# crud\_examples

@app.route('/', methods=['GET'])

This syntax is the part of the program that lets Flask know that this function, home, should be mapped to the path /.

The methods list (methods=['GET']), (methods=['POST']), (methods=['PUT']), (methods=['DELETE']) are keyword arguments that lets

Flask know what kind of HTTP requests are allowed

import flask — Imports the Flask library, making the code available to the rest of the application.

app = flask.Flask(\_\_name\_\_) — Creates the Flask application object, which contains data about the application and also methods

(object functions) that tell the application to do certain actions.The last line, app.run(), is one such method.

app.run(debug=True, port=8080) — A method that runs the application server, starts the debugerand tells it to use port namber 8080

@app.route() - It is a decorator(a function) which accepts arguments that gets called before a function. Going through all the files

we can see the arguments it accepted

The jsonify() function in flask returns a flask.Response() object that already has the appropriate content-type header

'application/json' for use with json responses.

Python Requests supports the entire restful API, i.e., all its methods – PUT, GET, DELETE, POST.

http://chroniclingamerica.loc.gov - is the base url if given in numbers for example https://127.0.0.1 it is the IP address

nomally it is difficult to remember numbers for a human brain so websites are given names

/search/pages/results/ - is the path

If we combine the base URL and the path together into one URL, we’ll have created a request to the Chronicling America API

that returns all available data in the database:

http://chroniclingamerica.loc.gov/search/pages/results/

The query parameters follow the ? in the request, and are seperated from one another by the & symbol.