Practice #1 Report - 20170481 Yuseung Lee (이유승)

“client.py” - Code Implementation

```
### 1. Request message from "/hello_world" ###
# Receive message by using GET method
msg = requests.get(url + "/hello_world").json()
print(msg["message"])
```

: Receive JSON response from the server by using “requests.get()” method, and use “json()” method to turn the JSON response into a Python dictionary.

```
### 2. Request "/hash" for SHA256 hashed value ###
# Put 'name' variable into a key-value pair
params = {'name': name}
```

: Put the ‘name’ variable into a key-value pair by using the Python dictionary.

```
# Receive hash value by using GET method
hash = requests.get(url + "/hash", params=params).json()
hash_res = hash["result"]
print(hash_res)
```

: Receive JSON response from the server by using “requests.get()” method. Add ‘name’ as a parameter by passing the above dictionary to the “params” argument in get() method. Use “json()” to turn the JSON response into a Python dictionary.

```
### 3. Request "/collatz" for next collatz number ###
# Put variables into key-value pairs
data = {"name": name, "hash": hash_res, "number": number}
```

: Put variables into key-value pairs by using the Python dictionary. Pass

```
# Receive collatz number by using POST method
collatz = requests.post(url + "/collatz", data=data).json()
```

: Receive JSON response from the server by using “requests.post()” method. Send data to the server by passing the above dictionary to the “data” parameter in “post()” method. Then turn the JSON response into a Python dictionary by using “json()”.

```
# Error Handling: If E1 or E2, print error message
if "error" in collatz:
   print(collatz['error'])
   res = 1
else:
   res = collatz['result']
   print(res)
```

: Check whether an error occurred in the server by checking if “error” key exists in the collatz dictionary. If an error exists, print out the error message.

```
# Keep requesting the next collatz number until the result is 1
while res != 1:
   data["number"] = res
   collatz = requests.post(url + "/collatz", data=data).json()
   res = collatz['result']
   print(res)
```

: Repeat the POST method until the received collatz number is 1.

“server.py” - Code Implementation

```
### Route function for "/hello_world" ###
@app.route('/hello_world', methods=['GET'])
def route_hello_world():
   return json.dumps({'message': 'hello_world'})
```

: Route function for “/hello_world”. If the client requests a GET method, return ‘hellow_world’ message in JSON format.

```
### Route function for "/hash" ###
@app.route('/hash', methods=['GET'])
def route_hash():
   name = request.args.get("name")
   HASHED_VALUE = hashlib.sha256(name.encode()).hexdigest()
   return json.dumps({'result': HASHED_VALUE})
```

: Route function for “/hash”. When the client requests a GET method with the ‘name’ argument, return the SHA256 hash value of that name in JSON format.

```
### Route function for "/collatz" ###
@app.route('/collatz', methods=['POST'])
def route_collatz():
   # Receive data from the client
   name = request.form.get("name")
   new_hash = request.form.get("hash")
   number = request.form.get("number")
```

: Route function for “/collatz”. The client requests a POST method with three parameters ‘name’, ‘hash’, and ‘number’. Store each value in a new variable.

```
   # Handle error E1: name and hash value do not match
   HASH_RES = hashlib.sha256(name.encode()).hexdigest()
   if HASH_RES != new_hash:
       return json.dumps({'error': 'HASH NOT MATCHED'})
```

: Error handling for E1. Derive the SHA256 hash value of ‘name’ and store it as HASH_RES. Then compare HASH_RES with the hash value from the client (new_hash). If these two are different, return an error message.

```
  # Handle error E2:
   if not number.isdigit():
       return json.dumps({'error': 'NUMBER NOT INTEGER'})
```

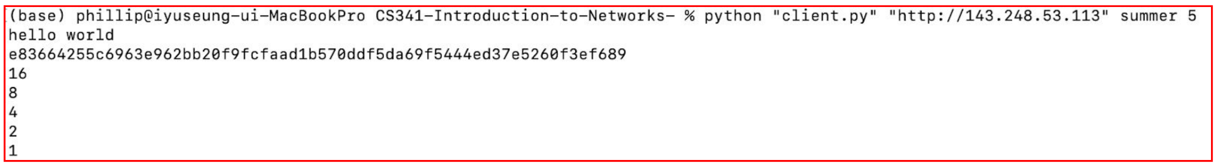
: Error handling for E2. Use isdigit() method to check whether ‘number’ is in an integer form. If not, then return an error message.

```
   new_number = int(number)

   if new_number % 2 == 1:
       NEXT_COLLATZ_NUMBER = 3 * new_number + 1
   else:
       NEXT_COLLATZ_NUMBER = new_number / 2

   return json.dumps({'result': int(NEXT_COLLATZ_NUMBER)})
```

: Since “request.form.get()” method receives data in string type, we need to change it to integer. Then do the calculations for the next collatz number. Return the next collatz number in JSON format.

Executing “client.py” on the given server, “http://143.248.53.113”

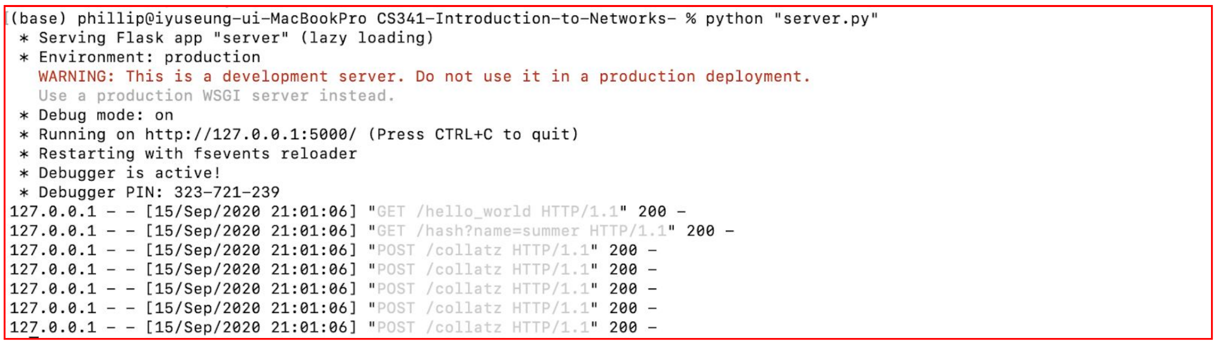
“client.py”



```
(base) phillip@iyuseung-ui-MacBookPro CS341-Introduction-to-Networks- % python "client.py" "http://143.248.53.113" summer 5
hello world
e83664255c6963e962bb20f9fcfaad1b570ddf5da69f5444ed37e5260f3ef689
16
8
4
2
1
```

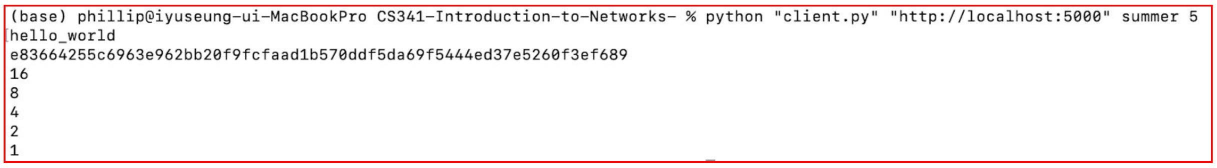
Executing “client.py” on localhost server, “http://localhost:5000”

“server.py”



```
(base) phillip@iyuseung-ui-MacBookPro CS341-Introduction-to-Networks- % python "server.py"
 * Serving Flask app "server" (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
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
 * Restarting with fsevents reloader
 * Debugger is active!
 * Debugger PIN: 323-721-239
127.0.0.1 - - [15/Sep/2020 21:01:06] "GET /hello_world HTTP/1.1" 200 -
127.0.0.1 - - [15/Sep/2020 21:01:06] "GET /hash?name=summer HTTP/1.1" 200 -
127.0.0.1 - - [15/Sep/2020 21:01:06] "POST /collatz HTTP/1.1" 200 -
127.0.0.1 - - [15/Sep/2020 21:01:06] "POST /collatz HTTP/1.1" 200 -
127.0.0.1 - - [15/Sep/2020 21:01:06] "POST /collatz HTTP/1.1" 200 -
127.0.0.1 - - [15/Sep/2020 21:01:06] "POST /collatz HTTP/1.1" 200 -
127.0.0.1 - - [15/Sep/2020 21:01:06] "POST /collatz HTTP/1.1" 200 -
```

“client.py”



```
(base) phillip@iyuseung-ui-MacBookPro CS341-Introduction-to-Networks- % python "client.py" "http://localhost:5000" summer 5
hello_world
e83664255c6963e962bb20f9fcfaad1b570ddf5da69f5444ed37e5260f3ef689
16
8
4
2
1
```