Python Programming: Flask Project

Final Project

First off, let's take a second to congratulate you for making it this far! We know we've packed a lot of knowledge into a relatively short time! Kudos for rocking it!

Prompt

We'd like you to have something tangible to show for having taken this class with us, so let's use your newly acquired `Flask` skills to make a mini-web app.

Got Ideas?

You are free to make a website about any topic you'd like as long as it meets all the requirements listed below. If you're having trouble coming up with a topic, here are a few to consider:

- Finding a fun API and basing it on that.
- Making an "About Me" site that's all about you.
- Making a portfolio website for one of your hobbies.
- Making a blog about a topic you find interesting.
- Making something you think would improve your life in some way.

Feel free to share resources and inspiration with your classmates!

Deliverables

You must have a Flask site running locally on your own machine. You will work individually on this project, but feel free to share inspiration, resources, or cool APIs that you find with your classmates.

Requirements

Your assignment **must** include:

1. At least three working routes with associated views.
   - Must include at least one `GET` and one `POST` route.
2. Data pulled from at least one API.
   - Get creative! Tons of free APIs exist. Ask your instructor or classmates for ideas.
3. Semantically clean HTML and CSS applied to a template — at least a small amount of styling.
   - Try to spend a little time styling your pages to make them look nice!
4. Core Python topics. At a minimum, this includes:
   - Dictionaries *or* sets *or* tuples.
   - `**args` *or* `kwargs` *or* `*kwargs`.
   - Basic debugging, such as a `try/except` block (only if necessary).
   - A class.
   - User input *or* reading from a file.
5. Comments, so another developer can easily see what your app does.

Bonus: Have extra time? Ask how to deploy a Flask site to a cloud service like Heroku! Or, try to follow the directions in this article.

Resources

Suggested Ways to Get Started

- **Begin with the end in mind.** Know where you want to go by planning ahead, so you don't waste time building things you don't need.
- **Read the docs** for whatever technologies or APIs you use. Most of the time there is a tutorial that you can follow! This isn't always the case, though learning to read documentation is crucial to your success as a developer.
- **Write pseudocode before you write actual code.** Thinking through the logic of something helps.

Useful Resources

- A List of Free APIs
- An Extremely Helpful Debugging Flowchart
- The Python Docs
- How to Use Keyword Args
- How to Use `*args` and `**kwargs`
- Using Chain and Other Itertools
- Python Sets Tutorial)
- Tuples Tutorial
- Writing a Great User Story
- Presenting Information Architecture (includes insight into wireframing)
- Heroku (platform for hosting your back end)

Evaluation

Your project will be evaluated based on the rubric below.

Rubric

Score	Expectations
0	Incomplete.
1	Does not meet expectations.
2	Meets expectations — good job!
3	Exceeds expectations — fantastic!

Here's an example of how the criteria works. Let's say your assignment was to cook a pizza:

Criteria	0 (Incomplete.)	1 (Does not meet expectations.)	2 (Meets expectations.)	3 (Exceeds expectations.)
Crust	No crust present. Submission is just cheese and sauce on a plate.	Pizza has a crust, but it is raw.	Crust is cooked thoroughly.	Crust is golden brown and the perfect thickness.
Cheese	No cheese present.	A small sprinkle of unmelted cheese in the middle of the pie.	Cheese covers the pizza from edge to edge but is not fully melted.	Cheese is delicious, plentiful, and melted to perfection.