Introduction to Flask

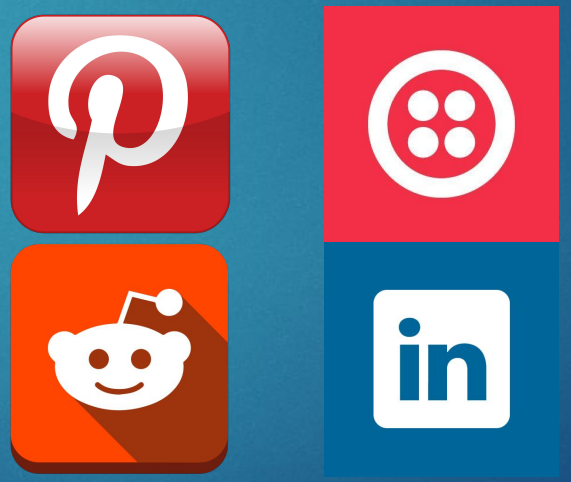
BY: HACK AT UCI

Prerequisites

- A computer capable of running Python
- A computer that has both Python and pip installed
- An interest in web development
- Clone the repository from github:
  - https://github.com/hackuci/learn-flask-template
- Run “pip install flask” in the command line
  - If you are having trouble, please raise your hand and we will get to you

What is Flask?

- A Python web framework that provides you with the tools to build web applications
- A framework that offers simplicity, flexibility, and fine-grained control of your application
- The framework that powers:
  - Pinterest
  - Twilio
  - Reddit
  - LinkedIn's Internal Stack



Key Learning Goals

- Defining and navigating between routes
- Jinjas2 templating for injecting data into HTML pages
- Processing data from forms in the backend
- Simple data storage using JSON files
- Building a fully functioning full stack application

Setting Up the Application

- Import Flask from flask:
  - from flask import Flask #And all other packages
- Initialize application:
  - app=Flask(__name__)
- Run the application:
  - app.run(debug=True)

Defining Routes

- Routes are defined by declaring a decorator with a route name as an argument and a function that modifies the behavior of the generic decorator
- Example:

```
@app.route("/route_name")
def function_name():
    return render_template ("index.html")
```

Jinjas2 Templating

- Template engine that allows one to inject data from the backend to the front end
- Similar to other templating engines such as Handlebars or EJS in Node
- No need to import Jinjas2 because it is used by default with Flask
- Let's learn how to use Python variables and data types in templates

Jinjas2 Templating Commands

- Backend Flask Commands
  - Injecting data to the HTML page inside a function

    user="Ryan"

    return render_template("index.html",user=user)
- Frontend Flask Commands
  - Using Python variables in the frontend

    {{ python_variable }}
  - Using Python conditional statements in the frontend

    {% if user %}

    {% endif %}
  - Using Python loops in the frontend

    {% for user in users %}

    {% endfor %}

Requests

- Requests are methods used by your computer to obtain or submit data to a server
- A request falls under the Hypertext Transfer Protocol (HTTP)
- The common types of requests are GET, PUT , POST , PATCH and DELETE
- The requests we will be focusing on are GET and POST

Handling Requests in Flask

- Import the request module from flask

  from flask import request
- Check the request method that was used

  if request.method=="POST"

  #Do stuff here
- Obtain data from the request form

  if request.method=="POST"

  data=request.form

JSON

- A data format that is clean, readable, and used primarily in communication between a web app and the server
- A format that models how data is stored in a NoSQL database such as MongoDB or Firebase
- The format that we will be using to store our user and task data

Reading and Writing to JSON Files

- Writing to a JSON File

```
with open("user.json","w") as f:
    json.dump(data,f)
```

- Reading a JSON File

```
with open("user.json","r") as f:
    data=json.load(f)
```

Areas for Improvement

- Adding an option to add as many tasks as you would like
- Checking to make sure the user is logged in before showing tasks
- More robust data persistence through a NoSQL or SQL database
- Text reminders for completing tasks
- Storing more details about events
- Smoother UI experience
- CSS styling of pages

Link to Repositories

- Finished Product: https://github.com/ryanluu12345/Flask-To-Do-Finished
- Incomplete Template: https://github.com/ryanluu12345/Flask-To-Do-Template