多媒體數據分析與應用

張家瑋 博士

助理教授

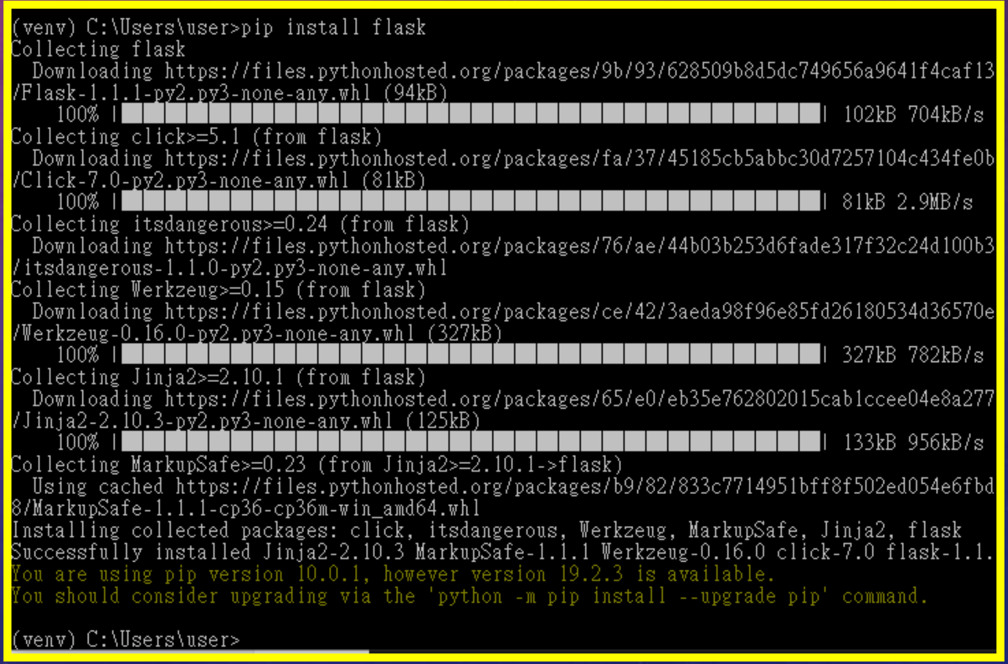
國立臺中科技大學資訊工程系

虛擬環境設定

- 前提在 Python 3.6 以上版本，使用內建的 python3-venv 套件。
  - python -m venv venv (虛擬環境名稱)

- Windows
  - .\venv\Scripts\activate.bat

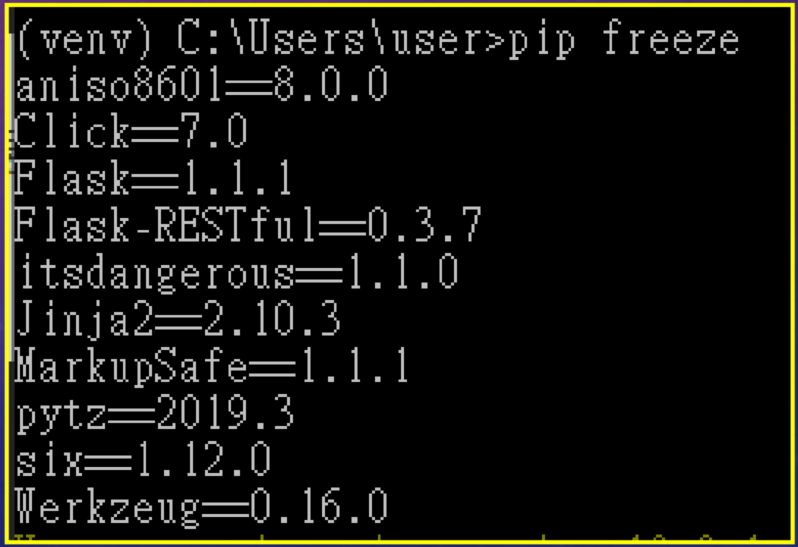
- Linux/macOS
  - source ./venv/bin/activate



```
(venv) C:\Users\user>pip install flask
Collecting flask
  Downloading https://files.pythonhosted.org/packages/9b/93/628509b8d5dc749656a9641f4caf13
/Flask-1.1.1-py2.py3-none-any.whl (94kB)
    100% |████████████████████████████████| 102kB 704kB/s
Collecting click>=5.1 (from flask)
  Downloading https://files.pythonhosted.org/packages/fa/37/45185cb5abbc30d7257104c434fe0b
/Click-7.0-py2.py3-none-any.whl (81kB)
    100% |████████████████████████████████| 81kB 2.9MB/s
Collecting itsdangerous>=0.24 (from flask)
  Downloading https://files.pythonhosted.org/packages/76/ae/44b03b253d6fade317f32c24d100b3
/itsdangerous-1.1.0-py2.py3-none-any.whl
Collecting Werkzeug>=0.15 (from flask)
  Downloading https://files.pythonhosted.org/packages/ce/42/3aeda98f96e85fd26180534d36570e
/Werkzeug-0.16.0-py2.py3-none-any.whl (327kB)
    100% |████████████████████████████████| 327kB 782kB/s
Collecting Jinja2>=2.10.1 (from flask)
  Downloading https://files.pythonhosted.org/packages/65/e0/eb35e762802015cab1ccee04e8a277
/Jinja2-2.10.3-py2.py3-none-any.whl (125kB)
    100% |████████████████████████████████| 133kB 956kB/s
Collecting MarkupSafe>=0.23 (from Jinja2>=2.10.1->flask)
  Using cached https://files.pythonhosted.org/packages/b9/82/833c7714951bff8f502ed054e6fbd
8/MarkupSafe-1.1.1-cp36-cp36m-win_amd64.whl
Installing collected packages: click, itsdangerous, Werkzeug, MarkupSafe, Jinja2, flask
Successfully installed Jinja2-2.10.3 MarkupSafe-1.1.1 Werkzeug-0.16.0 click-7.0 flask-1.1.
You are using pip version 10.0.1, however version 19.2.3 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

(venv) C:\Users\user>
```

安裝所需模組

1. pip install flask
2. pip freeze #觀看虛擬環境下已安裝的模組



```
(venv) C:\Users\user>pip freeze
aniso8601==8.0.0
Click==7.0
Flask==1.1.1
Flask-RESTful==0.3.7
itsdangerous==1.1.0
Jinja2==2.10.3
MarkupSafe==1.1.1
pytz==2019.3
six==1.12.0
Werkzeug==0.16.0
```

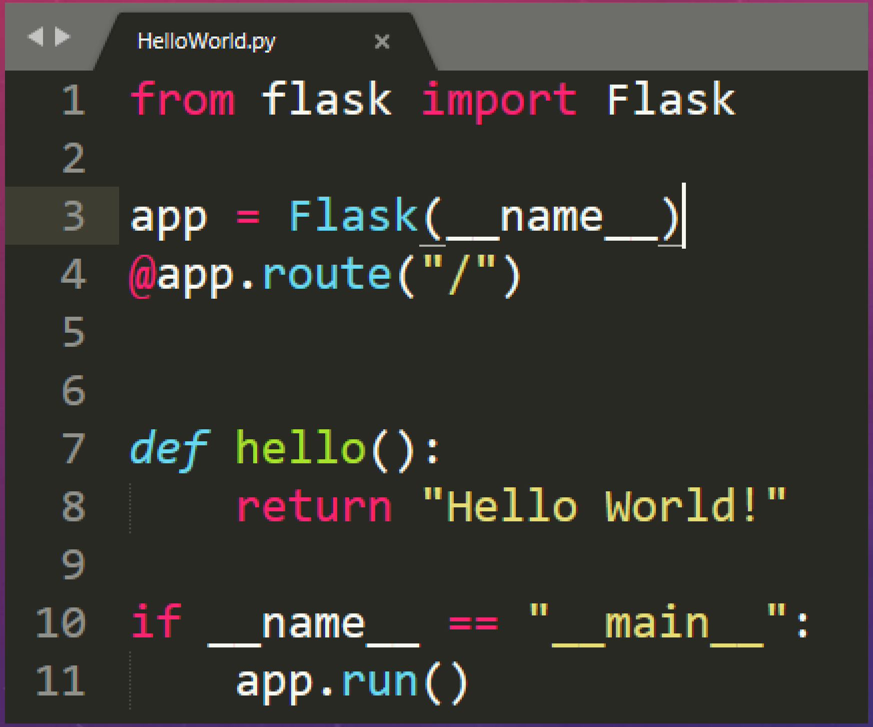
FLASK
HTTP
FLASK

FLASK

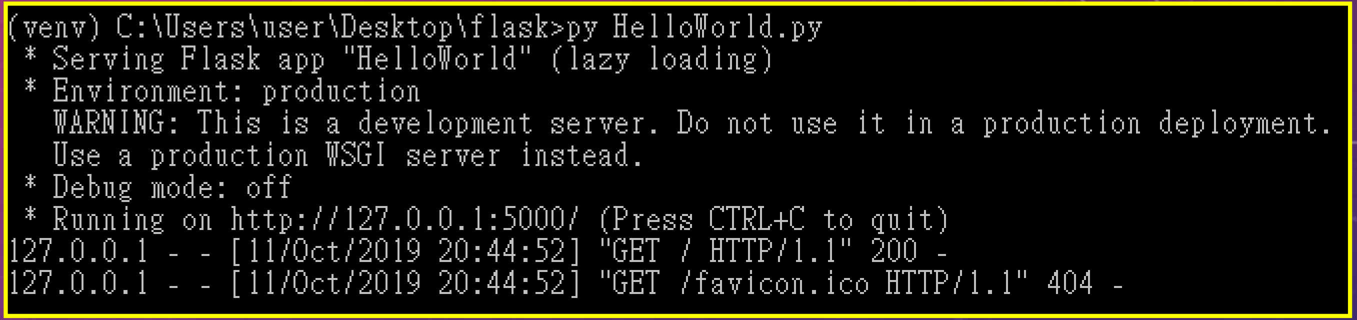
- Python 有許多能用於創建 Web 應用程式和 Web API 的框架，而輕量的 Flask 框架可以勝任Web API 的需求：管理 HTTP 請求和顯示資料內容。

- flask-restful 是針對 restful api 開發的一個flask的套件，建構在 flask 的輕薄短小的基礎下，flask-restful 可以在短短幾行內完成 restful api的開發。

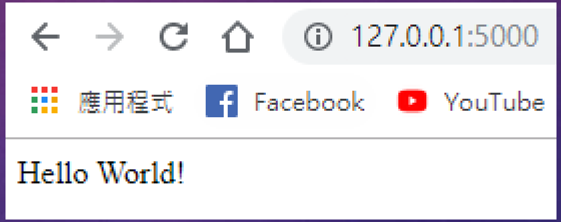
第一個 FLASK 範例


HelloWorld.py

```python
from flask import Flask

app = Flask(__name__)
@app.route("/")


def hello():
    return "Hello World!"

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

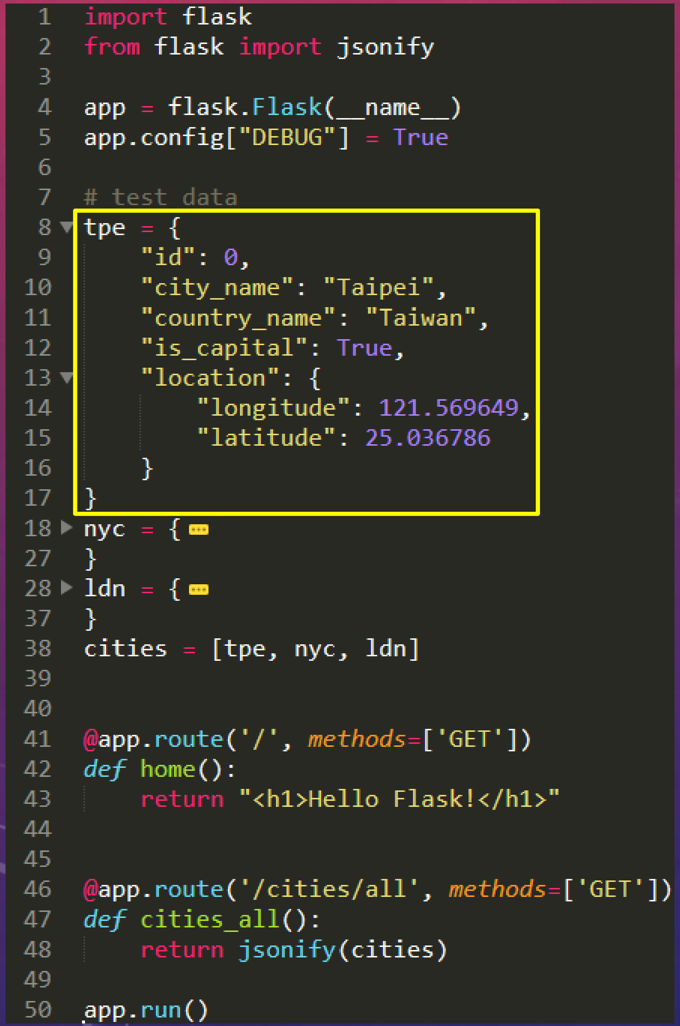


```
(venv) C:\Users\user\Desktop\flask>py HelloWorld.py
 * Serving Flask app "HelloWorld" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [11/Oct/2019 20:44:52] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [11/Oct/2019 20:44:52] "GET /favicon.ico HTTP/1.1" 404 -
```

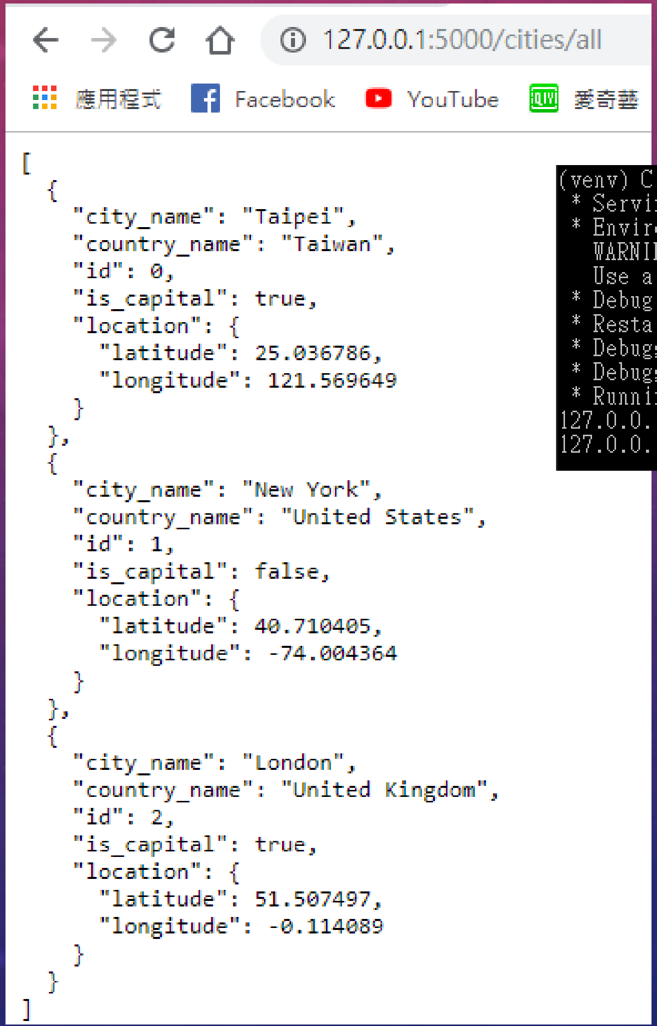


Hello World!

GET DATA BY WEB API



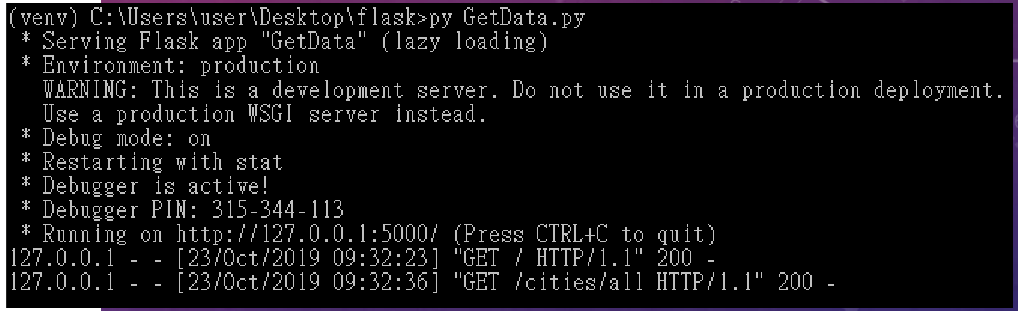
```python
import flask
from flask import jsonify

app = flask.Flask(__name__)
app.config["DEBUG"] = True

# test data
tpe = {
    "id": 0,
    "city_name": "Taipei",
    "country_name": "Taiwan",
    "is_capital": True,
    "location": {
        "longitude": 121.569649,
        "latitude": 25.036786
    }
}
nyc = {
}
ldn = {
}
cities = [tpe, nyc, ldn]


@app.route('/', methods=['GET'])
def home():
    return "<h1>Hello Flask!</h1>"


@app.route('/cities/all', methods=['GET'])
def cities_all():
    return jsonify(cities)

app.run()
```



127.0.0.1:5000/cities/all

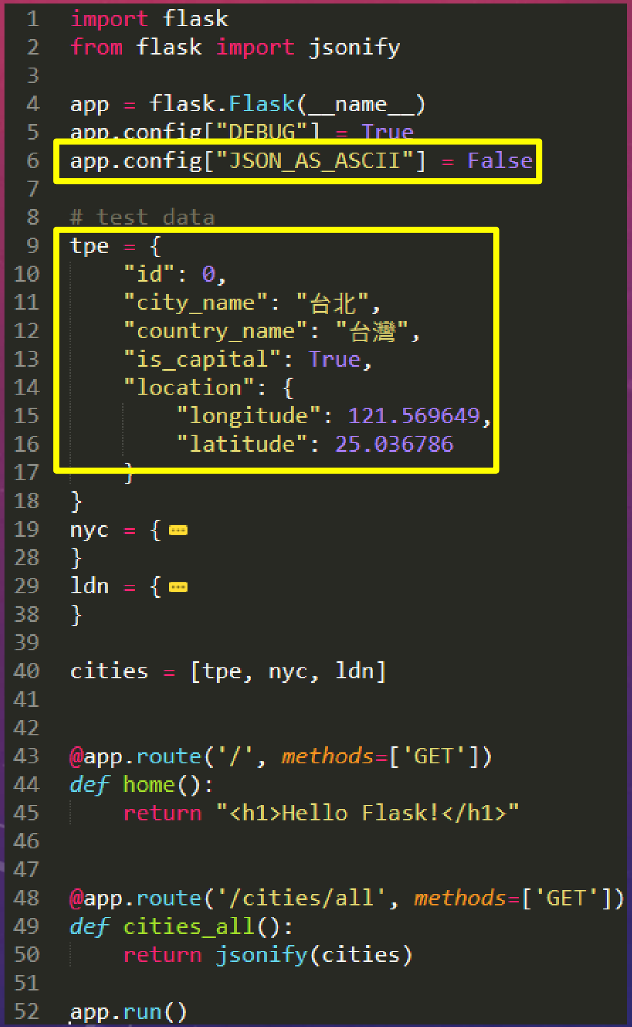
應用程式 Facebook YouTube 愛奇藝

```json
[
  {
    "city_name": "Taipei",
    "country_name": "Taiwan",
    "id": 0,
    "is_capital": true,
    "location": {
      "latitude": 25.036786,
      "longitude": 121.569649
    }
  },
  {
    "city_name": "New York",
    "country_name": "United States",
    "id": 1,
    "is_capital": false,
    "location": {
      "latitude": 40.710405,
      "longitude": -74.004364
    }
  },
  {
    "city_name": "London",
    "country_name": "United Kingdom",
    "id": 2,
    "is_capital": true,
    "location": {
      "latitude": 51.507497,
      "longitude": -0.114089
    }
  }
]
```

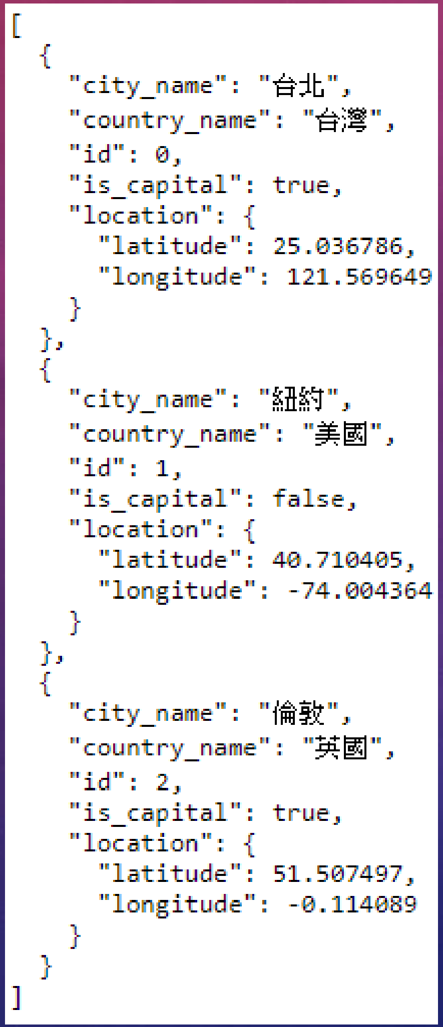


```
(venv) C:\Users\user\Desktop\flask>py GetData.py
 * Serving Flask app "GetData" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 315-344-113
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [23/Oct/2019 09:32:23] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [23/Oct/2019 09:32:36] "GET /cities/all HTTP/1.1" 200 -
```

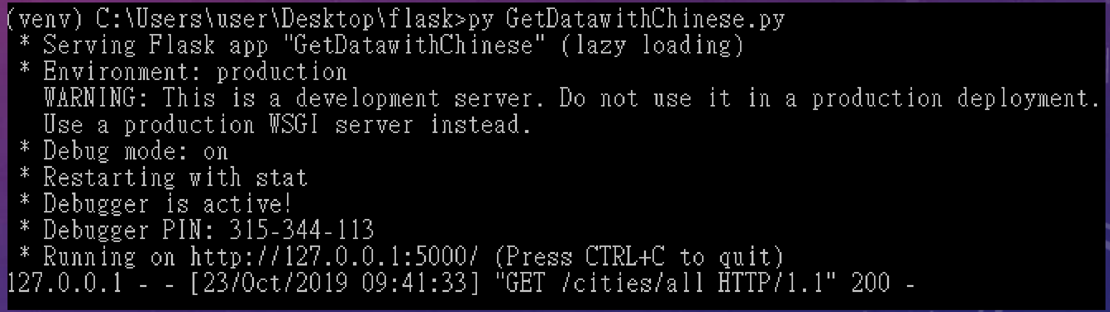
GET DATA WITH CHINESE



```python
import flask
from flask import jsonify

app = flask.Flask(__name__)
app.config["DEBUG"] = True
app.config["JSON_AS_ASCII"] = False

# test data
tpe = {
    "id": 0,
    "city_name": "台北",
    "country_name": "台灣",
    "is_capital": True,
    "location": {
        "longitude": 121.569649,
        "latitude": 25.036786
    }
}
nyc = {
}
ldn = {
}

cities = [tpe, nyc, ldn]


@app.route('/', methods=['GET'])
def home():
    return "<h1>Hello Flask!</h1>"


@app.route('/cities/all', methods=['GET'])
def cities_all():
    return jsonify(cities)

app.run()
```



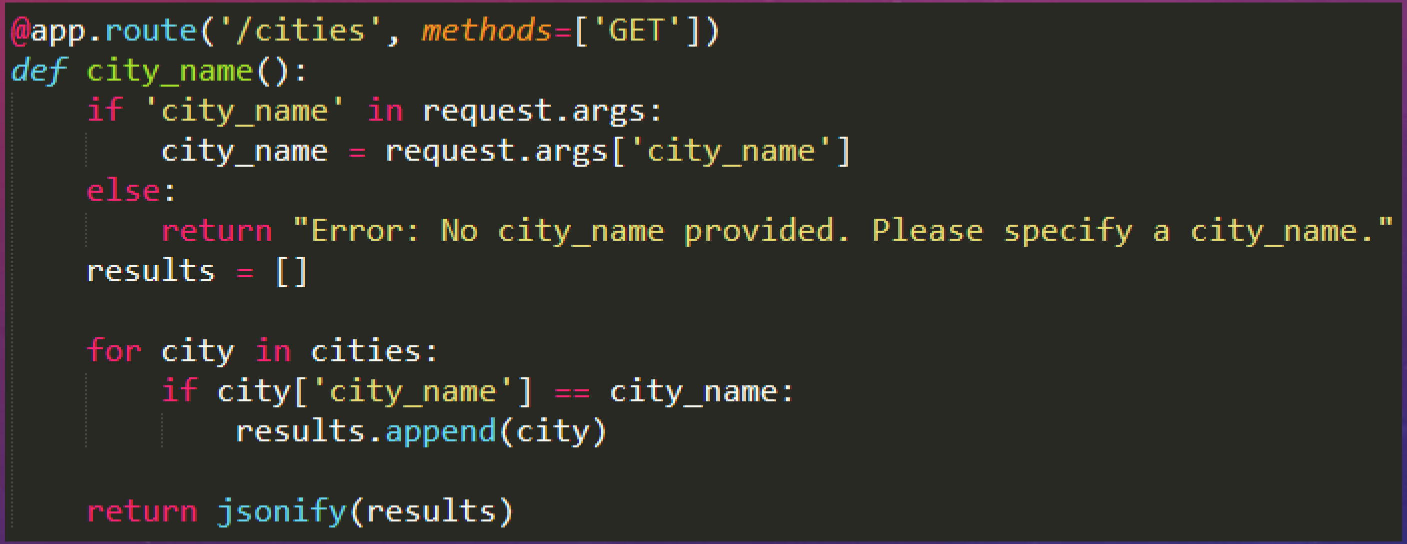
```
[
  {
    "city_name": "台北",
    "country_name": "台灣",
    "id": 0,
    "is_capital": true,
    "location": {
      "latitude": 25.036786,
      "longitude": 121.569649
    }
  },
  {
    "city_name": "紐約",
    "country_name": "美國",
    "id": 1,
    "is_capital": false,
    "location": {
      "latitude": 40.710405,
      "longitude": -74.004364
    }
  },
  {
    "city_name": "倫敦",
    "country_name": "英國",
    "id": 2,
    "is_capital": true,
    "location": {
      "latitude": 51.507497,
      "longitude": -0.114089
    }
  }
]
```



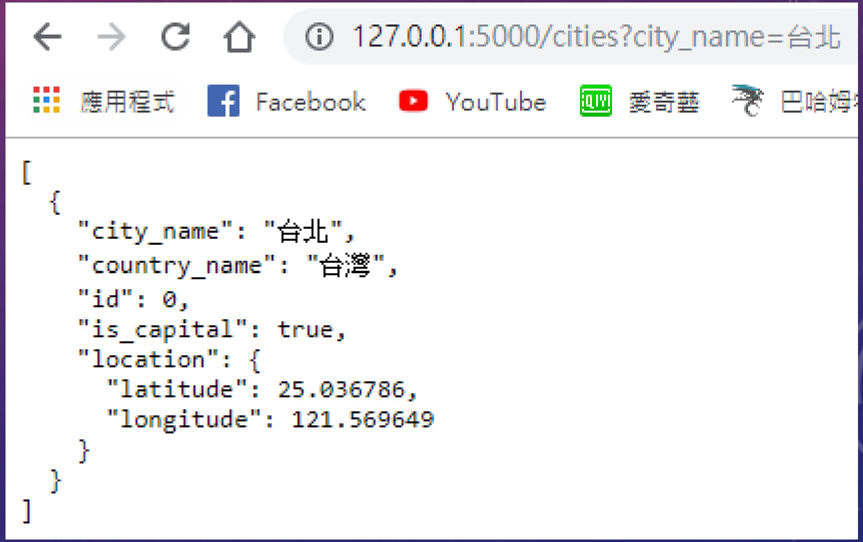
```
(venv) C:\Users\user\Desktop\flask>py GetDatawithChinese.py
 * Serving Flask app "GetDatawithChinese" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 315-344-113
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [23/Oct/2019 09:41:33] "GET /cities/all HTTP/1.1" 200 -
```

GET DATA BY NAME

from flask import request



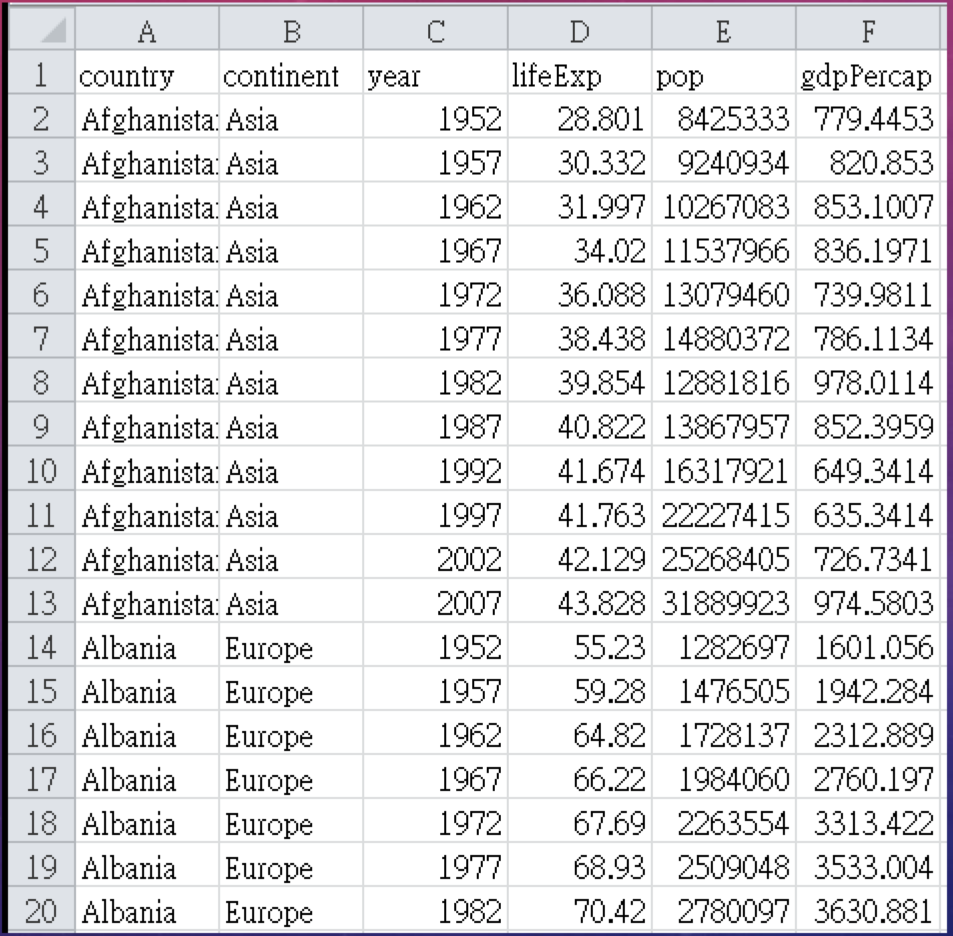
```python
@app.route('/cities', methods=['GET'])
def city_name():
    if 'city_name' in request.args:
        city_name = request.args['city_name']
    else:
        return "Error: No city_name provided. Please specify a city_name."
    results = []

    for city in cities:
        if city['city_name'] == city_name:
            results.append(city)

    return jsonify(results)
```



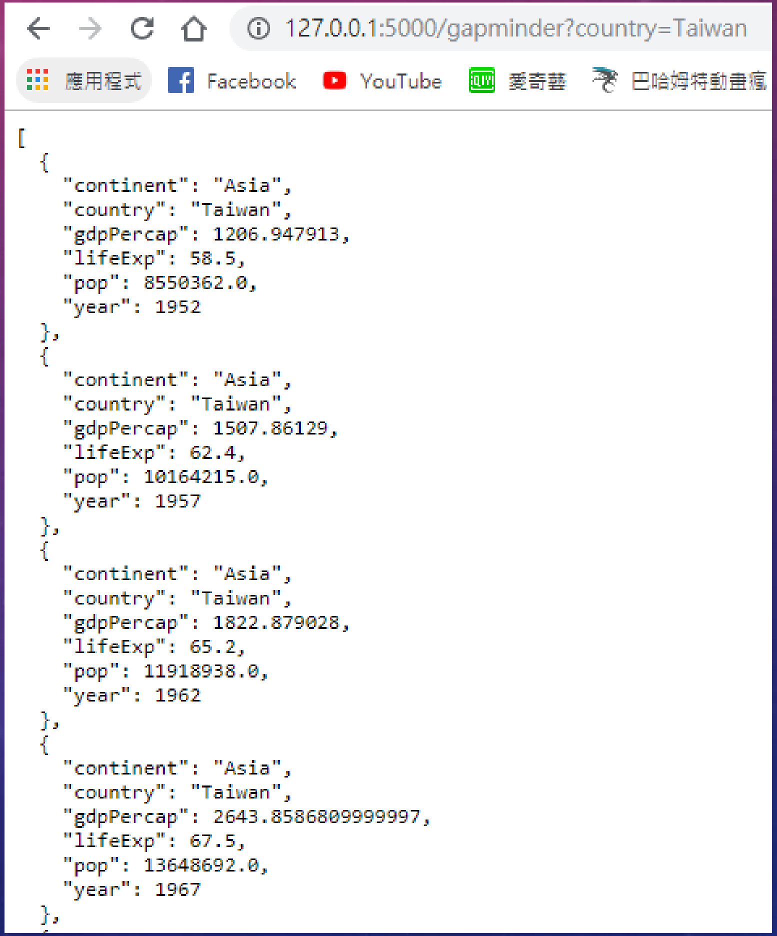
```
[
  {
    "city_name": "台北",
    "country_name": "台灣",
    "id": 0,
    "is_capital": true,
    "location": {
      "latitude": 25.036786,
      "longitude": 121.569649
    }
  }
]
```

GET DATA FROM CSV

gapminder.csv

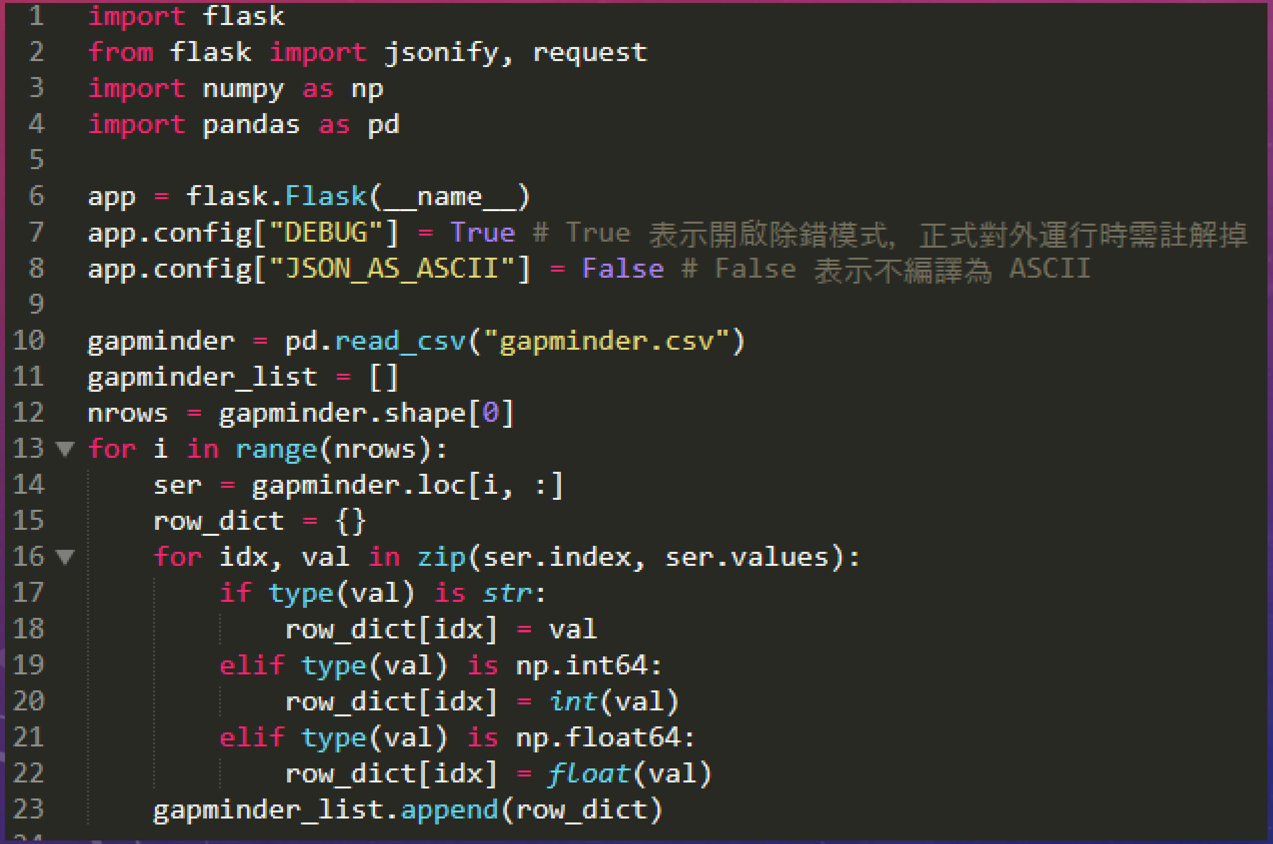


	A	B	C	D	E	F
1	country	continent	year	lifeExp	pop	gdpPercap
2	Afghanista	Asia	1952	28.801	8425333	779.4453
3	Afghanista	Asia	1957	30.332	9240934	820.853
4	Afghanista	Asia	1962	31.997	10267083	853.1007
5	Afghanista	Asia	1967	34.02	11537966	836.1971
6	Afghanista	Asia	1972	36.088	13079460	739.9811
7	Afghanista	Asia	1977	38.438	14880372	786.1134
8	Afghanista	Asia	1982	39.854	12881816	978.0114
9	Afghanista	Asia	1987	40.822	13867957	852.3959
10	Afghanista	Asia	1992	41.674	16317921	649.3414
11	Afghanista	Asia	1997	41.763	22227415	635.3414
12	Afghanista	Asia	2002	42.129	25268405	726.7341
13	Afghanista	Asia	2007	43.828	31889923	974.5803
14	Albania	Europe	1952	55.23	1282697	1601.056
15	Albania	Europe	1957	59.28	1476505	1942.284
16	Albania	Europe	1962	64.82	1728137	2312.889
17	Albania	Europe	1967	66.22	1984060	2760.197
18	Albania	Europe	1972	67.69	2263554	3313.422
19	Albania	Europe	1977	68.93	2509048	3533.004
20	Albania	Europe	1982	70.42	2780097	3630.881

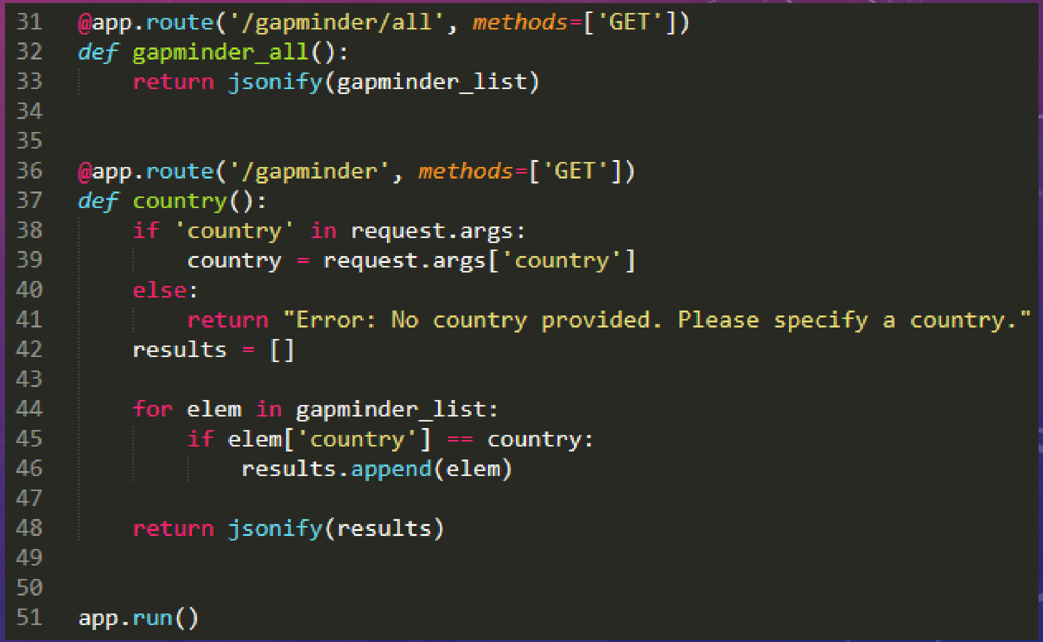


```
[
  {
    "continent": "Asia",
    "country": "Taiwan",
    "gdpPercap": 1206.947913,
    "lifeExp": 58.5,
    "pop": 8550362.0,
    "year": 1952
  },
  {
    "continent": "Asia",
    "country": "Taiwan",
    "gdpPercap": 1507.86129,
    "lifeExp": 62.4,
    "pop": 10164215.0,
    "year": 1957
  },
  {
    "continent": "Asia",
    "country": "Taiwan",
    "gdpPercap": 1822.879028,
    "lifeExp": 65.2,
    "pop": 11918938.0,
    "year": 1962
  },
  {
    "continent": "Asia",
    "country": "Taiwan",
    "gdpPercap": 2643.8586809999997,
    "lifeExp": 67.5,
    "pop": 13648692.0,
    "year": 1967
  },
```

GET DATA FROM CSV



```python
import flask
from flask import jsonify, request
import numpy as np
import pandas as pd

app = flask.Flask(__name__)
app.config["DEBUG"] = True # True 表示開啟除錯模式，正式對外運行時需註解掉
app.config["JSON_AS_ASCII"] = False # False 表示不編譯為 ASCII

gapminder = pd.read_csv("gapminder.csv")
gapminder_list = []
nrows = gapminder.shape[0]
for i in range(nrows):
    ser = gapminder.loc[i, :]
    row_dict = {}
    for idx, val in zip(ser.index, ser.values):
        if type(val) is str:
            row_dict[idx] = val
        elif type(val) is np.int64:
            row_dict[idx] = int(val)
        elif type(val) is np.float64:
            row_dict[idx] = float(val)
    gapminder_list.append(row_dict)
```



```python
@app.route('/gapminder/all', methods=['GET'])
def gapminder_all():
    return jsonify(gapminder_list)


@app.route('/gapminder', methods=['GET'])
def country():
    if 'country' in request.args:
        country = request.args['country']
    else:
        return "Error: No country provided. Please specify a country."
    results = []

    for elem in gapminder_list:
        if elem['country'] == country:
            results.append(elem)

    return jsonify(results)


app.run()
```

FLASK & SQLITE

F
L
A
S
K

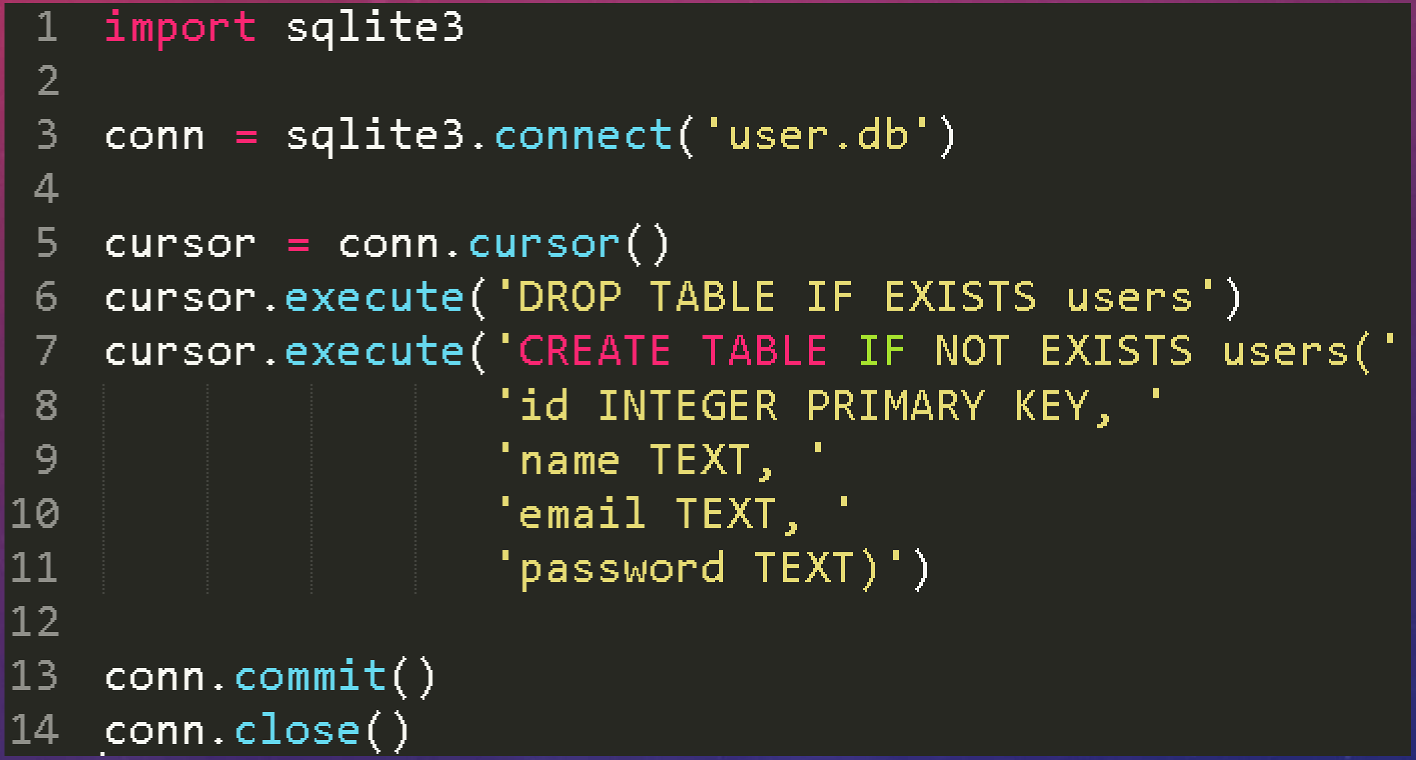
SQLITE

SQLITE

- Python 有許多能用於創建 Web 應用程式和 Web API 的框架，而輕量的 Flask 框架可以勝任Web API 的需求：管理 HTTP 請求和顯示資料內容。

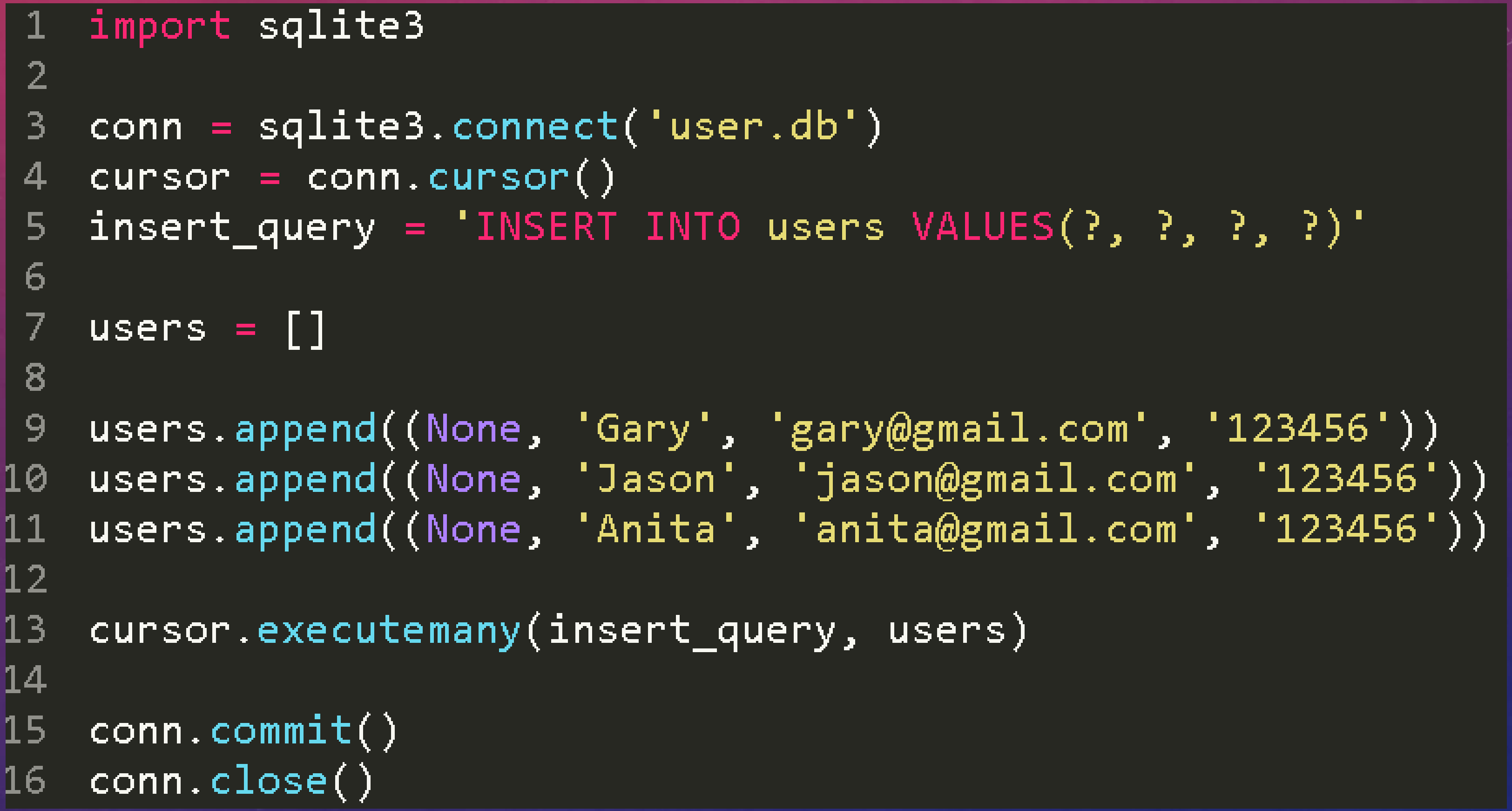
- flask-restful 是針對 restful api 開發的一個flask的套件，建構在 flask 的輕薄短小的基礎下，flask-restful 可以在短短幾行內完成 restful api的開發。

創建SQLITE資料庫與資料表



```python
import sqlite3

conn = sqlite3.connect('user.db')

cursor = conn.cursor()
cursor.execute('DROP TABLE IF EXISTS users')
cursor.execute('CREATE TABLE IF NOT EXISTS users('
               'id INTEGER PRIMARY KEY, '
               'name TEXT, '
               'email TEXT, '
               'password TEXT)')

conn.commit()
conn.close()
```

SQLITE 新增資料

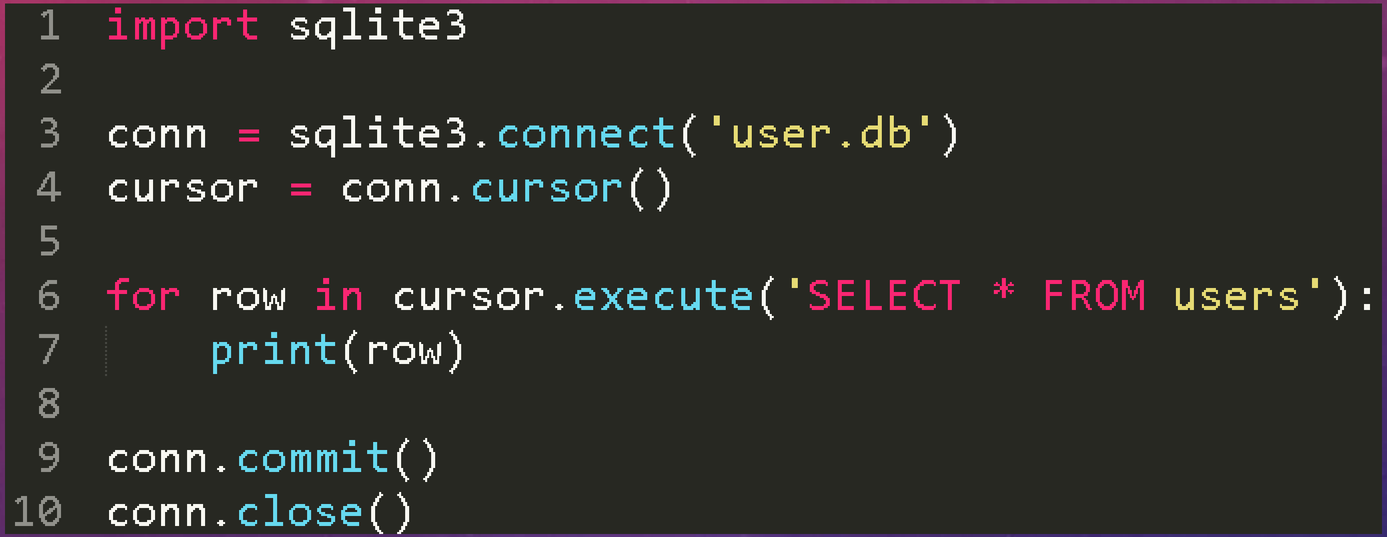


```python
import sqlite3

conn = sqlite3.connect('user.db')
cursor = conn.cursor()
insert_query = 'INSERT INTO users VALUES(?, ?, ?, ?)'

users = []

users.append((None, 'Gary', 'gary@gmail.com', '123456'))
users.append((None, 'Jason', 'jason@gmail.com', '123456'))
users.append((None, 'Anita', 'anita@gmail.com', '123456'))

cursor.executemany(insert_query, users)

conn.commit()
conn.close()
```

SQLITE 更新資料

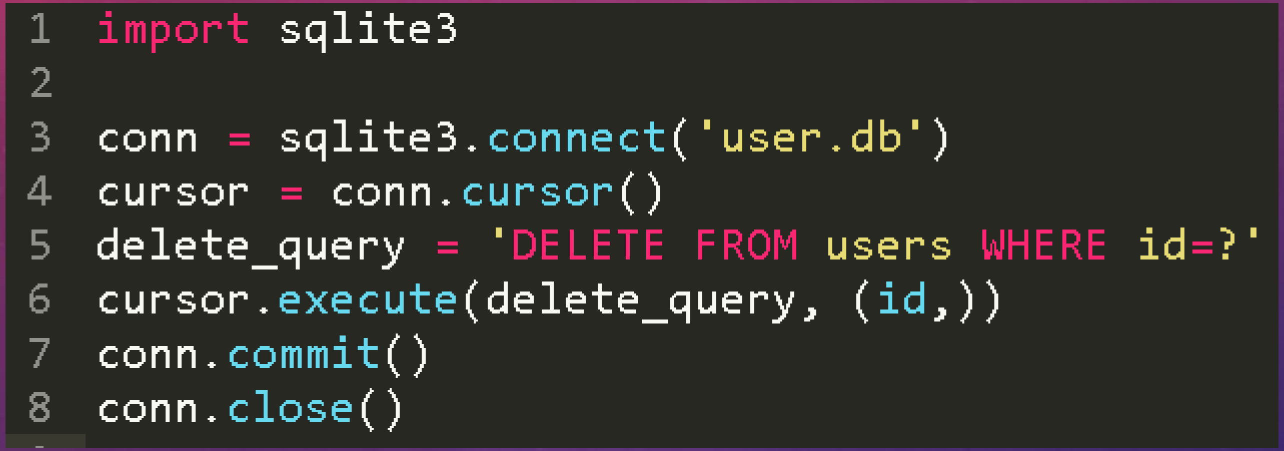
```python
import sqlite3

conn = sqlite3.connect('user.db')
cursor = conn.cursor()
update_query = 'UPDATE users SET name=?, email=?, password=? WHERE id=?'
cursor.execute(update_query, (name, email, password, uid))
conn.commit()
conn.close()
```

SQLITE 查詢資料



```python
import sqlite3

conn = sqlite3.connect('user.db')
cursor = conn.cursor()

for row in cursor.execute('SELECT * FROM users'):
    print(row)

conn.commit()
conn.close()
```

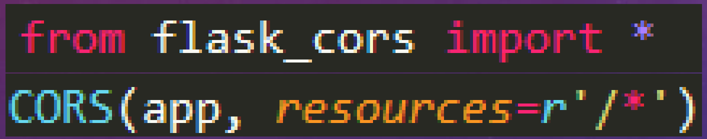
SQLITE 刪除資料



```python
import sqlite3

conn = sqlite3.connect('user.db')
cursor = conn.cursor()
delete_query = 'DELETE FROM users WHERE id=?'
cursor.execute(delete_query, (id,))
conn.commit()
conn.close()
```

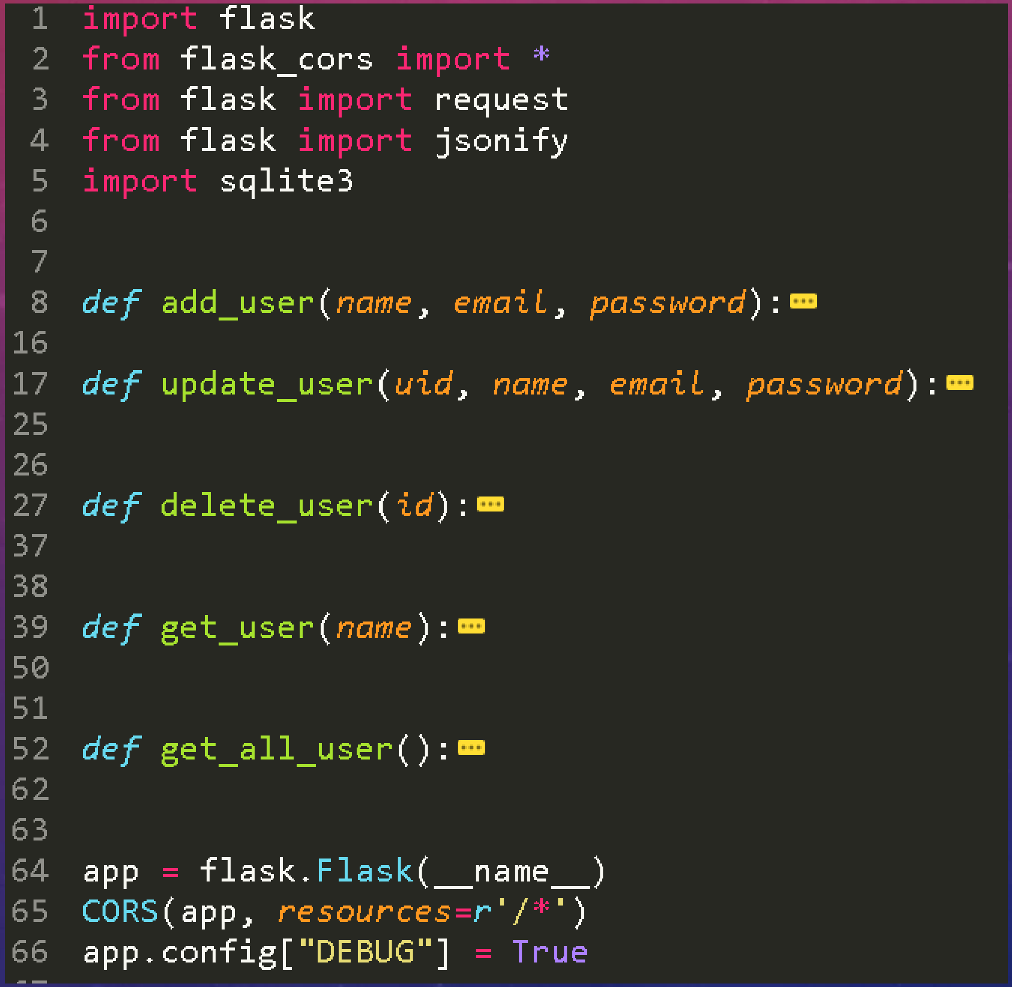
安裝所需模組

1. pip install flask
2. pip install flask-cors



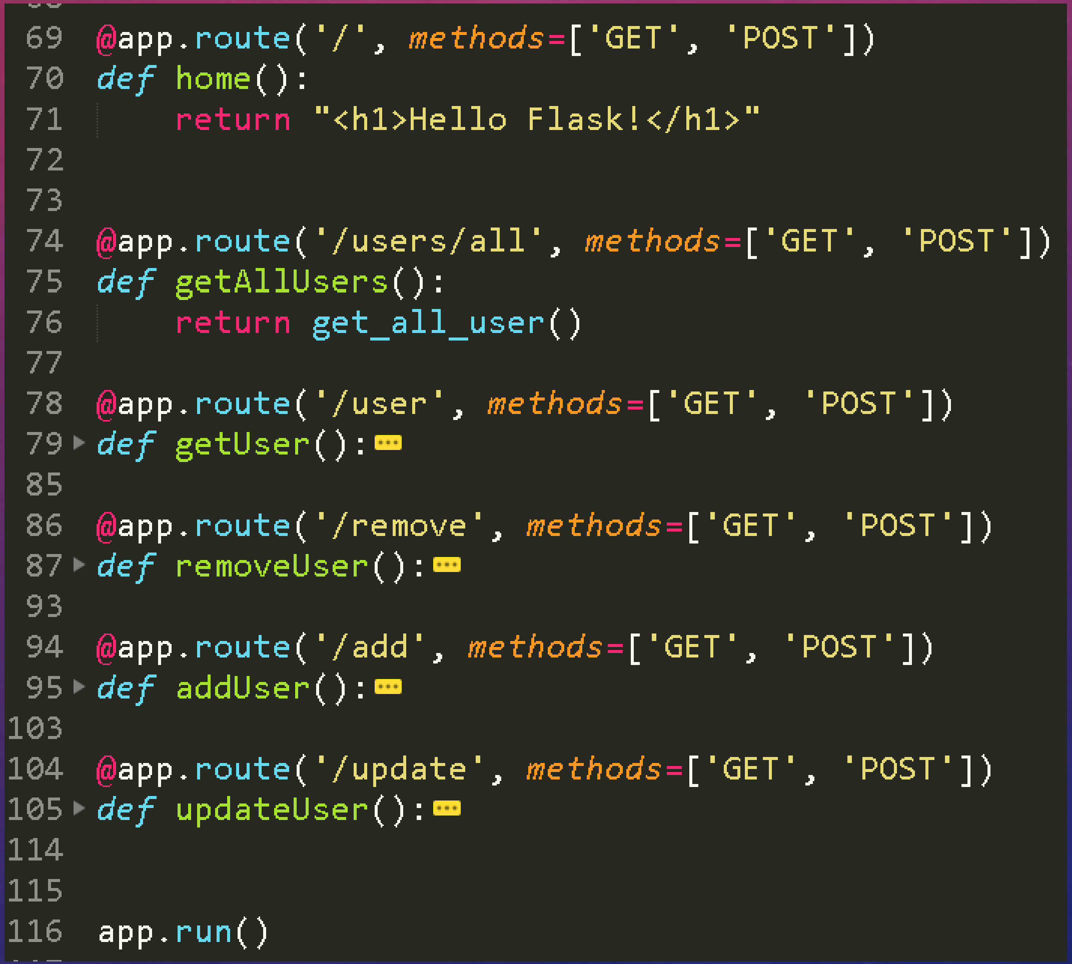
```
from flask_cors import *
CORS(app, resources=r'/*')
```

RESTFUL SQLITE



```python
import flask
from flask_cors import *
from flask import request
from flask import jsonify
import sqlite3


def add_user(name, email, password):

def update_user(uid, name, email, password):


def delete_user(id):


def get_user(name):


def get_all_user():


app = flask.Flask(__name__)
CORS(app, resources=r'/*')
app.config["DEBUG"] = True
```

RESTFUL SQLITE



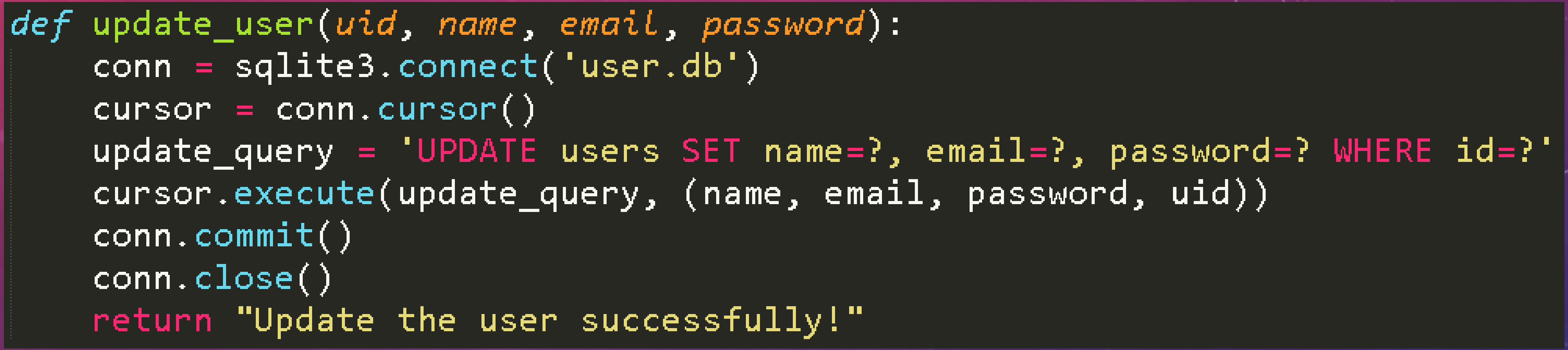
```
@app.route('/', methods=['GET', 'POST'])
def home():
    return "<h1>Hello Flask!</h1>"


@app.route('/users/all', methods=['GET', 'POST'])
def getAllUsers():
    return get_all_user()

@app.route('/user', methods=['GET', 'POST'])
def getUser():

@app.route('/remove', methods=['GET', 'POST'])
def removeUser():

@app.route('/add', methods=['GET', 'POST'])
def addUser():

@app.route('/update', methods=['GET', 'POST'])
def updateUser():


app.run()
```

RESTFUL SQLITE – ADD USER

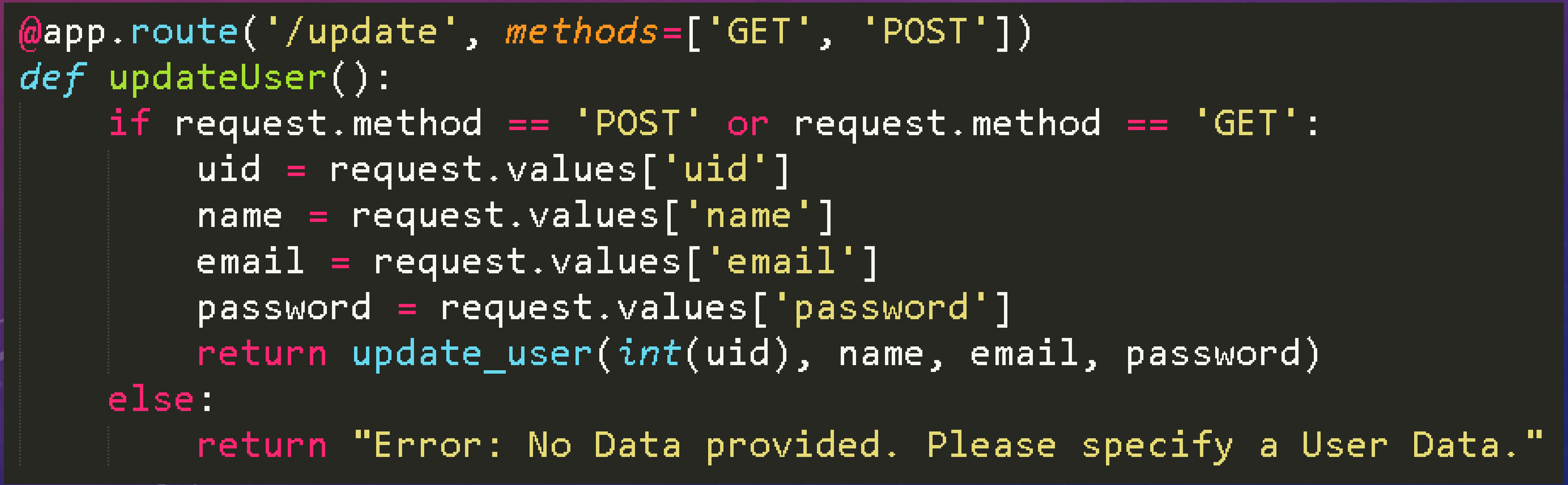
```python
def add_user(name, email, password):
    conn = sqlite3.connect('user.db')
    cursor = conn.cursor()
    insert_query = 'INSERT INTO users VALUES(?, ?, ?, ?)'
    cursor.execute(insert_query, (None, name, email, password))
    conn.commit()
    conn.close()
    return "Add the user successfully!"
```

```python
@app.route('/add', methods=['GET', 'POST'])
def addUser():
    if request.method == 'POST' or request.method == 'GET':
        name = request.values['name']
        email = request.values['email']
        password = request.values['password']
        return add_user(name, email, password)
    else:
        return "Error: No Data provided. Please specify a User Data."
```

RESTFUL SQLITE – UPDATE USER



```python
def update_user(uid, name, email, password):
    conn = sqlite3.connect('user.db')
    cursor = conn.cursor()
    update_query = 'UPDATE users SET name=?, email=?, password=? WHERE id=?'
    cursor.execute(update_query, (name, email, password, uid))
    conn.commit()
    conn.close()
    return "Update the user successfully!"
```



```python
@app.route('/update', methods=['GET', 'POST'])
def updateUser():
    if request.method == 'POST' or request.method == 'GET':
        uid = request.values['uid']
        name = request.values['name']
        email = request.values['email']
        password = request.values['password']
        return update_user(int(uid), name, email, password)
    else:
        return "Error: No Data provided. Please specify a User Data."
```

RESTFUL SQLITE – GET ALL USERS

```python
def get_all_user():
    users = {}
    conn = sqlite3.connect('user.db')
    cursor = conn.cursor()
    query_one_query = 'SELECT * FROM users'
    for item in cursor.execute(query_one_query):
        user = {'name': item[1], 'email': item[2], 'pwd': item[3]};
        users.update({item[0]:user})
    conn.close()
    return users
```

```python
@app.route('/users/all', methods=['GET', 'POST'])
def getAllUsers():
    return get_all_user()
```

CHAT_SERVER_WITH_SELECT & FLASK

F
L
A
S
K

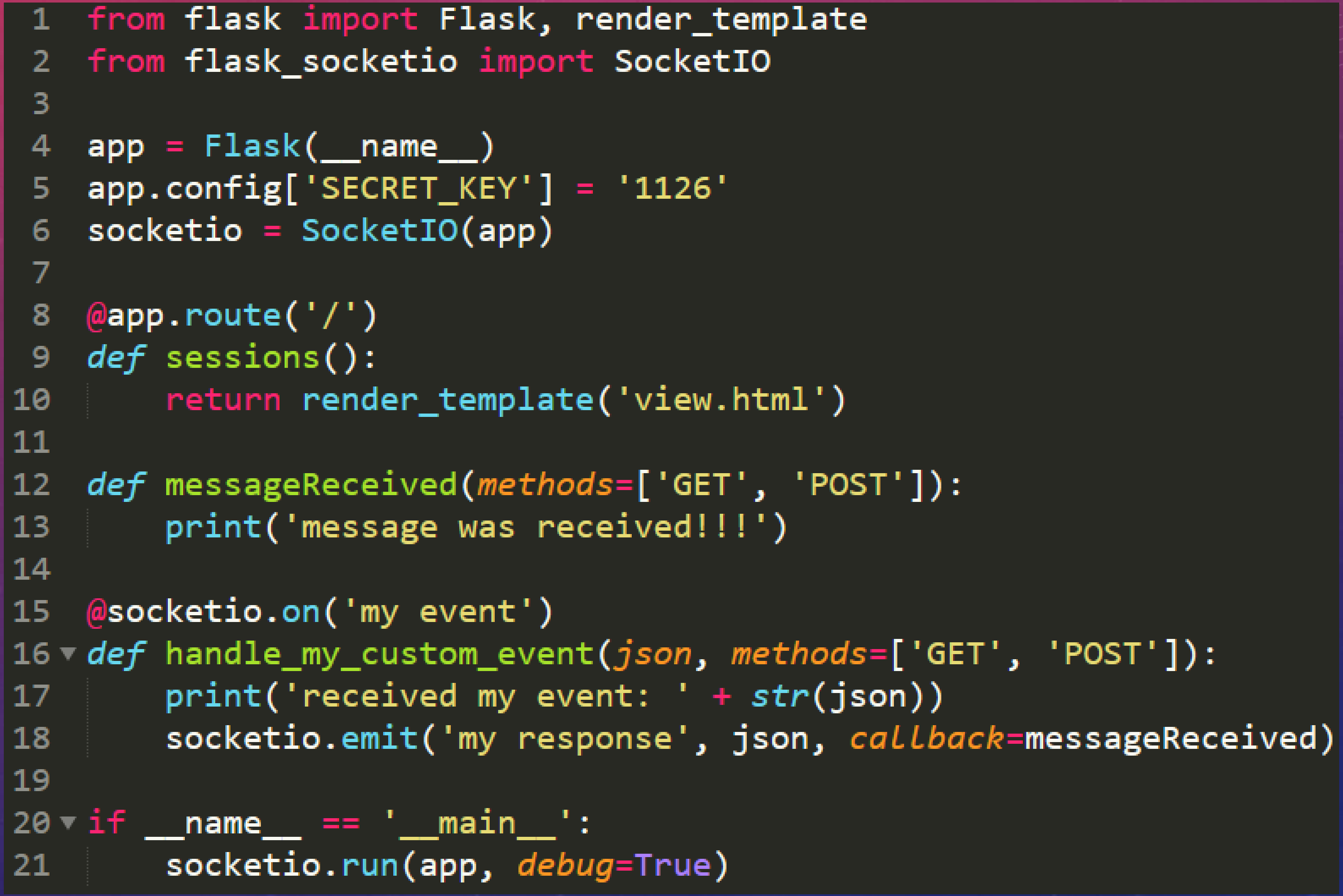
CHAT SERVER

模組

1. pip install flask
2. pip install flask-socketio

APP.PY

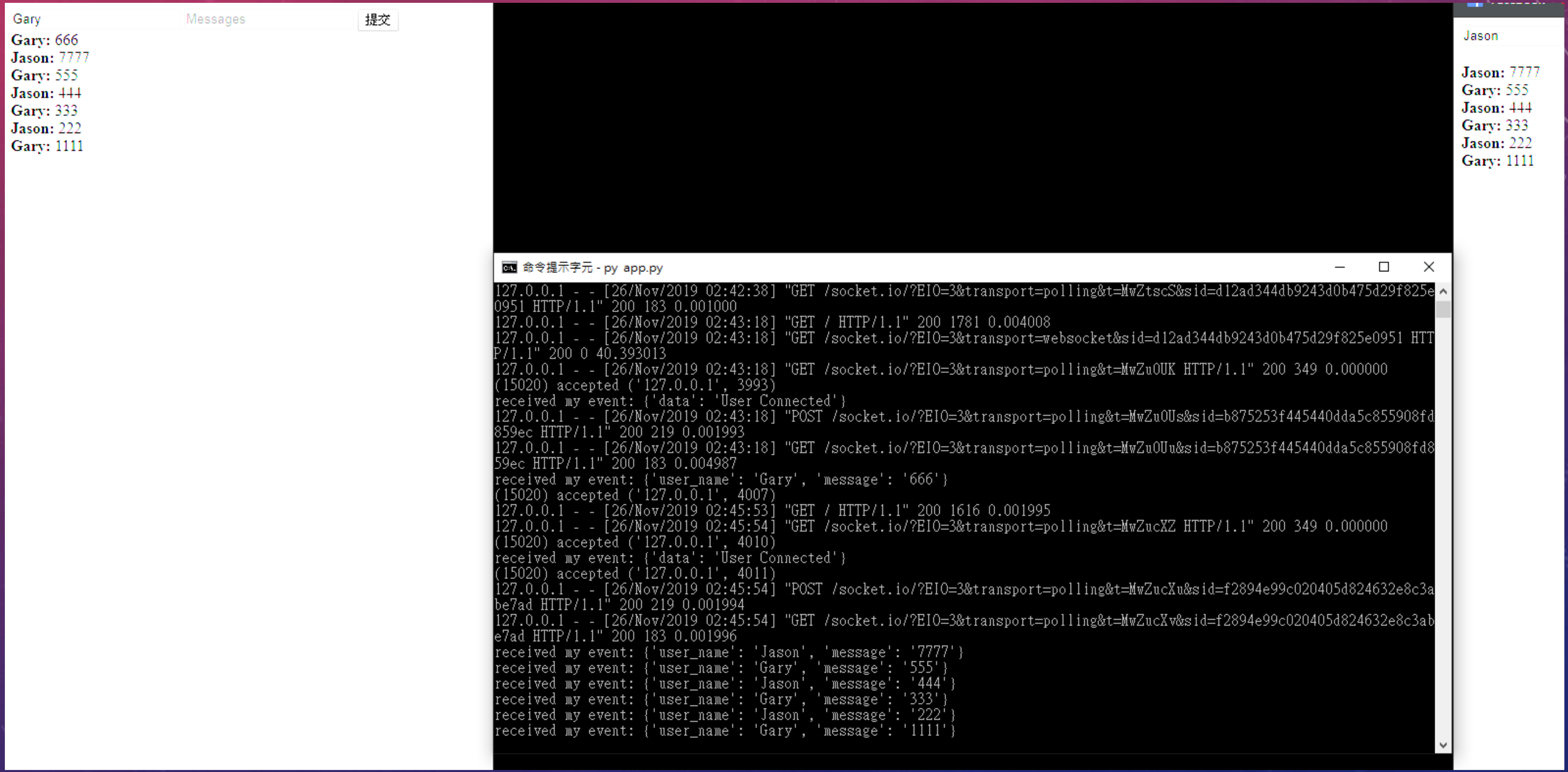
APP.PY



```python
from flask import Flask, render_template
from flask_socketio import SocketIO

app = Flask(__name__)
app.config['SECRET_KEY'] = '1126'
socketio = SocketIO(app)

@app.route('/')
def sessions():
    return render_template('view.html')

def messageReceived(methods=['GET', 'POST']):
    print('message was received!!!')

@socketio.on('my event')
def handle_my_custom_event(json, methods=['GET', 'POST']):
    print('received my event: ' + str(json))
    socketio.emit('my response', json, callback=messageReceived)

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

TEMPLATES/VIEW.HTML

VIEW.HTML

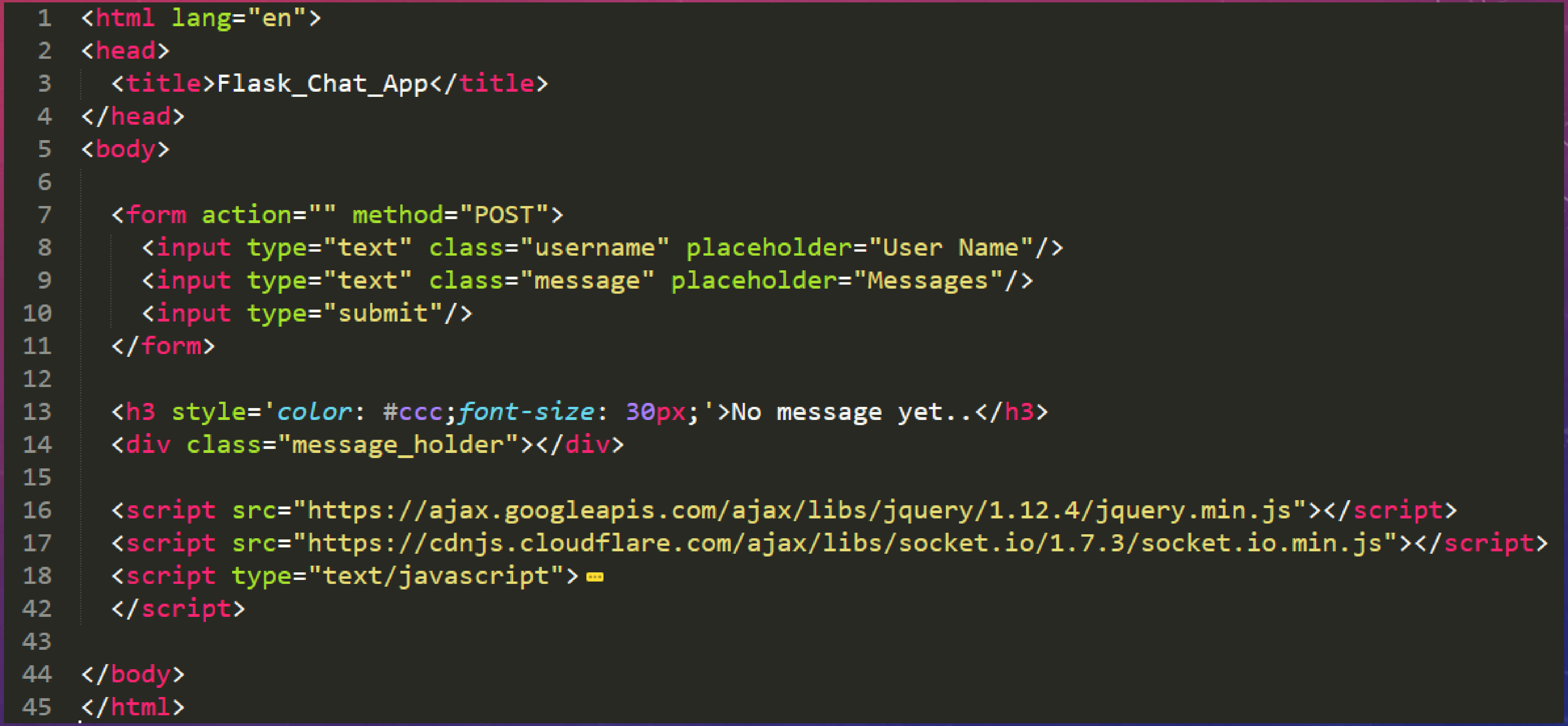


Gary: 666
Jason: 7777
Gary: 555
Jason: 444
Gary: 333
Jason: 222
Gary: 1111

```
127.0.0.1 - - [26/Nov/2019 02:42:38] "GET /socket.io/?EIO=3&transport=polling&t=MwZtscS&sid=d12ad344db9243d0b475d29f825e
0951 HTTP/1.1" 200 183 0.001000
127.0.0.1 - - [26/Nov/2019 02:43:18] "GET / HTTP/1.1" 200 1781 0.004008
127.0.0.1 - - [26/Nov/2019 02:43:18] "GET /socket.io/?EIO=3&transport=websocket&sid=d12ad344db9243d0b475d29f825e0951 HTT
P/1.1" 200 0 40.393013
127.0.0.1 - - [26/Nov/2019 02:43:18] "GET /socket.io/?EIO=3&transport=polling&t=MwZu0UK HTTP/1.1" 200 349 0.000000
(15020) accepted ('127.0.0.1', 3993)
received my event: {'data': 'User Connected'}
127.0.0.1 - - [26/Nov/2019 02:43:18] "POST /socket.io/?EIO=3&transport=polling&t=MwZu0Us&sid=b875253f445440dda5c855908fd
859ec HTTP/1.1" 200 219 0.001993
127.0.0.1 - - [26/Nov/2019 02:43:18] "GET /socket.io/?EIO=3&transport=polling&t=MwZu0Uu&sid=b875253f445440dda5c855908fd8
59ec HTTP/1.1" 200 183 0.004987
received my event: {'user_name': 'Gary', 'message': '666'}
(15020) accepted ('127.0.0.1', 4007)
127.0.0.1 - - [26/Nov/2019 02:45:53] "GET / HTTP/1.1" 200 1616 0.001995
127.0.0.1 - - [26/Nov/2019 02:45:54] "GET /socket.io/?EIO=3&transport=polling&t=MwZucXZ HTTP/1.1" 200 349 0.000000
(15020) accepted ('127.0.0.1', 4010)
received my event: {'data': 'User Connected'}
(15020) accepted ('127.0.0.1', 4011)
127.0.0.1 - - [26/Nov/2019 02:45:54] "POST /socket.io/?EIO=3&transport=polling&t=MwZucXu&sid=f2894e99c020405d824632e8c3a
be7ad HTTP/1.1" 200 219 0.001994
127.0.0.1 - - [26/Nov/2019 02:45:54] "GET /socket.io/?EIO=3&transport=polling&t=MwZucXv&sid=f2894e99c020405d824632e8c3ab
e7ad HTTP/1.1" 200 183 0.001996
received my event: {'user_name': 'Jason', 'message': '7777'}
received my event: {'user_name': 'Gary', 'message': '555'}
received my event: {'user_name': 'Jason', 'message': '444'}
received my event: {'user_name': 'Gary', 'message': '333'}
received my event: {'user_name': 'Jason', 'message': '222'}
received my event: {'user_name': 'Gary', 'message': '1111'}
```

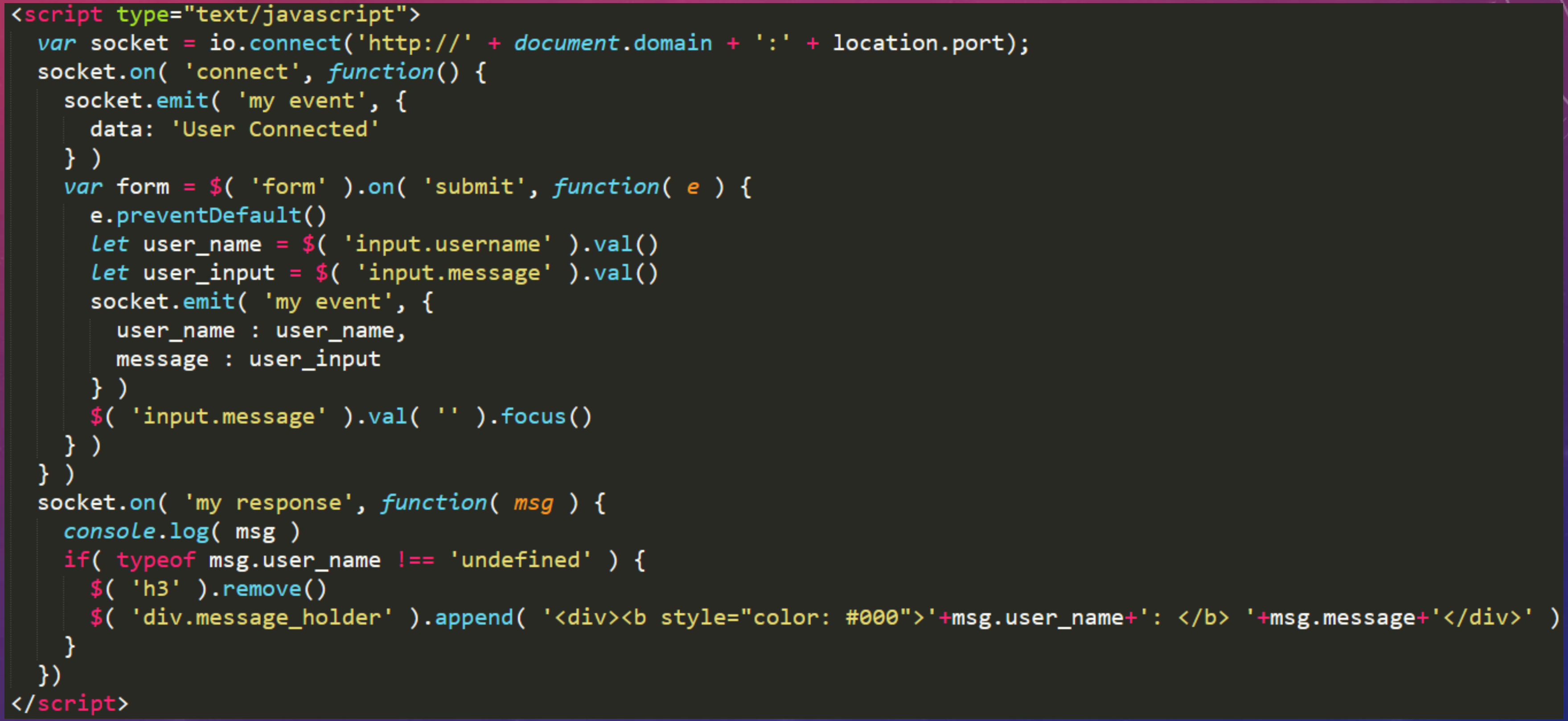
Jason: 7777
Gary: 555
Jason: 444
Gary: 333
Jason: 222
Gary: 1111

VIEW.HTML



```html
<html lang="en">
<head>
  <title>Flask_Chat_App</title>
</head>
<body>

  <form action="" method="POST">
    <input type="text" class="username" placeholder="User Name"/>
    <input type="text" class="message" placeholder="Messages"/>
    <input type="submit"/>
  </form>

  <h3 style='color: #ccc;font-size: 30px;'>No message yet..</h3>
  <div class="message_holder"></div>

  <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/1.7.3/socket.io.min.js"></script>
  <script type="text/javascript">
  </script>

</body>
</html>
```

VIEW.HTML



```
<script type="text/javascript">
  var socket = io.connect('http://' + document.domain + ':' + location.port);
  socket.on( 'connect', function() {
    socket.emit( 'my event', {
      data: 'User Connected'
    } )
    var form = $( 'form' ).on( 'submit', function( e ) {
      e.preventDefault()
      let user_name = $( 'input.username' ).val()
      let user_input = $( 'input.message' ).val()
      socket.emit( 'my event', {
        user_name : user_name,
        message : user_input
      } )
      $( 'input.message' ).val( '' ).focus()
    } )
  } )
  socket.on( 'my response', function( msg ) {
    console.log( msg )
    if( typeof msg.user_name !== 'undefined' ) {
      $( 'h3' ).remove()
      $( 'div.message_holder' ).append( '<div><b style="color: #000">'+msg.user_name+': </b> '+msg.message+'</div>' )
    }
  })
</script>
```

THANK YOU