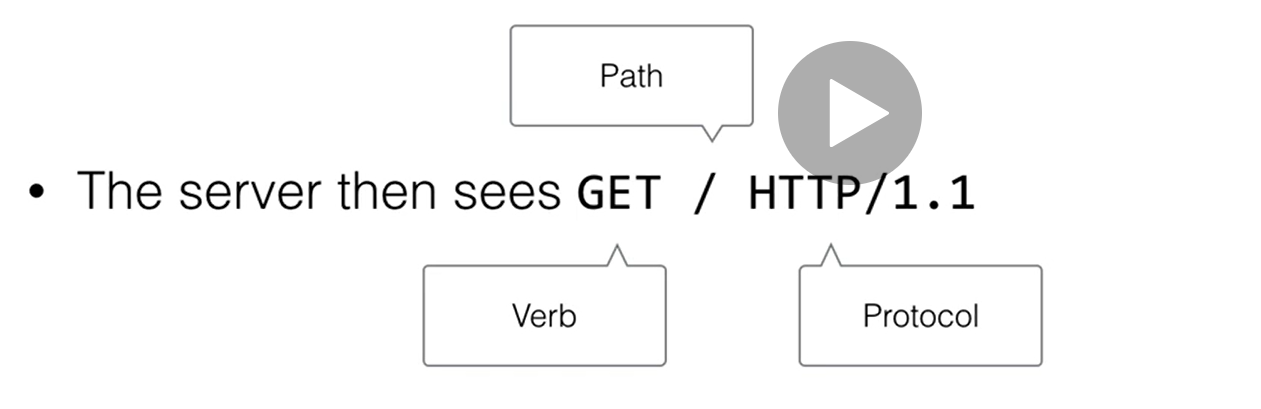
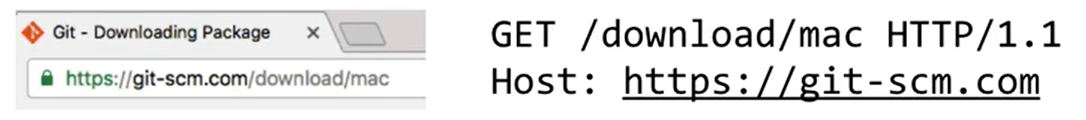
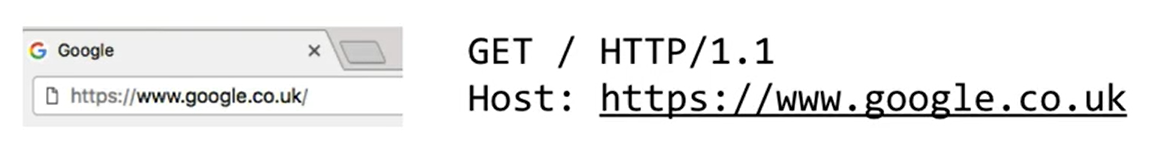
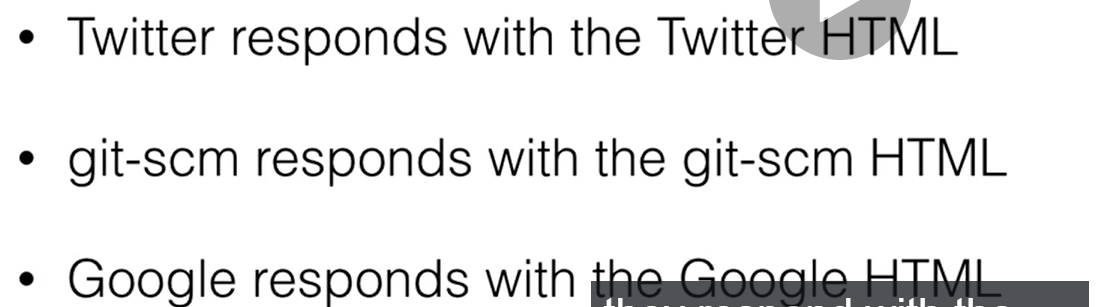
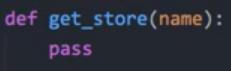
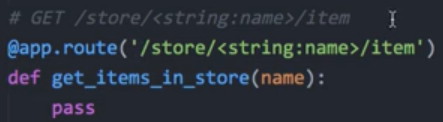
[https://github.com/schoolofcode-me/rest-api-sections/tree/master/section3](https://github.com/schoolofcode-me/rest-api-sections/tree/master/section3" \t "_blank)

1. Web Server

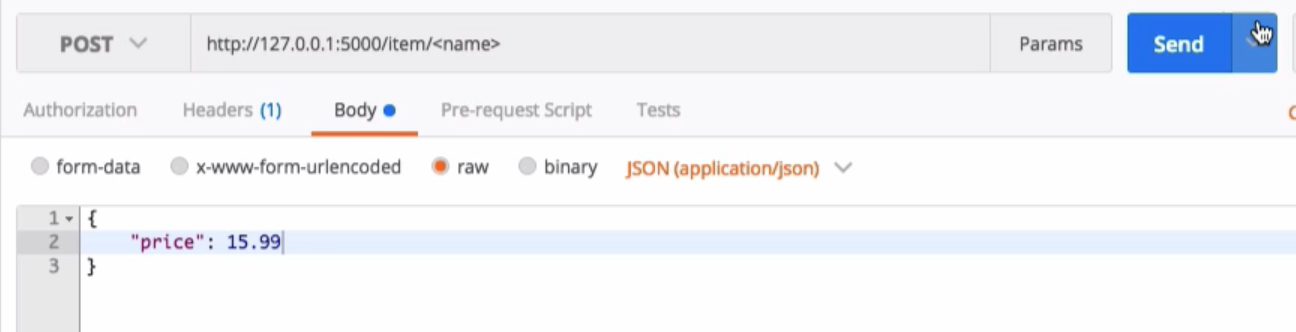
* A piece of software designed to accept incoming web requests
* When you go to a web server, you are sending some kind of request to the server
* When you go to a web site, the web’s server sees the following stuff
  + 
  + 
* Possible areas of error
  + The path is not found
  + HTTP is not supported (uses some other protocols)
  + Server is not available
  + May give HTML code back (normally what it do)
  + Give some text back
  + Nothing if the server is not configured
* Examples
  + 
    - Up to the **.com** is the **host name**
    - After the / is the **path** ---/login
  + ****
    - **Green – protocol**
    - **Black – host**
    - **Grey – path**
  + ****
    - In the case of homepage, it is just /
* Differences
  + Because of different requests and the different path in the request
  + The server will return different things (html code)
    - ****
* Extra stuff
  + Going to the website will always return you a GET request
  + Different HTTP verbs
    - GET
      * Receive something
      * When you go to Google home page, you are getting/retrieving something from Google Homepage
      * In the case of using API: **GET /item/1**
        + Retrieve an item named as 1
    - POST
      * You are sending something over to the server to create something
      * Like **POST /item**
      * You might need to send this item’s information along
    - PUT
      * You are making sure something is there
      * **PUT /item**
      * In this case, you may be sending some information over about the item to check whether an item with the information stated is present
      * If it is just there, it may just check the information with the newly-sent one
    - DELETE
      * **DELETE /item/1**
      * Delete this item with number 1
  + JSON
    - Json has a similar structure as dictionary with Key-Value pairs
    - But it is not a dictionary but a text but a long string
    - Javascript can only understand and send that kind of text
    - Flask has this library called jsonify which takes in dictionary and convert it to string
      * From flask import jsonify
      * **return jsonify(‘stores’: stores)**
        + jsonify will only take in a dictionary datatype so remember to convert accordingly if the structure is not a dictionary format
    - json always uses double quotes and never uses single quotes which is what Python dictionary usually looks like\
  + API
    - Usually **@app.route(“/”)** only does **GET** request which is like returning the information in json/txt format
    - However, to indicate that we want to post/create something you have to specify that the method is POST
      * ****
    - When you want to GET(find the store by typing in the name specifically in the url) by using the store’s name
      * ****
      * ****
        + It means that the when you trying to type **/store/<string:name>** you are actually trying to call the function **get\_store** with the name being passed in
    - Trying to get store/when the **url** keyed in **end with /store**
      * it is just getting normal store information
    - When you want to POST(create) Zsome items to a specific store
      * You will need the keyword which is the store’s name
      * But there is no specifications on which the items
      * 
        + In this case, the store’s name should be keyed in as string because you want to specify which store you want to create items into.
    - Get items from a specific store
      * ****
      * In this case, you need to specific which store you want to get items from.
* Usage of Postman
  + Order of creation
    - Create a collection first
    - Create a section first
    - Then create request
      * For request, you need to specify whether it is GET/POST
      * For get just,
      * For **POST** **ONLY!!!**
        + Go to **header**
        + To create new stuff in it, you need to specify the Key and Value pair as following

**Key : Content – Type**

**Value: application/json**

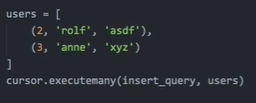
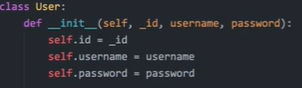
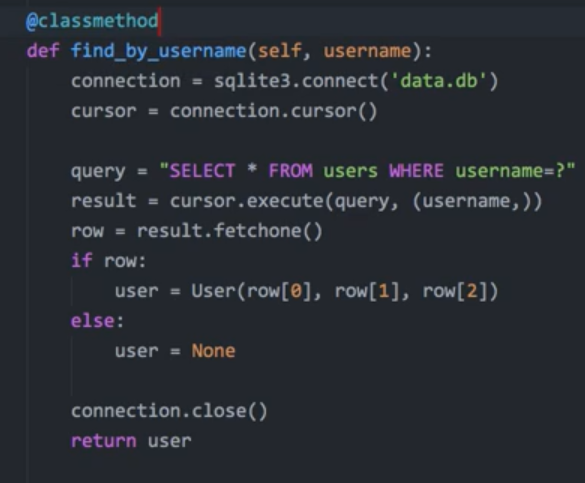
* + - * + **Body:**

Set to **raw**

****

Type in the structure you are expecting

So what this means is that when you create a new item, that item is going to be this price of 15.99

* + - * Enter the URL and press send
  + **PUT** method
    - It is almost the same as the POST method but you should indicate the updated value in the **raw** section
* Usage of Virtual Environment
  + Creation of virtual environment
    - py -m venv env
  + To activate
    - ****
  + To deactivate
    - ****
  + To delete
    - ****
    - <https://www.youtube.com/watch?v=APOPm01BVrk>
* Flask-Restful
  + No need to do jsonify
  + Status code
    - **Most common is 200**
    - Fail action is **404**
      * ****
    - Successfully created **201**
      * ****
    - Bad request ( such as creating a name for store that already exist)
      * ****
* SQLite
  + Code for this section
    - <https://github.com/schoolofcode-me/rest-api-sections/tree/master/section5>
  + You can just import this library **sqlite3**
    - Creation of database
      * ****
        + When this is executed in the command line then a **data.db** will be automatically created in the folder
    - Creation of cursor
      * ****
        + This is just like creation of a cursor that is always pointing at top of the folder
    - Table schema/format
      * ****
        + Specifying how the data is going to be stored
      * ****
        + Execute the creation of table
    - Insertion of value
      * Single Value
        + ****
      * Multiple Values
        + ****
    - End of creation & Insertion
      * ****
      * End the connection
    - Retrieval of result
      * ****
      * The select query is just SQL query
      * But it is saving the result into **select\_query**
      * Using a for loop to print out its content
    - Retrieval of result by certain condition
      * Define a class first
        + ****
      * Find user by username
        + ****
        + Connect to the **data.db**
        + Create a cursor
        + Create an SQL query
        + Execute the query with the given username and store the **rows returned** into variable **result**
        + **Result.fetechone()** will give you the rows that are returned in the result
        + If row exist the first value of the row **row[0]** will be **id** **row[1]** will be **name**, **row[2]** will be **password**

****

Can also be this kind of order sequence

* + Deploying using Heroku
    - Code
      * <https://github.com/schoolofcode-me/stores-rest-api>