COMP2322
Multi-thread Web Server
Individual project

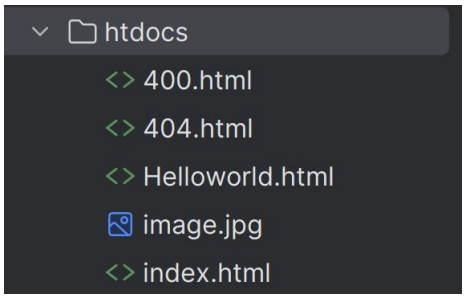
Name:LiuYuzhou
Student ID:201100602d

Part1:

Summary of design and implementation

Structure:(The name of the class form obey the camel case)

All the files on this sever:



Message classes:(4 types of response)

```
5 usages
class ResponseStates(object):
    OK = 'HTTP/1.1 200 OK\r\n'
    NOT_MODIFIED = 'HTTP/1.1 304 Not Modified\r\n'
    BAD_REQUEST = 'HTTP/1.1 400 Bad Request\r\n'
    NOT_FOUND = 'HTTP/1.1 404 Not Found\r\n'
```

The response status message are in this class, as we can see there are totally 4 status, which is 200 OK, 304 Not Modified, 404 Not Found.(shows in the picture above)

```
4 usages
class HeaderFields(object):
    status_code = ''
    last_modified_time = 'Last-Modified: '
    Accept_range = 'Accept-Ranges: bytes\r\n'
    Content_length = 'Content-Length: '
    Close_connection = "Connection: close\n\n"
    GMT = datetime.datetime.now()
    Alive_connection = "Connection: keep-alive\r\n"
    Time_out="Timeout: timeout=60\r\n"
    content_type = "Content-Type: text/html;charset=UTF-8\r\n"
```

The HeaderFileds class is used to store all the return header filed that might be used by each states, and aim to handle different type of bodies which will be send to the client. As we can see here there is keep-alive and the Time-out field which implement the bonus, during this time out

interval, our sever will be able to keep connect with the client, waiting if there are more messages will need to send.

Handler class:

```
1 usage
class HttpRequestHandler(object):
    def __init__(self):
        self.log_list = []
        self.modified_time = None
        self.last_access_time = None
        self.file_list = []
```

This Handler lass include all the method to handle the request,the last-access-time is the if-modified-since field. And file_list is aiming to store the file names, the modified time is used to store the time which is the time which the file last been modified. The functions for handle the http request message are included in this class.

The method include in this class is:

- async def add_log(self):
- def get_modified(self, filepath):
- def if_modified(self, file_path, header_dic: dict):
- def form_response(self, file_path, headers: HeaderFields, header_dict: dict, method: str):
- def form_error_response(self, file_path: str, headers: HeaderFields, method: str):
- def handle_request(self, url: str, header_dict: dict, method: str):
- def format_header(request: str): <- @staticmethod

Key function inside of the class:

1. The form_response function is used to form the 200 and 304 response to the client. It will add different header field according to different http status, but if the error happens here, the http outer class will detect the error, and then call the form_error_response function to generate the error response messages according to different error conditions. And you can see that we will also check the connection header in order to see whether it have keep-alive or close header.

```
3 usages
def form_response(self, file_path, headers: HeaderFields, header_dict: dict, method: str):
    """
    this function aims to get the response message
    :param file_path: the path which can be used to find the file
    :param headers: the class header field which can be used to form the response headers
    :param header_dict: the dictionary created by splitting the request message
    :param method: The method we are using, "HEAD" or "GET"
    :return: response_header(The headers of the response message), body(The body of the response message)
    """
    try:
        response_header = ''
        body = None
        headers.last_modified_time += self.get_modified(file_path)
        headers.status_code = self.if_modified(file_path, header_dict)
        # if it is 304, no need to get the size
        if headers.status_code.find("304") < 0:
            headers.Content_length += str(os.path.getsize(file_path)) + '\r\n'
        # if it is GET method and the status code is 200, we are going to read the body from the file
        if headers.status_code.find("200") >= 0 and method != "HEAD":...
        response_header += headers.status_code
        response_header += headers.last_modified_time
        response_header += "Cache-Control: max-age=3600\r\n"
        # if the code is 200 we need to have these header fields
        if "connection" in header_dict:
            if header_dict['connection'] == 'keep-alive':...
            else:
                response_header += "Connection: close\r\n"
        if headers.status_code.find("200") >= 0:...
        response_header += headers.GMT
        return response_header, body
```

2. This function is used to form the error response message, if the method be used is head, definitely we will return the none body, in order to avoid the useless read of the file. In this function, for sure we won't encounter file not found error, so we are not going to use try condition to avoid the error.

```
8 usages
def form_error_response(self, file_path: str, headers: HeaderFields, method: str):
    """
    this function aims to get the 404,400 Error response message
    :param file_path: the path which can be used to find the file
    :param headers: the class header field which can be used to form the response headers
    :param method: The method we are using, "HEAD" or "GET"
    :return: the 404 and 400 error message and the message body
    """
    response_header = ''
    body = None
    headers.Content_length += str(os.path.getsize(file_path)) + '\r\n'
    if file_path.find("400") >= 0:
        headers.status_code = ResponseStatus.BAD_REQUEST
    elif file_path.find("404") >= 0:
        headers.status_code = ResponseStatus.NOT_FOUND
    response_header += headers.status_code
    response_header += headers.content_type
    response_header += headers.Content_length
    response_header += headers.Close_connection
    if (method.find("GET") >= 0 and file_path.find("404") >= 0) or file_path.find("400") >= 0:
        fin = open(file_path)
        body = fin.read()
        fin.close()
    return response_header, body
```

Thread initialization:

```
1 usage
def start_sever(sever_host: str, sever_port: int):
    """
    act to listen to client, when a client try to connect to the sever, we will be able to receive in this function
    :param sever_host: IP of our host
    :param sever_port: port number of the sever
    :return: NULL
    """
    sever_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sever_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    sever_socket.bind((sever_host, sever_port))
    sever_socket.listen(1)
    while True:
        client_connection, client_address = sever_socket.accept()
        # print(client_address)
        request_thread = threading.Thread(target=single_thread, args=(client_connection, client_address[0],))
        request_thread.start()


1 usage
def main():
    sever_host = 'localhost'
    sever_port = 8000
    start_sever(sever_host, sever_port)


main()
```

When the program starts, the main function will call the start sever function which will be used to create a socket, and then waiting for the connection from the sever. Once it receive a connection from the client, it will create a connection socket, and then start a thread for that client.

```
1 usage
def single_thread(client_connection, client_address):
    """
    this function is aiming to call the asynchronize function
    :param client_connection: connection from the client
    :param client_address: The address of the client host
    :return: NULL
    """
    loop = asyncio.new_event_loop()
    asyncio.set_event_loop(loop)
    loop.run_until_complete(http_sever(client_connection, client_address))
```

The function single thread is used to resolve the synchronize problem, because we need to do synchronize to the written of file, so because the threading.thread()function cannot solve a function which need to be synchronized. To solve this problem, I created this function.

Bonus design:

```python
async def http_sever(client_connection, client_address, timeout=60):
    """..."""
    handler = HttpRequestHandler()
    handler.log_list.append(client_address)
    client_connection.settimeout(timeout)
    flag = False # this flag is used to detect whether we need to keep alive
    while True:
        if flag: # if the flag is true then now we can close the connection
            print("close")
            client_connection.close()
            break
        try:
            request = client_connection.recv(1024).decode()
        except socket.timeout:
            print("close2")
            client_connection.close()
            break
        try:
            if not request:
                client_connection.close()
                break
            http_method, url, header_dic, version = handler.format_header(request)
            if 'connection' in header_dic.keys():
                if header_dic['connection'] == 'close':
                    flag = True
```

Through the http_sever method we can set a timeout interval, so here if the time is out of 60 seconds, the connection will be closed, so the sever will keep-alive for this 60 seconds, and we will check whether the client ask us to keep alive, which means have keep-alive header field if not, we will not wait for timeout.

Handling Method functions: (HttpRequestHandler class)

Return type	Description
void	**async def add_log(self):** Write the record data of the user to the log.txt file(IP address,time,return status,file retrieved). This function will be synchronized for each thread.
str	**def get_modified(self, filepath):** This function will return the modified time of the file.
str	**def if_modified(self, file_path, header_dic: dict):** This function returns the http status, the aim of this function is to do time compare, if the modified is before the if-modified-since time, then the sever will set the response message to

	304 status, otherwise it will be 200.
2 return values: Str,(str or bytes)	**def form_response(self, file_path, headers: HeaderFields, header_dict: dict, method: str):** This function aims to form response for the correct states(form the http header message according to the type of the file, and also read the file and return the body message), if it get an Error, it will raise the FileNotFound error, case the handle_request method to turn to error handling stage.
2 return values: Str,(str or bytes)	**def form_error_response(self, file_path: str, headers: HeaderFields, method: str):** This function aims to form the error response and return it back to the handle request function, if a error File not found or the request method is wrong, we will come to this method, and form the error message, also get the status type here(400 or 404).
2 return values: Str,(str or bytes)	**def handle_request(self, url: str, header_dict: dict, method: str):** This function aims to handle the request from the client, in this function the file will be classified to two types: html, (png and jpg), and we will call different function to form the response in this function, normal case will be handled be the formed by form_response function, for the abnormal state, the form_error_response function will be able to form error message and pass back to this function.
4 return values: Str, Str, dict,Str	**def format_header(request: str):** This function is used to handle the request message, it will find the url, and method used from the first line, and split the rest of the string by ": ", and get a dictionary by using the first part as key, second part as value, this is aim to find the if-modified-since field, get the time from the user. Also it will be able to return the HTTP version, check if it have any problem.

Summary:with these three classes, the code's scalability is better.

Part2:

demonstration of executing program and screen captures

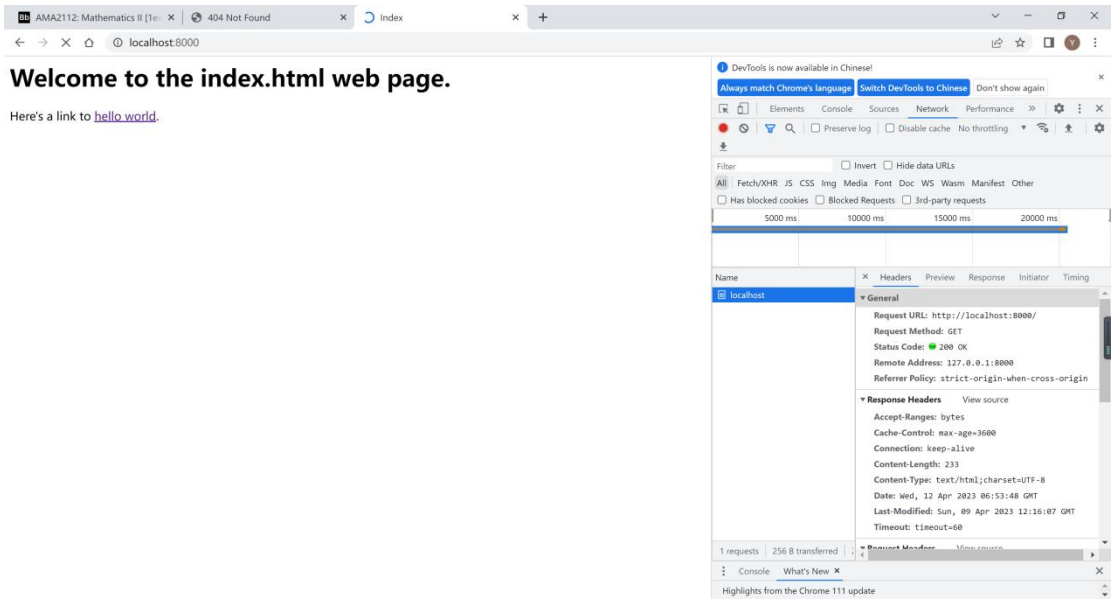
1. Demonstration:

This sever have provide 3 files in its cache, which is index.html, image.jpg, helloworld.html, and you may use HEAD or GET method to retrieve the files. GET /index.html, HEAD /index.html , GET / or HEAD / for retrieve index.html file, GET /image.jpg or HEAD /image.jpg to retrieve image.jpg file, GET /Helloworld.html HEAD /Helloworld.html to retrieve the Helloworld.html file.

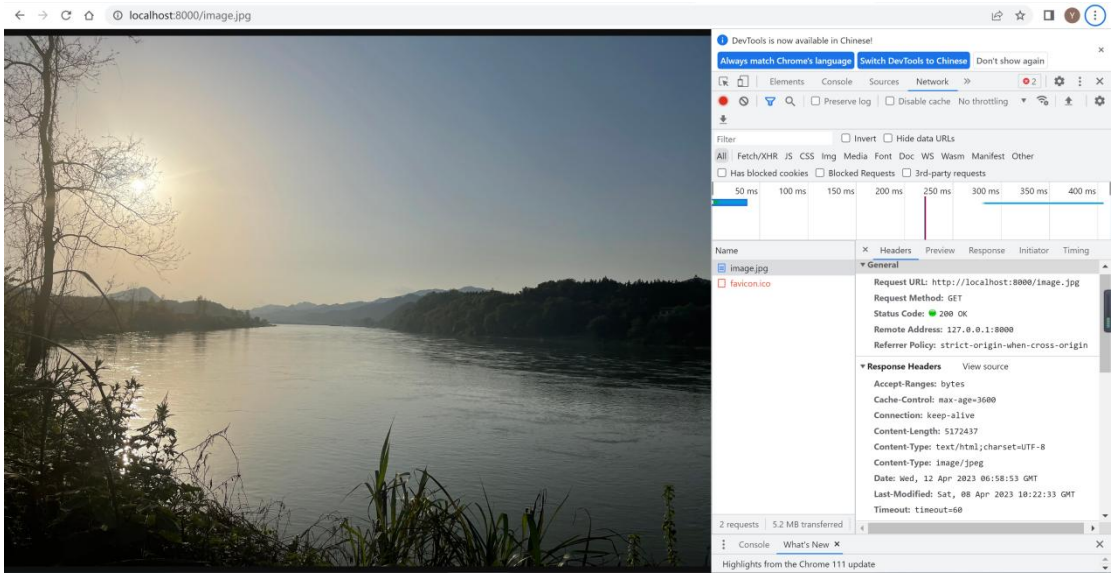
2. Screen captures:

The browser capture of the GET method:

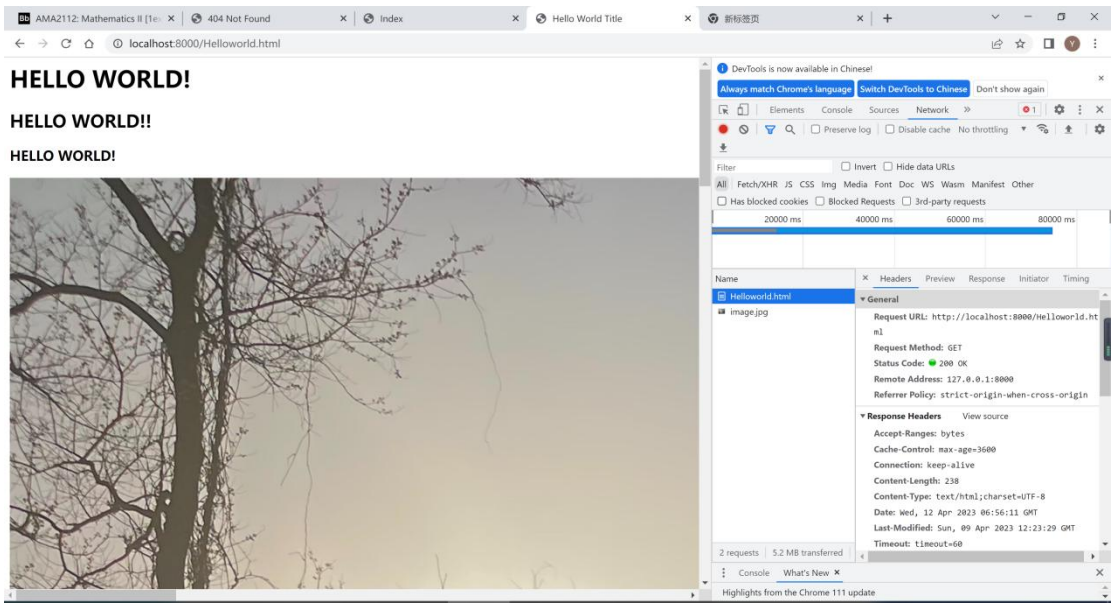
➢ with / or /index.html



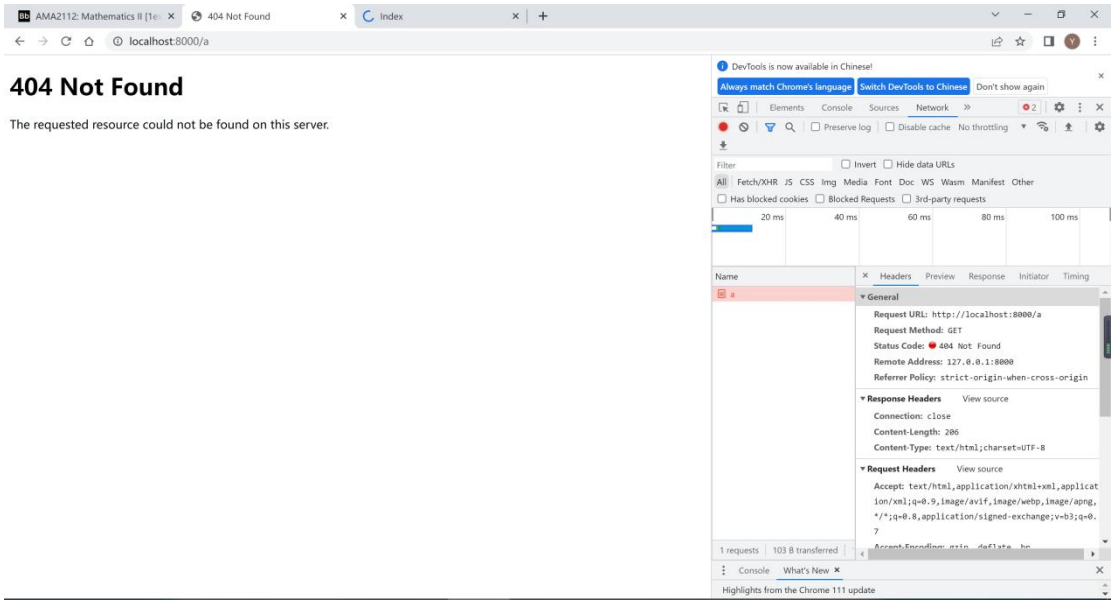
➢ Retrieve the image.jpg



➢ The hello world page



➢ 404 page if not Found



Test the method with a client:(All the print out are from the client side,first print out request)

➢ GET method:

- **404 Not found**

```
GET /index.jpg HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:87.0) Gecko/20100101 Firefox/87.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
If-Modified-Since: Tue, 05 Apr 2021 08:00:00 GMT


HTTP/1.1 404 Not Found
Content-Type: text/html;charset=UTF-8
Content-Length: 206
Connection: close


<!DOCTYPE html>
<html>
  <head>
    <title>404 Not Found</title>
  </head>
  <body>
    <h1>404 Not Found</h1>
    <p>The requested resource could not be found on this server.</p>
  </body>
</html>
```

- **304 Not Modified**

```
D:\Second_year_doncuments\comp2322\project\venv\Scripts\python.exe D:\Second_year_doncuments\comp2322\project\http\http_client.py
GET /index.html HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:87.0) Gecko/20100101 Firefox/87.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
If-Modified-Since: Tue, 10 Apr 2023 08:00:00 GMT


HTTP/1.1 304 Not Modified
Last-Modified: Sun, 09 Apr 2023 12:16:07 GMT
Cache-Control: max-age=3600
Connection: keep-alive
Timeout: timeout=60
Date: Wed, 12 Apr 2023 07:33:41 GMT




Process finished with exit code 0
```

- **200 OK**

```
GET /index.html HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:87.0) Gecko/20100101 Firefox/87.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
If-Modified-Since: Tue, 07 Apr 2023 08:00:00 GMT


HTTP/1.1 200 OK
Last-Modified: Sun, 09 Apr 2023 12:16:07 GMT
Cache-Control: max-age=3600
Connection: keep-alive
Timeout: timeout=60
Content-Length: 233
Accept-Ranges: bytes
Content-Type: text/html;charset=UTF-8
Date: Wed, 12 Apr 2023 07:07:08 GMT


<html> <head>
    <link rel="icon" href="data:,">
    <title>Index</title>
</head>
<body>
    <h1>Welcome to the index.html web page.</h1>
    <p>Here's a link to <a href="Helloworld.html">hello world</a>.</p>
</body>
```

➢ HEAD method:

- **200 OK**

```
D:\Second_year_doncuments\comp2322\project\venv\Scripts\python.exe D:\Second_year_doncuments\comp2322\project\http\http_client.py
HEAD /image.jpg HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:87.0) Gecko/20100101 Firefox/87.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
If-Modified-Since: Tue, 08 Apr 2023 08:00:00 GMT


HTTP/1.1 200 OK
Last-Modified: Sat, 08 Apr 2023 10:22:33 GMT
Cache-Control: max-age=3600
Connection: keep-alive
Timeout: timeout=60
Content-Length: 5172437
Accept-Ranges: bytes
Content-Type: text/html;charset=UTF-8
Content-Type: image/jpeg
Date: Wed, 12 Apr 2023 07:12:39 GMT



Process finished with exit code 0
```

- **304 Not Modified**

```
D:\Second_year_doncuments\comp2322\project\venv\Scripts\python.exe D:\Second_year_doncuments\comp2322\project\http\http_client.py
HEAD /image.jpg HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:87.0) Gecko/20100101 Firefox/87.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
If-Modified-Since: Tue, 11 Apr 2023 08:00:00 GMT


HTTP/1.1 304 Not Modified
Last-Modified: Sat, 08 Apr 2023 10:22:33 GMT
Cache-Control: max-age=3600
Connection: keep-alive
Timeout: timeout=60
Date: Wed, 12 Apr 2023 07:10:31 GMT




Process finished with exit code 0
```

- **404 Not Found**

```
D:\Second_year_doncuments\comp2322\project\venv\Scripts\python.exe D:\Second_year_doncuments\comp2322\project\http\http_client.py
HEAD /image. HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:87.0) Gecko/20100101 Firefox/87.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
If-Modified-Since: Tue, 08 Apr 2023 08:00:00 GMT


HTTP/1.1 404 Not Found
Content-Type: text/html;charset=UTF-8
Content-Length: 206
Connection: close




Process finished with exit code 0
```

- **Wrong method:**

- **400 Bad Request**

```
HA /index.html HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:87.0) Gecko/20100101 Firefox/87.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
If-Modified-Since: Tue, 10 Apr 2023 08:00:00 GMT


HTTP/1.1 400 Bad Request
Content-Type: text/html;charset=UTF-8
Content-Length: 257
Connection: close


<!DOCTYPE html>
<html>
  <head>
    <title>400 Bad Request</title>
  </head>
  <body>
    <h1>400 Bad Request</h1>
    <p>There may exist malformed request syntax, invalid request message framing, or deceptive request routing.</p>
  </body>
</html>

Process finished with exit code 0
```

➢ Wrong HTTP version(Only allowed 1.1 or 1.0):

- **400 Bad Request**

```
HEAD /index.html HTTP/11
Host: www.example.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:87.0) Gecko/20100101 Firefox/87.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
If-Modified-Since: Tue, 01 Apr 2023 08:00:00 GMT


HTTP/1.1 400 Bad Request
Content-Type: text/html;charset=UTF-8
Content-Length: 257
Connection: close


<!DOCTYPE html>
<html>
  <head>
    <title>400 Bad Request</title>
  </head>
  <body>
    <h1>400 Bad Request</h1>
    <p>There may exist malformed request syntax, invalid request message framing, or deceptive request routing.</p>
  </body>
</html>

Process finished with exit code 0
```

➢ Wrong message format:

- **400 Bad Request**

```
GET insdex.html
Host: www.example.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:87.0) Gecko/20100101 Firefox/87.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
If-Modified-Since: Tue, 01 Apr 2023 08:00:00 GMT


HTTP/1.1 400 Bad Request
Content-Type: text/html;charset=UTF-8
Content-Length: 257
Connection: close


<!DOCTYPE html>
<html>
  <head>
    <title>400 Bad Request</title>
  </head>
  <body>
    <h1>400 Bad Request</h1>
    <p>There may exist malformed request syntax, invalid request message framing, or deceptive request routing.</p>
  </body>
</html>

Process finished with exit code 0
```

```
D:\Second_year_doncuments\comp2322\project\venv\Scripts\python.exe D:\Second_year_doncuments\comp2322\project\http\http_client.py
GET index.html HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:87.0) Gecko/20100101 Firefox/87.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
If-Modified-Since: Tue, 01 Apr 2023 08:00:00 GMT


HTTP/1.1 400 Bad Request
Content-Type: text/html;charset=UTF-8
Content-Length: 257
Connection: close


<!DOCTYPE html>
<html>
  <head>
    <title>400 Bad Request</title>
  </head>
  <body>
    <h1>400 Bad Request</h1>
    <p>There may exist malformed request syntax, invalid request message framing, or deceptive request routing.</p>
  </body>
</html>

Process finished with exit code 0
```

➢ Bonus(Keep-alive):

- Return close and close the connection, if the request header is close.

```
D:\Second_year_doncuments\comp2322\project\venv\Scripts\python.exe D:\Second_year_doncuments\comp2322\project\http\http_client.py
GET /index.html HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:87.0) Gecko/20100101 Firefox/87.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: close
If-Modified-Since: Tue, 11 Apr 2023 08:00:00 GMT


HTTP/1.1 304 Not Modified
Last-Modified: Sun, 09 Apr 2023 12:16:07 GMT
Cache-Control: max-age=3600
Connection: close
Date: Thu, 13 Apr 2023 12:16:51 GMT




Process finished with exit code 0
```

- Keep connect with the sever if it asks to keep alive.

```
D:\Second_year_doncuments\comp2322\project\venv\Scripts\python.exe D:\Second_year_doncuments\comp2322\project\http\http_client.py
GET /index.html HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:87.0) Gecko/20100101 Firefox/87.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
If-Modified-Since: Tue, 11 Apr 2023 08:00:00 GMT


HTTP/1.1 304 Not Modified
Last-Modified: Sun, 09 Apr 2023 12:16:07 GMT
Cache-Control: max-age=3600
Connection: keep-alive
Timeout: timeout=60
Date: Thu, 13 Apr 2023 12:18:57 GMT
```