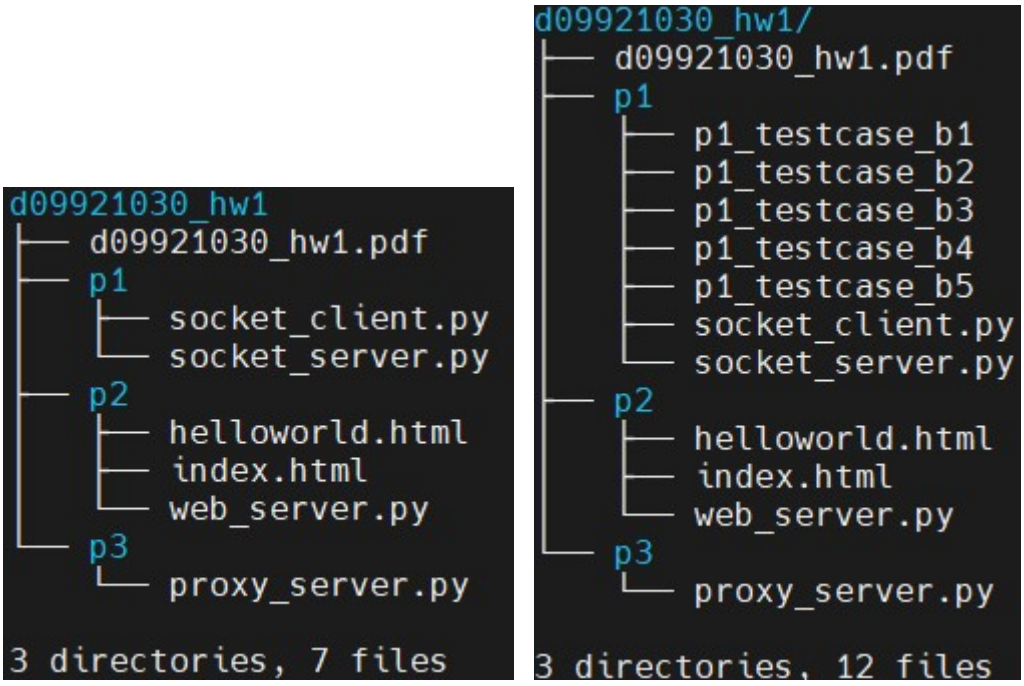
Homework1

Due date: 2023/10/28 21:00

Homework Policy: (Read before you start to work)

1. 作業請勿抄襲。疑似抄襲，助教會請你說明。如果認定抄襲，作業以零分計算。
2. 如果作業上遇到困難可以討論，但是程式碼的部分請自己完成，並且在程式第一行註明討論同學的姓名及學號。
3. 程式作業請於期限內至 NTU Cool 作業區上傳，格式為 zip 檔，檔名無限制。解壓縮後資料夾架構與名稱應恰如下圖所示，圖中學號為自己的學號：



(a) Without p1_bonus (b) With p1_bonus

Figure 1: Folder structure

4. 逾期繳交一天，分數 $\times\frac{2}{3}$；超過一天未滿兩天，分數 $\times\frac{1}{3}$；超過兩天則不予計分，請務必盡早開始，並努力完成。
5. 若助教無法編譯你的程式，以 0 分計算。繳交檔案格式錯誤、資料夾架構錯誤會扣 10 分。

6. 繳交程式碼時，除非題目特別指定 port number，否則請使用(自己學號末四碼 + 1023) % 65535 作為 port number。例如：學號為 d09921030，port number 即為 (1030+1023)%65535=2053。

7. 如有任何問題歡迎來信，請同時寄信給兩位助教。
   範例: [2023ICN] HW1
   信箱: 林致佑 d09921030@ntu.edu.tw、陳泓睿 r12942144@ntu.edu.tw

1. **Socket Programming - TCP** – 30%
   We are going to construct a TCP server and client system, as described in Figure 2, which illustrates the messages exchanged between the client and server.

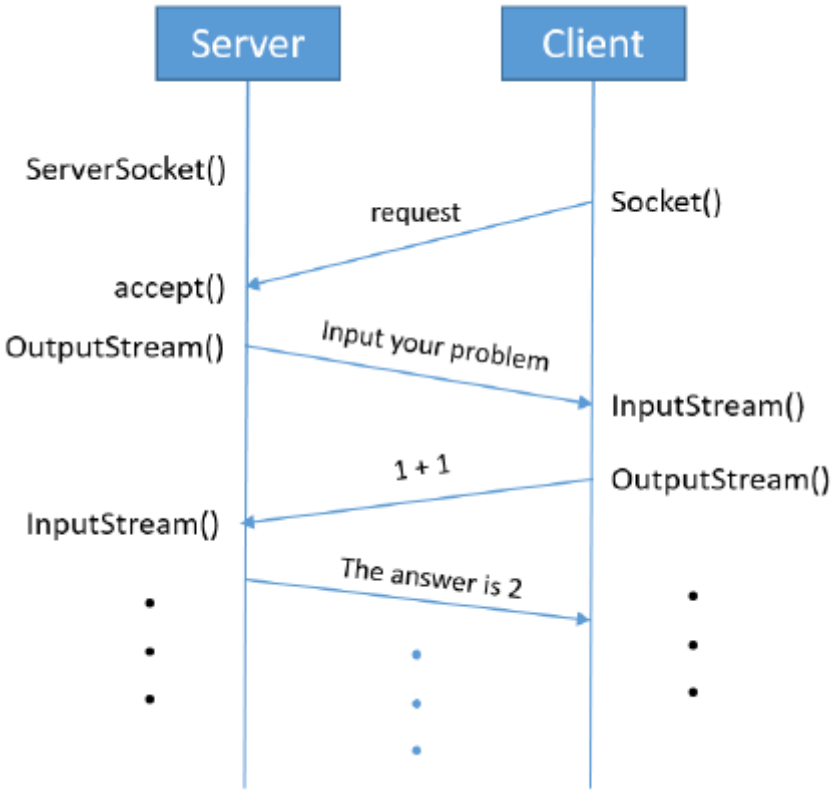


Figure 2: Message flow between the client and server.

Please refer to the following steps:

(a) Read the source code of `socket_server.py`, which works in a Python 3 environment.

(b) Complete the **TODO** part in `socket_server.py`. Modify the server to work as a calculator that supports basic operations such as addition (+), subtraction (-), multiplication (×), and division (÷).

(c) Complete the **TODO** part in `socket_client.py` so that it can:

   i. Read messages sent from the server.

   ii. Send messages (from `p1_testcase`) to the server to check whether the server functions as expected. The message flow between the client and server is illustrated in Figure 2.

(d) Compile and run `socket_server.py` first, and then execute your `socket_client.py`.

(e) Test it on your local machine. If you have done it correctly, the execution and output of `socket_client.py` should resemble Figure 3.

```
The Client is running..
Received the message from server:
Please input a question for calculation
Question: 1+1
Get the answer from server:
2.0
Do you wish to continue? (Y/N)
Response to server prompt: Y
Received the message from server:
Please input a question for calculation
Question: 2-4
Get the answer from server:
-2.0
Do you wish to continue? (Y/N)
Response to server prompt: Y
Received the message from server:
Please input a question for calculation
Question: 3*5
Get the answer from server:
15.0
Do you wish to continue? (Y/N)
Response to server prompt: Y
Received the message from server:
Please input a question for calculation
Question: 4/2
Get the answer from server:
2.0
Do you wish to continue? (Y/N)
Response to server prompt: N
```

Figure 3: Connect to server on local machine (127.0.0.1).

(f) When you finish (e), try to run your `socket_client.py` and connect to TA's computer with TA's IP address 140.112.42.104 and port 7777.

(g) (Bonus 5%) You can also implement other math functions. For example, matrix multiplication, integral, or Laplace's transform. Each function you add, you get 1 extra point in this problem. However, the existing math libraries are not allowed. The test cases you have created should be named as follows: `p1_testcase_b1`, `p1_testcase_b2`, ..., `p1_testcase_b5`. You should write these steps on your own and explain how TA can test your functions in `[student_id]_hw1.pdf`.

Submission and Grading Policy

(a) Please put your source code (including socket_server.py and socket_client.py) under [student_id]/p1.

(b) We provide `p1_testcase` for you to test your program. The result should be exact the same as `p1_testcase_golden`. Please make sure your program is available to save the result to a file `[student_id]_p1_client_result.txt`. TA will use this file to grade your program.

(c) Each operator will be worth 5%. If your program can successfully connect to the TA's computer, you will earn 10%.

(d) In `[student_id]_hw1.pdf`, you have to

- Briefly explain both the **TODO** part of socket_server.py and socket_client.py.
- Show the result of (c) in *Submission and Grading Policy* section.
- If you have done extra functions for bouns points, you should also give a brief explanation.

2. **Web server** – 25%
   (Ch.2 Socket Programming Assignment#1,#4 in textbook)
   In this assignment, you will develop a simple Python web server capable of processing basic requests. There should be two HTML files: one named 'index.html' and the other 'helloworld.html.' You will be able to access both HTML files through the web server. Additionally, you can access 'helloworld.html' from 'index.html.

   Please refer to the following steps:

   (a) Run the web server on the local machine and establish a connection socket.

   (b) Create two HTML files as described above.

   (c) Receive the HTTP request from this connection when contacted by a client (browser).

   (d) Parse the HTTP request to determine the specific file being requested.

   (e) Retrieve the requested file from the server's file system.

   (f) Create an HTTP response message consisting of the requested file preceded by header lines.

   (g) If the client sends an HTTP request for accessing other pages through the homepage (index.html), the web server will handle the request in the same way from step (c) to step (f).

   (h) Send the response over the TCP connection to the requesting browser. If a browser requests a file that is not present on your server, your web server should return a 404 Not Found error message. The message flow between the client, web browser, and web server is shown in Figure 4.

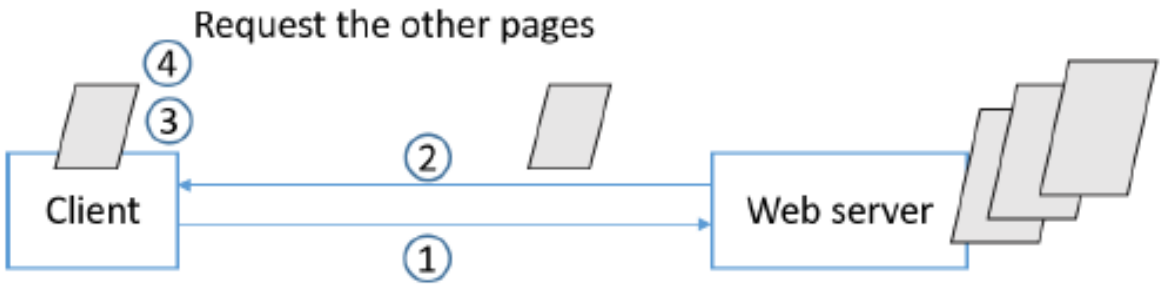


Figure 4: Message flow between the client and the web server.

Notes

(a) Read the source code `web_server.py`. Complete the **TODO** part for web server functions.

(b) **Ensure that HTML files are placed in the directory where the web server is running**, so that you can access HTML files through the browser.

(c) To test whether your program can access the HTML files, please use the URL: http://HOST:PORT/index.html

(d) To test the 404 NOT FOUND error, simply use the URL: http://HOST:PORT/no.html

(e) Show "Hello World!" in `helloworld.html`. Show your name and student ID in `index.html`.

(f) You can ignore the error message from "FAVICON.ICO".

Submission and Grading Policy

(a) Please put your source code (web_server.py) and HTML files under [student_id]/p2. Also, please screenshot your output results and put them into `[student_id]_hw1.pdf`.

(b) 15% Your program should be able to access HTML files through a local machine.

(c) 5% Your program should be able to show 404 NOT FOUND when the requested file is not in the web server.

(d) 5% Your program should be able to access `helloworld.html` through `index.html`.

(e) In `[student_id]_hw1.pdf`, you have to

- Briefly explain the **TODO** part of web_server.py.
- Show the results of (b), (c), and (d) in *Submission and Grading Policy* section.

```
Ready to serve...
('127.0.0.1', 14182) connected
client's request message:
 GET /INDEX.HTML HTTP/1.1
HOST: 127.0.0.1:2053
USER-AGENT: MOZILLA/5.0 (WINDOWS NT 10.0; WIN64; X64; RV:109.0) GECKO/20100101 FIREFOX/117.0
ACCEPT: TEXT/HTML,APPLICATION/XHTML+XML,APPLICATION/XML;Q=0.9,IMAGE/AVIF,IMAGE/WEBP,*/*;Q=0.8
ACCEPT-LANGUAGE: EN,ZH-TW;Q=0.5
ACCEPT-ENCODING: GZIP, DEFLATE, BR
CONNECTION: KEEP-ALIVE
UPGRADE-INSECURE-REQUESTS: 1
SEC-FETCH-DEST: DOCUMENT
SEC-FETCH-MODE: NAVIGATE
SEC-FETCH-SITE: NONE
SEC-FETCH-USER: ?1


Extract the filename: INDEX.HTML
Ready to serve...
```

(a) Access index.html

```
Ready to serve...
('127.0.0.1', 14728) connected
client's request message:
 GET /HELLOWORLD.HTML HTTP/1.1
HOST: 127.0.0.1:2053
USER-AGENT: MOZILLA/5.0 (WINDOWS NT 10.0; WIN64; X64; RV:109.0) GECKO/20100101 FIREFOX/117.0
ACCEPT: TEXT/HTML,APPLICATION/XHTML+XML,APPLICATION/XML;Q=0.9,IMAGE/AVIF,IMAGE/WEBP,*/*;Q=0.8
ACCEPT-LANGUAGE: EN,ZH-TW;Q=0.5
ACCEPT-ENCODING: GZIP, DEFLATE, BR
CONNECTION: KEEP-ALIVE
UPGRADE-INSECURE-REQUESTS: 1
SEC-FETCH-DEST: DOCUMENT
SEC-FETCH-MODE: NAVIGATE
SEC-FETCH-SITE: NONE
SEC-FETCH-USER: ?1


Extract the filename: HELLOWORLD.HTML
Ready to serve...
```

(b) Access helloworld.html

```
Ready to serve...
('127.0.0.1', 1132) connected
client's request message:
 GET /NO.HTML HTTP/1.1
HOST: 127.0.0.1:2053
USER-AGENT: MOZILLA/5.0 (WINDOWS NT 10.0; WIN64; X64; RV:109.0) GECKO/20100101 FIREFOX/117.0
ACCEPT: TEXT/HTML,APPLICATION/XHTML+XML,APPLICATION/XML;Q=0.9,IMAGE/AVIF,IMAGE/WEBP,*/*;Q=0.8
ACCEPT-LANGUAGE: EN,ZH-TW;Q=0.5
ACCEPT-ENCODING: GZIP, DEFLATE, BR
CONNECTION: KEEP-ALIVE
UPGRADE-INSECURE-REQUESTS: 1
SEC-FETCH-DEST: DOCUMENT
SEC-FETCH-MODE: NAVIGATE
SEC-FETCH-SITE: NONE
SEC-FETCH-USER: ?1


Extract the filename: NO.HTML
404 NOT FOUND
Ready to serve...
```

(c) Access a non-existing no.html

Figure 5: Access HTML files

3. **Proxy Server** – 35%
   In this assignment, you will develop a web proxy. The message flow is shown in Figure 6. The following are the tasks you need to complete.

   (a) Create a socket on the proxy server and receive data (request) from the client.

   (b) Parse the request to determine which file is being requested..

   (c) Check whether the requested file is on the local server. If yes, send the file back to the client, as you did in p2. If not, proceed to step (d).

   (d) Create another socket on the proxy server and connect it to the web server (the same step as you did in p2).

   (e) Request the web server for sending the file requested by the client.

   (f) The proxy server receives and read the response from the web server, then send the requested file back to the client. Also, **cache the file on the local proxy server**.

   (g) If the file is not found at the web server, return an error message 404 NOT FOUND.

   Notes

   (a) Read the source code `proxy_server.py`. Complete **TODO** part for proxy server functions.

   (b) The port of the proxy server should be set as **9999**.

   (c) Test your program by using different browsers to request web objects through your proxy.

   (d) Make sure to configure your web browser to use the proxy at first.

   (e) To test whether your program can access the HTML files, please use the URL: http://HOST:PORT/index.html

   (f) To test the 404 NOT FOUND error, simply use the URL: http://HOST:PORT/no.html

   (g) You might have to change the proxy setting of the browser or the local machine in this problem.

   (h) You can ignore the error message from "FAVICON.ICO".

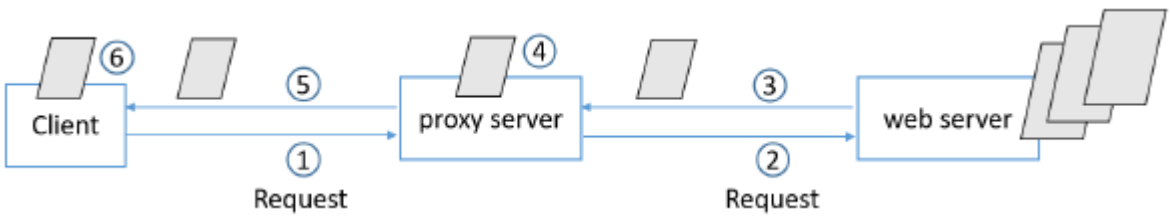


Figure 6: Message flow among Client-Proxy-Server.

Submission and Grading Policy

(a) Please put your source code (proxy_server.py) under [student_id]/p3. Also, please screenshot your output results and put them into `[student_id]_hw1.pdf`.

(b) Get 25% if your program can access HTML files (which is located at p2 directory) through the proxy server.

(c) Get 5% if your program can show 404 NOT FOUND when the requested file is not in neither the proxy server or the web server.

(d) Get 5% if you can access HTML files (which is located at p2 directory) through two different browsers.

(e) In `[student_id]_hw1.pdf`, you have to

- Briefly explain the **TODO** part of proxy_server.
- Show the results of (b), (c), and (d) in *Submission and Grading Policy* section.

```
index.html
/index.html
Read from cache
Ready to serve...
Received a connection from: ('127.0.0.1', 4507)
GET /favicon.ico HTTP/1.1
Host: 127.0.0.1:9999
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/117.0
Accept: image/avif,image/webp,*/*
Accept-Language: en,zh-TW;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Referer: http://127.0.0.1:9999/index.html
Sec-Fetch-Dest: image
Sec-Fetch-Mode: no-cors
Sec-Fetch-Site: same-origin
```

(a) Access index.html

```
HELLOWORLD.html
/HELLOWORLD.html
Host name is HELLOWORLD.html
Trying to connect to the web server
Connected successfully
GET /HELLOWORLD.html HTTP/1.0

Sent the request to the web server successfully
HTTP/1.1 200 OK
Content-Type: text/html

<!DOCTYPE html>
<html lang = "en">
<head>
        <meta charset ="utf-8">
        <title> Hello World! </title>
</head>
<body>
    Hello World
</body>
</html>
```

(b) Access helloworld.html

```
no.html
/no.html
Host name is no.html
Trying to connect to the web server
Connected successfully
GET /no.html HTTP/1.0

Sent the request to the web server successfully
HTTP/1.1 404 Not Found
```

(c) Access a non-existing no.html

Figure 7: Access HTML files

4. **Report** (`[student_id]_hw1.pdf`) – 10%

(a) If you have done the bonus part of p1, briefly explain your additional functions and how to test your implementations.

(b) For p1 3%:

- Post your results of p1.
- Briefly describe how you implement **TODO** part function in p1.

(c) For p2 3%:

- Post your results of p2.
- Briefly describe how you implement **TODO** part function in p2.

(d) For p3 4%:

- Post your results of p3.
- Briefly describe how you implement **TODO** part function in p3.