3. FLASK, HTML, TEMPLATING

CS490 – Software Engineering Principles

OBJECTIVES

- Frameworks
  - Flask
- HTML
- Images in HTML
- CSS

ADMINISTRIVIA

- If this is your first lecture, welcome!
  - Please look through lectures 1 and 2 on Canvas
  - Do Diagnostics Exam
- Project 1
  - Submit TBD
  - Use #project1 on Slack for questions!
  - Read the LATE POLICY in the syllabus! Hopefully you don't need it.
- Log into your AWS accounts, launch your environment, and then come back to Zoom to pay attention.

REMINDER!

- Have a question or problem? Search or post in #general. Someone else most likely has it too.
- Have a solution to a problem? Post it in #general. Someone else most likely needs it.

Still stuck? Use Google Search. Still stuck after 15 minutes? Reach out to me with what you tried and what didn't work. I'll help!

This is a senior-level class. We need to practice working in a corporate environment where teachers will not be your source of help and information!!

Why do we care?

- Who cares what the difference between a template and a framework is?
- When are we ever going to use HTML in real life?

Goal Motivations - Because...

In Software Engineering....

- One needs to know what tech options are available when deciding how to build software.
- HTML is a fundamental and universally used language.

In the tech industry...

- Communication is super important, and terminology is a keystone to communication.

For this course...

- You need Flask to finish Project 1, and you will be using a library in Project 2.

2.

FRAMEWORKS/FLASK

Systematic Approaches

This course will cover the following approaches:

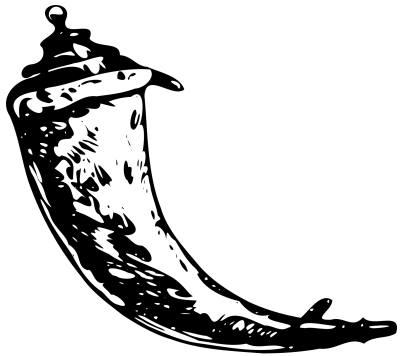
- Choosing a Development Process
- Determining a Programming Language(s)
- Formulating a Toolchain (cluster of technologies)
  - IDE
  - Framework

FRAMEWORK

Code that provides a structure which makes it easier for a developer to create an application (e.g. desktop, mobile, or web app)

Flask - Micro (Web) Framework

- Is a web framework that uses Python
- Helps with routing and fetching, and HTML/CSS for layout
- Almost makes development too easy...
  - "Hello, world" web app is about 10 lines of Python code, and nothing else!!
- Advertises itself as a "micro-framework"
  - The micro- prefix here just means it abstracts over very little (meaning you can decide what to abstract)



Flask

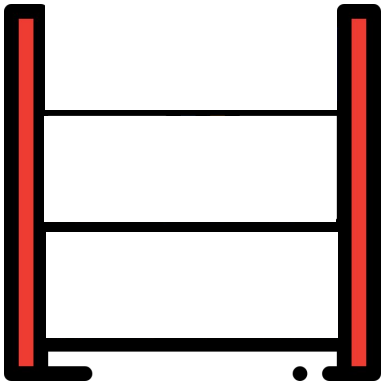
A LIBRARY VS. A FRAMEWORK

- We want this....

A LIBRARY VS. A FRAMEWORK

- Libraries help build using tools

Frameworks help structure



STILL CONFUSED?

- A framework calls into your code.
- Your code calls into a library.

Nonetheless, the distinction between these two terms is still a debate in the tech community.

Case in point...
https://www.quora.com/Whats-the-difference-between-a-library-and-a-framework
https://dev.to/renannobile/why-is-react-a-library-and-not-a-simple-framework-1mle

How do we get a website up and running using Flask on (AWS) Cloud9

?

INSTALLATION

```
vocstartsoft:~/environment/ $ pip install flask #try pip3 if pip does not work!
Collecting flask
  Downloading
https://files.pythonhosted.org/packages/9b/93/628509b8d5dc749656a9641f4caf13540e2cdec8527
6964ff8f43bbb1d3b/Flask-1.1.1-py2.py3-none-any.whl (94kB)
    100% |████████████████████████████████| 102kB 7.8MB/s
Collecting click>=5.1 (from flask)
  Downloading
https://files.pythonhosted.org/packages/fa/37/45185cb5abbc30d7257104c434fe0b07e5a195a6847
506c074527aa599ec/Click-7.0-py2.py3-none-any.whl (81kB)
    100% |████████████████████████████████| 81kB 7.7MB/s
Collecting Werkzeug>=0.15 (from flask)

...

Installed /usr/local/lib/python2.7/dist-packages/MarkupSafe-0.23-py2.7.egg
Finished processing dependencies for Flask
vocstartsoft:~/environment/ $
```

CREATE A DIRECTORY + FILE

```
vocstartsoft:~/environment/ $ mkdir scratch
vocstartsoft:~/environment/ ls
README.md  scratch
vocstartsoft:~/environment/ $ cd scratch/
vocstartsoft:~/environment/scratch $ touch lect3.py
```

USING FLASK FRAMEWORK

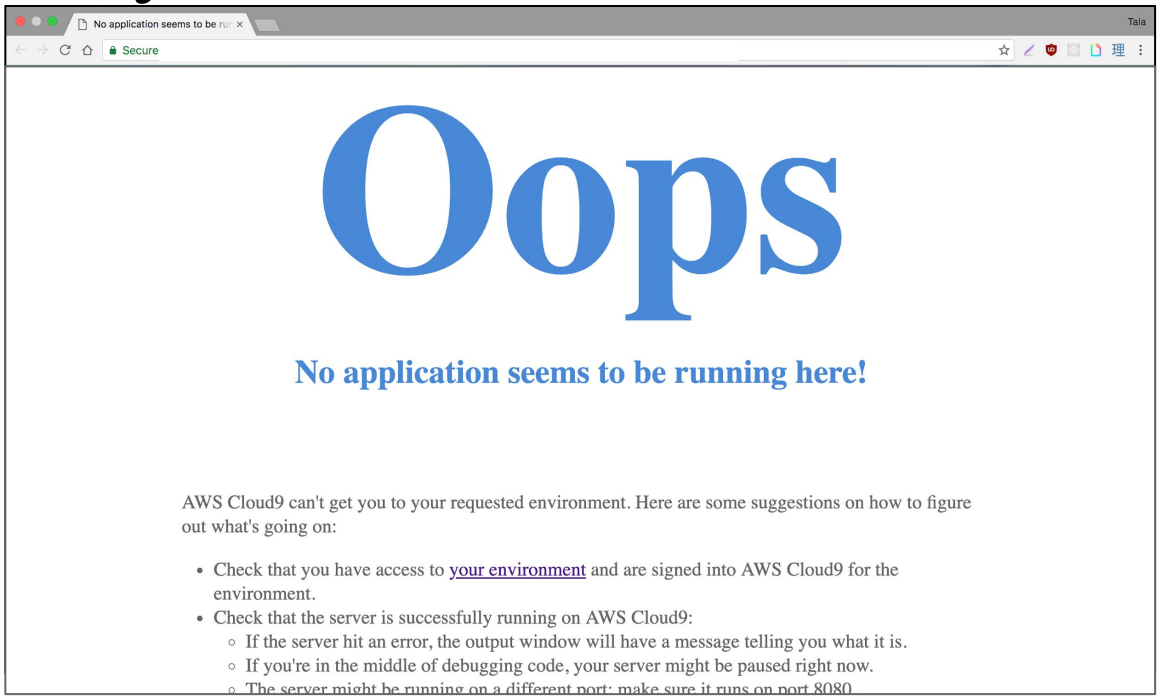
```
# lect3.py
import flask

app = flask.Flask(__name__)

@app.route('/') # Python decorator
def index():
    return "Hello, world!"

app.run()
```

RUN LECT3.PY

```python
# lect3.py
import flask

app = flask.Flask(__name__)

@app.route('/') # Python decorator
def index():
    return "Hello, world!"

app.run()
```

```
vocstartsoft:~/environment/scratch $ python lect3.py
 * Serving Flask app "lect3" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use
it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C
to quit)
```

A FAILURE?

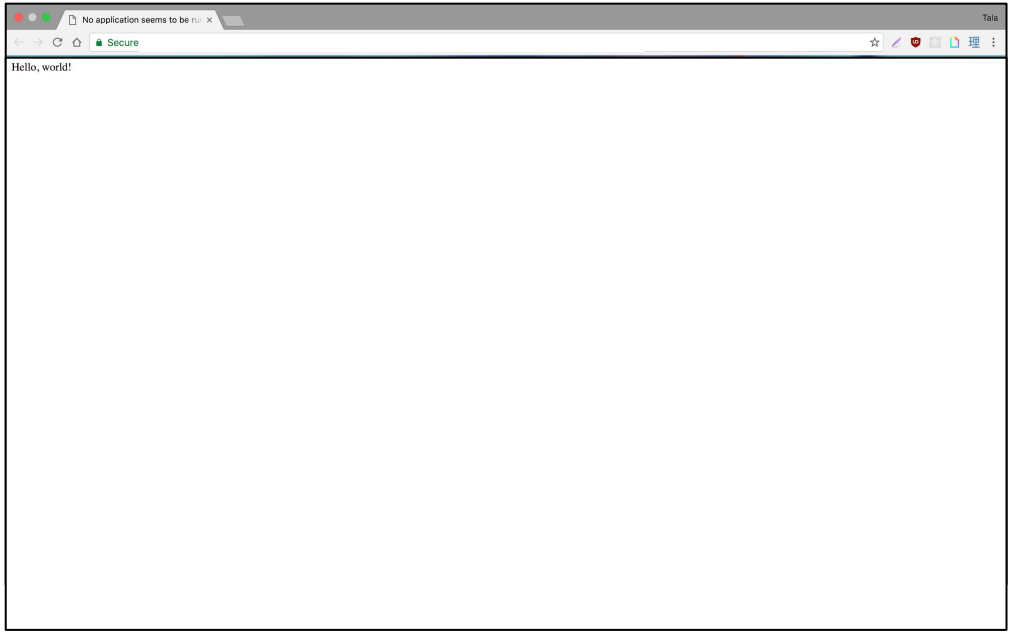

No application seems to be ru
Tala
Secure
Oops
No application seems to be running here!
AWS Cloud9 can't get you to your requested environment. Here are some suggestions on how to figure out what's going on:
Check that you have access to your environment and are signed into AWS Cloud9 for the environment.
Check that the server is successfully running on AWS Cloud9:
If the server hit an error, the output window will have a message telling you what it is.
If you're in the middle of debugging code, your server might be paused right now.
The server might be running on a different port; make sure it runs on port 8080

RUN THE APP ON PORT 8080

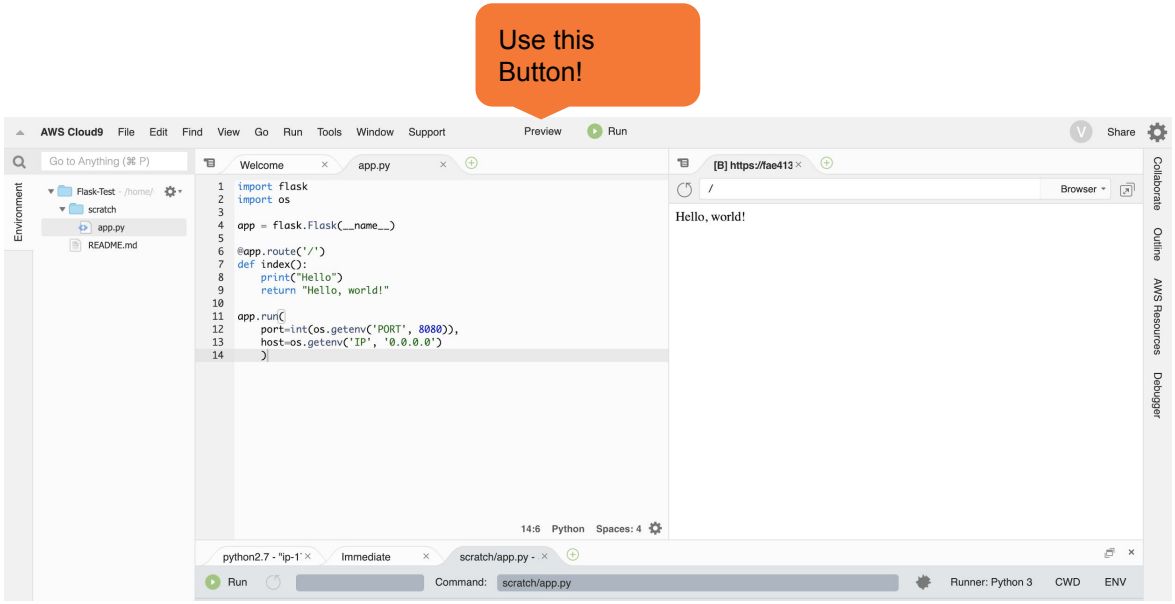
```python
# lect3.py
import flask
import os

app = flask.Flask(__name__)

@app.route('/')
def index():
    return "Hello, world!"

app.run(
    port=int(os.getenv('PORT', 8080)),
    host=os.getenv('IP', '0.0.0.0')
)
```

```
vocstartsoft:~/environment/scratch $ python lect3.py
 * Serving Flask app "lect3" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use
it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:8080/ (Press CTRL+C to
quit)
vocstartsoft:~/environment/scratch $
```

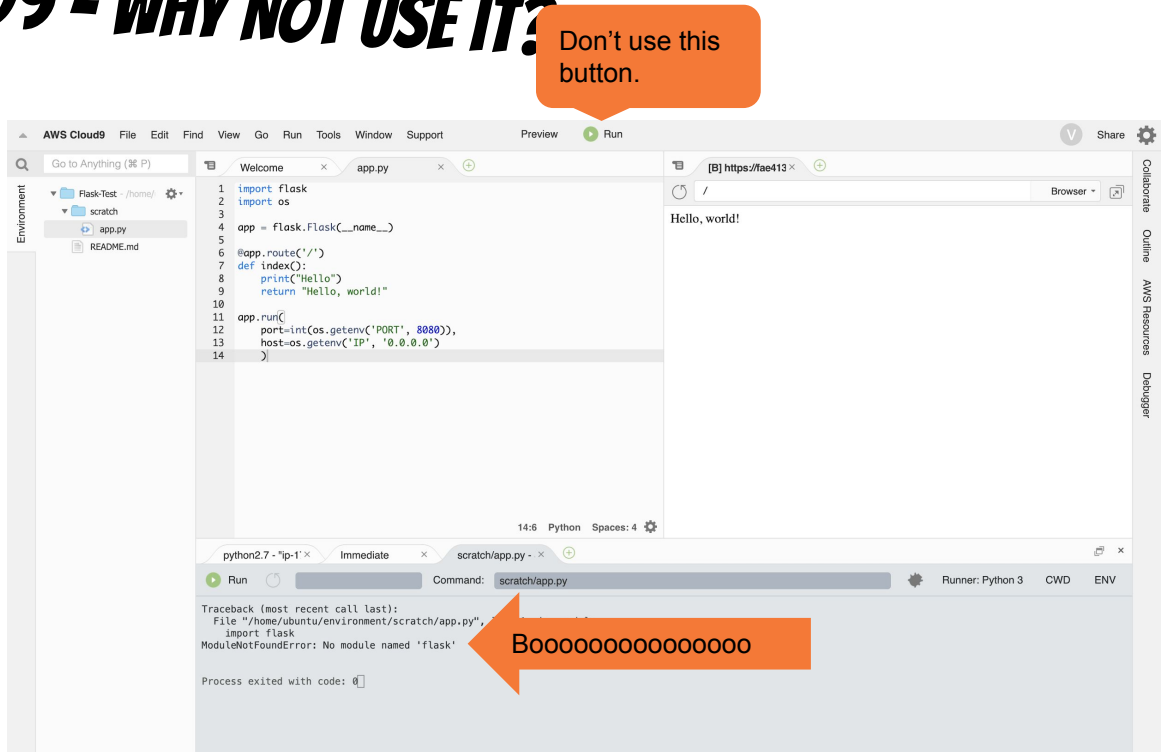
HELLO, WORLD! 😊



CLOUD9 - WHY USE IT?



Cloud9 - Why not use it?



ADD IN A PRINT STATEMENT

```python
# lect3.py
import flask
import os

app = flask.Flask(__name__)

@app.route('/') # "Python decorator"
def index():
    print("This is a debug statement!")
    return "Hello, world!"

app.run(
    port=int(os.getenv('PORT', 8080)),
    host=os.getenv('IP', '0.0.0.0')
)
```

```
vocstartsoft:~/environment/scratch $ python lect3.py
 * Serving Flask app "lect3" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use
it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:8080/ (Press CTRL+C
to quit)
This is a debug statement!
10.240.0.179 - - [09/Aug/2019 03:51:34] "GET /
HTTP/1.1" 200 -
vocstartsoft:~/environment/scratch $
```

PROTIP: 'DEBUG = TRUE' SAVES LIVES. USE IT!

```python
# lect3.py
import flask
import os

app = flask.Flask(__name__)

@app.route('/') # "Python decorator"
def index():
    print("This is a debug statement!")
    return "Hello, world!"

app.run(
    port=int(os.getenv('PORT', 8080)),
    host=os.getenv('IP', '0.0.0.0'),
    debug=True
)
```

```
vocstartsoft:~/environment/scratch $ python lect3.py
 * Serving Flask app "lect3" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use
it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:8080/ (Press CTRL+C
to quit)
This is a debug statement!
10.240.0.179 - - [09/Aug/2019 03:51:34] "GET /
HTTP/1.1" 200 -
vocstartsoft:~/environment/scratch $
```

OH-LEH-DO-IT (10 MINUTES)

Head to canvas, and carefully follow the directions in the 3.1 assignment.

3.

Flask... and HTML

Systematic Approaches

This course will cover the following approaches:

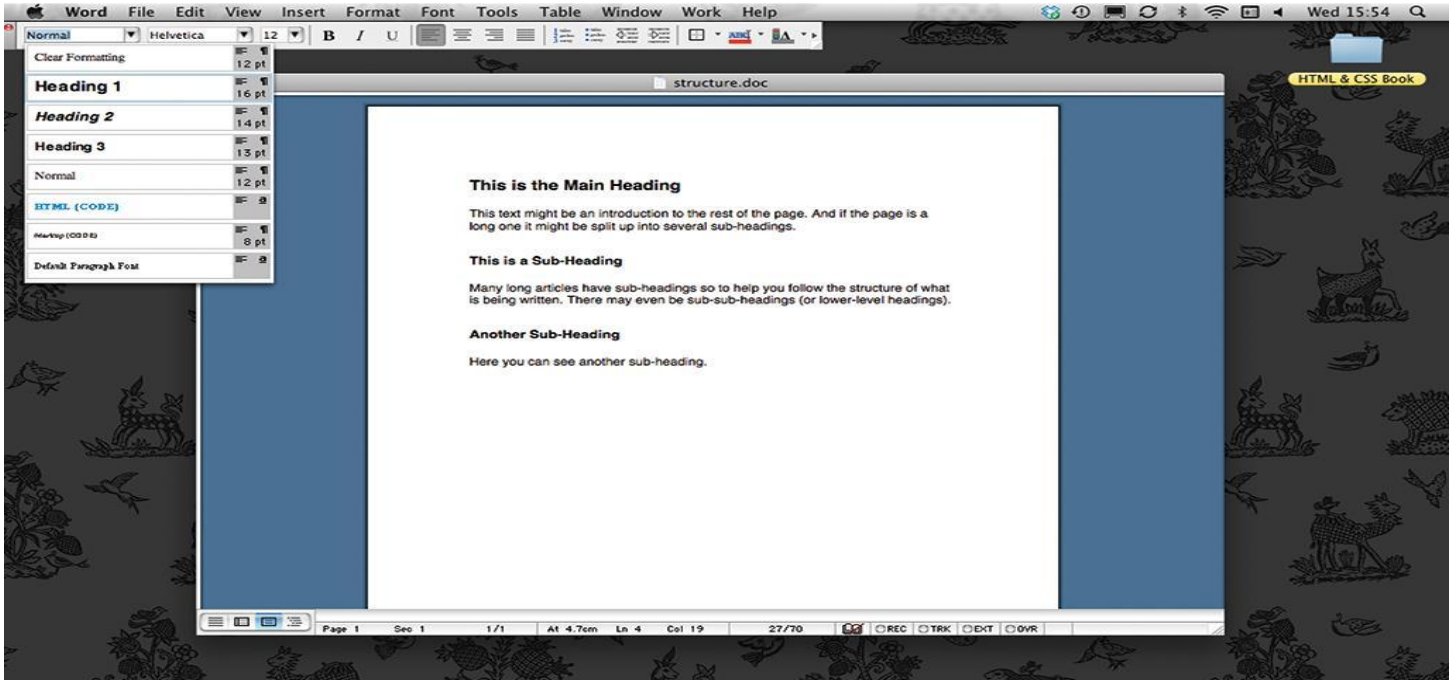
- Choosing a Development Process
- Determining a Programming Language(s)
- Formulating a Toolchain (cluster of technologies)
  - IDE
  - Framework
  - HTML

HTML

or hypertext markup language, is a markup (not programming) language that describes the type of content being displayed on a webpage.

This content can be a paragraph, heading, section, list, image and so on...

STRUCTURE IN WORD DOCS



HTML: PAGE STRUCTURE

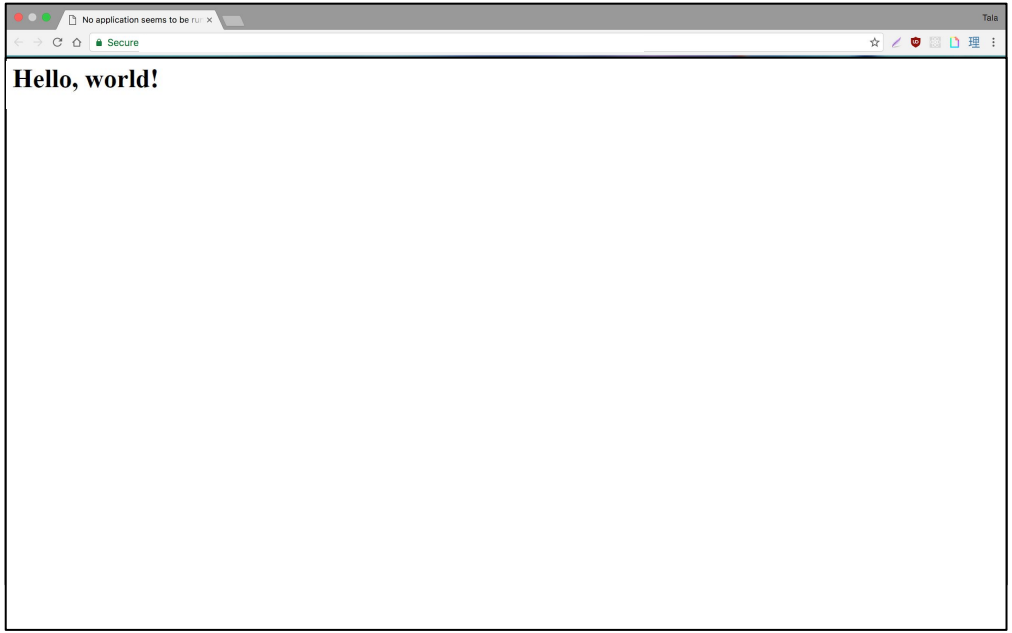
```
<html>
  <body>
    <h1>This is the Main Heading</h1>
    <p>This text might be an introduction to
       the rest of the page.</p>
    <h2>This is a Sub-Heading</h2>
    <p>Many long articles have sub-headings
       to help you follow the structure.</p>
    <h2>Another Sub-Heading</h2>
    <p>Here you can see another.</p>
  </body>
</html>
```

USING HTML

```python
# lect3.py
import flask
import os

app = flask.Flask(__name__)

@app.route("/")
def index():
    print("This is a debug statement!")
    return "<h1>Hello, world!</h1>"

app.run(
    port=int(os.getenv("PORT", 8080)),
    host=os.getenv("IP", "0.0.0.0")
)
```

```
vocstartsoft:~/environment/scratch $ python lect3.py
 * Serving Flask app "lect3" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use
it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:8080/ (Press CTRL+C
to quit)
This is a debug statement!
10.240.0.179 - - [09/Aug/2019 03:51:34] "GET /
HTTP/1.1" 200 -
vocstartsoft:~/environment/scratch $
```

Hello, world! 😊 (as a header)



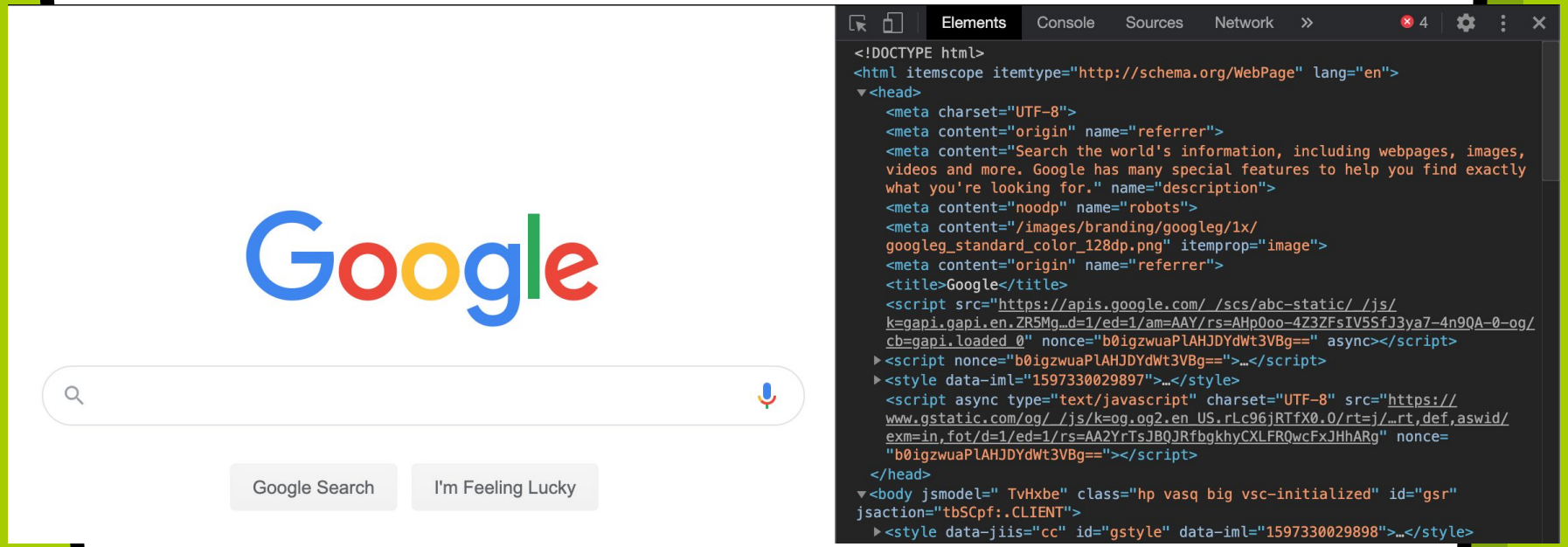
Practice (10 minutes)

Write a web app named randhtml.py that, on every load, displays a random number in an HTML header tag.

4.

TEMPLATES

GOOGLE.COM: AS SIMPLE AS RAND.PY?



THIS DOESN'T SEEM EFFICIENT...

```
import flask
import random
import os

app = flask.Flask(__name__)

@app.route('/')
def index():
    return '<html itemscope="" itemtype="http://schema.org/WebPage" lang="en"><head><meta
content="Search the world's information, including webpages, images, videos and more. Google has many
special features to help you find exactly what you're looking for." name="description"><meta
content="noodp" name="robots"><meta
content="/logos/doodles/2017/bessie-colemans-125th-birthday-5751652702224384-hp.gif"
itemprop="image"><link href="/images/branding/product/ico/googleg_lodp.ico" rel="shortcut icon"><meta
content="Bessie Coleman’s 125th birthday! #GoogleDoodle" property="og:description">...'

app.run(
    port=int(os.getenv('PORT', 8080)),
    host=os.getenv('IP', '0.0.0.0')
)
```

TEMPLATES FTW!!!

- Templates enable you to put your HTML into nicely formatted files
  - No need to stuff all HTML into a crazy string
- Flask has a built-in templating language called Jinja2
  - This will be relevant when you Google for documentation!
- Instead of putting HTML in your python file, just tell Flask to render a template

The Power of Micro-Frameworks

Remember how we discussed that Flask is a micro-framework? As in, it abstracts over very little (AKA gives the user control over what to abstract}?

Jinja2 is a great example of this abstraction.

Flask allows a user to determine how they want to render HTML, whether it be manually or using a template engine such as Jinja2, Mako, etc.

With that, let's get started and make our app with templates!

CREATING OUR TEMPLATES FOLDER

```
vocstartsoft:~/environment $ pwd # displays the full path
~/environment
vocstartsoft:~/environment $ cd lect3 # navigate to lect3 directory
vocstartsoft:~/environment/lect3 $ mkdir templates # create a new directory called templates
vocstartsoft:~/environment/lect3 $ ls # shows what files/dirs exist in a directory
app.py  templates/
vocstartsoft:~/environment/lect3 $ cd templates
vocstartsoft:~/environment/lect3/templates $ touch index.html # create a file called index.html
vocstartsoft:~/environment/lect3/templates $ ls
index.html
vocstartsoft:~/environment/lect3/templates $ cd .. # go up one directory
vocstartsoft:~/environment/lect3/ $
```

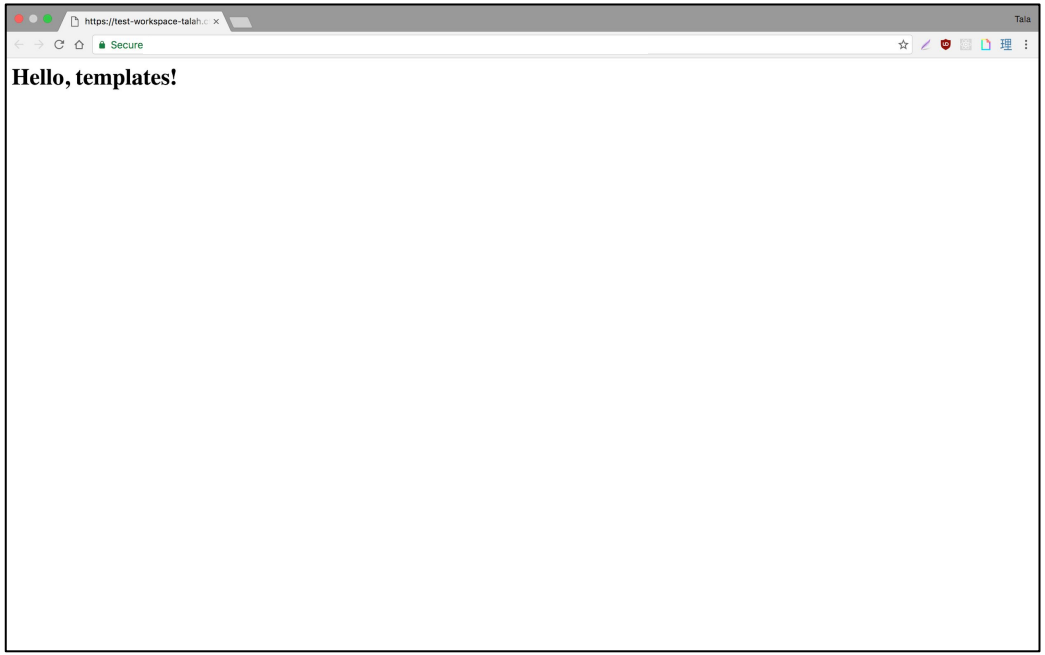
APP.PY CAN STAY SIMPLE...

```python
# app.py
import flask
import os

app = flask.Flask(__name__)

@app.route('/')
def index():
    return flask.render_template("index.html")

app.run(
    port=int(os.getenv('PORT', 8080)),
    host=os.getenv('IP', '0.0.0.0'),
    debug=True
)
```

```html
<!-- templates/index.html -->
<h1>Hello, templates!</h1>
```

This is the important difference!

SUCCESS 😎

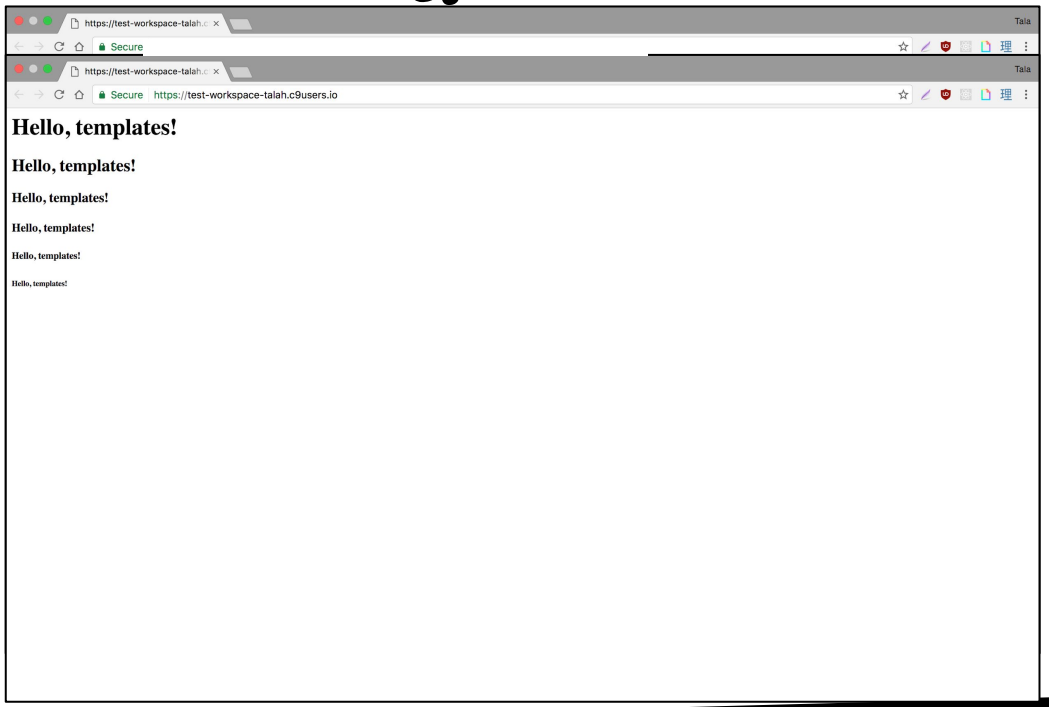


... WHILE INDEX.HTML GROWS

```python
# app.py
import flask
import os

app = flask.Flask(__name__)

@app.route('/')
def index():
    return flask.render_template("index.html")

app.run(
    port=int(os.getenv('PORT', 8080)),
    host=os.getenv('IP', '0.0.0.0'),
    debug=True
)
```

```html
<!-- templates/index.html -->
<h1>Hello, templates!</h1>
<h2>Hello, templates!</h2>
<h3>Hello, templates!</h3>
<h4>Hello, templates!</h4>
<h5>Hello, templates!</h5>
<h6>Hello, templates!</h6>
```

WHILE THE CODE GROWS!



Flask as a framework

Remember when we said that a framework calls the code when it's ready?

Well, we told Flask what to render (and from where) with the line :

```
flask.render_template("index.html")
```

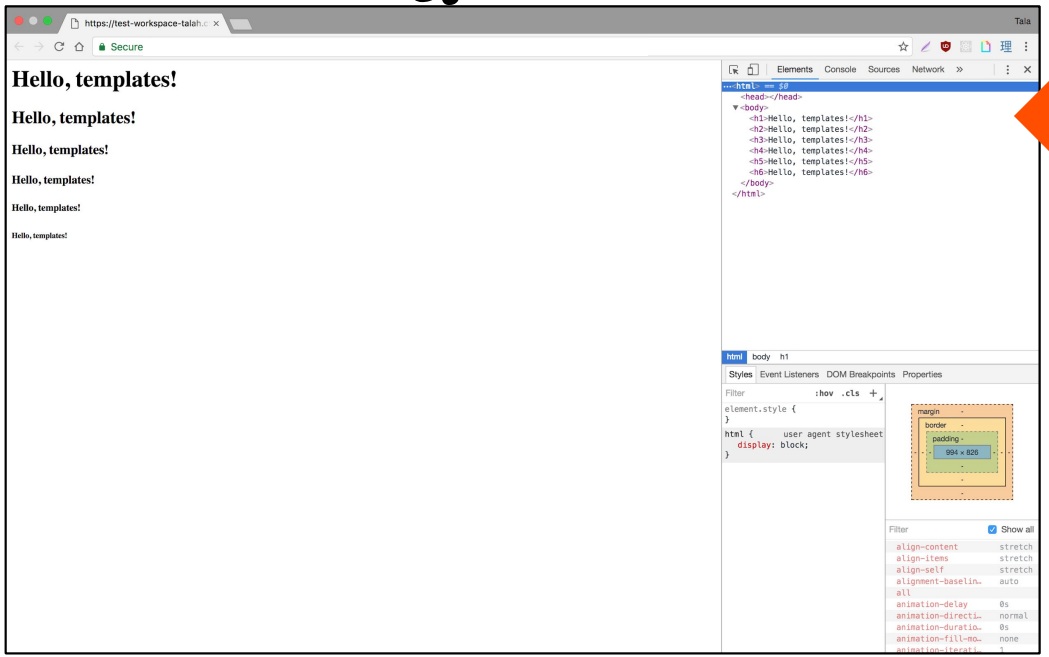
... and Flask automagically ran our code when it was ready!

"BAD" HTML

```python
# app.py
import flask
import os

app = flask.Flask(__name__)

@app.route('/')
def index():
    return flask.render_template("index.html")

app.run(
    port=int(os.getenv('PORT', 8080)),
    host=os.getenv('IP', '0.0.0.0'),
    debug=True
)
```

```html
<!-- templates/index.html -->
<h1>Hello, templates!</h1>
<h2>Hello, templates!</h2>
<h3>Hello, templates!</h3>
<h4>Hello, templates!</h4>
<h5>Hello, templates!</h5>
<h6>Hello, templates!</h6>
```

WHILE THE CODE GROWS!



GOOD (ENOUGH) HTML

```python
# app.py
import flask
import os

app = flask.Flask(__name__)

@app.route('/')
def index():
    return flask.render_template("index.html")

app.run(
    port=int(os.getenv('PORT', 8080)),
    host=os.getenv('IP', '0.0.0.0'),
    debug=True
)
```

```html
<!-- templates/index.html -->
<html>
  <head>
  </head>
  <body>
    <h1>Hello, templates!</h1>
    <h2>Hello, templates!</h2>
    <h3>Hello, templates!</h3>
    <h4>Hello, templates!</h4>
    <h5>Hello, templates!</h5>
    <h6>Hello, templates!</h6>
  </body>
</html>
```

OH-LEH-DO-IT

Head to canvas, and carefully follow the directions in the 4.1 assignment. You will be reproducing what you've just learned.

We will continue lecture at 10:30a (even if you don't finish, which is okay!)

How do we get a Python variable to show up in our HTML file

?

REMEMBER THIS DO-NOW?

```python
# rand.py
import flask
import random
import os

app = flask.Flask(__name__)

@app.route('/random')
def index():
    r = random.randint(1, 20)
    return '<h1>' + str(r) + '</h1>'

app.run(
    port=int(os.getenv('PORT', 8080)),
    host=os.getenv('IP', '0.0.0.0'),
    debug=True
)
```

PYTHON AND HTML

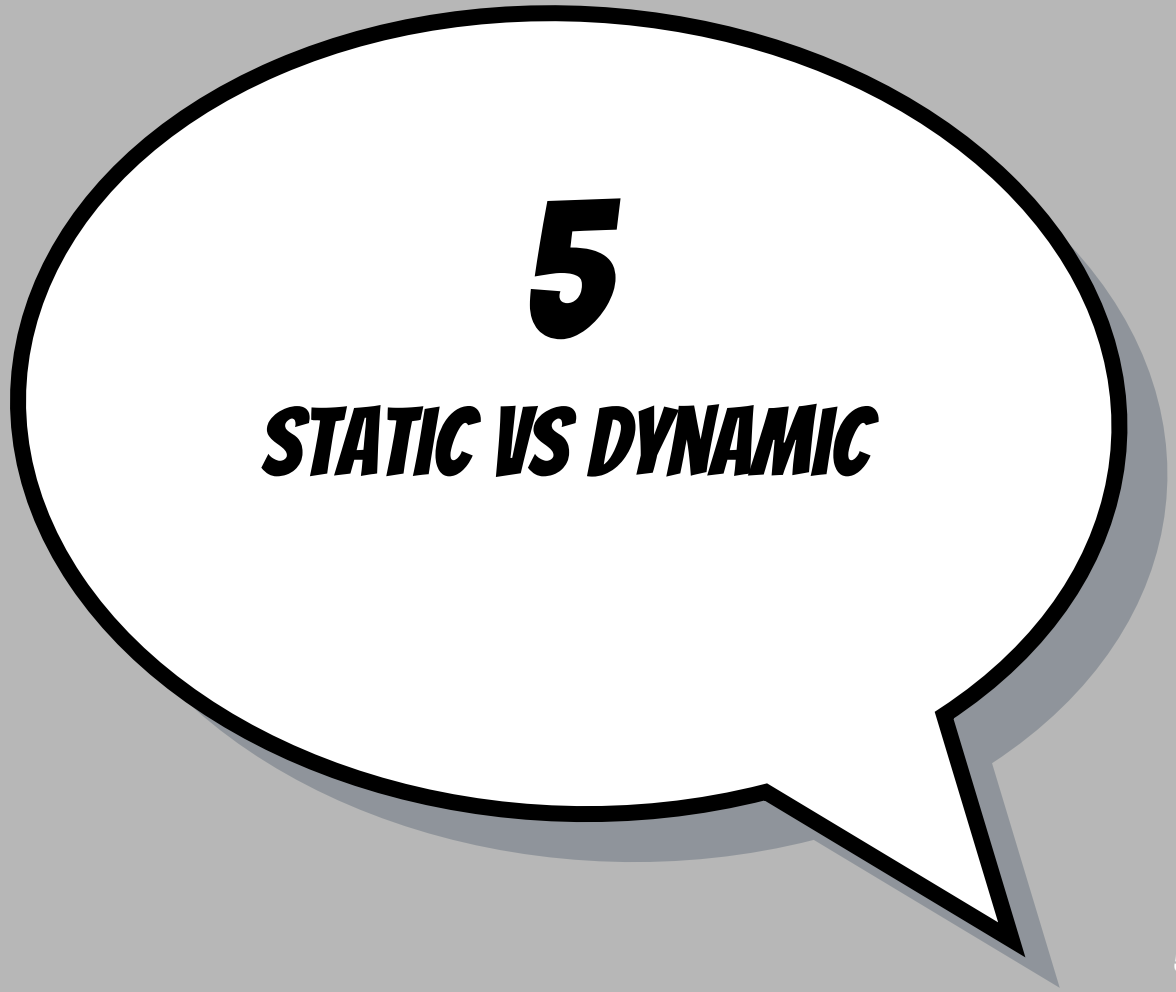
```python
# app.py
import flask, random, os

app = flask.Flask(__name__)

@app.route('/')  # we'll use the default page
def index():
    r = random.randint(1, 20)
    return flask.render_template(
        "index.html",
        random_num=r # if this is confusing, look up 'python kwaargs'
    )

app.run(
    port=int(os.getenv('PORT', 8080)),
    host=os.getenv('IP', '0.0.0.0'),
    debug=True
}
```

```html
<!-- templates/index.html -->
<html>
  <head>
  </head>
  <body>
    <h1>{{ random_num }}</h1>
  </body>
</html>
```

Python and HTML (Multiple Variables)

```python
# app.py
import flask, random, os

app = flask.Flask(__name__)

@app.route('/')  # we'll use the default page
def index():
    num_one=random.randint(1, 20)
    num_two=random.randint(1, 20)
    return flask.render_template(
        "index.html",
        random_num_one=num_one,
        random_num_two=num_two
    )

app.run(
    port=int(os.getenv('PORT', 8080)),
    host=os.getenv('IP', '0.0.0.0'),
    debug=True
)
```

```html
<!-- templates/index.html -->
<html>
  <head>
  </head>
  <body>
    <h1>{{ random_num_one }}
and {{ random_num_two }} </h1>
  </body>
</html>
```


5
STATIC VS DYNAMIC

Static versus Dynamic Web Pages

- Static web pages are provided by the server as originally stored. No additional processing is done.
  - For us, this happens when your view function is just render_template without any additional parameters.
  - "hello, templates!" page was a static web page
- Dynamic web pages are provided by the server after processing is done to create a response.
  - This can include server fetching data from a database, talking to other services, or even generating random numbers
  - Our random number page was a dynamic web page

Static versus Dynamic Web Pages

- A dynamic web page is not related to an "interactive" webpage
  - 'Dynamic' does not say anything about the client!!

- It's possible to make interactive web apps that make the client do all the heavy lifting: http://madebyevan.com/webgl-water/

6.

HTML - IMAGES

RESOURCES

Different types of data that are served in addition to a web page.

*For example: PDFs, images, songs, etc.

STATIC RESOURCES

Different types of data that are provided by the server as they're stored.

STATIC IMAGES

are images that are provided by the server as they're stored.

Now, let's add a static image to our app!

CREATE AND NAVIGATE TO THE STATIC FOLDER

```
vocstartsoft:~/environment/ $ cd lect3
vocstartsoft:~/environment/lect3 $ ls
app.py  templates/
vocstartsoft:~/environment/lect3 $ mkdir static # remember this creates a directory called
static
vocstartsoft:~/environment/lect3 $ ls
app.py  static/  templates/
vocstartsoft:~/environment/lect3 $ cd static
vocstartsoft:~/environment/lect3/static $
```

STATIC IMAGES

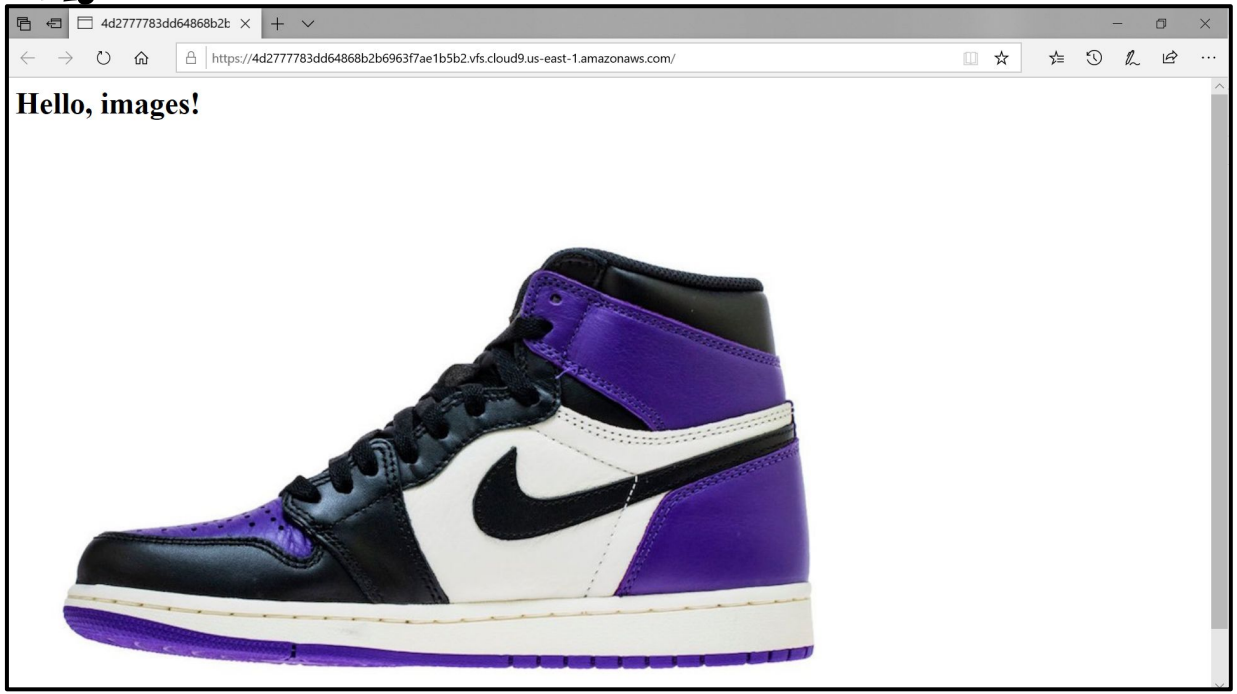
```
vocstartsoft:~/environment/lect3/static $ wget
https://www.kicksonfire.com/wp-content/uploads/2018/09/AIR-JORDAN-1-3-1.jpg # downloads a
picture from a URL
--2019-08-25 21:49:49--
https://www.kicksonfire.com/wp-content/uploads/2018/09/AIR-JORDAN-1-3-1.jpg
Resolving www.kicksonfire.com (www.kicksonfire.com)... 151.139.244.25
Connecting to www.kicksonfire.com (www.kicksonfire.com)|151.139.244.25|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 62965 (61K) [image/jpeg]
Saving to: ‘AIR-JORDAN-1-3-1.jpg’

AIR-JORDAN-1-3-1.jpg
100%[=====================================================================>]  61.49K
--.-KB/s    in 0s

2019-08-25 21:49:52 (307 MB/s) - ‘AIR-JORDAN-1-3-1.jpg’ saved [62965/62965]

vocstartsoft:~/environment/lect4/static $ ls
AIR-JORDAN-1-3-1.jpg
```

STATIC IMAGES

```python
# app.py
import flask
import os

app = flask.Flask(__name__)

@app.route('/')
def index():
    return flask.render_template("index.html")

app.run(
    port=int(os.getenv('PORT', 8080)),
    host=os.getenv('IP', '0.0.0.0'),
    debug=True
)
```

```html
<!-- templates/index.html -->
<html>
  <head>
  </head>
  <body>
    <h1>Hello, images!</h1>
    <img src="/static/AIR-JORDAN-1-3-1.jpg" />
  </body>
</html>
```

VOILA!


4d2777783dd64868b2b
https://4d2777783dd64868b2b6963f7ae1b5b2.vfs.cloud9.us-east-1.amazonaws.com/
Hello, images!

THIS EQUALS THAT

```
<!-- templates/index.html -->
<html>
  <head>
  </head>
  <body>
    <h1>Hello, images!</h1>
    <img src= "https://www.kicksonfire.com/
wp-content/uploads/2018/09/AIR-JORDAN-1-3-1
.jpg"/>
  </body>
</html>
```

```
<!-- templates/index.html -->
<html>
  <head>
  </head>
  <body>
    <h1>Hello, images!</h1>
    <img src="/static/AIR-JORDAN-1-3-1.jpg" />
  </body>
</html>
```

HTML: PASSING THE TORCH

- I'm not going to teach you how to use HTML, so you're going to have to do research on your own.
- But HTML is pretty simple to pick up, here are some resources:
  - http://www.w3schools.com/html/
  - http://html.com/
- Post on Slack or come to Office Hours if you need help!

PRACTICE TEMPLATES

Head to canvas, and carefully follow the directions and start on 3.2 assignment (if you've finished 3.1). You will be reproducing what you've just learned, adding an image!