**TCP/IP connection (python)**

The primary socket API functions and methods in this module are:

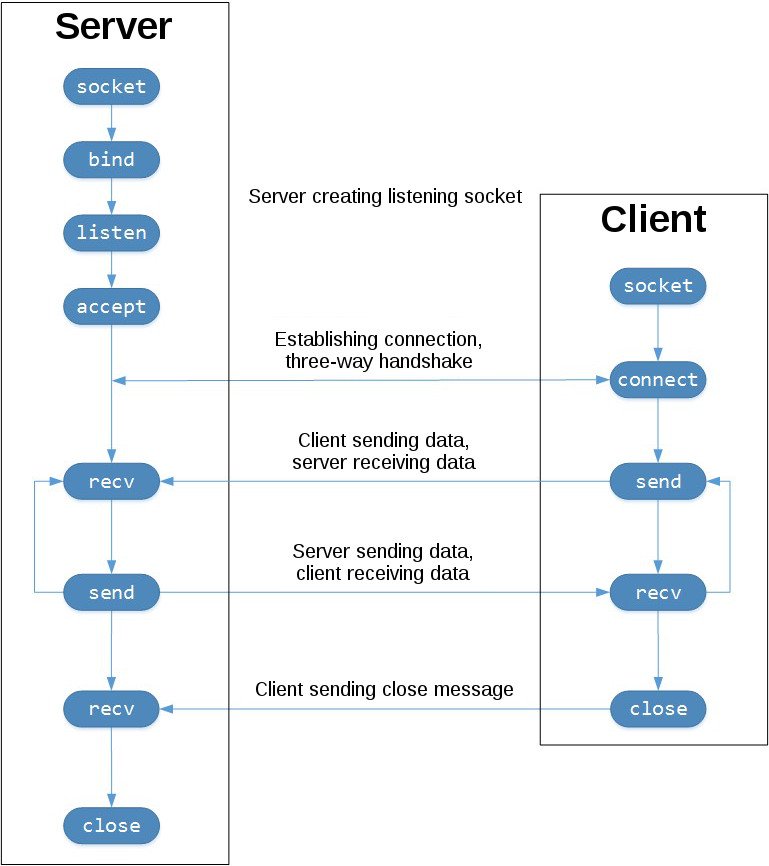
* socket()
* bind()
* listen()
* accept()
* connect()
* connect\_ex()
* send()
* recv()
* close()

**Why should you use TCP? The Transmission Control Protocol (TCP):**

* **Is reliable:** packets dropped in the network are detected and retransmitted by the sender.
* **Has in-order data delivery:** data is read by your application in the order it was written by the sender.

In contrast, [User Datagram Protocol (UDP)](https://en.wikipedia.org/wiki/User_Datagram_Protocol) sockets created with socket.SOCK\_DGRAM aren’t reliable, and data read by the receiver can be out-of-order from the sender’s writes.

**TCP/IP socket workflow**



**TCP/IP Connection Example**

**Server/Receiver >> server.py**

import socket

HOST = '127.0.0.1' # Standard loopback interface address (localhost)

PORT = 65432 # Port to listen on (non-privileged ports are > 1023)

With socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) as s:

s.bind((HOST, PORT))

s.listen()

conn, addr = s.accept()

with conn:

print('Connected by', addr)

while True: #l

data = conn.recv(1024)

if not data:

break

conn.sendall(data)

**Client/Sender >> client.py**

import socket

HOST = '127.0.0.1' # The server's hostname or IP address

PORT = 65432 # The port used by the server

with socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) as s:

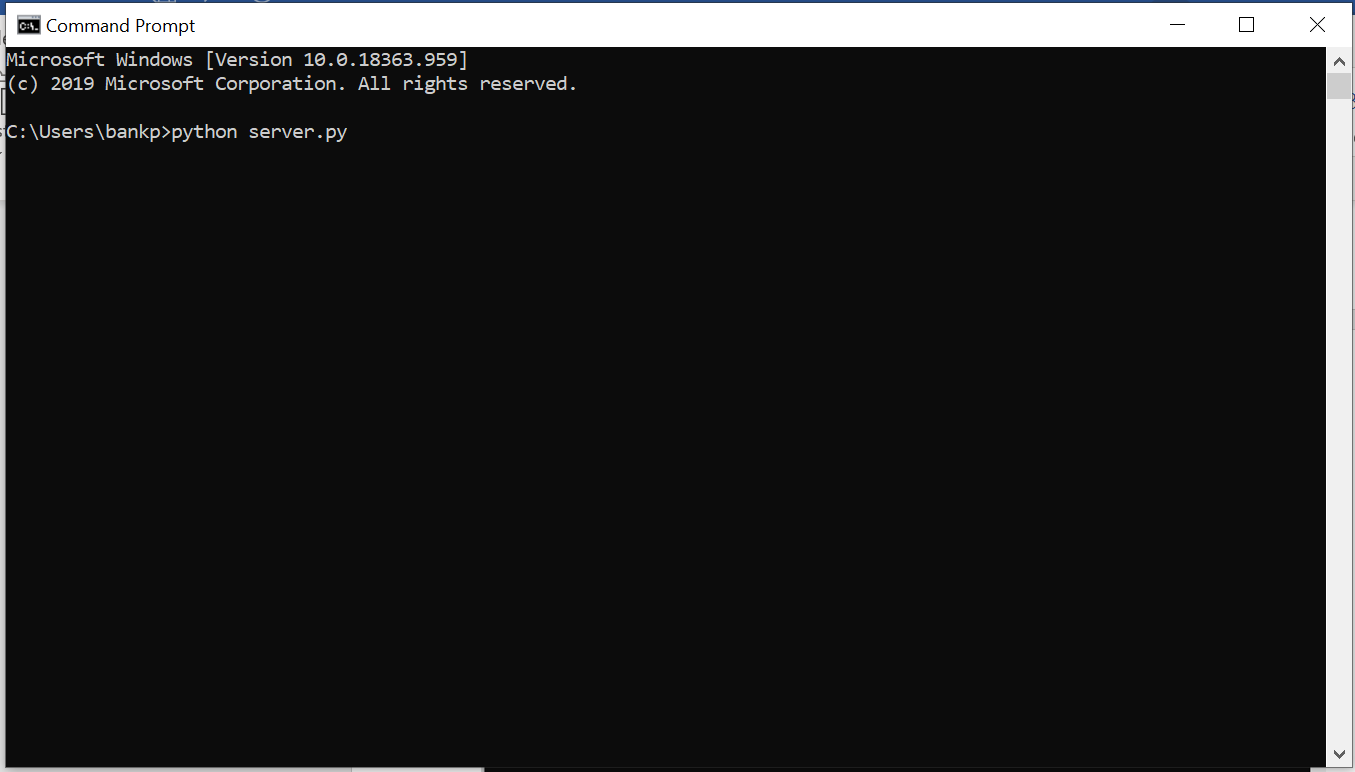
s.connect((HOST, PORT))

s.sendall(b'Hello, world')

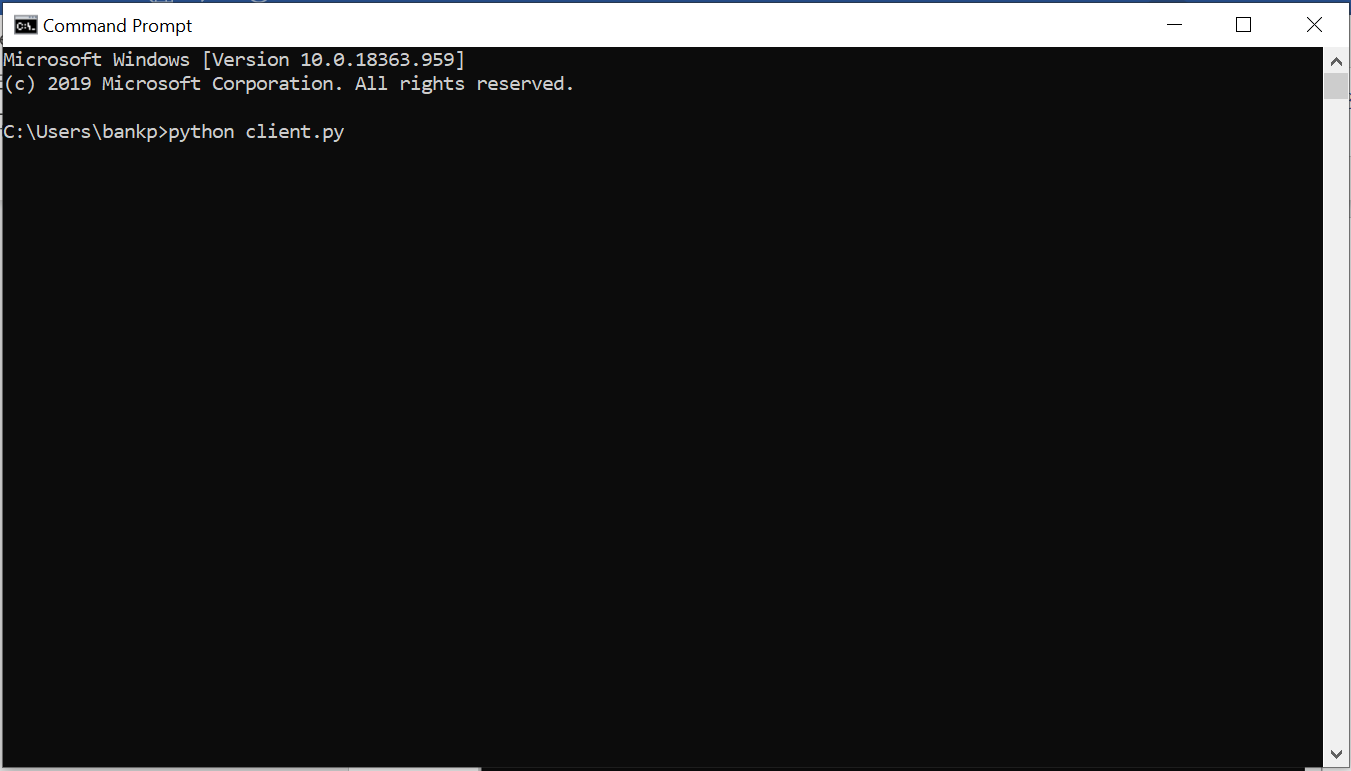
data = s.recv(1024)

print('Received', repr(data))

Run these codes separately using cmd

For server.py code

For client.py code



To streaming twitter hashtags, you need to import this

from twython import TwythonStreamer

class TweetStreamer(TwythonStreamer):

def on\_success(self, data):

#this def will work when new subscribed tweet has been posted

#you can put any code in this field

#for example, I want to print tweet message

print(data[‘text’])

def on\_error(self, status\_code, data):

print (status\_code)

self.disconnect()

consumer\_key = 'X5gCNKpqL6NUsLvBJjIYIm4Pz'

consumer\_secret = 'UrlJVaRjpWmZrMjmWH7uADCBCzXwdoceIWVv2ftC34z85wpmMR'

access\_token = '1267471839929880577-ks8UGM7ydtxJ8xWJpHtv55tUCVGh7Z'

access\_token\_secret = 'QjKJ2FLe3R6gGtGD17lyBauOAqfptcnK75B5Sk8qZRSYN'

streamer = TweetStreamer(consumer\_key, consumer\_secret,

access\_token, access\_token\_secret)

#this code below uses to stream new update from your tracking, in this example we track for #covid19 when the new tweet on twitter is happen it will be sent to our script through on\_success(self,data) >> print(data[‘text’])

streamer.statuses.filter(track = '#covid19')

Combine TCP/IP Connection with Twitter Streaming

from twython import TwythonStreamer

import time

import socket

import json #now we keep data in json form (dict)

class TweetStreamer(TwythonStreamer):

def on\_success(self, data):

#establishing tcp/ip connection when new tweet is occurred

s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

localIP = '127.0.0.1'

localPort = 3001

#keep our data in json form

jsontext = json.dumps(data)

#encode our data to utf-8, so it can be sent through this protocol.

texttomatlab = jsontext.encode('utf-8')

#show what we sent in python ide

print('PYTHON SENT:/n ' + jsontext)

#connect with destination machine

s.connect((localIP, localPort))

#send data to destination machine

s.sendall(texttomatlab)

#backup information to python ide

Tweet\_list.append(jsontext)

def on\_error(self, status\_code, data):

print (status\_code)

self.disconnect()

consumer\_key = 'X5gCNKpqL6NUsLvBJjIYIm4Pz'

consumer\_secret = 'UrlJVaRjpWmZrMjmWH7uADCBCzXwdoceIWVv2ftC34z85wpmMR'

access\_token = '1267471839929880577-ks8UGM7ydtxJ8xWJpHtv55tUCVGh7Z'

access\_token\_secret = 'QjKJ2FLe3R6gGtGD17lyBauOAqfptcnK75B5Sk8qZRSYN'

#next

streamer = TweetStreamer(consumer\_key, consumer\_secret,access\_token, access\_token\_secret)

tweet\_list = []

hashtag = ‘asltest’ #input your hashtag here

streamer.statuses.filter(track = '#{hashtag}'.format(hashtag = hashtag))