An Introduction to Web Development using Flask

Mojola Balogun

HACKRICE

Overview

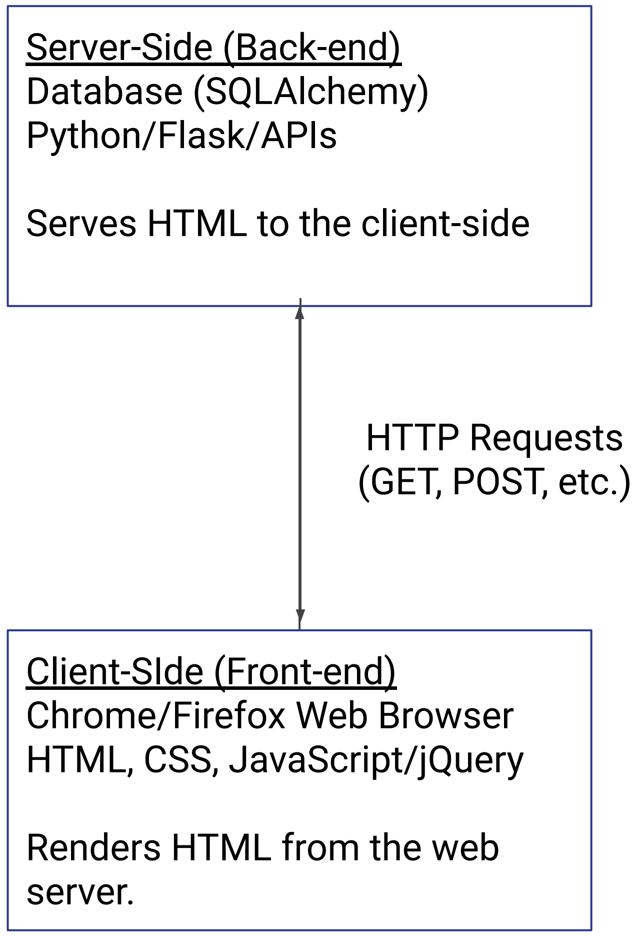
- Introduction
    - Flask
    - Pre-Reqs
    - Setup
- Hello World
    - The Basics
    - Variable Routing
- List Sort
    - Demo/Detailed Overview
    - Fix the Bug!
- Calculator
    - Demo
    - Using Flask at HackRice
- Summary
- Closing

HACKRICE

Introduction: Flask

Flask is a Python microframework that provides useful functions for building web applications.

Flask allows you to not worry about the low-level, nitty-gritty details about developing a back-end for your web application like protocols, routing, thread management, sockets, etc.

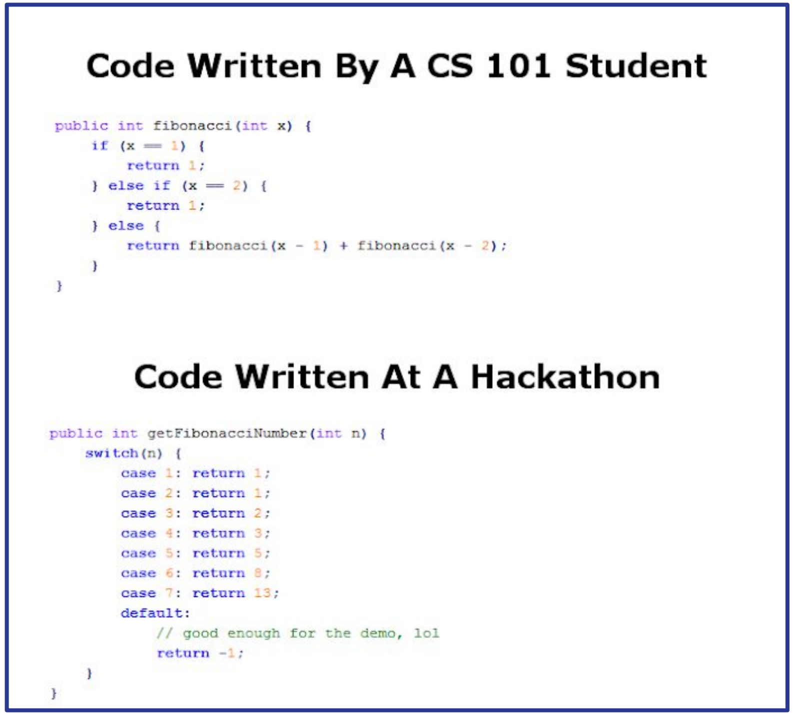

Server-Side (Back-end)
Database (SQLAlchemy)
Python/Flask/APIs
Serves HTML to the client-side
HTTP Requests
(GET, POST, etc.)
Client-SIde (Front-end)
Chrome/Firefox Web Browser
HTML, CSS, JavaScript/jQuery
Renders HTML from the web server.

Introduction: Pre-Reqs for this Workshop

A basic familiarity with Python (so if you've taken COMP 140, you're in good shape)

Familiarity with HTML, CSS and Javascript

A desire to learn fast, build things, and code!



Code Written By A CS 101 Student

```
public int fibonacci(int x) {
    if (x == 1) {
        return 1;
    } else if (x == 2) {
        return 1;
    } else {
        return fibonacci(x - 1) + fibonacci(x - 2);
    }
}
```

Code Written At A Hackathon

```
public int getFibonacciNumber(int n) {
    switch(n) {
        case 1: return 1;
        case 2: return 1;
        case 3: return 2;
        case 4: return 3;
        case 5: return 5;
        case 6: return 8;
        case 7: return 13;
        default:
            // good enough for the demo, lol
            return -1;
    }
}
```

HACKRICE

Introduction: Setup Python & Flask

Install Python. Flask is compatible with both Python 2 or 3, so either one is fine. This workshop will be done in Python 3.

Download: python.org

Install Flask using PIP (Comes with you installation of Python)

```
>> sudo pip install Flask
```

You can use any text-editor. I recommend VSCode :)

HACKRICE

Hello World: Example

/mojolabalogun/hackrice9-flask-workshop/hello_world/hello_world.py

```
from flask import Flask
app = Flask(__name__)
@app.route(“/”)
def hello():
    return “Hello, World!”

if __name__ == “__main__”:
    app.run()
```

This is all the code you need to start serving html web pages to the browser. That’s the beauty of Flask!

HACKRICE

Running the Flask App

Execute the following commands to run the code:

```
>> cd /directory/with/project
>> export FLASK_APP=hello_world.py
>> flask run
```

If successful, you should see the following:

```
* Serving Flask app “hello_world”
* Running on http://127.0.0.1:5000/
```

HACKRICE

A Technical Overview

- We started up a local flask server which is listening at IP address 127.0.0.1, port 5000
- We visited this page and the browser sent a request to our flask server
- Because we visited 127.0.0.1:5000/ (emphasis on the "/"), the method bound to "/" was invoked
- The hello() method in hello world.py was executed and the return value is sent to the browser as HTML
- The browser received the HTML and rendered the webpage

HACKRICE

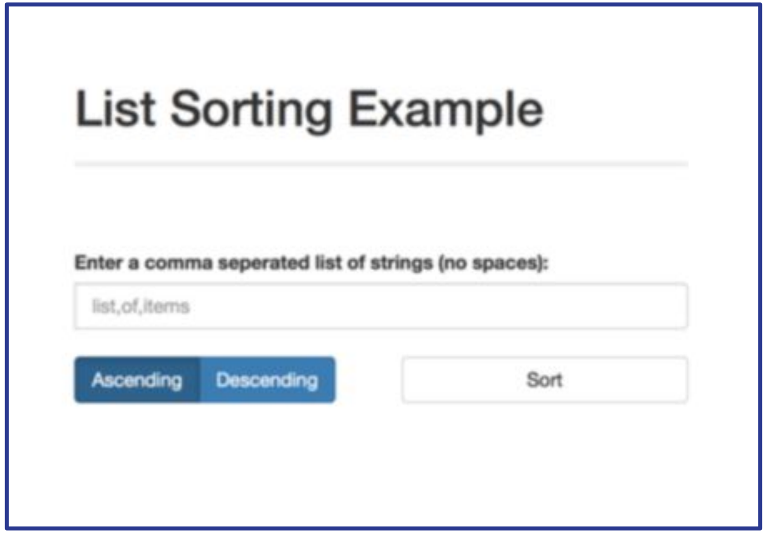
Hello World: Using Variables when Routing

- Flask allows you to access variables from your url when routing!
    - /mojolabalogun/hackrice9-flask-workshop/hello_world/hello_user.py
- Using the same method we used to run ‘hello_world.py’, try running ‘hello_user.py’.
- See if you can figure out what’s going on!
    - You declare the form of the url in app.route(), enclosing the variable within a less-than and greater-than sign: <[variable]>
    - You can then accept that variable as a parameter within your function
- After running the application, try typing in ‘localhost:5000/anything’ into your browser. What happens?

HACKRICE

List Sort: Demo

Visit github.com/mojolabalogun/hackrice9-flask-workshop/list_sort/


List Sorting Example

Enter a comma seperated list of strings (no spaces):

Author: Prudhvi Boyapalli

HACKRICE

List Sort: Detailed Overview (1/3)

- Observe the project structure of the project. Flask requires you to follow this project structure.
  - **parent_directory**
    - **static**
      - **scripts** (Javascript Files)
      - **styles** (CSS Files)
    - **templates** (HTML Files)
    - app.py
- When running Flask functions, like render_template(), it assumes this structure.

Quick Detour

- Remember how I said that the web server and the browser communicate via HTTP Requests. Well, what exactly are they?
- HyperText Transfer Protocol is the data transfer protocol used on the WWW
- GET is the most common HTTP method and is used to get files from a server. Multiple GET requests are made by your browser every time you visit a webpage to retrieve all the necessary resources needed to render that page.
- POST is used to send data to a server, most commonly via form submits, or any other user input. All the data is hidden within a JSON object sent to the server.
- There are other HTTP request methods: www.developer.mozilla.org/en-US/docs/Web/HTTP/Methods

HACKRICE

List Sort: Detailed Overview (2/3)

Retrieving resources and rendering the webpage

1. User enters the address into their browser
2. Browser sends a GET request to the address, in this case: 127.0.0.1:5000/
3. The flask method bound to "/" is invoked and returns index.html
4. Browser processes the HTML file, which instructs it to load CSS and JavaScript files
5. Browser sends additional GET requests to the location indicated within the HTML files
6. Flask serves the static CSS and JavaScript files
7. Browser now has all necessary files and renders the web page

HACKRICE

List Sort: Detailed Overview (3/3)

Send data to the server, back-end computation, and return results

1. User enters information into the input fields and clicks the Sort button
2. The button’s onclick is invoked and invoking the corresponding JS function
3. The input information is read from the input fields and stored in an JSON object
4. A POST request is created and sent to "/sort"
5. Flask server receives the request; so, the method bound to "/sort" is invoked, data from the request object is accessed, the list is sorted, and a response object is created with the results in JSON format and returned
6. Back in the JavaScript function, the response is parsed
7. The necessary HTML to display the sorted list is created and inserted into page

HACKRICE

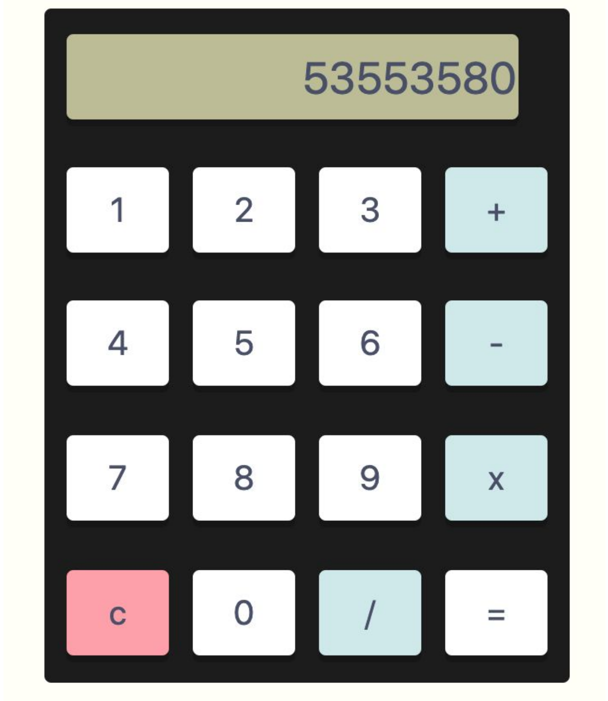
List Sort: Fix the Bug!

- Try running ‘list sort.py’. You should notice that nothing appears when you access localhost:5000/ in your web browser
- Is the function that’s called when you access the base URL returning anything?
- See if you can fix this problem :)
- After fixing the bug, walk through the code to see if you can understand what’s going on
- If you run into trouble running the code, flag down one of the instructors and we’ll help you out

HACKRICE

Calculator: Demo

Visit www.github.com/mojolabalogun/hackrice9-flask-workshop/calculator/



HACKRICE

Calculator: Using Flask at HackRice

- This application was made in under an hour using Flask, Python, HTML, CSS and Javascript
  - The HTML was trivial: a form with a textbox for the display and 16 inputs for the calculator buttons. The CSS was written by Emmanuel Ndubuisi. And for the Flask, I only used concepts from this workshop!
- At HackRice, you won't have the time to worry about the details of all the technologies you are working with. Find what applies to your project and use what works

Summary: What did you learn?

- Setting up a development environment for Flask
- Basic & and variable routing
- GET and POST requests
- Accessing request data
- Building and serving a “complex” web app with HTML, CSS, JavaScript and Python
- How to use Flask at HackRice!

HACKRICE

Next Steps

- What you learned today could enable you to build a wide-range of web applications. But there are a lot of other things you can do with Flask:
    - URL Building
    - Additional HTTP Methods
    - HTML templates with placeholders, complex templating language
    - File uploads
    - Reading and writing cookies
    - Redirects and error handling
    - And more!
- Use the resources provided at the end of the slides to learn these things!

HACKRICE

Closing

- Here are some resources you can use to continue exploring web development in Python using Flask:
  - https://palletsprojects.com/p/flask/
  - https://www.fullstackpython.com/flask.html
  - https://www.tutorialspoint.com/flask/index.htm
- Oh, and come to HackRice!
  - Contact me (mojola.balogun@rice.edu) or any of the HackRice mentors for help during the Hackathon

HACKRICE

Thanks for coming!

HACKRICE