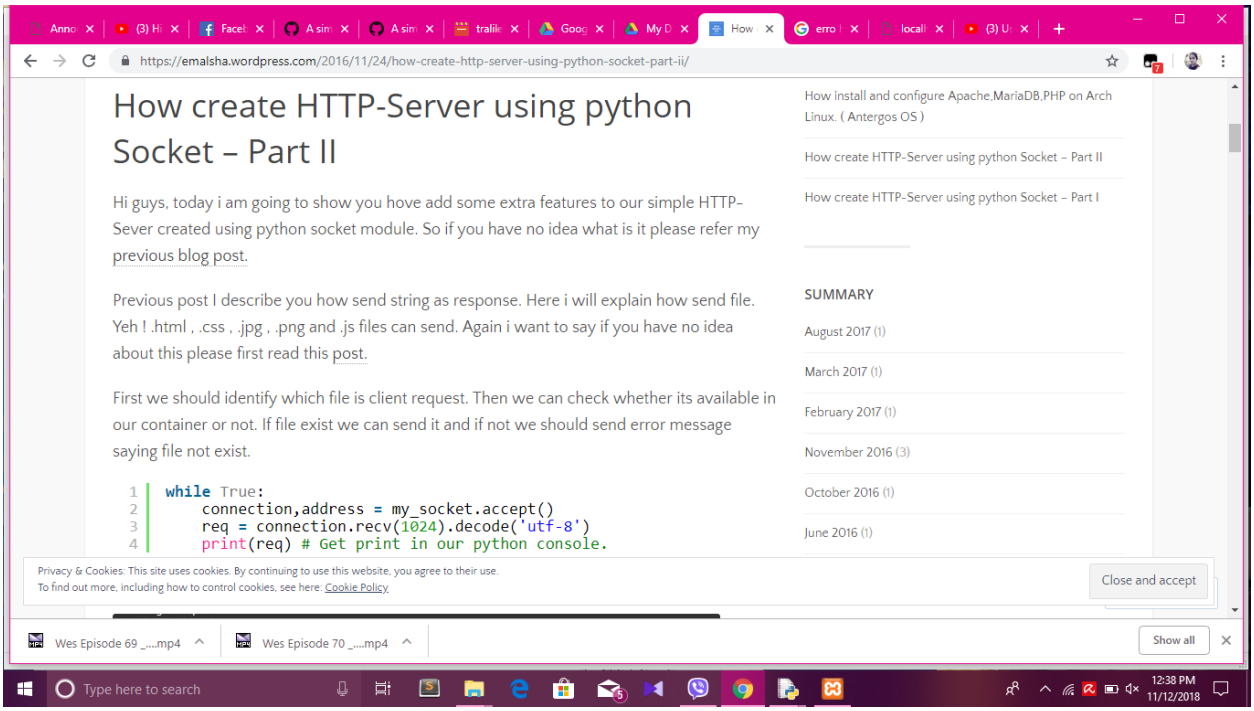


How create HTTP-Server using python Socket – Part II

Hi guys, today i am going to show you hove add some extra features to our simple HTTP-Sever created using python socket module. So if you have no idea what is it please refer my previous blog post.

Previous post I describe you how send string as response. Here i will explain how send file. Yeh ! .html , .css , .jpg , .png and .js files can send. Again i want to say if you have no idea about this please first read this post.

First we should identify which file is client request. Then we can check whether its available in our container or not. If file exist we can send it and if not we should send error message saying file not exist.

```
while True:
    connection,address = my_socket.accept()
    req = connection.recv(1024).decode('utf-8')
    print(req) # Get print in our python console.
```

How install and configure Apache,MariaDB,PHP on Arch Linux. (Antergos OS)

How create HTTP-Server using python Socket – Part II

How create HTTP-Server using python Socket – Part I

SUMMARY

August 2017 (1)

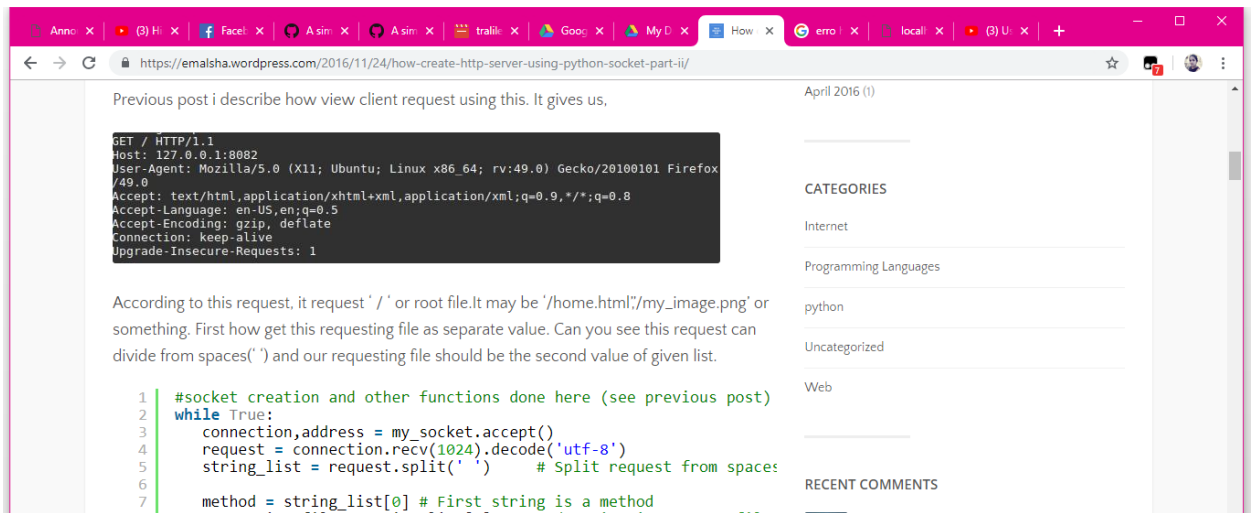
March 2017 (1)

February 2017 (1)

November 2016 (3)

October 2016 (1)

June 2016 (1)



Previous post i describe how view client request using this. It gives us,

```
GET / HTTP/1.1
Host: 127.0.0.1:8082
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:49.0) Gecko/20100101 Firefox
/49.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

According to this request, it request ' / ' or root file.It may be '/home.html","/my_image.png' or something. First how get this requesting file as separate value. Can you see this request can divide from spaces(' ') and our requesting file should be the second value of given list.

```
#socket creation and other functions done here (see previous post)
while True:
    connection,address = my_socket.accept()
    request = connection.recv(1024).decode('utf-8')
    string_list = request.split(' ')     # Split request from spaces

    method = string_list[0] # First string is a method
    requesting_file = string_list[1] #Second string is request file

    print('Client request ',requesting_file)

    connection.close()
```

April 2016 (1)

CATEGORIES

Internet

Programming Languages

python

Uncategorized

Web

RECENT COMMENTS

emalsha on How install and configure Apac...

dag on How install and configure Apac...

Follow

This code will give this ,

```
emalsha@emalsha-Inspiron-3537:~/Desktop$ python3 sample.py
Serving on port  8082
Client requset  /home.html
```

OK , now we can get what file we want as well as which method. If we want to give some conditions according to request method, you have to put if condition. But I don't want this kind of condition in here.

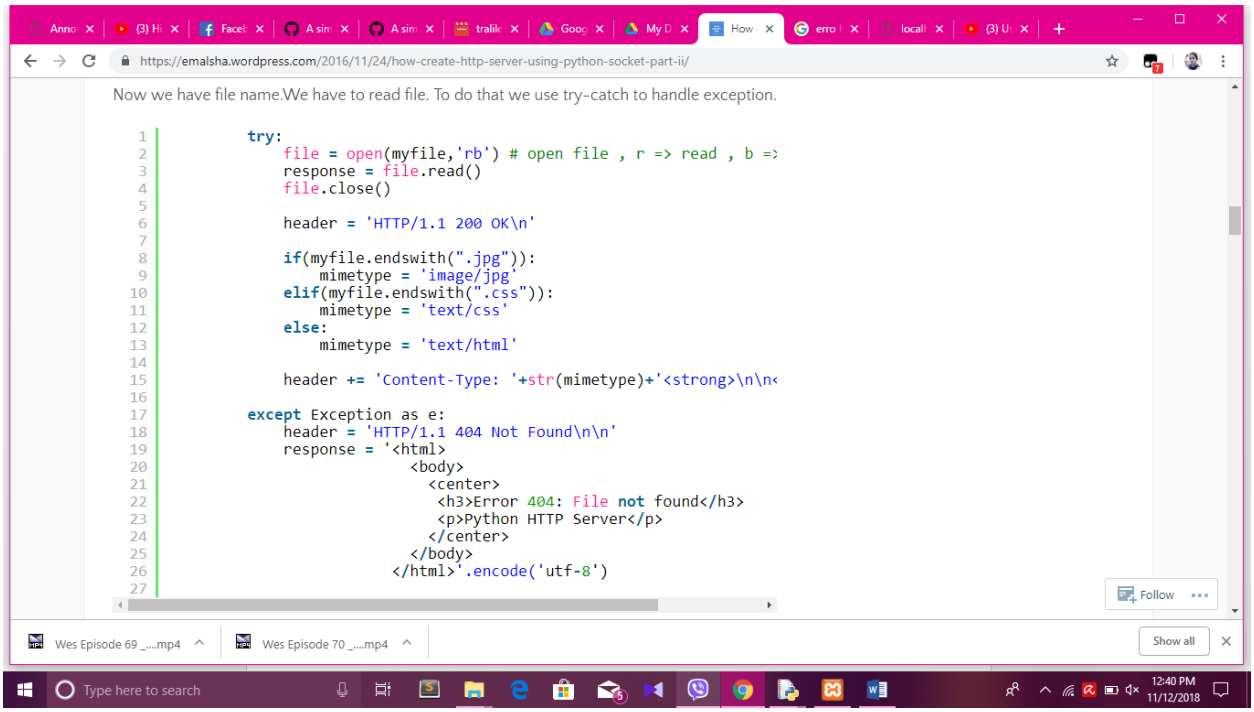
You know sometimes request came like this,

http://www.sample.com/myfile?er='234'

this get request comes with data, but we don't want data in here. Then i split it from ' ? '.

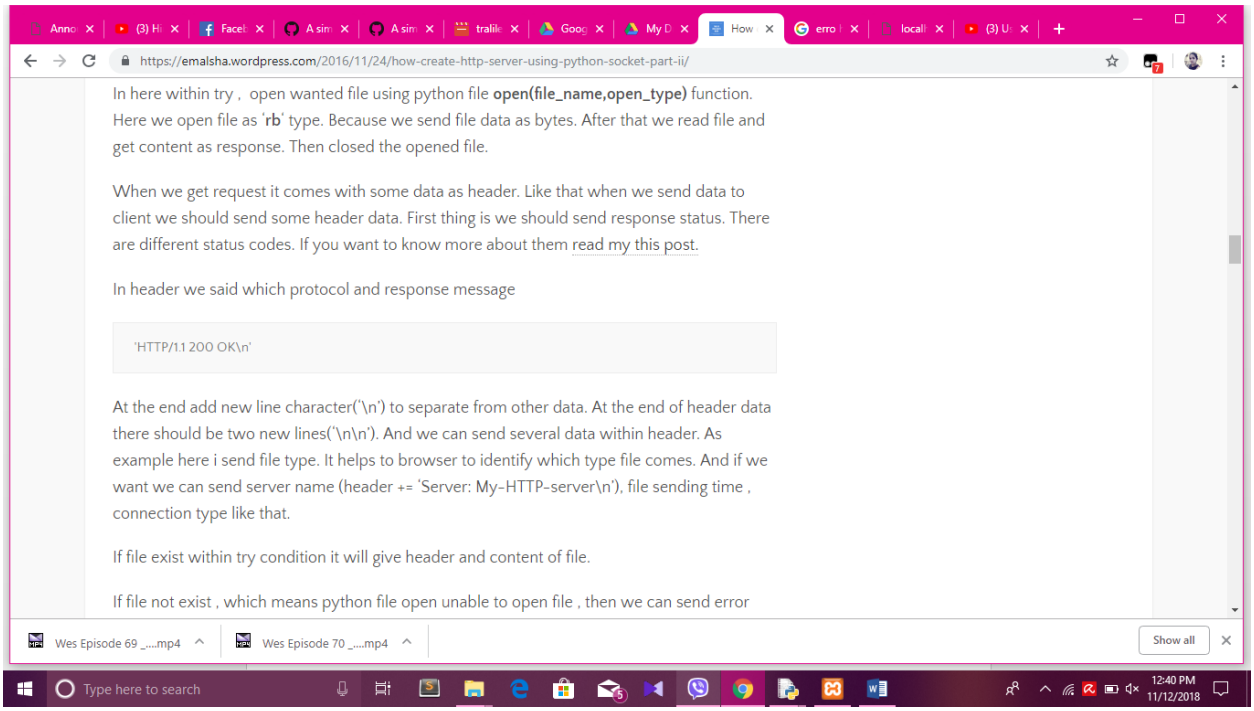
```
1   myfile = requesting_file.split('?')[0] # After the "?" symbol nc
2
3   myfile = myfile.lstrip('/')
4   if(myfile == '/'):
5       myfile = 'index.html'    # Load index file as default
```

hey! , I used **lstrip()** to remove ' / ' from file name. And when i get a request calling ' / ', I want to serve index.html file. Therefore i assign index.html as myfile.



Now we have file name.We have to read file. To do that we use try-catch to handle exception.

```
1           try:
2               file = open(myfile,'rb') # open file , r => read , b =>
3               response = file.read()
4               file.close()
5
6               header = 'HTTP/1.1 200 OK\n'
7
8               if(myfile.endswith(".jpg")):
9                   mimetype = 'image/jpg'
10              elif(myfile.endswith(".css")):
11                  mimetype = 'text/css'
12              else:
13                  mimetype = 'text/html'
14
15              header += 'Content-Type: '+str(mimetype)+'<strong>\n\n<
16
17          except Exception as e:
18              header = 'HTTP/1.1 404 Not Found\n\n'
19              response = '<html>
20                            <body>
21                              <center>
22                               <h3>Error 404: File not found</h3>
23                               <p>Python HTTP Server</p>
24                              </center>
25                            </body>
26                          </html>'.encode('utf-8')
27
```



In here within try , open wanted file using python file **open(file_name,open_type)** function. Here we open file as '**rb**' type. Because we send file data as bytes. After that we read file and get content as response. Then closed the opened file.

When we get request it comes with some data as header. Like that when we send data to client we should send some header data. First thing is we should send response status. There are different status codes. If you want to know more about them read my this post.

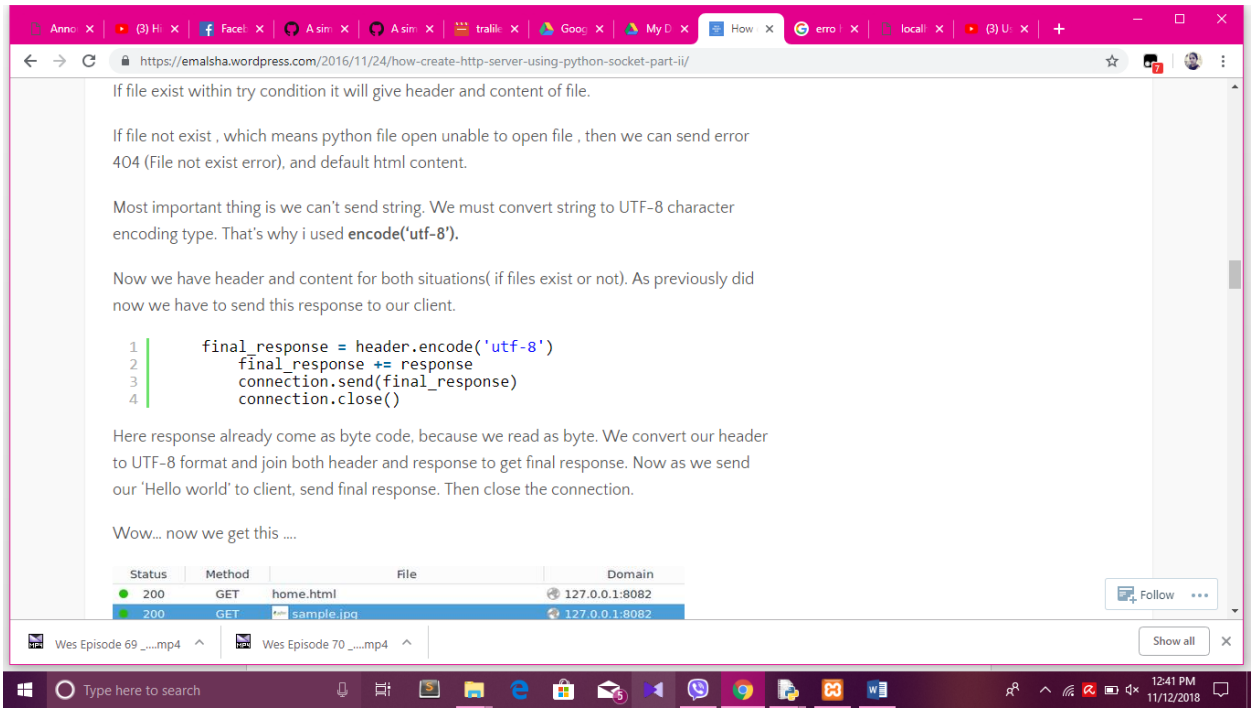
In header we said which protocol and response message

```
'HTTP/1.1 200 OK\n'
```

At the end add new line character('\n') to separate from other data. At the end of header data there should be two new lines('\n\n'). And we can send several data within header. As example here i send file type. It helps to browser to identify which type file comes. And if we want we can send server name (header += 'Server: My-HTTP-server\n'), file sending time , connection type like that.

If file exist within try condition it will give header and content of file.

If file not exist , which means python file open unable to open file , then we can send error



If file exist within try condition it will give header and content of file.

If file not exist , which means python file open unable to open file , then we can send error 404 (File not exist error), and default html content.

Most important thing is we can't send string. We must convert string to UTF-8 character encoding type. That's why i used **encode('utf-8').**

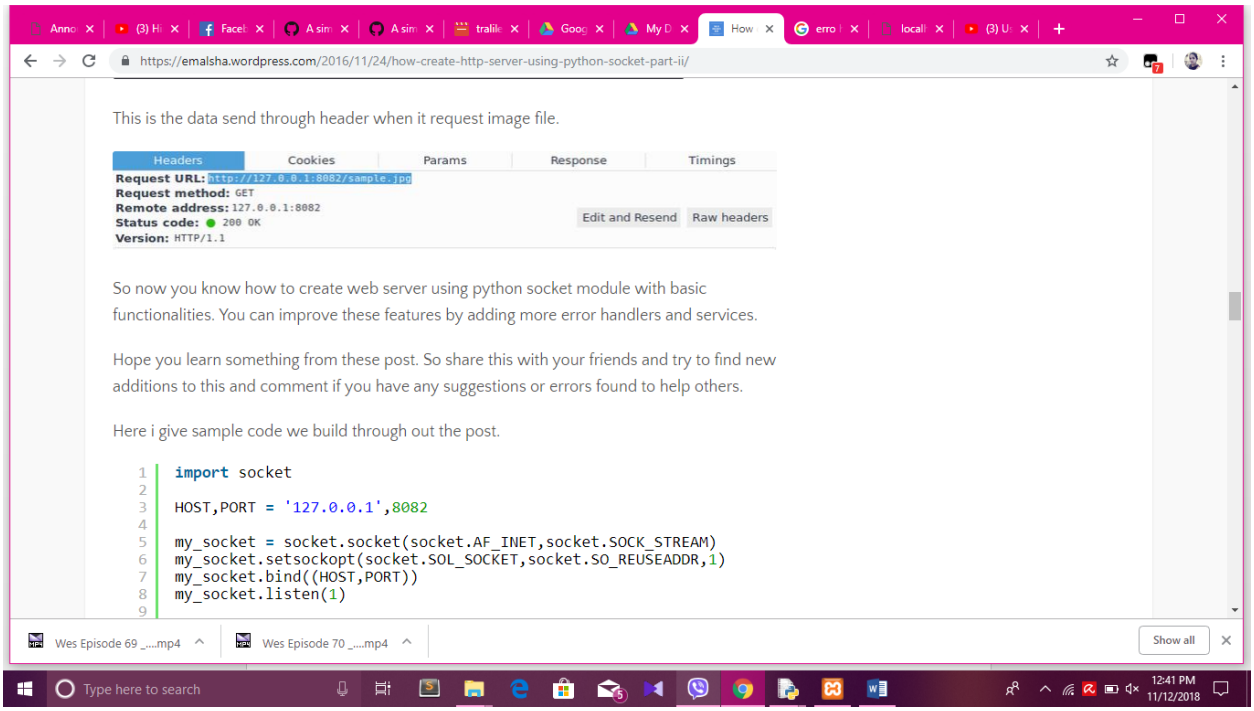
Now we have header and content for both situations(if files exist or not). As previously did now we have to send this response to our client.

```
1    final_response = header.encode('utf-8')
2        final_response += response
3        connection.send(final_response)
4        connection.close()
```

Here response already come as byte code, because we read as byte. We convert our header to UTF-8 format and join both header and response to get final response. Now as we send our 'Hello world' to client, send final response. Then close the connection.

Wow... now we get this ....

Status	Method	File	Domain
200	GET	home.html	127.0.0.1:8082
200	GET	sample.jpg	127.0.0.1:8082



This is the data send through header when it request image file.

Headers | Cookies | Params | Response | Timings

Request URL: http://127.0.0.1:8082/sample.jpg
Request method: GET
Remote address: 127.0.0.1:8082
Status code: 200 OK
Version: HTTP/1.1

Edit and Resend | Raw headers

So now you know how to create web server using python socket module with basic functionalities. You can improve these features by adding more error handlers and services.

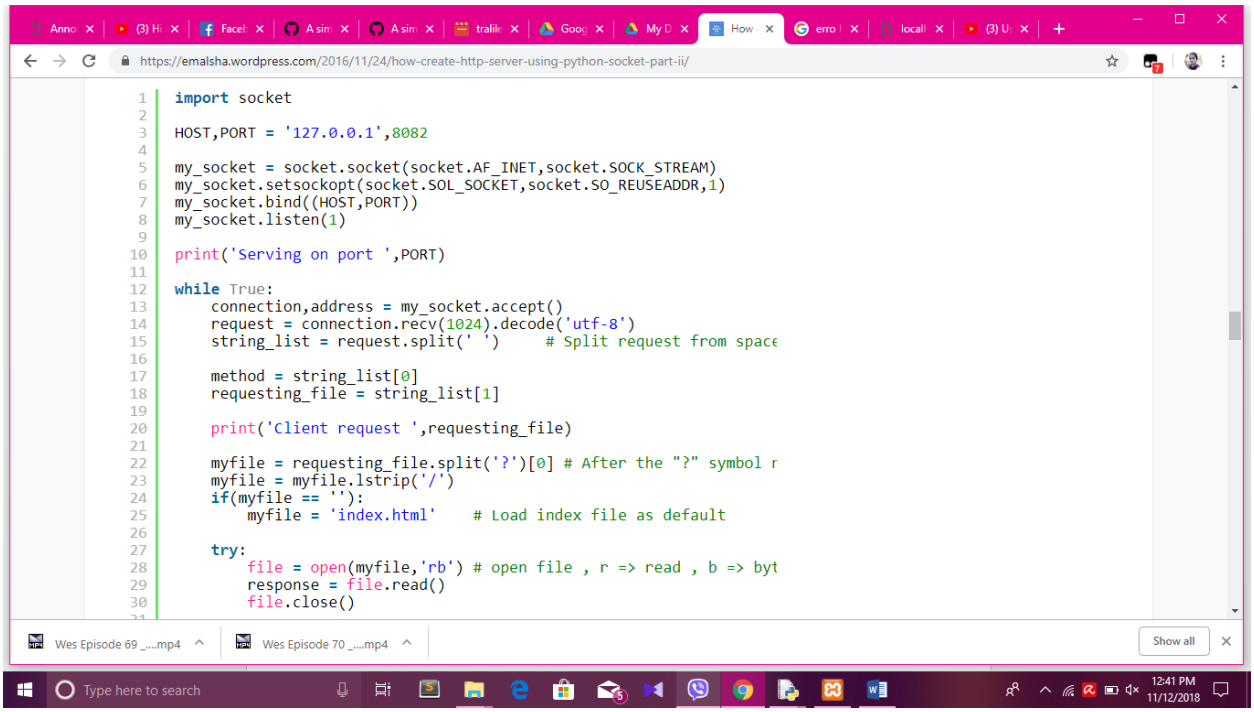
Hope you learn something from these post. So share this with your friends and try to find new additions to this and comment if you have any suggestions or errors found to help others.

Here i give sample code we build through out the post.

```
1  import socket
2
3  HOST,PORT = '127.0.0.1',8082
4
5  my_socket = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
6  my_socket.setsockopt(socket.SOL_SOCKET,socket.SO_REUSEADDR,1)
7  my_socket.bind((HOST,PORT))
8  my_socket.listen(1)
9
```

Wes Episode 69 _....mp4 | Wes Episode 70 _....mp4 | Show all

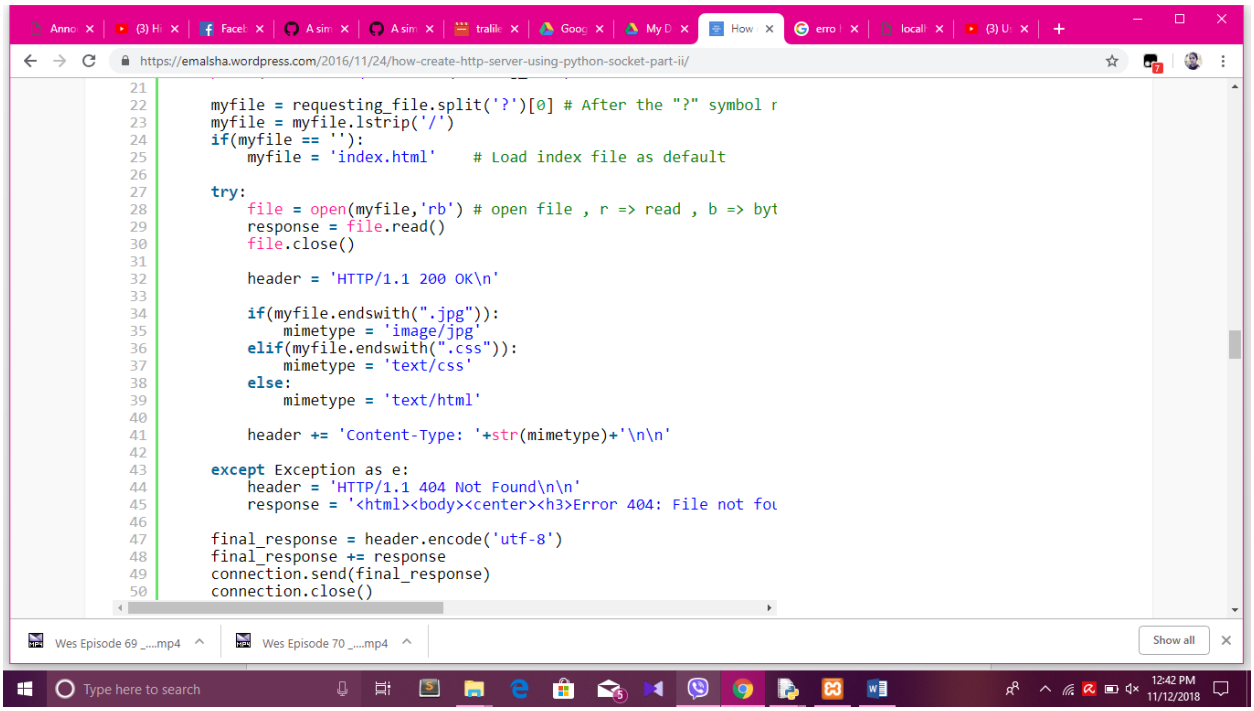
Type here to search 12:41 PM 11/12/2018



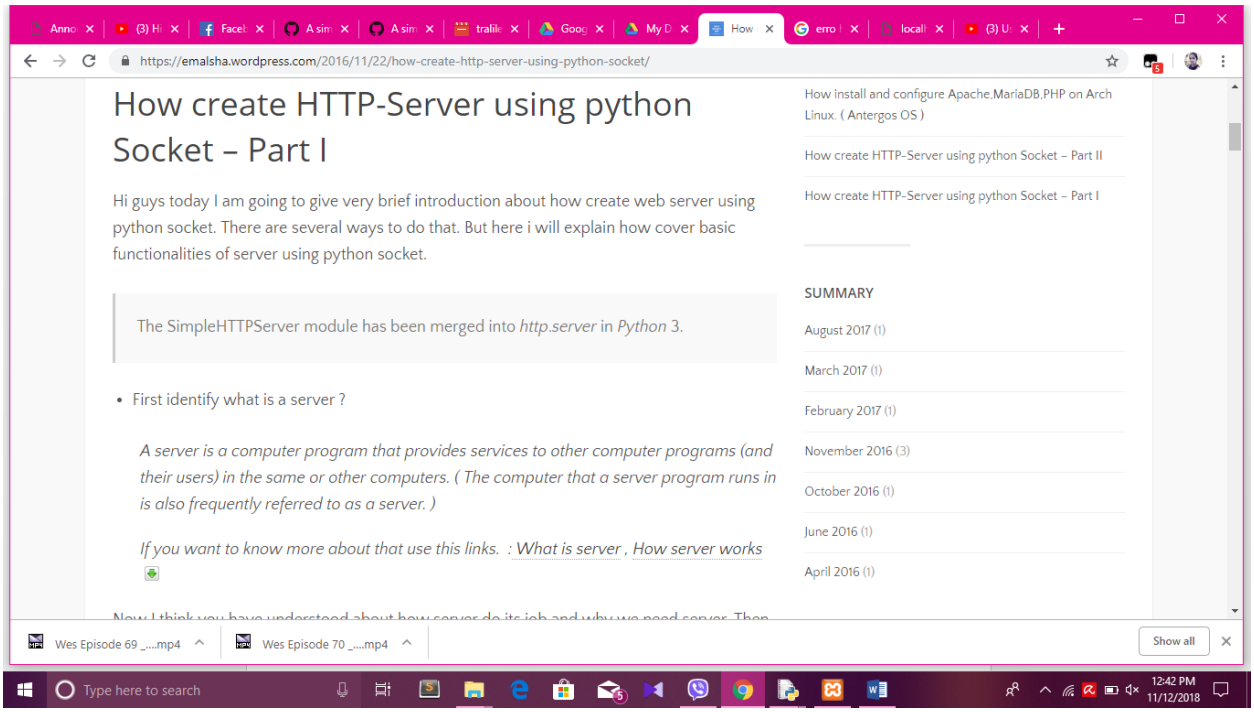
```
1  import socket
2
3  HOST,PORT = '127.0.0.1',8082
4
5  my_socket = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
6  my_socket.setsockopt(socket.SOL_SOCKET,socket.SO_REUSEADDR,1)
7  my_socket.bind((HOST,PORT))
8  my_socket.listen(1)
9
10 print('Serving on port ',PORT)
11
12 while True:
13     connection,address = my_socket.accept()
14     request = connection.recv(1024).decode('utf-8')
15     string_list = request.split(' ')     # Split request from space
16
17     method = string_list[0]
18     requesting_file = string_list[1]
19
20     print('Client request ',requesting_file)
21
22     myfile = requesting_file.split('?')[0] # After the "?" symbol r
23     myfile = myfile.lstrip('/')
24     if(myfile == ''):
25         myfile = 'index.html'    # Load index file as default
26
27     try:
28         file = open(myfile,'rb') # open file , r => read , b => byt
29         response = file.read()
30         file.close()
```

Wes Episode 69 _....mp4 | Wes Episode 70 _....mp4 | Show all

Type here to search 12:41 PM 11/12/2018



```
    myfile = requesting_file.split('?')[0] # After the "?" symbol r
    myfile = myfile.lstrip('/')
    if(myfile == ''):
        myfile = 'index.html'    # Load index file as default

    try:
        file = open(myfile,'rb') # open file , r => read , b => byt
        response = file.read()
        file.close()

        header = 'HTTP/1.1 200 OK\n'

        if(myfile.endswith(".jpg")):
            mimetype = 'image/jpg'
        elif(myfile.endswith(".css")):
            mimetype = 'text/css'
        else:
            mimetype = 'text/html'

        header += 'Content-Type: '+str(mimetype)+'\n\n'

    except Exception as e:
        header = 'HTTP/1.1 404 Not Found\n\n'
        response = '<html><body><center><h3>Error 404: File not fou

    final_response = header.encode('utf-8')
    final_response += response
    connection.send(final_response)
    connection.close()
```



How create HTTP-Server using python Socket – Part I

Hi guys today I am going to give very brief introduction about how create web server using python socket. There are several ways to do that. But here i will explain how cover basic functionalities of server using python socket.

> The SimpleHTTPServer module has been merged into *http.server* in *Python* 3.

- First identify what is a server ?

  A server is a computer program that provides services to other computer programs (and their users) in the same or other computers. (The computer that a server program runs in is also frequently referred to as a server.)

  If you want to know more about that use this links. : What is server , How server works

How install and configure Apache,MariaDB,PHP on Arch Linux. (Antergos OS)

How create HTTP-Server using python Socket – Part II

How create HTTP-Server using python Socket – Part I

SUMMARY

August 2017 (1)

March 2017 (1)

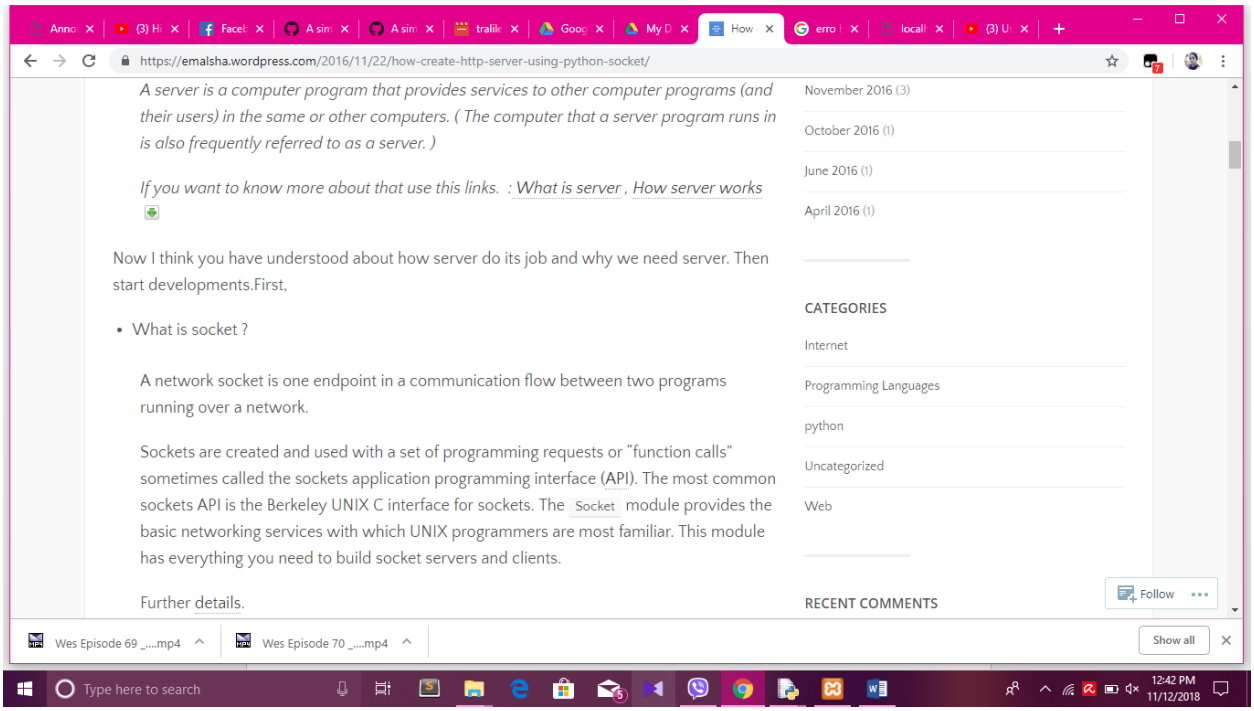
February 2017 (1)

November 2016 (3)

October 2016 (1)

June 2016 (1)

April 2016 (1)



A server is a computer program that provides services to other computer programs (and their users) in the same or other computers. (The computer that a server program runs in is also frequently referred to as a server.)

If you want to know more about that use this links. : What is server , How server works

Now I think you have understood about how server do its job and why we need server. Then start developments.First,

- What is socket ?

A network socket is one endpoint in a communication flow between two programs running over a network.

Sockets are created and used with a set of programming requests or "function calls" sometimes called the sockets application programming interface (API). The most common sockets API is the Berkeley UNIX C interface for sockets. The `Socket` module provides the basic networking services with which UNIX programmers are most familiar. This module has everything you need to build socket servers and clients.

Further details.

November 2016 (3)

October 2016 (1)

June 2016 (1)

April 2016 (1)

CATEGORIES

Internet

Programming Languages

python

Uncategorized

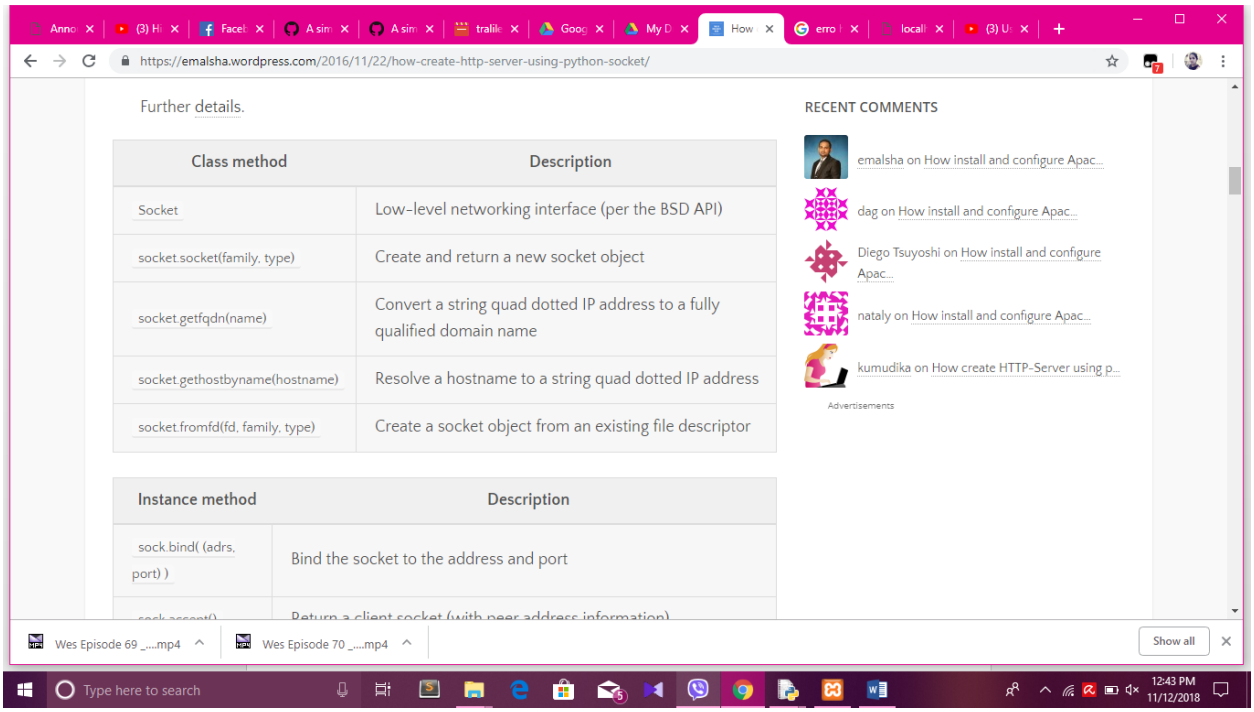
Web

RECENT COMMENTS

Follow

Wes Episode 69 _....mp4 | Wes Episode 70 _....mp4 | Show all

Type here to search 12:42 PM 11/12/2018



Further details.

Class method	Description
Socket	Low-level networking interface (per the BSD API)
socket.socket(family, type)	Create and return a new socket object
socket.getfqdn(name)	Convert a string quad dotted IP address to a fully qualified domain name
socket.gethostbyname(hostname)	Resolve a hostname to a string quad dotted IP address
socket.fromfd(fd, family, type)	Create a socket object from an existing file descriptor

Instance method	Description
sock.bind((adrs, port))	Bind the socket to the address and port
sock.accept()	Return a client socket (with peer address information)

RECENT COMMENTS

emalsha on How install and configure Apac...

dag on How install and configure Apac...

Diego Tsuyoshi on How install and configure Apac...

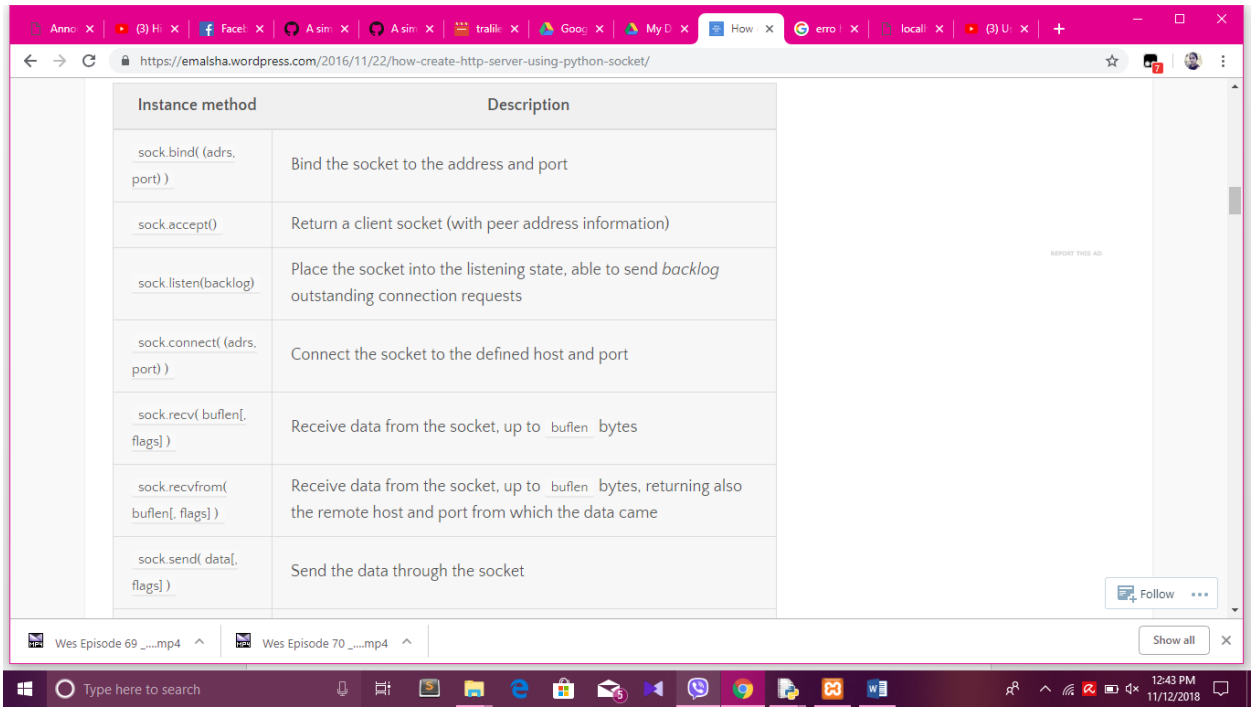
nataly on How install and configure Apac...

kumudika on How create HTTP-Server using p...

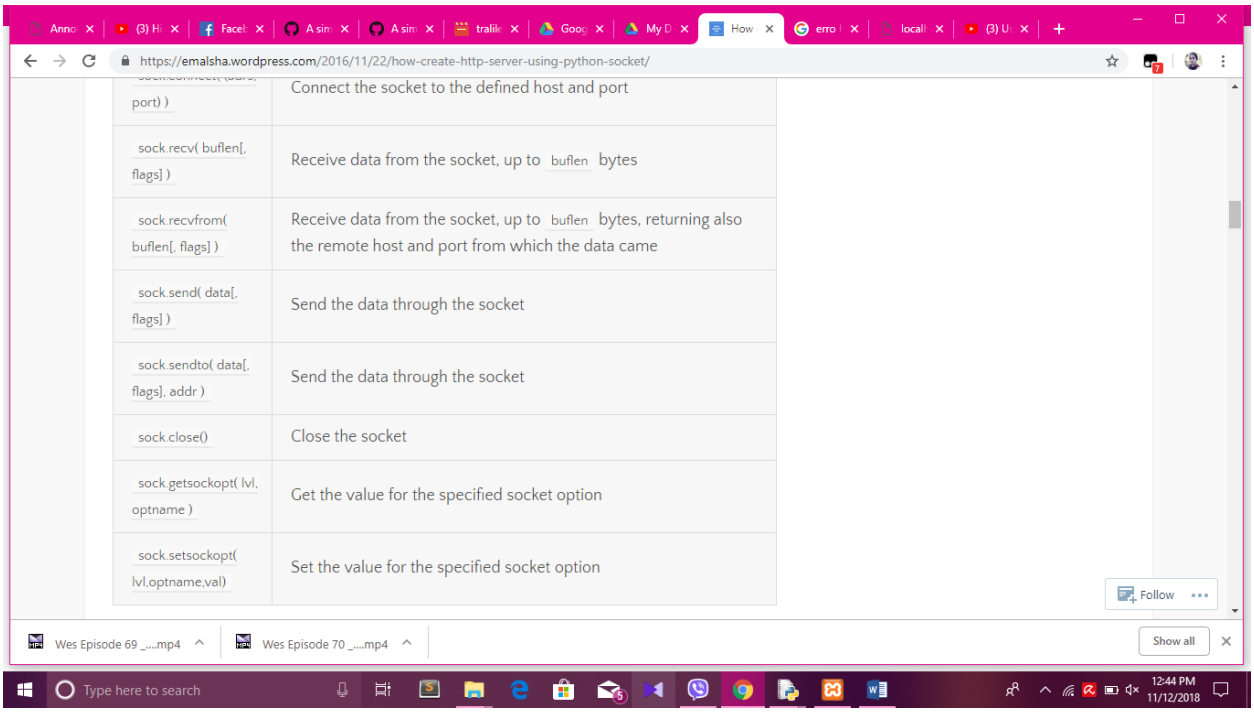
Advertisements

Wes Episode 69 _....mp4 | Wes Episode 70 _....mp4 | Show all

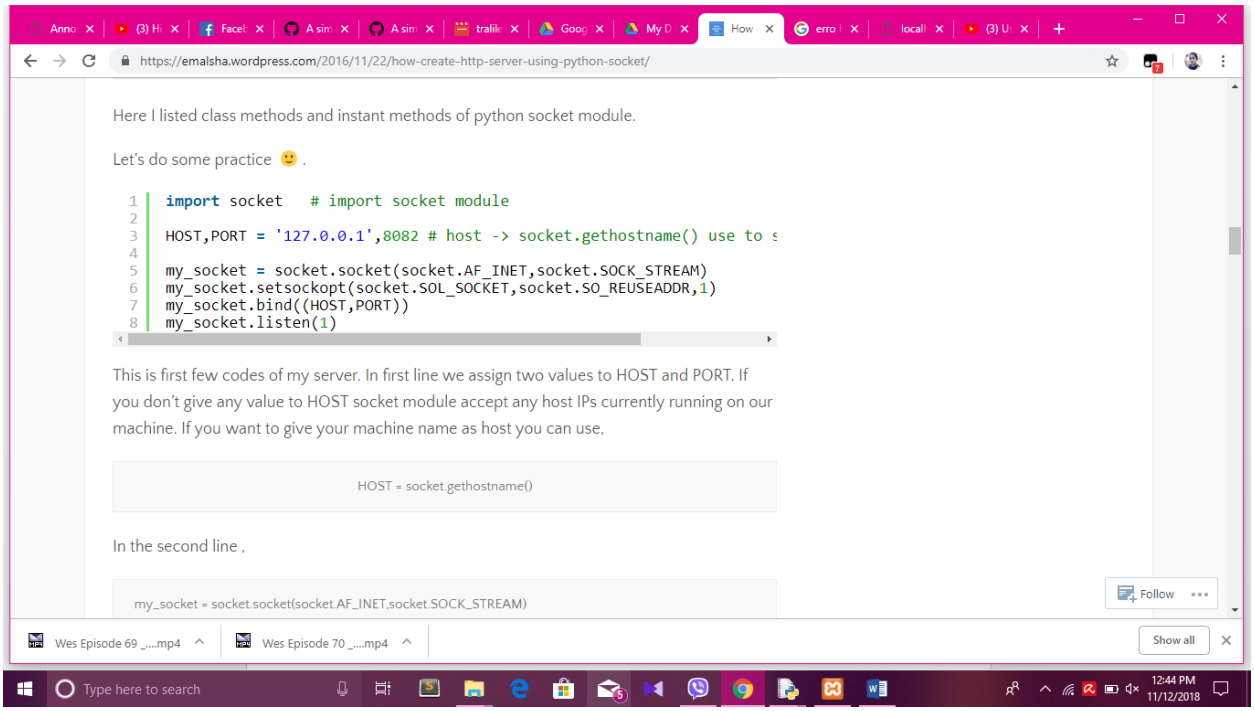
Type here to search 12:43 PM 11/12/2018



Instance method	Description
sock.bind((adrs, port))	Bind the socket to the address and port
sock.accept()	Return a client socket (with peer address information)
sock.listen(backlog)	Place the socket into the listening state, able to send *backlog* outstanding connection requests
sock.connect((adrs, port))	Connect the socket to the defined host and port
sock.recv(buflen[, flags])	Receive data from the socket, up to buflen bytes
sock.recvfrom(buflen[, flags])	Receive data from the socket, up to buflen bytes, returning also the remote host and port from which the data came
sock.send(data[, flags])	Send the data through the socket



port))	Connect the socket to the defined host and port
sock.recv(buflen[, flags])	Receive data from the socket, up to buflen bytes
sock.recvfrom(buflen[, flags])	Receive data from the socket, up to buflen bytes, returning also the remote host and port from which the data came
sock.send(data[, flags])	Send the data through the socket
sock.sendto(data[, flags], addr)	Send the data through the socket
sock.close()	Close the socket
sock.getsockopt(lvl, optname)	Get the value for the specified socket option
sock.setsockopt(lvl,optname,val)	Set the value for the specified socket option



Here I listed class methods and instant methods of python socket module.

Let's do some practice 🙂 .

```
1  import socket   # import socket module
2
3  HOST,PORT = '127.0.0.1',8082 # host -> socket.gethostname() use to s
4
5  my_socket = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
6  my_socket.setsockopt(socket.SOL_SOCKET,socket.SO_REUSEADDR,1)
7  my_socket.bind((HOST,PORT))
8  my_socket.listen(1)
```

This is first few codes of my server. In first line we assign two values to HOST and PORT. If you don't give any value to HOST socket module accept any host IPs currently running on our machine. If you want to give your machine name as host you can use,

```
HOST = socket.gethostname()
```

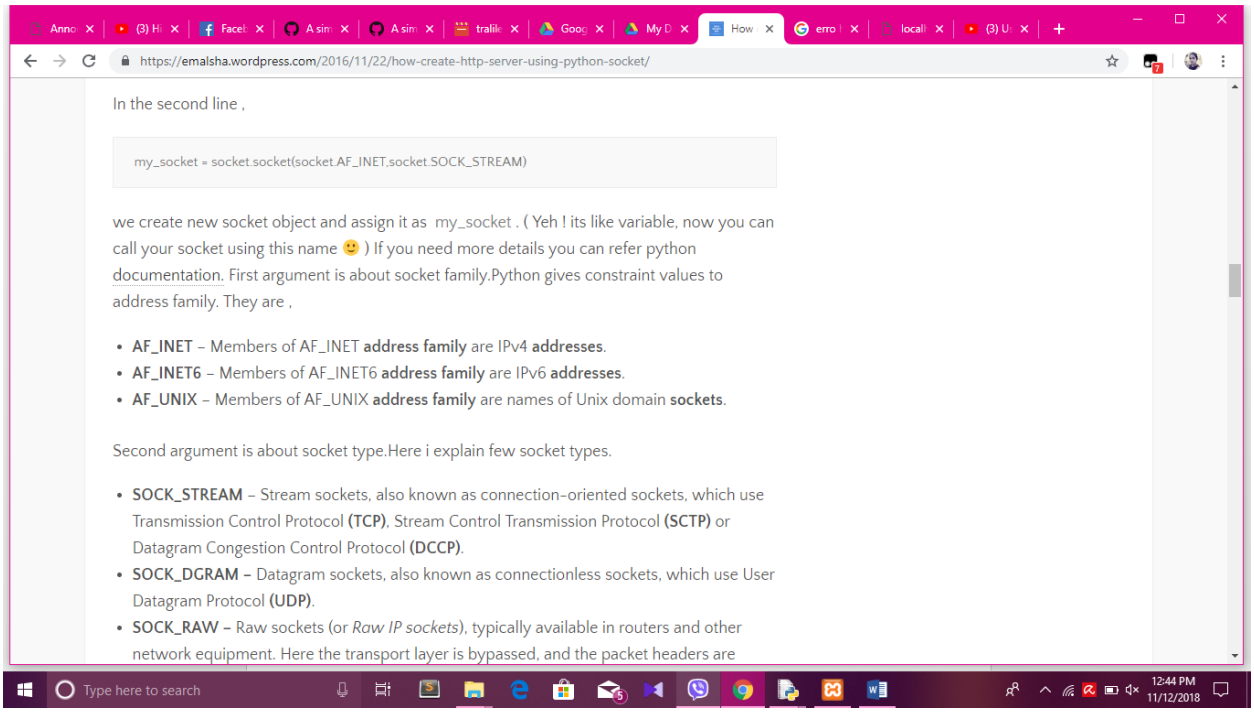
In the second line ,

```
my_socket = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
```

Follow

Wes Episode 69 _....mp4 Wes Episode 70 _....mp4 Show all

Type here to search 12:44 PM 11/12/2018



In the second line ,

```
my_socket = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
```

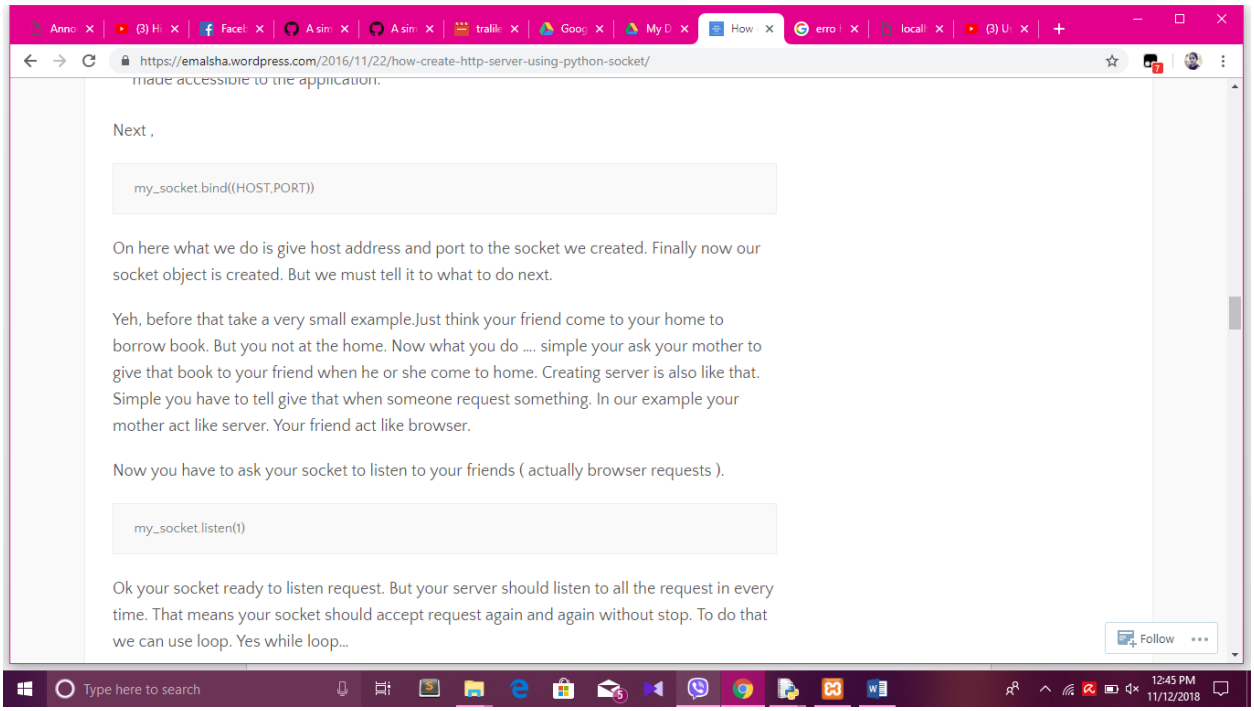
we create new socket object and assign it as my_socket . (Yeh ! its like variable, now you can call your socket using this name 🙂) If you need more details you can refer python documentation. First argument is about socket family.Python gives constraint values to address family. They are ,

- **AF_INET** – Members of AF_INET **address family** are IPv4 **addresses**.
- **AF_INET6** – Members of AF_INET6 **address family** are IPv6 **addresses**.
- **AF_UNIX** – Members of AF_UNIX **address family** are names of Unix domain **sockets**.

Second argument is about socket type.Here i explain few socket types.

- **SOCK_STREAM** – Stream sockets, also known as connection-oriented sockets, which use Transmission Control Protocol **(TCP)**, Stream Control Transmission Protocol **(SCTP)** or Datagram Congestion Control Protocol **(DCCP)**.
- **SOCK_DGRAM –** Datagram sockets, also known as connectionless sockets, which use User Datagram Protocol **(UDP)**.
- **SOCK_RAW –** Raw sockets (or *Raw IP sockets*), typically available in routers and other network equipment. Here the transport layer is bypassed, and the packet headers are

Type here to search 12:44 PM 11/12/2018



made accessible to the application.

Next ,

```
my_socket.bind((HOST,PORT))
```

On here what we do is give host address and port to the socket we created. Finally now our socket object is created. But we must tell it to what to do next.

Yeh, before that take a very small example.Just think your friend come to your home to borrow book. But you not at the home. Now what you do .... simple your ask your mother to give that book to your friend when he or she come to home. Creating server is also like that. Simple you have to tell give that when someone request something. In our example your mother act like server. Your friend act like browser.

Now you have to ask your socket to listen to your friends (actually browser requests).

```
my_socket.listen(1)
```

Ok your socket ready to listen request. But your server should listen to all the request in every time. That means your socket should accept request again and again without stop. To do that we can use loop. Yes while loop...

Follow

we can use loop. Yes while loop...

```
while True:
    connection,address = my_socket.accept()
    req = connection.recv(1024).decode('utf-8')
    print(req) # Get print in our python console.
```

accept() , accept request and return two valuable information. First socket connection and client socket address. Here after we use this connection to send or serve what client requested. When we get request this connection contain about request details. To get this data we can use

```
connection.recv(1024)
```

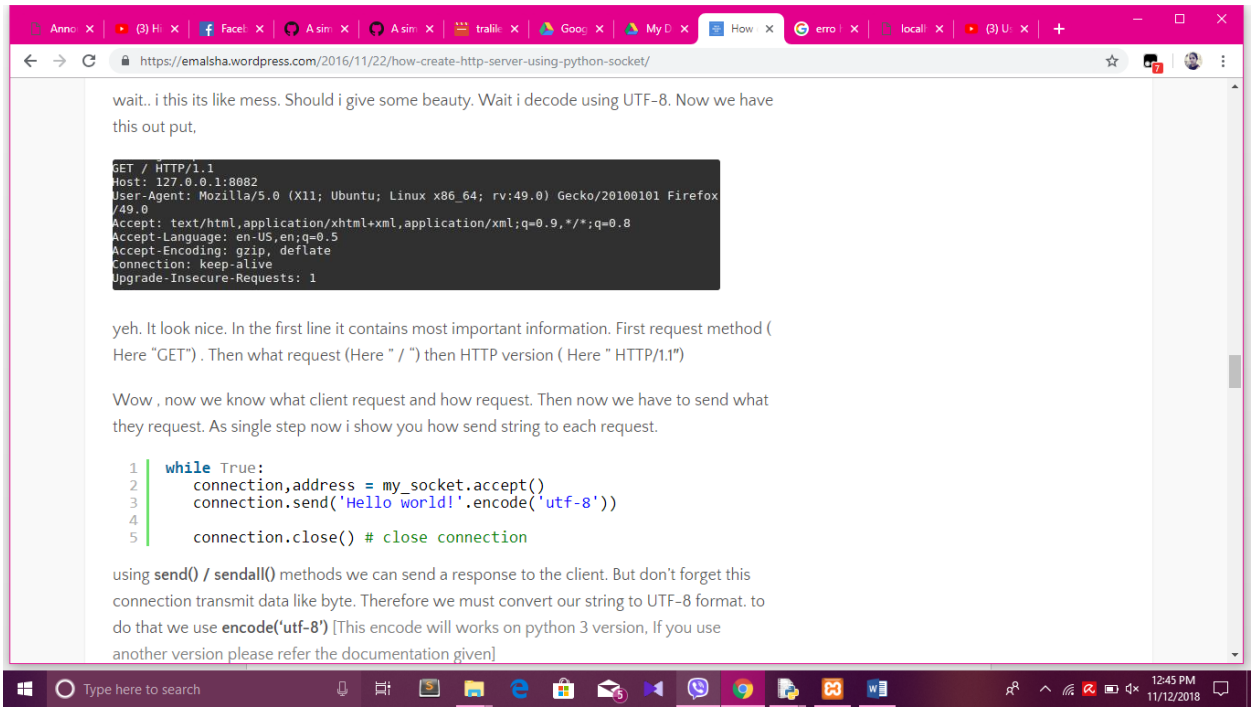
recv() return the request details and we said get maximum 1024 bytes from connection request. But its look like this,

```
Serving on port  8082
b'GET / HTTP/1.1\r\nHost: 127.0.0.1:8082\r\nUser-Agent: Mozilla/5.0 (X11; Ubuntu; L
inux x86_64; rv:49.0) Gecko/20100101 Firefox/49.0\r\nAccept: text/html,application/
xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\nAccept-Language: en-US,en;q=0.5\r\nAcc
ept-Encoding: gzip, deflate\r\nConnection: keep-alive\r\nUpgrade-Insecure-Requests:
 1\r\n\r\n'
```

wait.. i this its like mess. Should i give some beauty. Wait i decode using UTF-8. Now we have this out put,

```
GET / HTTP/1.1
```

Follow



wait.. i this its like mess. Should i give some beauty. Wait i decode using UTF-8. Now we have this out put,

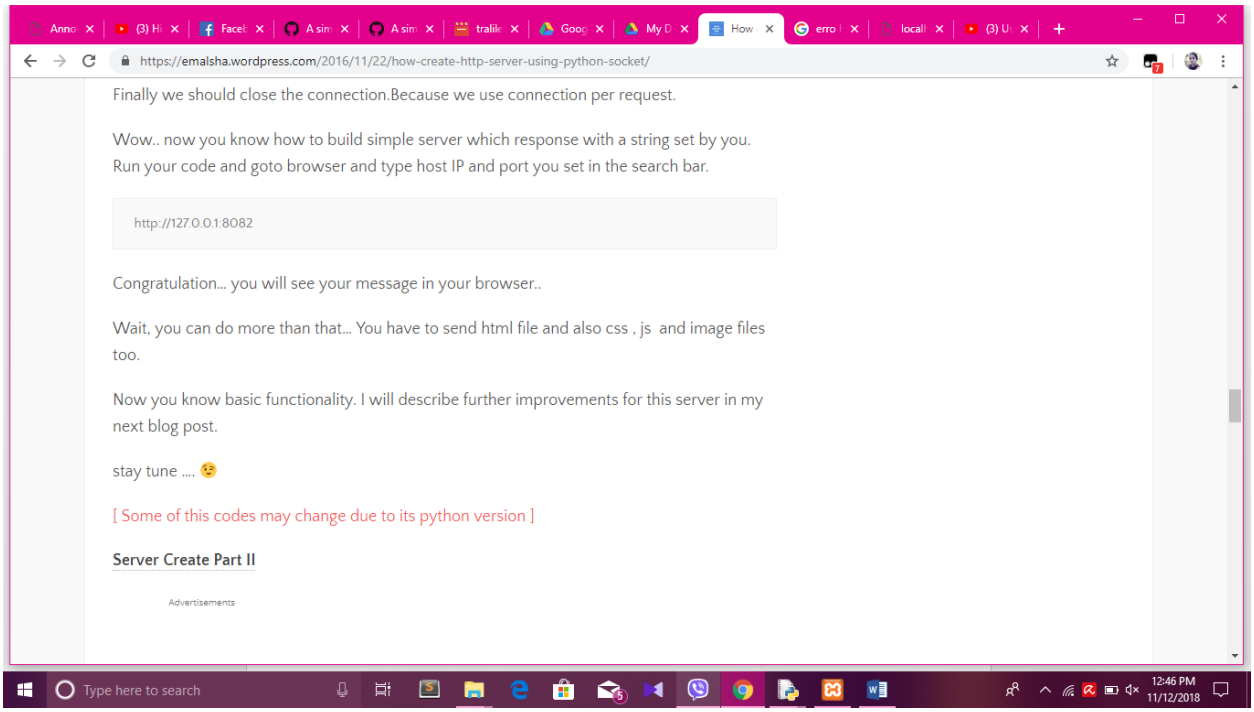
```
GET / HTTP/1.1
Host: 127.0.0.1:8082
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:49.0) Gecko/20100101 Firefox
/49.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

yeh. It look nice. In the first line it contains most important information. First request method (Here "GET") . Then what request (Here " / ") then HTTP version (Here " HTTP/1.1")

Wow , now we know what client request and how request. Then now we have to send what they request. As single step now i show you how send string to each request.

```
while True:
    connection,address = my_socket.accept()
    connection.send('Hello world!'.encode('utf-8'))

    connection.close() # close connection
```

using **send() / sendall()** methods we can send a response to the client. But don't forget this connection transmit data like byte. Therefore we must convert our string to UTF-8 format. to do that we use **encode('utf-8')** [This encode will works on python 3 version, If you use another version please refer the documentation given]



Finally we should close the connection.Because we use connection per request.

Wow.. now you know how to build simple server which response with a string set by you. Run your code and goto browser and type host IP and port you set in the search bar.

```
http://127.0.0.1:8082
```

Congratulation... you will see your message in your browser..

Wait, you can do more than that... You have to send html file and also css , js and image files too.

Now you know basic functionality. I will describe further improvements for this server in my next blog post.

stay tune .... 😉

[Some of this codes may change due to its python version]

Server Create Part II