Data Representation
Lab 05.2: Flask: intro to web server

Lecturer: Andrew Beatty

Get a web server to run :

- Create a simple app-server with flask
- Serve static web pages
- Get pre-written REST-server to run (and test it with curl)
- The server serves out a RESTfull api that performs CRUD operations on cars

Action	Method	URL	Sample params	Sample return
Get all	GET	/cars	none	{ cars:[{…},{…},{…}] }
Find by id	GET	/cars/*reg*	none	{car: { "reg":"12 D 1234", "make":"Fiat", "model":"Punto", "price":3000 } }
Create	POST	/cars	{ "reg":"12 D 1234", "make":"Fiat", "model":"Punto", "price":3000 }	{ "reg":"12 D 1234", "make":"Fiat", "model":"Punto", "price":3000 }
Update	PUT	/cars/*reg*	{ "price":3000 }	{ "reg":"12 D 1234", "make":"Fiat", "model":"Punto", "price":3000 }
delete	DELETE	/cars/*reg*	none	{ "done":true }

1. Create a directory called Week05 (this is to contain this week's work)
2. In that directory, create a sub-directory called server.
3. In the directory called server, write a simple app server with flask that serves hello world (call it simpleserver.py

```
#!flask/bin/python
from flask import Flask

app = Flask(__name__)

@app.route('/')
def index():
    return "Hello, World!"

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

4. Run it (from the directory called server)

```
python simpleserver.py
```

You should get an output like this.

```
* Serving Flask app "simpleserver" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a
production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 598-374-362
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

If you don't it might be because the flask module needs to be loaded. Install that

```
pip install flask
```

5. Test this by opening up the browser and opening the link `http://127.0.0.1:5000/`. You should see Hello World

Serve static pages

6. Get the app server to be able to serve static pages by adding the following line below the import flask line

```
app = Flask(__name__,
            static_url_path='',
            static_folder='../')
```

The static folder points to the directory that contains the html pages.

7. Test this code by making a static html file called index.html in the week05 directory, (ie the parent of the server directory.
8. Open the file in the browser by going to `http://127.0.0.1:5000/index.html`. You should see your file

Get REST server running

9. In the Server directory create a file called restserver.py. copy all the code from my file on githup and paste it into your file.

https://github.com/andrewbeattycourseware/dataRepresentation/blob/master/code/week05-REST/server/b_restserver.py

10. Test it: Close the simple server (ctrl-c) and run the rest server.

```
python restserver.py
```

11. You should get the same output that you got when you ran the simple server.
12. Open the browser and check you are getting the static page.

```
http://127.0.0.1:5000/index.html
```

13. Check you can get the cars

```
http://127.0.0.1:5000/cars
```

14. Check that all the REST functionality is working with the following CURL commands (in the command line.
    a. Get all

```
curl -i http://localhost:5000/cars
```

    b. Find by id

```
curl -i http://localhost:5000/cars/test
```

    c. Create

```
curl -i -H "Content-Type:application/json" -X POST -
d '{"reg":"12 D 1234","make":"Fiat","model":"Punto","price":3000}' http://localhost:50
00/cars
```

    d. Update

```
curl -i -H "Content-Type:application/json" -X PUT -
d '{"make":"Fiesta"}' http://localhost:5000/cars/181%20G%201234
```

    e. delete

```
curl -i -X DELETE http://localhost:5000/cars/181%20G%201234
```