Femmehacks
Intermediate Web Dev

Let’s Build your Website!

Set up a repo

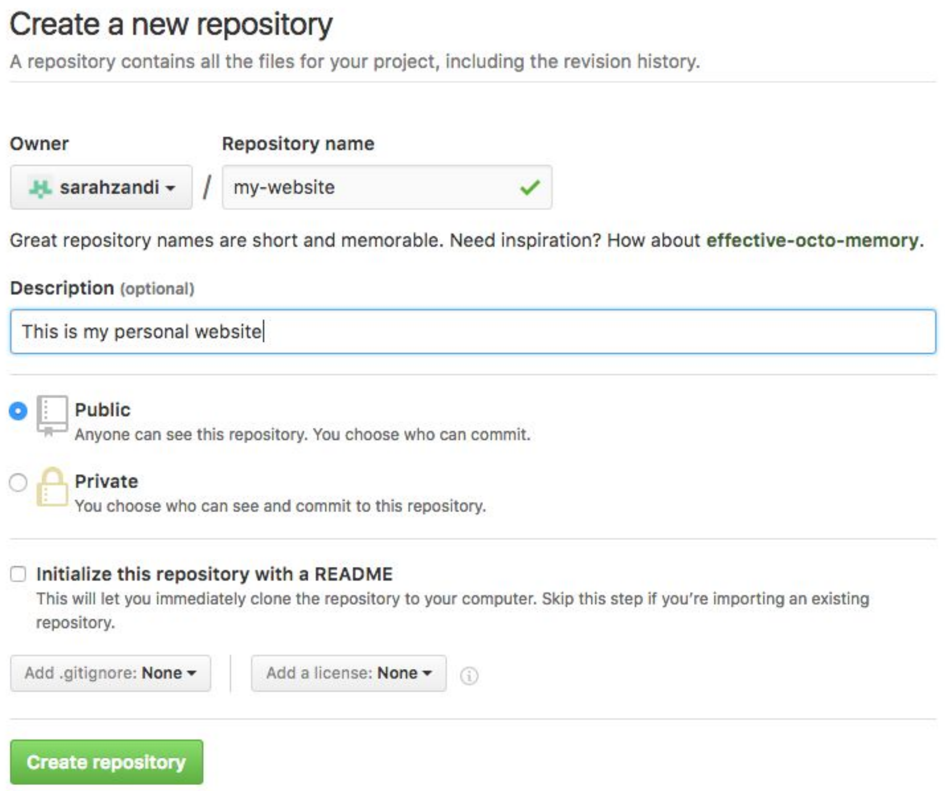
Click “Start a Project” and create your repository

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)



Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: sarahzandi / Repository name: my-website ✓

Great repository names are short and memorable. Need inspiration? How about **effective-octo-memory**.

Description (optional)

This is my personal website

● **Public**
Anyone can see this repository. You choose who can commit.

○ **Private**
You choose who can see and commit to this repository.

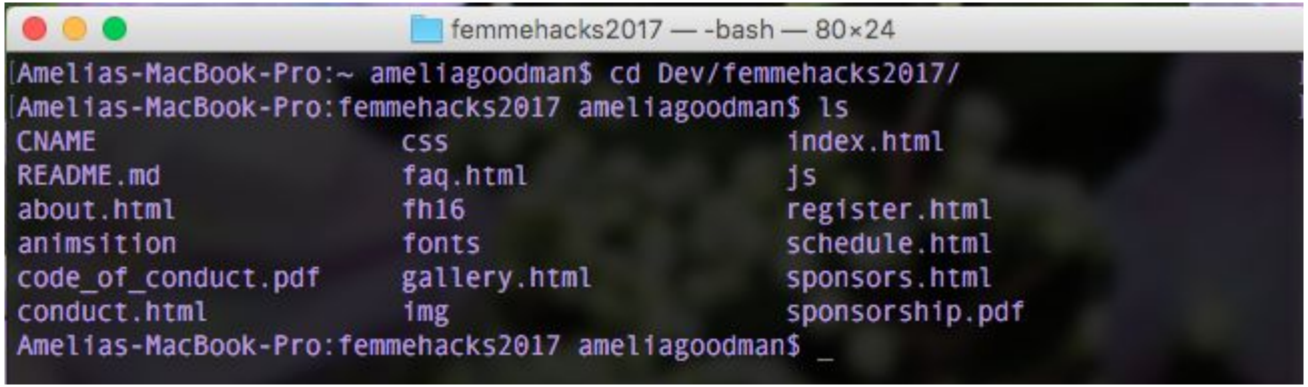
☐ **Initialize this repository with a README**
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** | Add a license: **None**

Open up Terminal

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

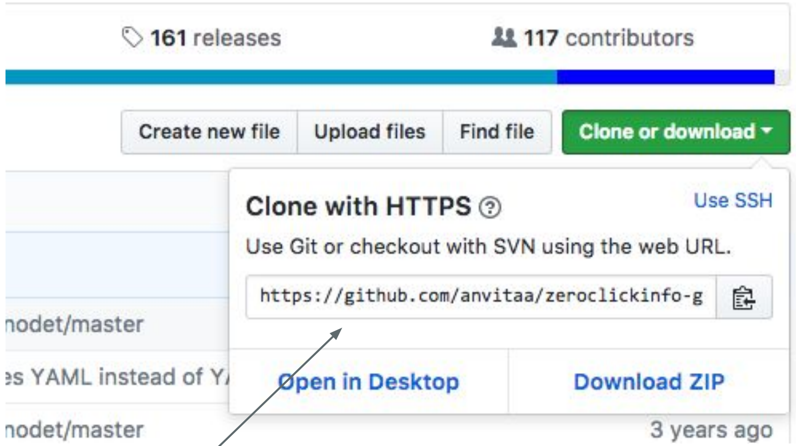
```
cd ~/Desktop
mkdir FemmehacksDemo
cd FemmehacksDemo
ls
```



```
Amelias-MacBook-Pro:~ ameliagoodman$ cd Dev/femmehacks2017/
Amelias-MacBook-Pro:femmehacks2017 ameliagoodman$ ls
CNAME                   css                     index.html
README.md               faq.html                js
about.html              fh16                    register.html
animsition              fonts                   schedule.html
code_of_conduct.pdf     gallery.html            sponsors.html
conduct.html            img                     sponsorship.pdf
Amelias-MacBook-Pro:femmehacks2017 ameliagoodman$ _
```

Tell Github where your code is

Copy the url mentioned in your repo



Clone with HTTPS

Use Git or checkout with SVN using the web URL.

```
https://github.com/anvitaa/zeroclickinfo-g
```

```
git clone https://github.com/blahblahblah
```

Pick a text editor

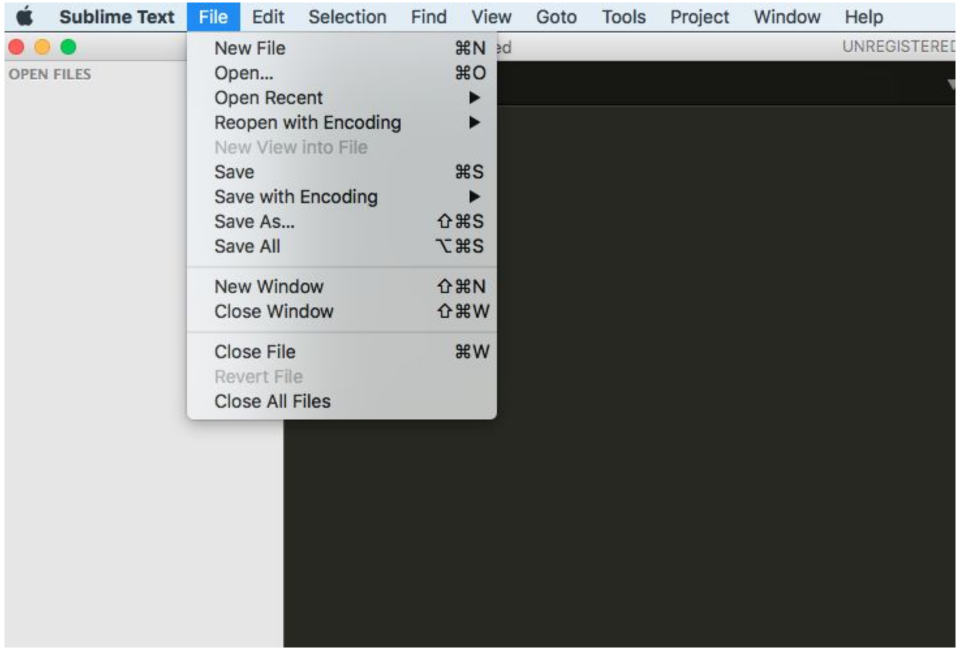
I like Sublime. You can download it at https://www.sublimetext.com/3

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

```
<!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

```
<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

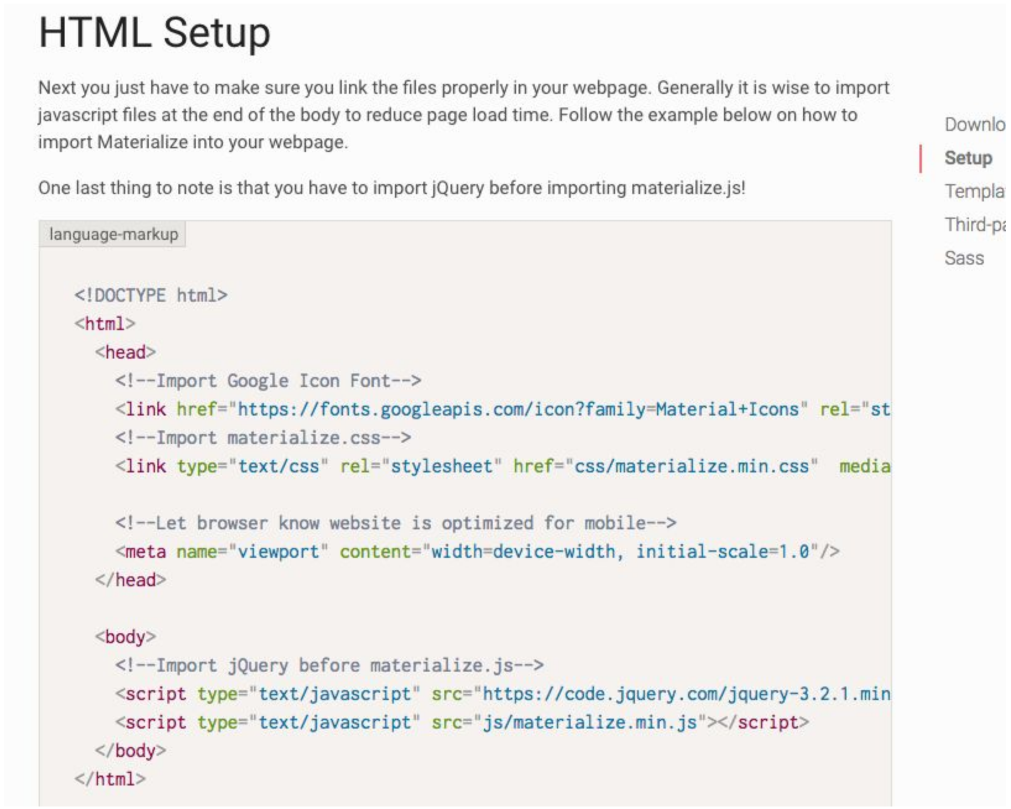
- A whole section of your website
  - ex: about me - all blue
- Solution: use class attribute
  - Multiple elements with shared properties

```
<div id="unique-section">

    <ul>
      <li>Poodle</li>
      <li>Golden Retreiver</li>
    </ul>

</div>

<span class="regular-section">

    <p>Words 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



HTML Setup

Next you just have to make sure you link the files properly in your webpage. Generally it is wise to import javascript files at the end of the body to reduce page load time. Follow the example below on how to import Materialize into your webpage.

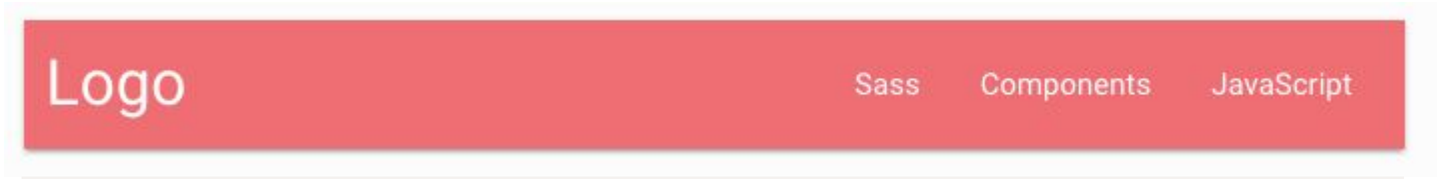
One last thing to note is that you have to import jQuery before importing materialize.js!

language-markup

```
<!DOCTYPE html>
<html>
  <head>
    <!--Import Google Icon Font-->
    <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="st
    <!--Import materialize.css-->
    <link type="text/css" rel="stylesheet" href="css/materialize.min.css"  media

    <!--Let browser know website is optimized for mobile-->
    <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  </head>

  <body>
    <!--Import jQuery before materialize.js-->
    <script type="text/javascript" src="https://code.jquery.com/jquery-3.2.1.min
    <script type="text/javascript" src="js/materialize.min.js"></script>
  </body>
</html>
```

Let’s get a navbar

http://materializecss.com/navbar.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 ...
```

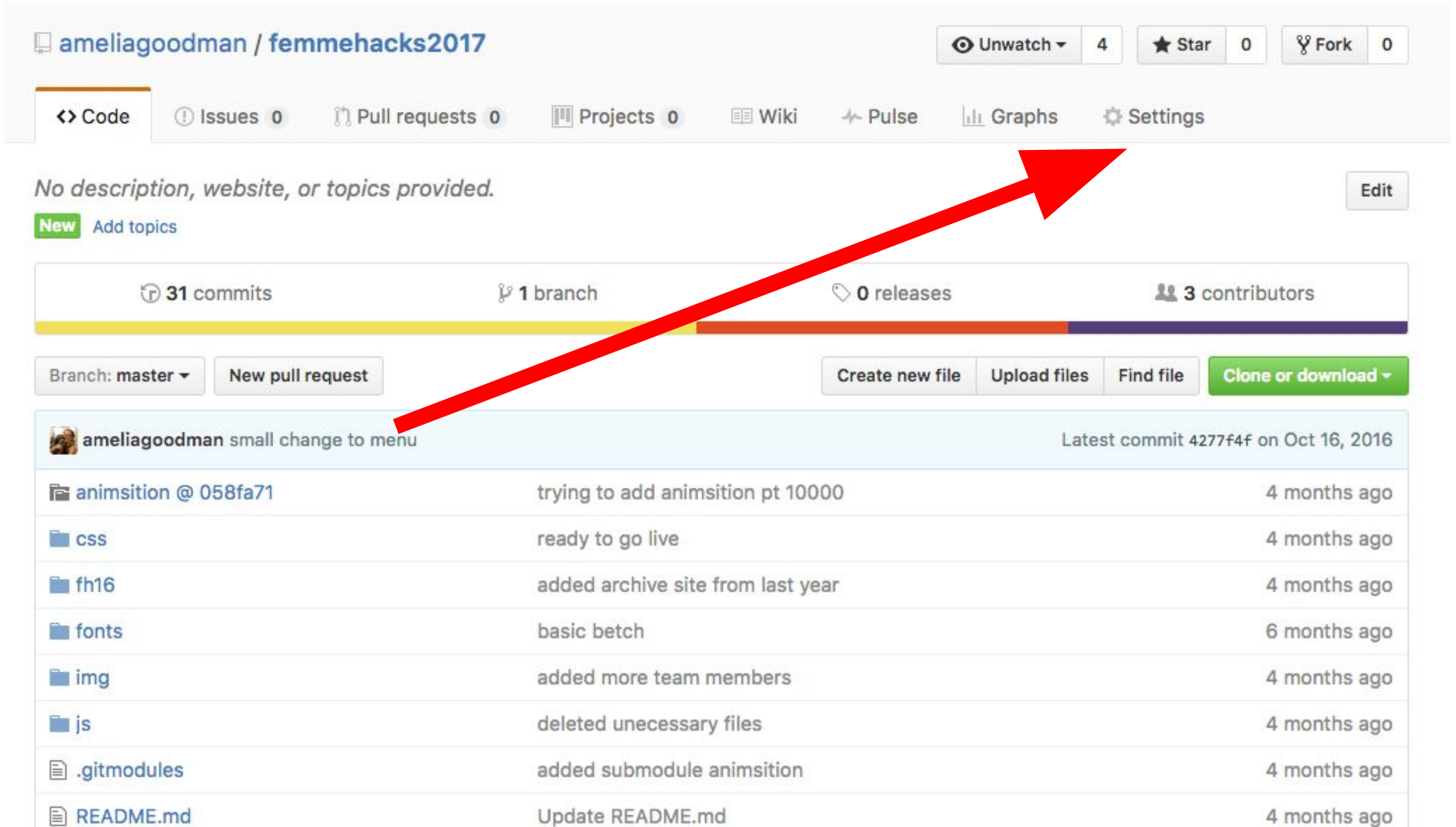
Tell github you are ready to commit the changes

```
git commit -m “changed the readme”
```

Send your changes to the online folder

```
git pull origin master
git push origin master
```

Go to settings



ameliagoodman / femmehacks2017

Unwatch 4 | Star 0 | Fork 0

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Pulse | Graphs | Settings

No description, website, or topics provided. Edit

New Add topics

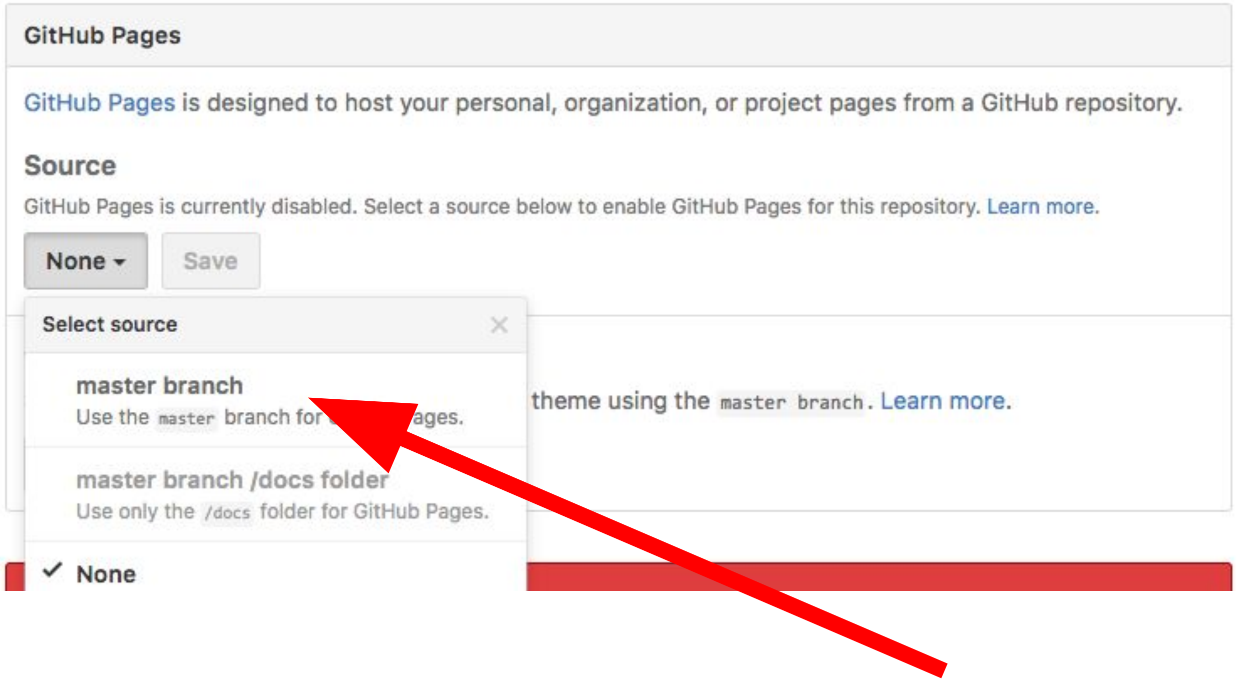
31 commits | 1 branch | 0 releases | 3 contributors

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

ameliagoodman small change to menu — Latest commit 4277f4f on Oct 16, 2016

animsition @ 058fa71	trying to add animsition pt 10000	4 months ago
css	ready to go live	4 months ago
fh16	added archive site from last year	4 months ago
fonts	basic betch	6 months ago
img	added more team members	4 months ago
js	deleted unecessary files	4 months ago
.gitmodules	added submodule animsition	4 months ago
README.md	Update README.md	4 months ago

Enable Github Pages



GitHub Pages

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

Source

GitHub Pages is currently disabled. Select a source below to enable GitHub Pages for this repository. Learn more.

None
Save
Select source
master branch
Use the master branch for ages.
master branch /docs folder
Use only the /docs folder for GitHub Pages.
✓ None
theme using the master branch. Learn more.

By clicking “Master branch” and saving

Your username here

Now go to ______.github.io

Freestyle time!
Use the mentors to help improve your website

Welcome to Flask

Overview

Flask is a microframework for created 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:

```
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:

```
<!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:

```
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)

```
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:

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

3. Define a function (you can pick what to name it)
4. Get data from the POST

```
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:

```
<form method="post" action="/receiver">

  <input type="text" name="username">

  <button type="submit">Upload</button>

</form>
```

Relevant Links

https://galaxydatatech.com/2018/03/31/passing-data-html-page/

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

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