Python Network Programming

Edge & Cloud Device Communication

【110上】嵌入式系統技術實驗課程

TA: 陳翰群 hanz1211.ee09@nycu.edu.tw

Network Protocol

- There are 2 types of methods to communicate through network
  - Stateless: HTTP, RESTful API
    - Request webpage
    - Submit data, form
    - Download files
    - Almost every modern website and APIs are designed this way
  - Stateful: WebSocket, SocketIO
    - Online chat room
    - Real-time streaming (this is another technology called WebRTC)

RESTful

- REST: Representational State Transfer
  - A networking programming style designed 20 years ago
  - Use URLs & HTTP verb (request methods) to specify actions
    - GET
    - POST
    - DELETE
    - PUT
    - PATCH

RESTful

- URL

$$\overset{\text{Protocol: HTTP Secure}}{\text{https}}\text{://}\underbrace{\text{e3.nycu.edu.tw}}_{\text{Domain Name or IP address}}\text{:}\overset{\text{Port number (HTTPS default is 443, can be neglected)}}{\text{443}}\underbrace{\text{/course/view.php}}_{\text{Resource Path}}\text{?}\underbrace{\text{id=29640}}_{\text{Query String}}$$

- Uniform Interface
  - Resource ID inside requests
  - NYCU E3 as example:
    - All courses info are under /course/view.php, you can change id in query string to see other courses
    - All UI icons are under /theme/image.php

Make HTTP GET Request

- In Python, use requests library

```
pip install requests
```

- json_request.py

```
import requests

if __name__ == '__main__':
    response = requests.get('https://jsonplaceholder.typicode.com/users')
    print(response.status_code)
    my_users = response.json()
    print( len(my_users))
```

  - This will perform a GET request to that URL, the response is an object
  - The .status_code should be 200 (OK) if success
  - Use .json() to obtain response data as Python object, in this case you will get a list of length 10, every member is a dict contains a dummy user data
- If you want to try different data, you can see their API definition on https://jsonplaceholder.typicode.com/

Make HTTP POST Request

- json_request.py

```python
my_data = {
    …
}
response = requests.post('https://jsonplaceholder.typicode.com/posts', json=my_data)
print(response.status_code)
print(response.json())
```

- This will perform a POST request to that URL, the server will echo the data you sent
- By parameter json=my_data, the library will convert Python List & Dict to JSON and send
  - Most Python basic data structure can be directly convert to json format
- The .status_code should be 201 (Created) if success
- Use .json() to obtain response data as Python object, in this case you will get a list of length 10, every member is a dict contains a dummy user data

Setup a REST API Server

- Install server package

```
pip install fastapi
```

- Install ASGI server

```
pip install uvicorn
```

  - Compared to traditional WSGI (Python Web Server Gateway Interface), ASGI (Asynchronous Server Gateway Interface) can make async-capable Python web app
  - With gateway interface, our app will not be block if it’s handling other requests

Run API Server

- api_server.py
- Define your server and its functions

```
from fastapi import FastAPI

app = FastAPI()
```

```
@app.get("/")
def read_root():
    return {"Hello": "World"}
```

Python decorator, this tell app how to handle GET request at http://localhost:8000/

- Start app with Uvicorn ASGI server, running on http://localhost:8000

```
import uvicorn

if __name__ == '__main__':
    uvicorn.run(app, host="localhost", port=8000)
```

  - Visit localhost:8000 with your browser, you shall see {“Hello”: “World”} from server
  - The default browser HTTP action is GET

Handle Query String

- api_server.py
- This is a simple calculator on server
  - You can use multiple decorator on one function
- Here we specify parameter type, which will handle invalid input for us
- For example, if we want to calculate 123*456
  - http://localhost:8000/calc/mul?a=123&b=456
    - This will use the first path, and pass a and b as query parameter
  - http://localhost:8000/calc/mul/123/456
    - This will use the second path, a and b are a part of URL path

```python
@app.get("/calc/{operation}")
@app.get("/calc/{operation}/{a}/{b}")
def calculator(operation: str, a: int, b: int):
    if operation == 'add':
        return a + b
    elif operation == 'sub':
        return a - b
    elif operation == 'mul':
        return a * b
    elif operation == 'div':
        return a / b
    else:
        return "Unknown operation"
```

Optional Field

- api_server.py
- This is an API return a type of food
- Here we can specify parameter type as optional
  - Remember to import from typing library

```python
from typing import Optional
```

- For example,
  - http://localhost:8000/food/
    - This will get random food, since selection is none by default
  - http://localhost:8000/food/avocado
    - This only return avocado 🥑

```python
@app.get("/food/")
@app.get("/food/{selection}")
def get_random_food(selection: Optional[str] = None):
    if selection is None:
        name, emoji = random.choice(list(foods.items()))
    else:
        name = selection
        emoji = foods[name]
    return {
        "message": f"Have some {name} and enjoy 🥰",
        "food": emoji,
    }
```

Handle POST Requests

- api_server.py
- This will add new food to the server data
- Here we specify the parameter type as dict
  - In older Python, you may need to use Dict instead

```python
from typing import Dict
```

```python
@app.post("/food", status_code=201)
def create_new_food(msg: dict):
    foods[msg['name']] = msg['emoji']

    return {
        "message": f"Successfully add {msg['name']} to menu",
        "number of food": len(foods)
    }
```

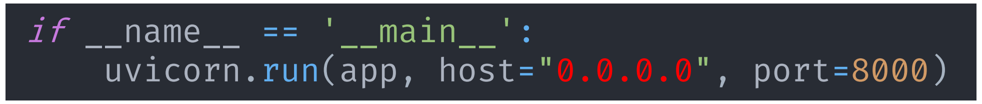
- api_client.py
  - The last example add hamburger to server's menu using requests.post

```python
msg = {
    "name": "hamburger",
    "emoji": "🍔",
}
response = requests.post(f'{SERVER_URL}/food', json=msg)
print(f'Server response: {response.status_code}, {response.json()}')
```

  - You will see server status code 201 instead of 200 because we specified in @app.post decorator

Expose Server to the Internet

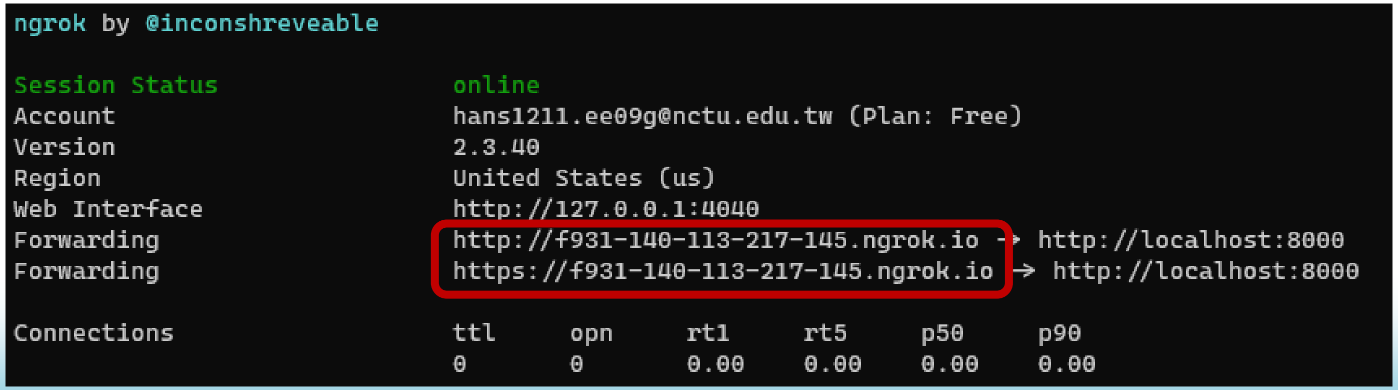
- If you want to access server on other machines, make sure to change the IP to 0.0.0.0 in server code



```
if __name__ == '__main__':
    uvicorn.run(app, host="0.0.0.0", port=8000)
```

- This will accept all incoming request at port 8000
- If the other device is in the same network, you can access this server with IP 192.168.x.x
  - This is local network IP address, which is not visible to outside
  - Use command ipconfig on Windows, ifconfig on MacOS/Linux to check your local IP
- If you want to access this server outside local network, use ngrok to expose local port
  - https://ngrok.com/download
  - After install, use command in terminal: ngrok http 8000 to receive a URL that is available everywhere
  - If you are not sign up, the service only last for 2 hours, after that, URL will be expire
  - You can signup for free and receive a token string, to have no time limit

```
ngrok authtoken <yourtoken>
```



```
ngrok by @inconshreveable

Session Status                online
Account                       hans1211.ee09g@nctu.edu.tw (Plan: Free)
Version                       2.3.40
Region                        United States (us)
Web Interface                 http://127.0.0.1:4040
Forwarding                    http://f931-140-113-217-145.ngrok.io -> http://localhost:8000
Forwarding                    https://f931-140-113-217-145.ngrok.io -> http://localhost:8000

Connections                   ttl     opn     rt1     rt5     p50     p90
                              0       0       0.00    0.00    0.00    0.00
```