

Flask

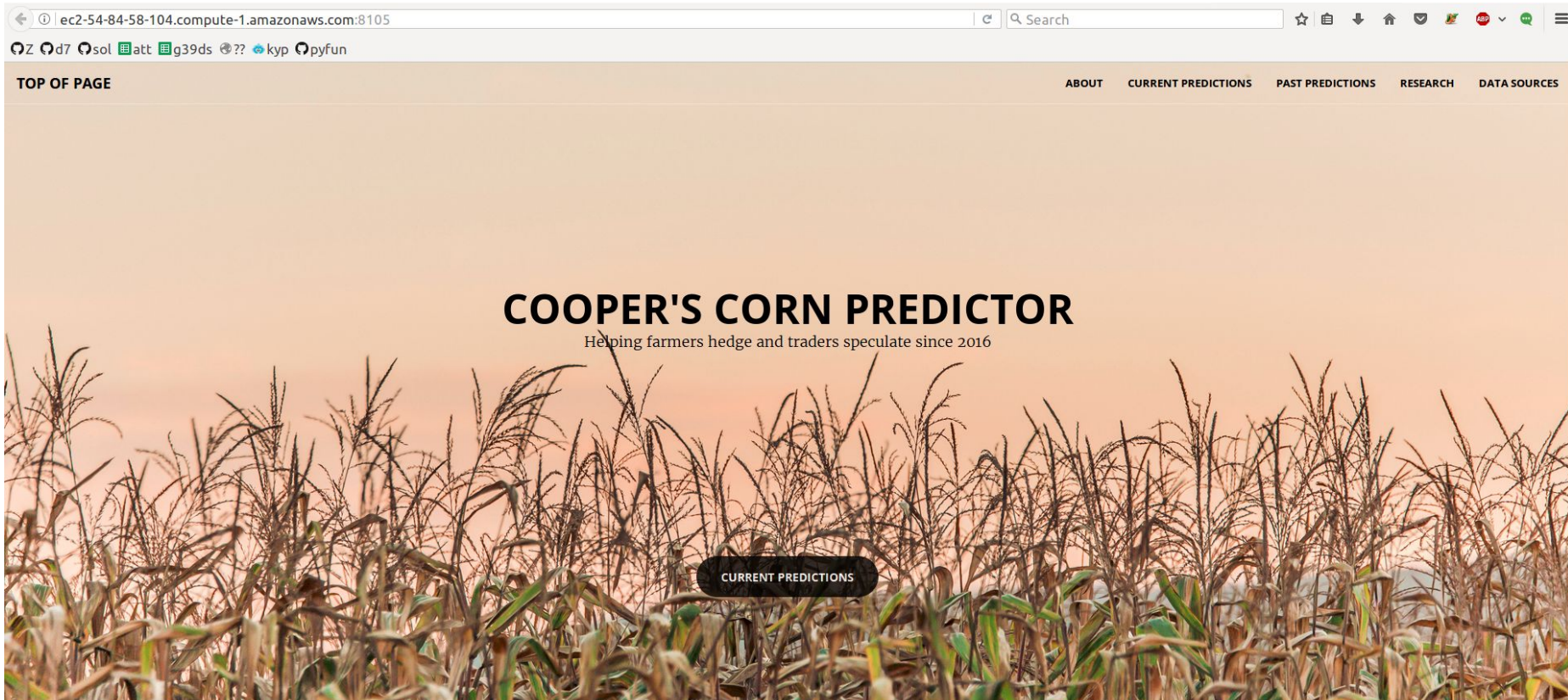
(i.e. how we do Web Development in a day in the DSI)

One website made with Flask



A DSI capstone website made with Flask

cooperscorn.com



https://github.com/cstainbrook/Corn-Futures-Capstone/tree/master/web_app

Objectives

- Review World-Wide-Web (www) and Uniform Resource Locator(URL)
- Review client-server relationship and GET and POST requests
- Review web browsers, HTML and CSS
- Introduce Flask
- Structure of a Flask application
- Use of templates (Jinja2)
- Resources

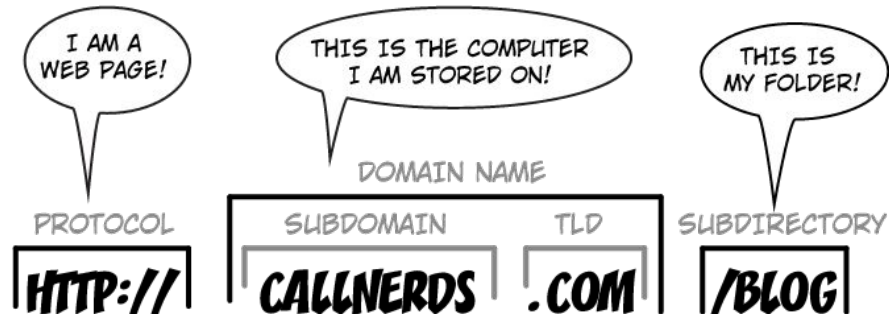
The WWW, and finding information with the URL

The World Wide Web (WWW) is an information space where documents are identified by Uniform Resource Locators (URLs), interlinked by hypertext links, and can be accessed via the Internet.

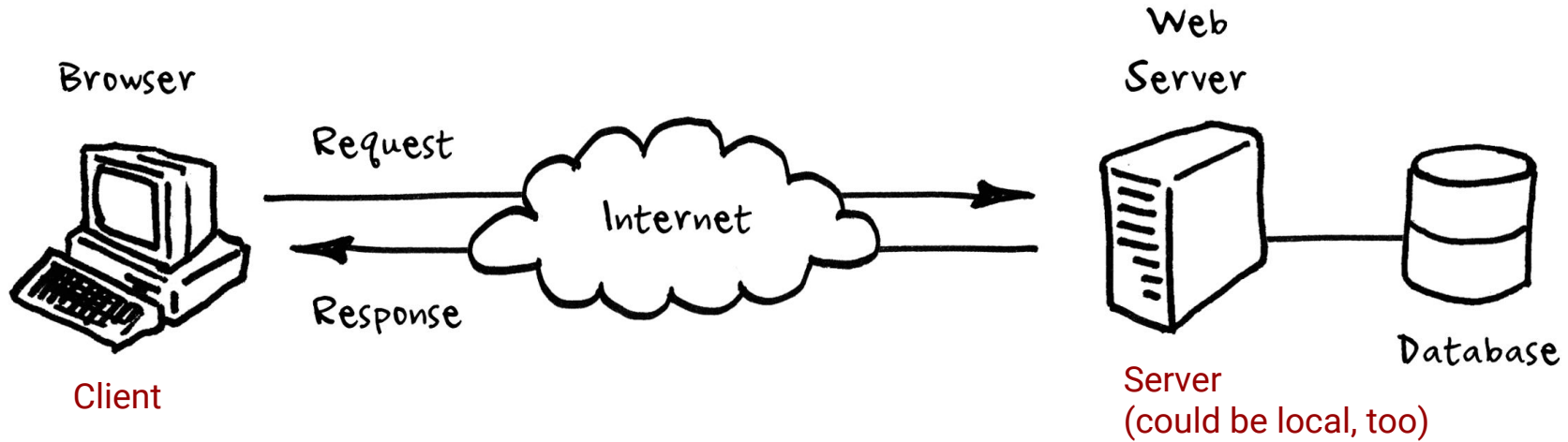
Information is stored on web servers all over the world.



A Uniform Resource Locator (URL), (a.k.a. web address) is the location of a web resource on a computer network and the mechanism for retrieving it.



Client-server relationship and GET and POST requests



An HTTP GET request method retrieves information from the server. Some data can be passed within the URL's query string, specifying (for example) search terms, date ranges, etc.

In a POST request, an arbitrary amount of data of any type can be sent to the server in the body of the request message.

-- Wikipedia

Confidential and/or lengthy information should be sent using POST.

https://developer.mozilla.org/en-US/docs/Learn/HTML/Forms/Sending_and_retrieving_form_data

Web Browser

A web browser sends, requests, receives, and renders data on the internet and WWW.

Typical steps are:

- A user inputs a URL into the address bar
- A GET request is sent to the appropriate web server
- The web server responds with HTML (Hypertext Markup Language) and associated content (CSS - Cascading Style Sheets, images, video, etc.)
- The browser's layout engine renders the content (including HTML and CSS)

HTML (Hypertext Markup Language)

HTML is written using **tags**, where the tags describe content:

Tag	Description
<h1> - <h6>	Heading
<p>	Paragraph
<i>	Italic
	Bold
<a>	Anchor (links)
 & 	Unordered List & List Item
<blockquote>	Blockquote
	Image
<div>	Division

Example HTML:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Simple HTML document</title>
</head>
<body>
  <h1>Hello World!</h1>
</body>
</html>
```

Other examples:

<http://www.tutorialrepublic.com/html-examples.php>

CSS (Cascading Style Sheets)

Hard-coding formatting information into HTML is unreasonable for large websites. Formatting is placed in CSS. The HTML document references the CSS document, and uses classes and IDs defined in that document.

Demo:

https://www.w3schools.com/css/css_intro.asp

HTML file

```
<!DOCTYPE html>
<link rel="stylesheet" type="text/css" href="mystyle.css">
<title>Page Title</title>

<h1 class="red_text">This is a red heading using a class.</h1>
<p id="blue_text">This is a blue paragraph using an ID.</p>
```

CSS file

```
.red_text {
    color: red
}

#blue_text {
    color: blue
}
```

Flask

Flask is a web micro-framework. It provides you with the minimal tools and libraries to build a web application in Python. It's a micro-framework in that it only has two dependencies:

- [Werkzeug](#) - a WSGI (Web Server Gateway Interface) utility library for interface between web servers and web applications for the Python programming language.
- [Jinja2](#) - its html template engine

A simple Flask web application

app.py:

```
#!/usr/bin/env python

import flask

# Create the application.
APP = flask.Flask(__name__)

@APP.route('/')
def index():
    """ Displays the index page accessible at '/' """
    return flask.render_template('index.html')

if __name__ == '__main__':
    APP.debug=True
    APP.run()
```

templates/index.html

```
<!DOCTYPE html>
<html lang='en'>
<head>
    <meta charset="utf-8" />
    <title>Hello world!</title>
    <link type="text/css" rel="stylesheet"
        href="{{ url_for('static',
            filename='hello.css')}}" />
</head>
<body>
```

It works!

```
</body>
</html>
```

Folder structure:

```
web_app/
  app.py
  templates/
    index.html
  static/
    hello.css
```

Jinja2

Jinja2 is a templating language for Python, modeled after Django. Besides allowing HTML to be removed from the Flask app and placed into a template, it allows data to be passed-in, allowing the web page to dynamically change with the data it is given. It has many other advantages (see website).

```
<title>{% block title %}{% endblock %}</title>
<ul>
  {% for user in users %}
    <li><a href="{{ user.url }}">{{ user.username }}</a></li>
  {% endfor %}
</ul>
```

<http://jinja.pocoo.org/docs/2.9/>

Resources

- Flask_worksheet_morning.ipynb
 - Step-by-step investigation of Flask syntax and use cases with a lot of documentation
- <https://www.fullstackpython.com/flask.html>
 - Good overview of what you can do
- <https://realpython.com/blog/python/flask-by-example-part-1-project-setup/>
 - Flask with PostgreSQL database deployed on Heroku with many front end refinements
- <https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-now-with-python-3-support>
 - Another excellent step-by-step guide
- <https://www.fullstackpython.com/jinja2.html>
 - Jinja2 formatting examples
- <http://www.tutorialrepublic.com/html-examples.php>
 - HTML examples for about everything