url_for('static',
filename='css/template.css') }}">

</head>
  <body>

<header>
<div class="container">

<h1 class="logo">First Web App</h1> <strong><nav>

<ul class="menu">
<li><a href="{{ url_for('home') }}">Home</a></li> <li><a href="{{
url_for('about') }}">About</a></li>

          </ul>
        </nav></strong>
      </div>
```

```
    </header>
{% block content %}
{% endblock %}
 </body>
</html>
```

**Line 7:** Here we are giving the path to where the template.css is located. Now see the changes: http://127.0.0.1:5000/about.


First Web App

- Home
- About

My First Try Using Flask

Flask is Fun