Let’s Build your Website!
Set up a repo
Click “Start a Project” and create your repository

A repository is where all of your code gets stored.

https://github.com/

Instead of “my-website,” name it your own name, like “adalovelace” or “ameliaearheart”

(this is what will show in the URL when you publish it later)
Open up Terminal

Go to your Desktop or folder of choice in Terminal using "cd"

cd ~/Desktop
mkdir FemmehacksDemo
cd FemmehacksDemo
ls
Tell Github where your code is

Copy the url mentioned in your repo

```
git clone https://github.com/blahblahblah
```
Pick a text editor

I like Sublime. You can download it at https://www.sublimetext.com/3

Open the application once it downloads

Open the folder where the github repo is saved

Create a new file using CTRL+N

http://bit.ly/2DXg0M9
Boilerplate HTML

- Start with this every time
- `<head>` = metadata
- `<body>` = actual content

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>TITLE HERE</title>
    <meta name="description" content="DESCRIBE YOUR WEBSITE">
    <meta name="keywords" content="KEY, WORDS, HERE">
  </head>

  <body>
  
  </body>
</html>
```
Headers and Paragraphs

- `<h1> ... <h6> Text! </h1>` etc.
  - Headers
- `<p> Text here </p>`
  - Paragraph tags
  - Enclose text in a formatted paragraph
- `<br>`
  - Line `break`
  - Like hitting “enter”
  - Doesn’t have a closing tag

```html
<body>
  <h1>Puppies for Philly</h1>
  <h3>Bow wow wow wow</h3>
  <p>Puppies are PAWsitively cool!</p>
  <br>
  <p>Life would be RUFF, without puppies.</p>
</body>
```
id and class in HTML

- Many `<ul>`, but one special list
  - ex: navigation bar
- Solution: use id attribute
  - Single, unique element
- A whole section of your website
  - ex: about me - all blue
- Solution: use class attribute
  - Multiple elements with shared properties

```html
<div id="unique-section">
  <ul>
    <li>Poodle</li>
    <li>Golden Retreiver</li>
  </ul>
</div>

<span class="regular-section">
  <p>Words words words words words.</p>
  <p>More words.</p>
</span>

<div class="regular-section">
  <p>LALALALALLAL</p>
  <p>More LALALALALAL.</p>
</div>
```
Make it pretty

http://materializecss.com/getting-started.html

Include the right code on your html page, so that it knows where to find the css
Let’s get a navbar

http://materializecss.com/navbar.html

```html
<nav>
  <div class="nav-wrapper">
    <a href="#" class="brand-logo">Logo</a>
    <ul id="nav-mobile" class="right hide-on-med-and-down">
      <li><a href="sass.html">Sass</a></li>
      <li><a href="badges.html">Components</a></li>
      <li><a href="collapsible.html">JavaScript</a></li>
    </ul>
  </div>
</nav>
```
Image card

```html
<div class="row">
  <div class="col s12 m7">
    <div class="card">
      <div class="card-image">
        <img src="images/sample-1.jpg">
        <span class="card-title">Card Title</span>
      </div>
      <div class="card-content">
        <p>I am a very simple card. I am good at containing small bits of information. I am convenient because I require little markup to use effectively.</p>
      </div>
      <div class="card-action">
        <a href="#">This is a link</a>
      </div>
    </div>
  </div>
</div>
```
Adding files and committing

Tell github which files you want to update and track

```
git add .
```

OR

```
git add file.html file2.txt ...
```

Tell github you are ready to commit the changes

```
git commit -m "changed the readme"
```

Send your changes to the online folder

```
git pull origin master
git push origin master
```
Go to settings
Enable Github Pages

By clicking “Master branch” and saving
Now go to _____.github.io

Your username here
Freestyle time!
Use the mentors to help improve your website
Welcome to Flask
Overview

Flask is a microframework for creating a web app in Python.

Flask depends on Jinja, a template language that renders the pages your application serves, allowing you to dynamically generate HTML pages.
Reference Code: https://github.com/saniyah-shaikh/fh20-intermediate
WARNING
if you copy/paste code from this presentation, you will likely need to delete and retype any quotation marks so they use the right characters.
Installation

Follow the instructions at https://flask.palletsprojects.com/en/1.1.x/installation/#install-flask

1. Install virtualenv
2. Activate the environment
3. Run `pip install Flask` to install Flask in the activated environment

If you get pip not found, Google Anaconda, and download and install the right version for your computer, then try again.
Making an app

Here’s the code in app.py for the most minimal app:

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

@app.route('/

def hello_world():
    return 'Hello, World!'

if __name__ == "__main__":
    app.run()
```
Additional Considerations

Jinja templates cannot perform calculations or call APIs. Jinja templates should only have simple if statements and loops that access already existing information in the data structures you have passed through from Python.

Make sure to wrap code statements like so: `{% for x in ["a", "b"]: %} and variables like {{var}}`
Running

Run using `python [file]`, where file is the name of your python file (including .py extension).

It will print out a URL to view the app - you can paste this into your browser.

If you see `python: can't open file '[file]': [Errno 2] No such file or directory`, you might be in the wrong directory. Use the `cd` and `ls` commands to move directories until `ls` shows you the file you are trying to run.

Alternatively, make [file] the complete file path - you can look this up in your file explorer, or by dragging the file into a terminal window.
Passing in data

In your template file (ex: index.html), you should have something like:

```html
<!DOCTYPE html>
<html>
<head>
<title>My Website</title>
</head>
<body>
<p>My name is {{name}} and my favorite hackathon is {{hackathon}}.</p>
</body>
</html>
```
Passing in data

Change the import statement to import the render_template function, like so:

```python
from flask import Flask, render_template
```

Now, change the return statement to return rendered HTML based on your template (index.html), and some data (a name and a hackathon)

```python
return render_template('index.html', name="Saniyah", hackathon="Femmehacks")
```

After running the code, you should see the HTML appear in localhost. You can do this with any kind of data - strings, ints, lists, dictionaries, objects...
Handling POST requests

1. Import request: `from flask import request`

2. Create a new route which will only respond to POST requests:

```python
@app.route('/receiver', methods = ['POST'])
```

3. Define a function (you can pick what to name it)

4. Get data from the POST

```python
username = request.form.get("username")
```

5. Do whatever you want with the data! You can use `render_template` as before to return an HTML file, but will need to add “GET” to the methods.
Sending POST requests

This will be done entirely from your html - here is an example, which should go within the body of your html file:

```html
<form method="post" action="/receiver">
  <input type="text" name="username">
  <button type="submit">Upload</button>
</form>
```
Relevant Links


https://stackoverflow.com/questions/22947905/flask-example-with-post

https://teamtreehouse.com/library/using-forms-for-post-requests