JSON

Let's represent a structure that contains a list of locations, where each location has a suburb and postcode:

```
{
        "locations": [
        {
            "suburb" : "Kensington",
            "postcode" : 2033
        },
        {
            "suburb" : "Mascot",
            "postcode" : 2020
        },
        {
            "suburb" : "Sydney CBD",
            "postcode" : 2000
        }
    ]
}
```

Note:

- No trailing commas allowed
- Whitespace is ignored

Flask

Lightweight HTTP web server built in python

flask1.py

```
1 from flask import Flask
2 APP = Flask(__name__)
3
4 @APP.route("/")
5 def hello():
6     return "Hello World!"
7
8 if __name__ == "__main__":
9     APP.run()
```

```
1 $ python3 flask1.py
```

Restful API & "CRUD"

A *RESTful API* is an application program interface (*API*) that uses HTTP requests to GET, PUT, POST and DELETE data. These 4 methods describe the "nature" of different API requests.

GET, PUT, POST, DELETE are HTTP Methods

Method	**Operation**
POST	**C**reate
GET	**R**ead
PUT	**U**pdate
DELETE	**D**elete

Input & Output

Different CRUD properties require different approaches for input. All output are the same.

Inputs are either:

- GET: via URL and "request.args"
- PUT|POST|DELETE: via post-data and via "request.get_json()"
- All outputs should be packaged up as JSON
- (JSON discussed later)

crud.py

```
1  from flask import Flask, request
2  from json import dumps
3
4  APP = Flask(__name__)
5
6  @APP.route("/one", methods=['GET'])
7  def one():
8      return dumps({
9          '1': request.args.get('data1'),
10         '2': request.args.get('data1'),
11     })
12
13 @APP.route("/two", methods=['POST'])
14 def two():
15     data = request.get_json()
16     return dumps({
17         '1': data['data1'],
18         '2': data['data2'],
19     })
20
21 if __name__ == '__main__':
22     APP.run()
```

Wrapping Iteration 1

```python
def channels_listall_v1():
    return datastore.get_all_channels()
```

```python
from json import dumps
from flask import Flask, request
from search import search_fn
APP = Flask(__name__)

@APP.route('/channels', methods=['GET'])
def search_channels():
    # check auth
    return dumps(channels_listall_v1())

if __name__ == '__main__':
    APP.run()
```

GET http://localhost:80... POST http://localhost:8... No Environment

http://localhost:8080/getpeople?min_age=50&name=bob Save

GET http://localhost:8080/getpeople?min_age=50&name=bob Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
min_age	50			
name	bob			
Key	Value	Description		

Body Cookies Headers (4) Test Results Status: 200 OK Time: 8 ms Size: 147 B Save Response

Pretty Raw Preview Visualize JSON

```
1 []
```

GET http://localhost:80... POST http://localhost:8... No Environment

http://localhost:8080/addperson

Save

POST http://localhost:8080/addperson Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
{
    "name": "Steve",
    "dob": "17/07/1990"
}
```

Body Cookies Headers (4) Test Results Status: 200 OK Time: 12 ms Size: 147 B Save Response

Pretty Raw Preview Visualize JSON

```
{}
```

Notes

Important things:

- HIGHLY recommend lab05 simple

Automarking Improvements

- (Iteration 2 video) you can check which asserts failed
- 9/10 times errors come from misreading the spec, have had two groups improve scores by misunderstanding how certain functions work
- Good style and clean code, use of helper functions REALLY HELP (another team couldn't improve autotest, so they fixed their style which made it easier to see their bug)
- Thorough tests should find bugs; have someone who didn't deal with a function's tests or implementations write further tests if score isn't 100

- I'll get marking done soon and hand over my feedback hopefully today or tomorrow
- Sorry if I was too critical last week and seemed like I was focusing on style too much, I'd like to help improve your logic within that time, but most of the marking criteria deals with style and git practices, it's what sets you apart as Software Engineers
- **Multiple get requests**
- **Only a 3 autotests (leaderboard)**

HTTP Method	CRUD Action
GET	Retrieve a resource
POST	Create a resource
PUT	Update a resource
DELETE	Delete a resource