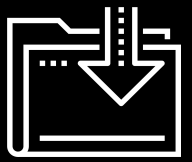
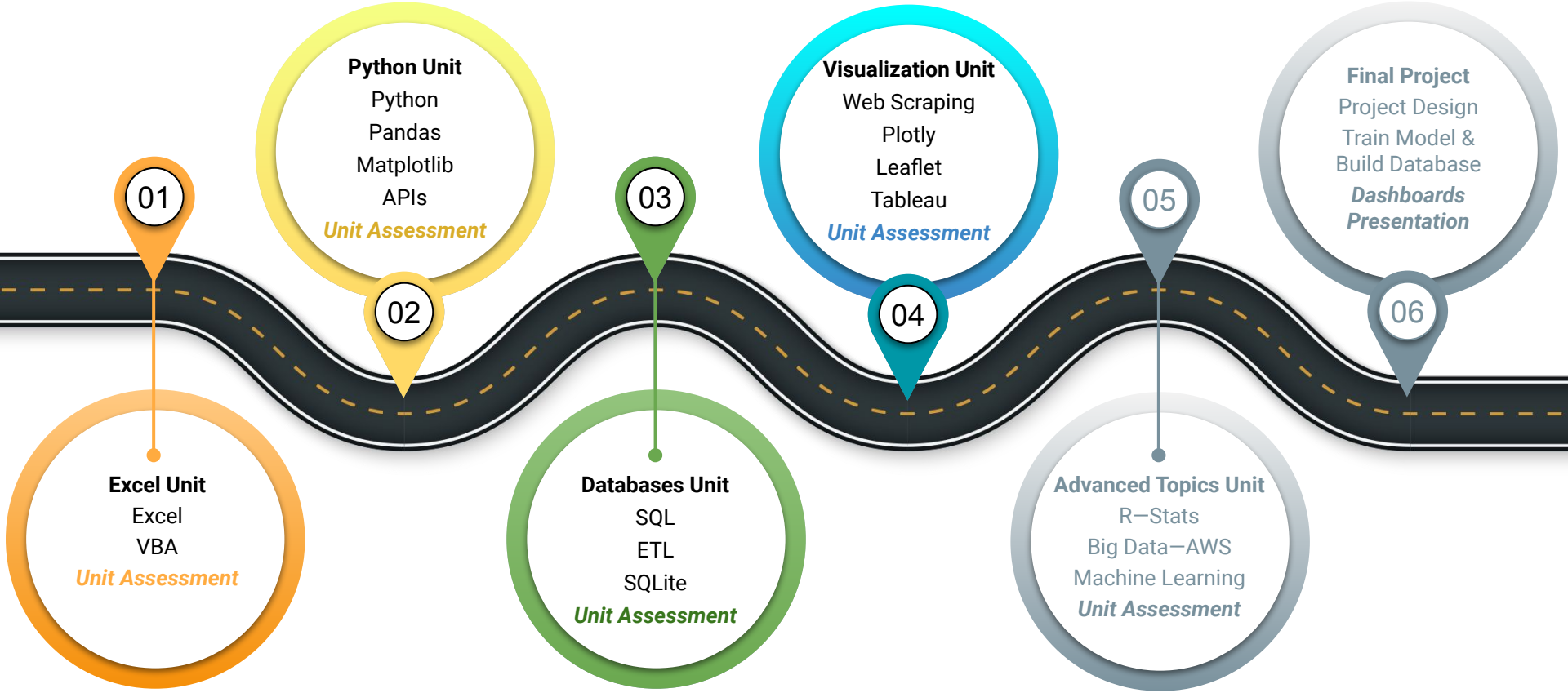
Web Scraping With HTML & CSS

Data Boot Camp

Lesson 10.2



The Big Picture



Module 10

This Week:
Web Scraping with HTML & CSS

This Week: Web Scraping with HTML & CSS

By the end of this week, you'll know how to:

- Gain familiarity with and use HTML elements, as well as class and id attributes, to identify content for web scraping
- Use Beautiful Soup and Splinter to automate a web browser and perform a web scrape
- Create a MongoDB database to store data from the web scrape
- Create a web application with Flask to display the data from the web scrape
- Create an HTML/CSS portfolio to showcase projects
- Use Bootstrap components to polish and customize the portfolio

This Week's Challenge

Using the skills learned throughout the week, create a web app that displays scraped images of Mars' hemispheres, complete with titles.

Module 10

Today’s Agenda

Today's Agenda

By completing today's activities, you'll learn the following skills:

01 Flask templates

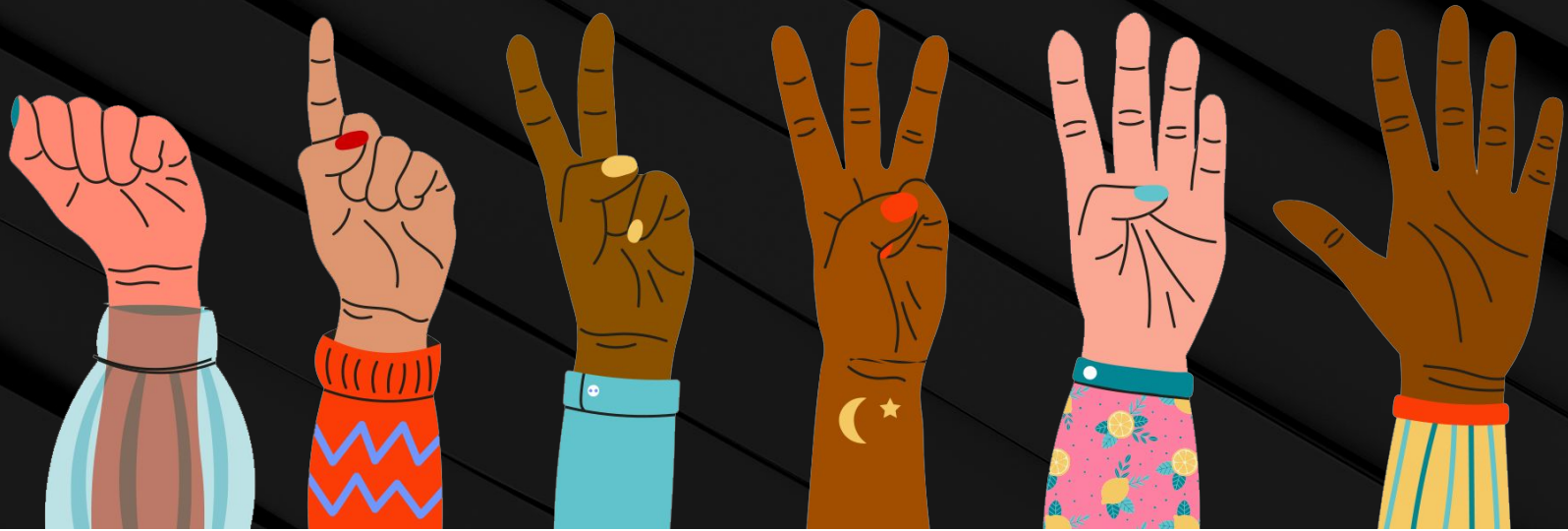
02 Saving data with PyMongo

03 Creating a web page using this module's tools

Make sure you've downloaded any relevant class files!

FIST TO FIVE:

How comfortable do you feel with this topic?

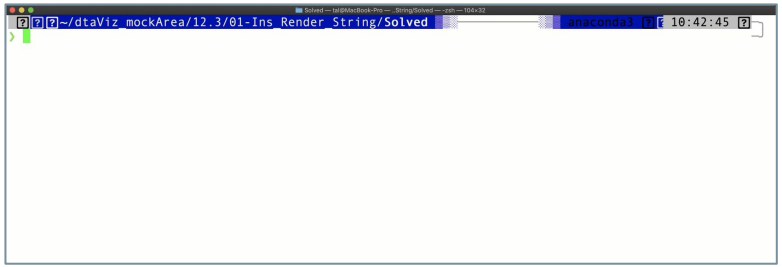


Instructor Demonstration

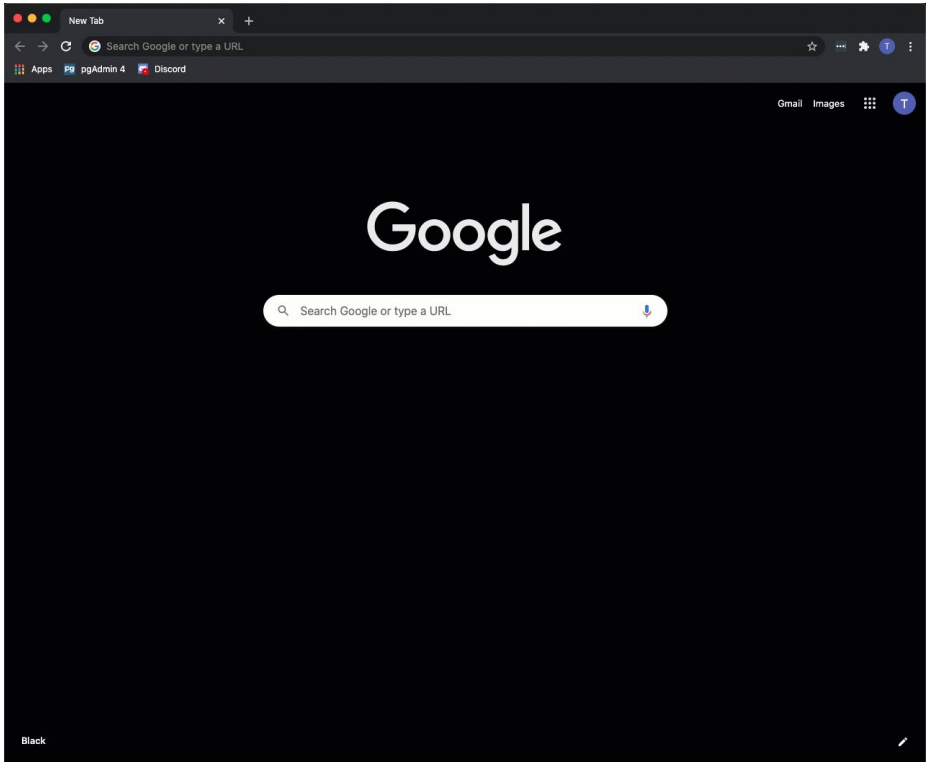
Intro to Flask Render

Intro to Flask Render

How it works—the basics of rendering a template with Flask!



1. In your CLI, navigate to the appropriate folder.
2. Run: `python app.py`
3. Open your browser and visit

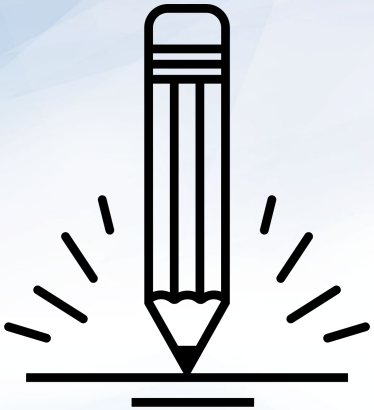


Intro to Flask Render

```
/app.py
/templates
    /index.html
```

```python
# import necessary libraries
from flask import Flask, render_template

# create instance of Flask app
app = Flask(__name__)


# create route that renders index.html template
@app.route("/")
def echo():
    return render_template("index.html", text="Serving up cool text from the Flask server!!")


if __name__ == "__main__":
    app.run(debug=True)
```

```html
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Templates 101</title>
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
</head>

<body>
  <div class="container">

    <div class="jumbotron text-center">
      <!-- Render our data -->
      <h1>{{ text }}</h1>
    </div>

  </div>
</body>

</html>
```



Activity: Rendering a String with Flask

In this activity, you will use Flask to render a welcome message on their page.

Suggested Time:
15 minutes

Let's Review

Instructor Demonstration

Rendering a List and a Dictionary

Rendering a List

```
/app.py
/templates
    /index.html
```

```python
# import necessary libraries
from flask import Flask, render_template

# create instance of Flask app
app = Flask(__name__)


# create route that renders index.html template
@app.route("/")
def index():
    team_list = ["Jumpers", "Dunkers", "Dribblers", "Passers"]
    return render_template("index.html", list=team_list)


if __name__ == "__main__":
    app.run(debug=True)
```

```html
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Teams!</title>
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
</head>

<body>
  <div class="container text-center">
    <h1 class="jumbotron">Team Rosters</h1>
    <div>
      <ul style="list-style: none;">

        {% for name in list %}
        <li>{{ name }}</li>
        {% endfor %}

      </ul>
    </div>
  </div>
</body>

</html>
```

```
{% for name in list %}
  <li>{{ name }}</li>
{% endfor %}
```

Rendering a List and a Dictionary

Suggested Time:

20 minutes

Instructor Demonstration

Scrape, Save, and Render Data

Scrape, Save, and Render Data

/app.py
/scrape_craigslist.py
/templates
    /index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Hot Finds</title>
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
</head>

<body>
  <div class="container">

    <div class="jumbotron text-center">
      <h1>Hot Finds On Craigslist</h1>
      <p><a class="btn btn-primary btn-lg" href="/scrape" role="button">Find An Awesome Deal!</a></p>
    </div>

    <!-- Craigslist Listings -->
    <div class="row" id="craiglist-listings">
      <div class="col-md-12">
        <h4 class="heading">{{listings.price}} {{listings.headline}}</h4>
        <small>{{listings.hood}}</small>
      </div>
    </div>

  </div>
</body>

</html>
```

```
from splinter import Browser
from bs4 import BeautifulSoup


def init_browser():
    # @NOTE: Replace the path with your actual path to the chromedriver
    executable_path = {"executable_path": "/usr/local/bin/chromedriver"}
    return Browser("chrome", **executable_path, headless=False)


def scrape():
    browser = init_browser()
    listings = {}

    url = "https://raleigh.craigslist.org/search/hhh?max_price=1500&availabilityMode=0"
    browser.visit(url)

    html = browser.html
    soup = BeautifulSoup(html, "html.parser")

    listings["headline"] = soup.find("a", class_="result-title").get_text()
    listings["price"] = soup.find("span", class_="result-price").get_text()
    listings["hood"] = soup.find("span", class_="result-hood").get_text()

    return listings
```

```
from flask import Flask, render_template, redirect
from flask_pymongo import PyMongo
import scrape_craigslist

app = Flask(__name__)

# Use flask_pymongo to set up mongo connection
app.config["MONGO_URI"] = "mongodb://localhost:27017/craigslist_app"
mongo = PyMongo(app)

# Or set inline
# mongo = PyMongo(app, uri="mongodb://localhost:27017/craigslist_app")


@app.route("/")
def index():
    listings = mongo.db.listings.find_one()
    return render_template("index.html", listings=listings)


@app.route("/scrape")
def scraper():
    listings = mongo.db.listings
    listings_data = scrape_craigslist.scrape()
    listings.update({}, listings_data, upsert=True)
    return redirect("/", code=302)


if __name__ == "__main__":
    app.run(debug=True)
```

Scrape and Render

Suggested Time:

30 minutes

Questions?