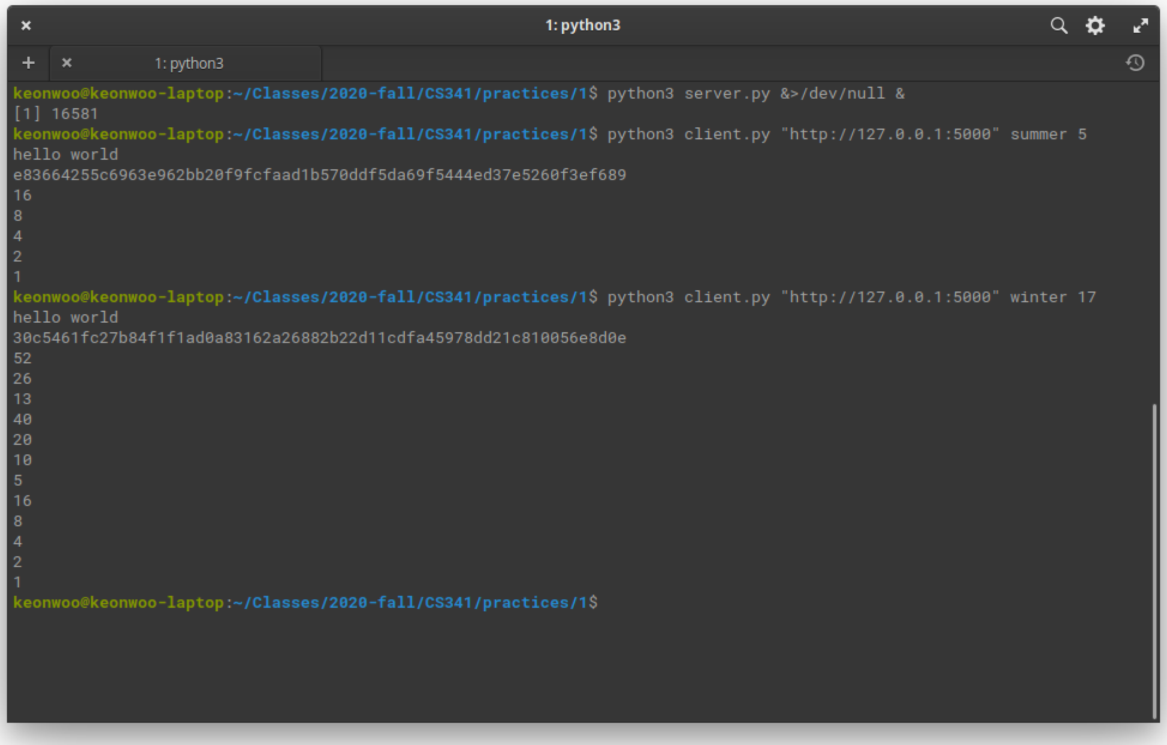
**20170058 Keonwoo Kim** · September 3, 2020
CS341 Introduction to Computer Networks

# Practice #1



```
keonwoo@keonwoo-laptop:~/Classes/2020-fall/CS341/practices/1$ python3 server.py &>/dev/null &
[1] 16581
keonwoo@keonwoo-laptop:~/Classes/2020-fall/CS341/practices/1$ python3 client.py "http://127.0.0.1:5000" summer 5
hello world
e83664255c6963e962bb20f9fcfaad1b570ddf5da69f5444ed37e5260f3ef689
16
8
4
2
1
keonwoo@keonwoo-laptop:~/Classes/2020-fall/CS341/practices/1$ python3 client.py "http://127.0.0.1:5000" winter 17
hello world
30c5461fc27b84f1f1ad0a83162a26882b22d11cdfa45978dd21c810056e8d0e
52
26
13
40
20
10
5
16
8
4
2
1
keonwoo@keonwoo-laptop:~/Classes/2020-fall/CS341/practices/1$
```

Figure 1: The screenshot of working server and client.

The following is the code of the file `server.py`, with detailed and descriptive comments added.

```python
# server.py

from flask import Flask, render_template, request
import json
import hashlib

# create a Flask app
```

```python
app = Flask(__name__)

@app.route('/hello_world', methods=['GET'])
def route_hello_world():
    # send the hello world message to the client
    return json.dumps({'message': 'hello world'})

@app.route('/hash', methods=['GET'])
def route_hash():
    # get the name from the GET url parameters
    name = request.args.get('name')

    # hash the name parameter
    hash = hash_sha256(name)

    # send it to the client
    return json.dumps({'result': hash})

@app.route('/collatz', methods=['POST'])
def route_collatz():
    # get the name and the hash value from the POST data
    name = request.form.get('name')
    hash_in_request = request.form.get('hash')

    # hash the name parameter and compare with the requested one
    hash = hash_sha256(name)
    if hash != hash_in_request:
        return json.dumps({'error': 'HASH NOT MATCHED'})

    # get the number from the POST data
    number = request.form.get('number')
    # try to convert it to an integer
    try:
        # if it is an integer,
        # send the result of the collatz function to the client
        return json.dumps({'result': collatz(int(number))})
    except ValueError:
        # elsewise, show the error to the client
        return json.dumps({'error': 'NUMBER NOT INTEGER'})

# hash function
def hash_sha256(text):
    return hashlib.sha256(text.encode()).hexdigest()

# collatz function
def collatz(number):
    if number % 2 == 0:
        return int(number / 2)
    return 3 * number + 1

# run the Flask app
if __name__ == '__main__':
    app.run()
```

The following is the code of the file `client.py`, with detailed and descriptive comments added.

```python
# client.py

import sys
import requests

# get url, name, and number parameters from the terminal
url = sys.argv[1]
name = sys.argv[2]
number = int(sys.argv[3])

# send a GET request to /hello_world and shows the message the client got
hello_world_response = requests.get(url + '/hello_world')
print(hello_world_response.json().get('message'))

# send a GET request to /hash with the name parameter
# and shows the result the client got
hash_response = requests.get(url + '/hash', params={'name': name})
hash = hash_response.json().get('result')
print(hash)

# while the client got 1
while number != 1:
    # send a POST request to /collatz
    # with the POST data including `name`, `hash` and `number`
    collatz_response = requests.post(
        url + '/collatz', data={'name': name, 'hash': hash, 'number': number})

    # convert the response data into a python dict
    collatz_response_dict = collatz_response.json()

    # if the response has the `error' field, raise an Exception
    if 'error' in collatz_response_dict:
        raise Exception('ERROR: ' + collatz_response_dict.get('error'))

    # save the collatz result into `number`, print it,
    # and repeat the process
    number = collatz_response_dict.get('result')
    print(number)
```