

# a guide to flask

## Make a Web App With Flask [🔗](#)

### What is Flask? [🔗](#)

Flask is a lightweight Python web framework. You just write Python functions that return web pages, making backend development way easier.

**Before we start...** This guide assumes you know basic git and have Python installed on your machine.

### Setup [🔗](#)

1. Create an empty directory and open a terminal.
2. Create a new file called `requirements.txt` and write `flask` inside it.
3. Run the following in your terminal:

```
pip install -r requirements.txt
# If that doesn't work:
pip3 install -r requirements.txt
```

4. Create a new file called `app.py` and write the following:

### Routes [🔗](#)

Routes are what you'll need to display HTML in the browser.

1. In your project directory, create a folder called `templates`, and inside it create an `index.html` file.
2. Back in `app.py`, below `app = Flask(__name__)`, add your routes.
3. Run the app:

```
python3 app.py
```

4. Visit `http://127.0.0.1:5000/` — you should see a blank white page.
5. Go to your `index.html` file and write the following:

```
<p>Hello World!</p>
```

6. Back at `http://127.0.0.1:5000/`, you should now see **"Hello World!"**

## Jinja Templating [🔗](#)

Flask lets you pass Python variables into HTML using Jinja Templating.

1. In `app.py`, edit the `index()` function — create a `user_name` variable and pass it when rendering:

```
@app.route("/")
def index():
    user_name = "Mustafa"
    return render_template("index.html", user_name=user_name)
```

2. In `index.html`, edit the `<p>` tag to display the variable:

```
<p>Hello {{ user_name }}!</p>
```

3. Reload your tab — you should see whatever name you set in `user_name`.
4. Try it on another route — in `app.py`, add a `/testing` route:

```
@app.route("/testing")
def testing():
    user_name = "Mustafa"
    return render_template("index.html", user_name=user_name)
```

5. Visit `http://127.0.0.1:5000/testing` to see it in action. Going back to `/` will still show your original name.

## Templates

As your site grows, use a **base template** to avoid repeating HTML across pages.

1. In the `templates` directory, create `base.html`:

```
<!DOCTYPE html>
<html>
  <head><title>My Site</title></head>
  <body>
    {% block content %}{% endblock %}
  </body>
</html>
```

2. Edit `index.html` to extend it:

```
{% extends "base.html" %}
{% block content %}
  <p>Hello {{ user_name }}!</p>
{% endblock %}
```

3. Create `testing.html` in the same directory:

```
{% extends "base.html" %}
{% block content %}
  <p>Testing page — Hello {{ user_name }}!</p>
{% endblock %}
```

4. In `app.py`, update the `/testing` route to use `testing.html`:

```
return render_template("testing.html", user_name=user_name)
```

5. Visit `http://127.0.0.1:5000/testing` to see the new template in action.

## Adding CSS

1. Create a `static` directory in your project root, and inside it create `styles.css`.

2. Link the stylesheet in `base.html`:

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

3. Test it out — try changing the background color in `styles.css`:

```
body {
  background-color: #f0f0f0;
}
```

4. Reload your page to see the result.

## Deploying Your Project

You can use free hosting platforms such as [Vercel](#) to host your project.

*Written by Mustafa*

