12.3 Lesson Summary - Rendering Your Data With Flask

We have previously described how data is exchanged in a server-client model over a network. The client will make a request of the server and the server responds by sending data to the client. When you are browsing the internet, your browser is the client receiving data from the web servers hosting the sites you are browsing. Flask enables us to create a server to distribute the webpages of a website to clients.

Concept: A Flask App works by providing data in response to HTTP requests made on a URL specified in an app route decorator. The simplest Flask App that returns the string "Hello World" when someone browses to the root URL (making an HTTP Get request) would require the following code:

*from flask import Flask*

*app = Flask(\_\_name\_\_)*

*@app.route("/")*

*def echo():*

*return "Hello World"*

*if \_\_name\_\_ == "\_\_main\_\_":*

*app.run(debug=True)*

Concept: You'll often wish to use Flask to serve HTML webpages to clients. You'll usually want to include data from the server or a database with these webpages. You can use HTML **templates** to blend server data with HTML webpage data. For example if you had the following text in your HTML: *<h1>{{ data\_to\_insert }}</h1>* you would use the following code to inject text into the HTML text:

*from flask import Flask, render\_template*

*app = Flask(\_\_name\_\_)*

*@app.route("/")*

*def home():*

*return render\_template("index.html", data\_to\_insert="Display this in the webpage")*

* Activity: 01-Ins\_Render\_String, 02-Stu\_Render\_String

Concept: You are able to inject more than simple strings into your HTML, you can also inject **Lists**. So if you were passing a list variable into your HTML template using a *render\_template* statement in your Python you would use the following code in your HTML template to display the contents of that list:

*{{data\_to\_insert}}*

*{% for name in list %}*

*<li>{{ name }}</li>*

*{% endfor %}*

* Activity: 03-Ins\_Render\_List, 04-Stu\_Render\_List

Concept: You can inject **dictionaries** into your HTML templates using the following code in your HTML:

*<ul style="list-style: none;">*

*<li>{{ dict.item\_1 }}</li>*

*<li>{{ dict.item\_2 }}</li>*

*</ul>*

* Activity: 05-Ins\_Render\_Dict, 06-Stu\_Render\_Dict

Concept: Most webpages are generated by taking data stored into a **database** and data that is generated by the server and combining that data with an HTML template.

* Activity: 07-Ins\_Render\_From\_Mongo, 08-Stu\_Render\_From\_Mongo, 09-Ins\_Scrape\_And\_Render, 10-Stu\_Scrape\_Weather