# Socket Programming

**Name: Piyush Zode**

**ID: XXXXXXXXXX**

**Title:** Socket Programming

**Requirements:**

1. Python 2.7.11
2. Windows CMD Prompt
3. Browser
4. IDE: Even notepad can be used. I used Visual Studio Code.

**Running the code:**

1. **Running the Code with browser as the client:**

Open a command prompt for running the server and open a browser for client.

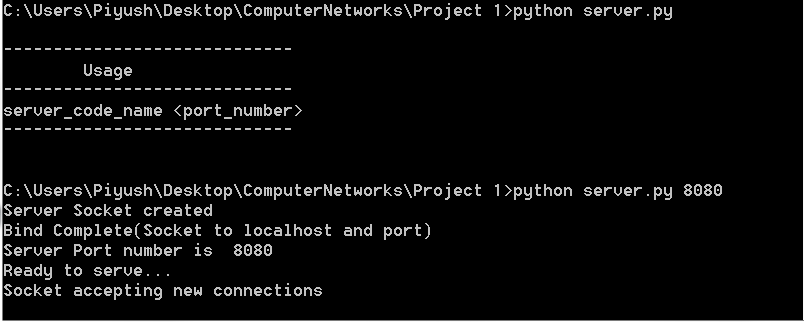
**A] At Server Side:**

**i.** The server is multithreaded and handles multiple requests from clients concurrently. The server works with HTTP GET messages. If the requested file exists at the server, it responds with a HTTP/1.1 200 OK” together with the requested page to the client, otherwise it sends a corresponding error message, “HTTP/1.1 404 Not Found” or “HTTP/1.1 400 Bad Request”.

**ii.** Below is the command for running the server code:

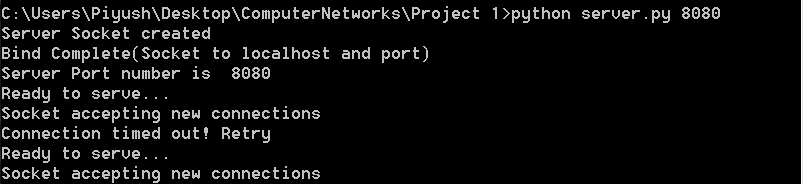
**python server.py 8080**

If the user doesn’t run the server code properly, a usage gets printed as below:



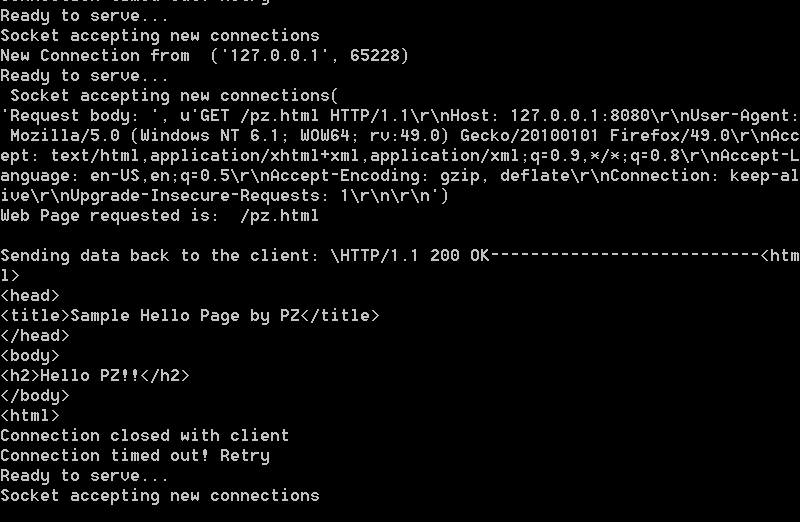
**iii**. Now, once we start the server, the port at which the server gets started gets printed and the server is ready to accept any new connections on that port.

**iv**. Now, there is a timeout of 25 seconds which has been kept for the server after which the server again tries to accept new connections from the server.

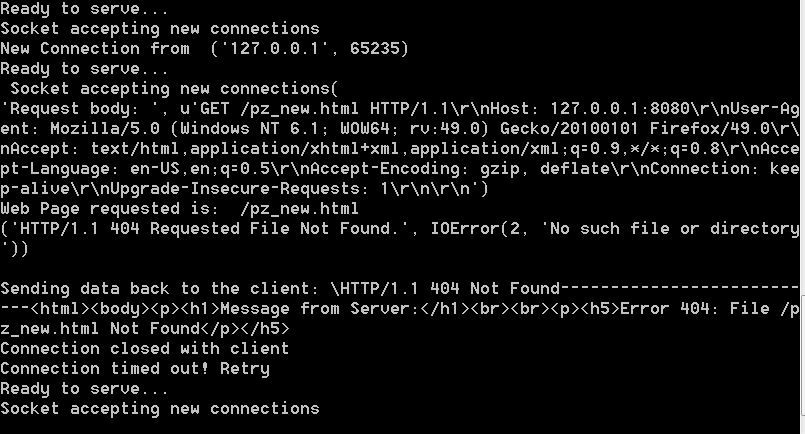


**v**. If the client connects to the server, the server logs the information about the client and checks if the file is present or not in the server and replies back to the server with the appropriate response message and the connection with the client is closed.

**a**. **When file is present at the server:**



**b.** **When file is not present at the server:**



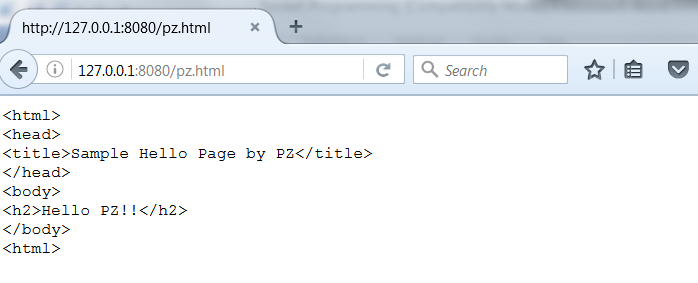
**vi.** The server will keep on running and accept new client requests until it is terminated and it will log every client request.

**B] At Client Side[Browser]:**

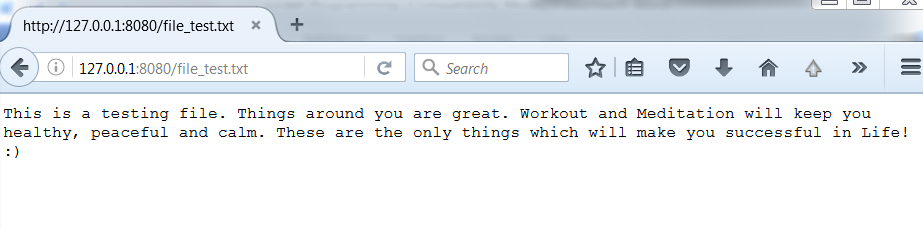
**i.** The client initiates a connection to the server via a socket and requests any page on the server.

**ii.** Open any browser and request for any file:

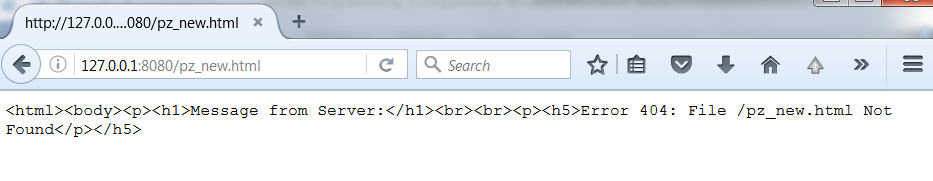
1. **Requested file is pz.html**



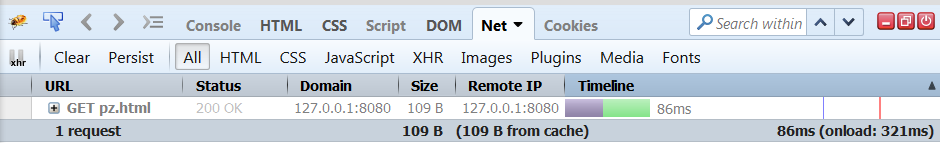
1. **Requested file is file\_test.txt**



1. **If the file is not present at the server:**



**iii.** TheRTTcan be seen in firebug for Mozilla Firefox.



1. **Running the Code with a standalone client through the Command Line:**

Open 2 Command prompts. One for Client and other for Server.

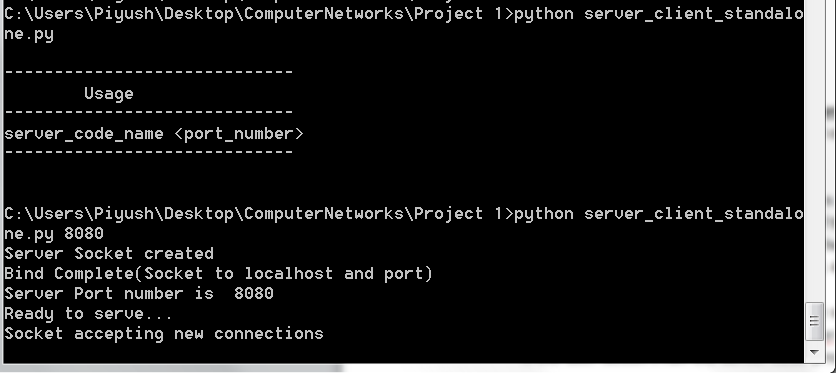
**A] At Server Side:**

**i.** The server is multithreaded and handles multiple requests from clients concurrently. The server works with HTTP GET messages. If the requested file exists at the server, it responds with a HTTP/1.1 200 OK” together with the requested page to the client, otherwise it sends a corresponding error message, “HTTP/1.1 404 Not Found” or “HTTP/1.1 400 Bad Request”.

**ii.** Below is the command for running the server code:

**python server\_client\_standalone.py 8080**

If the user doesn’t run the server code properly, a usage gets printed as below:

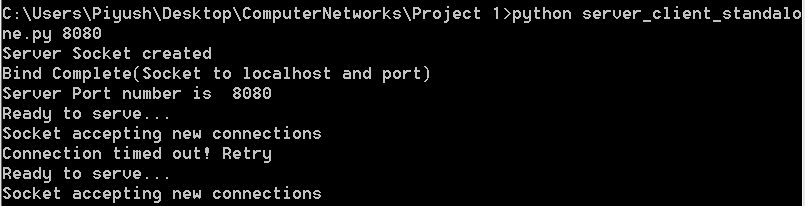


**iii**. Now, once we start the server, the port at which the server gets started gets printed and the server is ready to accept any new connections on that port.

**iv**. Now, there is a timeout of 25 seconds which has been kept for the server after which the server again tries to accept new connections from the server.

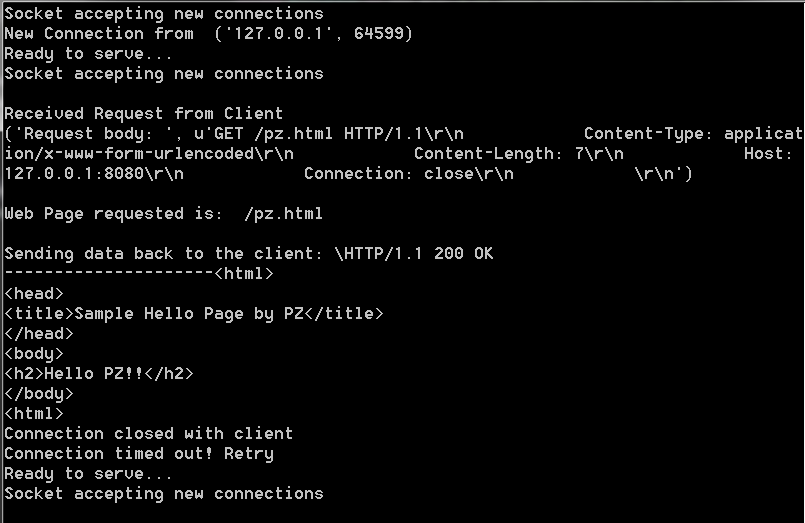
**v**. If the client connects to the server, the server logs the information about the client and checks if the file is present or not in the server and replies back to the server with the appropriate response message and the connection with the client is closed.

**a**. **When file is present at the server:**

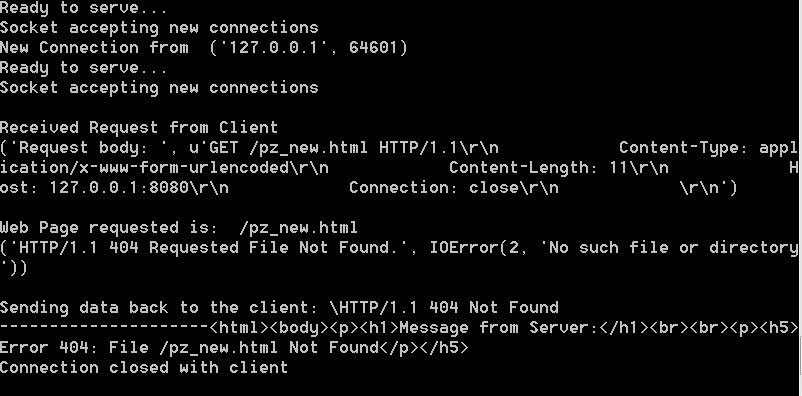


**v**. If the client connects to the server, the server logs the information about the client and checks if the file is present or not in the server and replies back to the server with the appropriate response message and the connection with the client is closed.

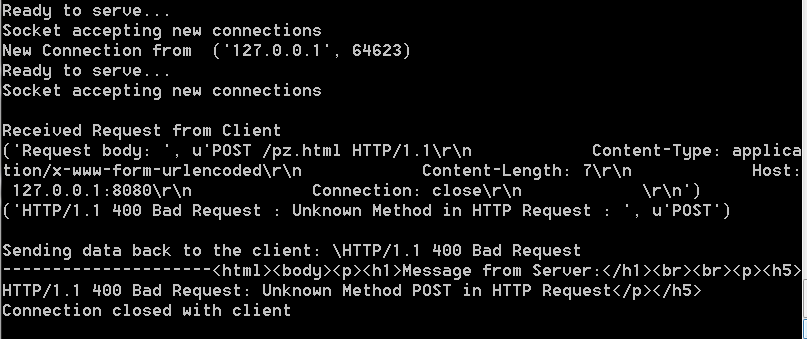
**a**. **When file is present at the server:**



**b.** **When file is not present at the server:**



**c. When there is Bad Request:**



**vi.** The server will keep on running and accept new client requests until it is terminated and it will log every client request and the server displays the client information along with the new port created for handling the client requests.

Socket family: AF\_INET

Socket Type: Sock\_Stream

Protocol: IPPROTO\_TCP

**B] At Client Side:**

**i.** The client initiates a connection to the server via a socket and requests any page on the server.

**ii.** Below is the command for running the client code:

**client\_code -i ip\_address [-p port\_number] [-f filename] [-m method]**

There are 4 command line options which can be set which can be set:

-i -> IP Address[Required]

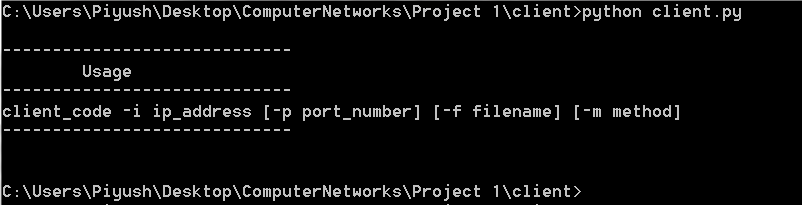
-p -> Port Number

-f -> Filename

-m -> Method[GET]

**python client.py –i 127.0.0.1 –p 8080 –f pz.html –m GET**

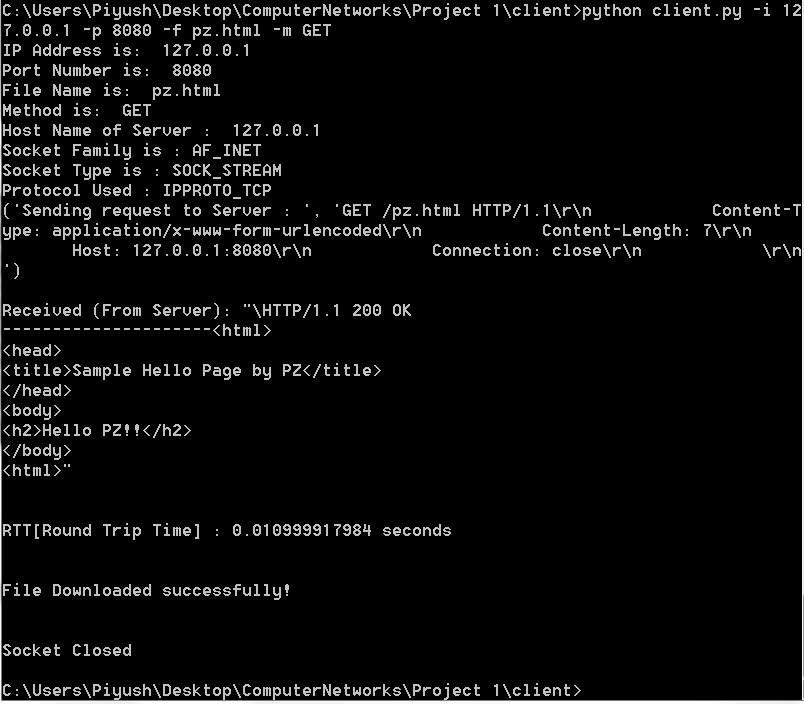
If the user doesn’t run the client code properly, a usage gets printed as below:

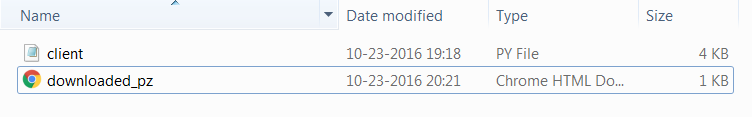


**iii.** The client initiates a connection and requests any page on the server.

1. **When the file is present on the server:**

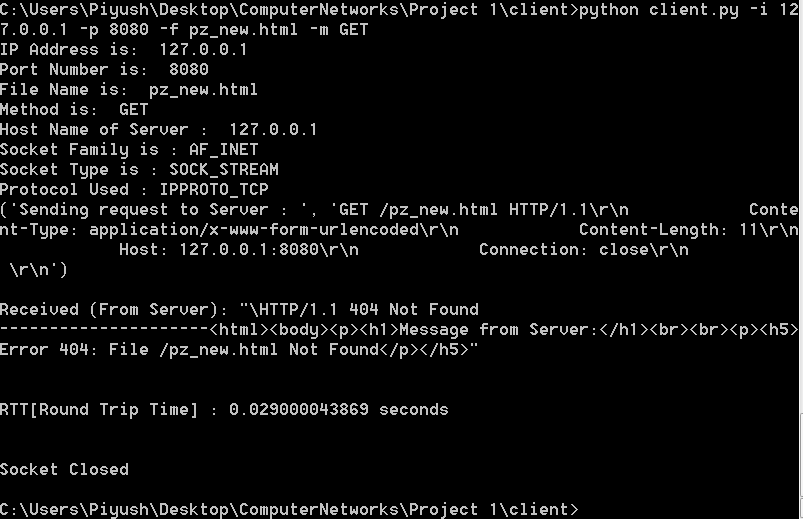
The server responds with a 200 OK message with the file requested. The file gets downloaded in the same folder where the client is present. The downloaded filename name will have the prefix ‘downloaded\_’. The response message from the server also gets printed.





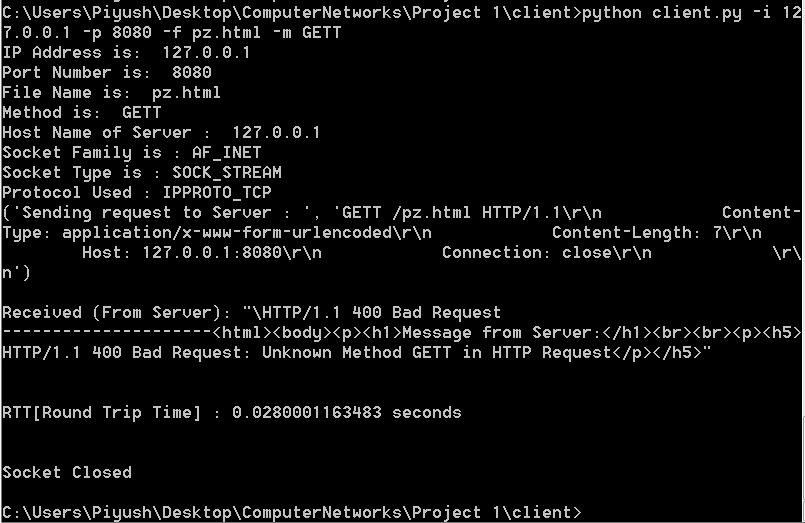
1. **When the requested file is not present at the server:**

The server responds with a 404 Not Found message and the socket is closed.

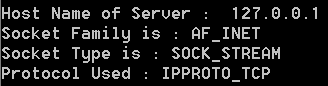


1. **When a bad request is made:**

The server responds with a 400 Bad Request message and the socket is closed.



**iv.** The client also displays the server details and also shows the **RTT[Round Trip Time]** for every request made which can be seen in the above screenshots. The RTT is calculated as the difference between the time when the request is made from the client and time when the response is received at the client.



**References:**

1. <http://www.binarytides.com/python-socket-server-code-example/>
2. <http://stackoverflow.com/questions/1410723/is-there-a-way-to-reopen-a-socket>
3. <https://docs.python.org/2/library/getopt.html>
4. <https://pymotw.com/2/socket/addressing.html>