# DropBox Clone : Server-Client System

**Project 2**

Niharika Vadlamudi
Sivani Pentapati

April 28 2020

## 1 Dropbox Clone : Problem Overview

In this project, we aim to create a file sharing protocol similar to Dropbox, with support for download and upload for files along with indexed searching.The generalised file sharing approaches use FTTP protocol for file transfer,but here we have implemented our own server-client from scratch thus imitating the former's functionality.We have the following functions to work with :

- **getIndex**: Details of all files in the corresponding directory.
- **fileHash**: MD5 Hash and checksum details of file or set of files in directory.
- **fileDownload**: Transfer of files from given directory to destination directory.
- **caching** : To ease the process of fileDownload,we maintain Cache folder.
- **serverLogs :** Log files that consists of history of commands exists here.

## 2 getIndex() functionality

The entire algorithm of getIndex , can be summarised as follows :

- 
- The client.py is free to choose between shortlist and longlist.
- At the server, we check if the folder exists and there is access for it
- Based on the number of inputs of client , we call the function with or without bonus part .
- Note : 'longlist' along with bonus will be valid ,only if the file is of '.txt' extension.
- If any of the arguments turn out to be invalid then immediately the client connection is closed and system exits.
- If the connection breaks in between , the program simply prints error message and quits.
- If no failure occurs while dealing with the file , we see that the client waits for new command with dollar symbol as prompt.

### 2.1 Code : getIndex

The following is the code snippet , that can be found in server.py :

```
1
2 def check(file,keyword):
3     with open(file) as f:
4         datafile = f.readlines()
5     for line in datafile:
6         if keyword in line:
7             return True
8     return False
```

```
9
10
11  def indexGet(flagType,startTime=0,endTime=0,dataPath='',bonus=None,keyWord='Programmer'):
12      #Result
13      res=[]
14      fileNames=[]
15
16      #Error Testing .
17      try :
18          assert(os.path.exists(dataPath))
19      except:
20          res=[]
21          res.append("Invalid Directory")
22          data=json.dumps(res,indent=4)
23          return(data)
24      if(bonus==None):
25          fileNames=os.listdir(dataPath)
26      else:
27          if(bonus!=None):
28              fileNames=[file for file in os.listdir(dataPath) if file.endswith(str(bonus))]
29
30      #Case 1 : Shortlist , bonus invariant.
31      if(flagType=="shortlist"):
32          os.chdir(dataPath)
33          for f in fileNames:
34              statObj=os.stat(str(f))
35              startTime=startTime.strip('[').strip(']')
36              endTime=endTime.strip('[').strip(']')
37
38              stmp=datetime.strptime(startTime,'%Y-%m-%d %H:%M:%S')
39              endtmp=datetime.strptime(endTime,'%Y-%m-%d %H:%M:%S')
40              curtmp=datetime.strptime(str(datetime.fromtimestamp(int(os.path.getmtime(f)))),'%Y-%m
    -%d %H:%M:%S')
41              dt_object=datetime.fromtimestamp(statObj.st_mtime)
42
43              if( (curtmp>stmp) and (curtmp<endtmp)) :
44                  res.append ( {  "Name: " : f[:f.find('.')],
45                                  "Size: " : statObj.st_size,
46                                  "Timestamp: " : dt_object.strftime('%Y-%m-%d %H:%M:%S'),
47                                  "File Format: ": f[f.find('.'):]
48                               } )
49
50      #Case 2 : Longlist , bonus invariant.
51      if(flagType=="longlist" and bonus==".txt"):
52          os.chdir(dataPath)
53          for f in fileNames:
54              isOpen=check(str(dataPath)+'/'+f,keyWord)
55              if(isOpen):
56                  statObj=os.stat(str(dataPath)+'/'+f)
57                  dt_object =datetime.fromtimestamp(statObj.st_mtime)
58                  res.append ( {  "Name: " : f[:f.find('.')],
59                                  "Size: " : statObj.st_size,
60                                  "Timestamp: " : dt_object.strftime('%Y-%m-%d %H:%M:%S'),
61                                  "File Format: ": f[f.find('.'):]
62                               } )
63
64      #Case 3: Longlist, NO bonus
65      #Case 2 : Longlist , bonus invariant.
66      if(flagType=="longlist" and bonus==None):
67          os.chdir(dataPath)
68          for f in fileNames:
69              statObj=os.stat(str(dataPath)+'/'+f)
70              dt_object =datetime.fromtimestamp(statObj.st_mtime)
71              res.append ( {  "Name: " : f[:f.find('.')],
72                              "Size: " : statObj.st_size,
73                              "Timestamp: " : dt_object.strftime('%Y-%m-%d %H:%M:%S'),
74                              "File Format: ": f[f.find('.'):]
75                           } )
76
77      data=json.dumps(res,indent=4)
78      return(data)
```

## 2.2 Results

Note : So, here we test out our function using a directory containing images,files,etc

- Takes only JPG Query : **indexGet shortlist 2000-03-19 10:00:00 2020-01-12 10:00:00**

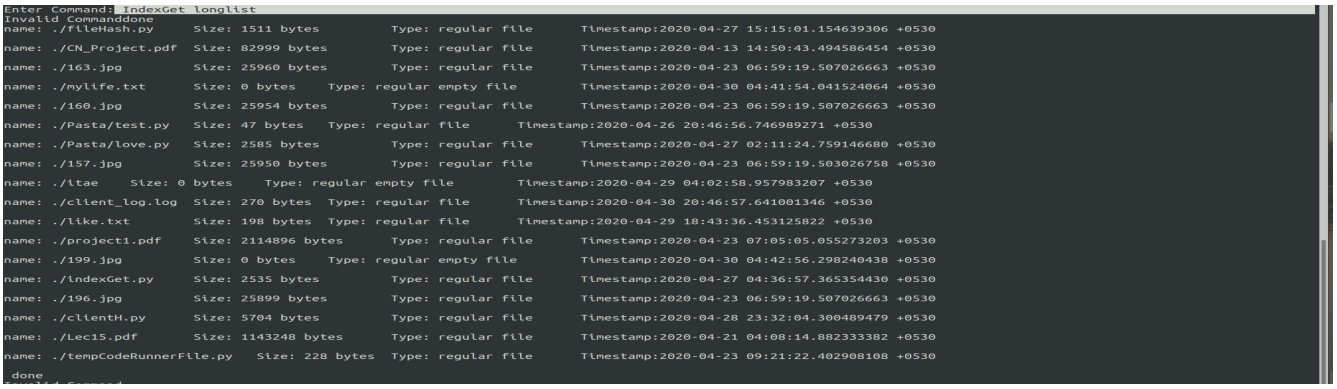


Figure 1: Query 1

- Query: **indexGet longlist**

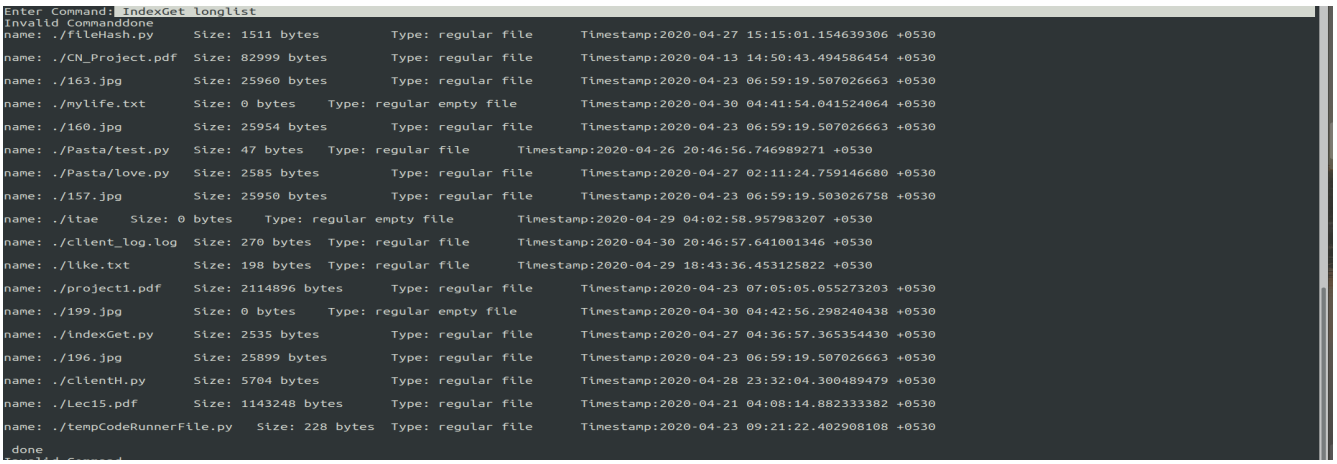


Figure 2: Query 2

## 2.3 Bonus: indexGet()

Note : The source code for these functions is the folder Bonus-IndexGet, please refer to README.txt file for running the codes and getting the outputs. Note : The keyword is always set to 'Programmer' , you can edit it out in the main code for different choice.

Note : So, here we test out our function using a directory containing images,files,etc

- Query : **indexGet longlist .txt**

```
niharika@niharika:~/Desktop/FML/Bonus-IndexGet$ python3 clientBonus.py
Enter PORT number :2003
$indexGet longlist .txt
Vachesindi...
RECIEVED ON CLIENT
[{'Name: ': 'like', 'Size: ': 198, 'Timestamp: ': '2020-04-29 18:43:36', 'File F
ormat: ': '.txt'}]
$
```

Figure 3: Query 9

- Query: **indexGet shortlist 2000-03-19 10:00:00 2020-01-12 10:00:00 .jpg**

```
$indexGet shortlist 2020-03-10 10:00:10 2030-01-01 10:00:00 .jpg
Vachesindi...
RECIEVED ON CLIENT
[{'Name: ': '163', 'Size: ': 25960, 'Timestamp: ': '2020-04-18 14:52:15', 'File
Format: ': '.jpg'}, {'Name: ': '160', 'Size: ': 25954, 'Timestamp: ': '2020-04-1
8 14:52:16', 'File Format: ': '.jpg'}, {'Name: ': '157', 'Size: ': 25950, 'Times
tamp: ': '2020-04-18 14:52:08', 'File Format: ': '.jpg'}, {'Name: ': '199', 'Siz
e: ': 0, 'Timestamp: ': '2020-04-30 04:42:56', 'File Format: ': '.jpg'}, {'Name:
 ': '196', 'Size: ': 0, 'Timestamp: ': '2020-04-30 22:29:24', 'File Format: ': '
.jpg'}]
$
```

Figure 4: Query 10

## 3 fileHash() functionality

The entire algorithm of getIndex , can be summarised as follows :

- 
- The client.py is free to choose between 'checkall' and 'verify'
- At the server, we check if the folder exists and there is access for it , and then we check if there is file access too.
- Based on the number of inputs of client , we call the function.
- 'checkall' gets all the files in the directory and produces the hash values , and 'verify' does the hash value for a single file .
- If any of the arguments turn out to be invalid then immediately the client connection is closed and system exits.
- If the connection breaks in between , the program simply prints error message and quits.
- If no failure occurs while dealing with the file , we see that the client waits for new command with dollar

symbol as prompt.

## 3.1 Code : fileHash()

The following is the code snippet , that can be found in server.py . The function hash() generates block-wise hash of each file part.

```
1
2  def hash(file,chunk=1024):
3      md5_hash=hashlib.md5()
4      with open(file,'rb') as f:
5          for byte_block in iter(lambda:f.read(chunk),b""):
6              md5_hash.update(byte_block)
7      return(md5_hash.hexdigest())
8
9  def fileHash(flagType,fileName=None,dirPath=None):
10
11     if(flagType=='verify'):
12         filePath=dirPath+'/'+str(fileName)
13
14     res=[]
15     fileNames=[]
16
17     if(dirPath!=None and flagType=="checkall"):
18         fileNames=glob.glob(dirPath+'/'+'*')
19
20     if(flagType=='verify'):
21         hashval=hash(filePath)
22         res.append({"File Hash : ":hashval,"Modified Timestamp : ":datetime.fromtimestamp(os.path.
       getmtime(filePath)).strftime("%m/%d/%Y, %H:%M:%S")})
23
24     if(flagType=='checkall'):
25         os.chdir(dirPath)
26         for file in fileNames:
27             if(os.path.isfile(file)):
28                 hashmap=hash(file)
29                 res.append({"File Name : ":basename(file),"File Hash : ":hashmap,"Modified
       Timestamp : ":datetime.fromtimestamp(os.path.getmtime(file)).strftime("%m/%d/%Y, %H:%M:%S")})
30
31     #Dump into json file .
32     data=json.dumps(res,indent=4)
33     return(data)
```

## 3.2 Results : fileHash()

Note : So, here we test out our function using a directory containing images,files,etc

- Query : **FileHash checkall**

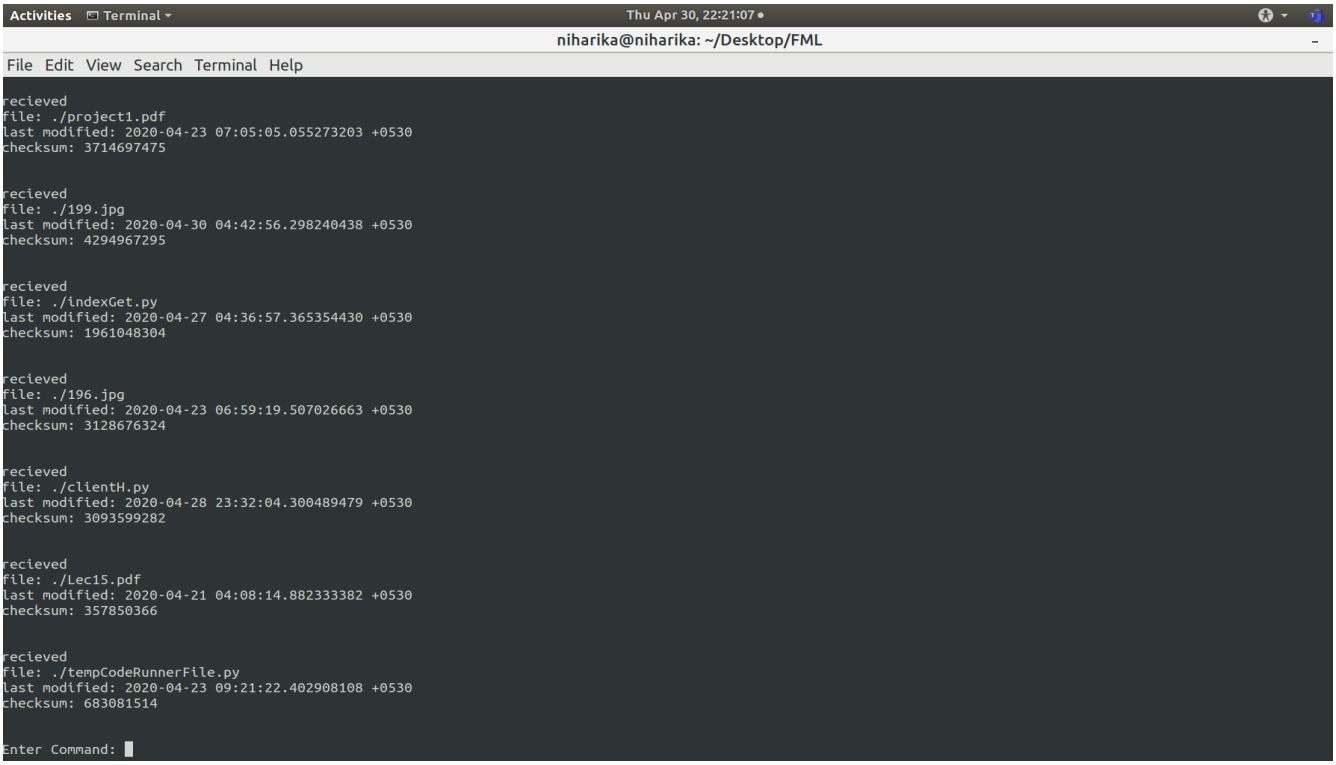


Figure 5: Query 3

- Query: **FileHash verify 199.jpg**

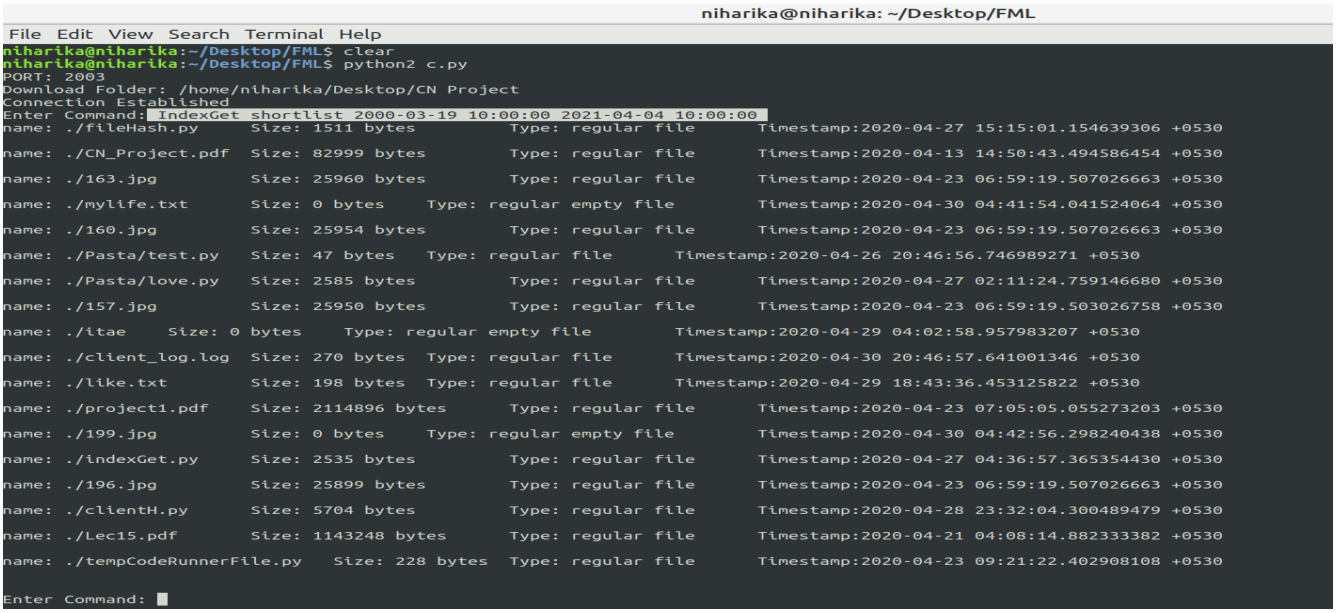


Figure 6: Query 4

# 4 fileDownload() functionality

Transmission Control Protocol is a standard that defines how to establish and maintain a network conversation via which application programs can exchange data. The socket is created using socket.socket and the socket type is specified as socket.SOCKSTREAM. The default protocol that is used is the TCP. The file is directly sent from server to client without going through any intermediate server. In TCP it is a 3-way Handshake Protocol : 1 Server is waiting for a connection 2 Client sends the arguments 3 Server sends the data 4 Client replies with a received signal.

In the case of UDP Protocol i.e User Datagram Protocol is a connection less protocol which is useful for communication which requires efficient communication and does not have a problem with packet loss. The Socket is created using socket.socket() and the socket type is specified as socket.SOCKDGRAM. UDP sends messages called datagrams and is considered best-effort mode of communication. UDP is efficient in communication as it does not require connection setup. i.e no handshaking dialogues.There are no handshaking dialogues which makes it suitable for time-sensitive applications.But UDP is not reliable as the datagrams dropped in the network may not be detected.The datagrams received and read by the reader may be out of order from the writes.

The entire working algorithm of fileDownload() function can be described as :

- The client.py is free to choose between TCP and UDP
- At the server, we check if the folder exists and there is access for it , and then we check if there is file access too.
- Based on the number of inputs of client , we call the function - we take the both filename,its parent folder on the server side , and on the client end we make sure to take the destination folder , where the downloaded file will be placed.
- The handshake procedure for TCP and UDP are different, and if the user opts for UDP we create a UDP socket whose UDP Port Number is fixed.
- If any of the arguments turn out to be invalid then immediately the client connection is closed and system exits.
- If the connection breaks in between , the program simply prints error message and quits.
- If no failure occurs while dealing with the file , we see that the client waits for new command with dollar symbol as prompt.

## 4.1 Code : fileDownload()

The following is the code snippet , that can be found in server.py :

```
1
2  def file_send(s, args):
3      inp = args.split(" ")
4      flag = inp[1]
5      filename = " ".join(inp[2:])
6      err = os.popen('ls "' + filename + '"').read().split('\n')[0]
7      if err == "":
8          s.send("No Such File or Directory")
9          return
10     if flag == "UDP":
11         print("req UDP")
12         s.send("recieved")
13         s.recv(1024)
14         ncs = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
15         nport = create_port(ncs)
16         s.send(str(nport))
17         data, addr = ncs.recvfrom(1024)
18         if data == "recieved":
19             try:
20                 f = open(filename, "rb")
21                 byte = f.read(1024)
22                 while byte:
23                     ncs.sendto(byte, addr)
24                     data, addr = ncs.recvfrom(1024)
```

```
                    if data != "recieved":
                        break
                    byte = f.read(1024)
                ncs.sendto("done", addr)
            except:
                print "Bad Connection Error"
                return

    elif flag == "TCP":
        s.send("recieved")
        s.recv(1024)
        try:
            f = open(filename, "rb")
            byte = f.read(1024)
            while byte:
                s.send(byte)
                if s.recv(1024) != "recieved":
                    break
                byte = f.read(1024)
            s.send("done")
        except:
            print "Bad Connection Error"
            return
    else:
        print "Wrong Arguments"
        return
    hash = os.popen('md5sum "' + filename + '"').read().split()[0]
    s.send(hash)
    cmd = "stat --printf 'name: %n \tSize: %s bytes\t Timestamp:%z\n' " + filename
    res = os.popen(cmd).read()
    if s.recv(1024) == 'sendme':
        s.send(res)
        print "Done"
```

## 4.2 Results : fileDownload()

Note : So, here we test out our function using a directory containing images,files,etc

- Query : **FileDownload TCP 196.jpg**

```
Enter Command: FileDownload TCP 196.jpg
FileDownload TCP 196.jpg
('196.jpg', 'TCP')
download requested
recieved

name: 196.jpg   Size: 0 bytes    Timestamp:2020-04-30 22:29:24.167686409 +0530

md5hash:  d41d8cd98f00b204e9800998ecf8427e
Successfulluy Downloaded
Enter Command:
```

Figure 7: Query 5

- Query: **FileDownload UDP Lec15.pdf**

```
FileDownload UDP Lec15.pdf
('Lec15.pdf', 'UDP')
download requested
recieved

name: Lec15.pdf         Size: 0 bytes    Timestamp:2020-04-30 22:32:49.215356018 +0530

md5hash:  d41d8cd98f00b204e9800998ecf8427e
Successfulluy Downloaded
Enter Command:
```

Figure 8: Query 6

# 5 caching() functionality

The entire concept of caching can be summarised as :

- When we call for fileDownload(), we will check the size of the cache folder , if the cache folder is filled , then we sort the existing files , with their modification/access time .
- Now, if the access time of the folder is quite high (old) , we shall remove the files using - os.unlink() functionality.
- Now,since we have space cleared out we try to fit in the new file here , if then also the space is insufficient we store it in the destination folder provided by the client.

Note: The size of the client folder is tunable ,as per the user input in the server.py settings part .

## 5.1 Code : caching()

The following is the code snippet , that can be found in server.py .

```
def get_cache_size():
  val = os.popen("du -sb cache/").read().split()[0]
  val = int(val) - 4096
  return val


def update_cache(name, exist_flag):
  global file_requests, file_id
  filename = name
  file_size = get_file_size(filename, True)
  if not exist_flag:
    remove_list = cache_policy(s, filename)
    cache_modify(remove_list)
  if filename not in file_requests:
    if not bool(file_requests):
      file_id = 1
      file_requests[filename] = [file_size]
      file_requests[filename].append(file_id)
    else:
      file_id += 1
      file_requests[filename] = [file_size]
      file_requests[filename].append(file_id)
    file_requests[filename].append(filename)
  else:
    file_id += 1
    file_requests[filename][1] = file_id
  return
```

## 5.2 Results: caching()

The code section isn't functioning fully, and runs into error hence, we have mentioned the code , but there are no results for it .We have mentioned the code,to emphasise that we have put our efforts into it .

# 6 Conclusion

We learnt a great deal about the functionality of sockets, and how we can connect to any computer given its Host IP Address. We learnt that sending messages across sockets , need serialisation which is why JSON,XML,etc formats are very handy . We got introduced to various Python libraries that expanded our horizon , and enabled us to write cleaner and readable codes.

# 7 References


- https://stackoverflow.com/questions/21120947/catching-keyboardinterrupt-in-python-during-program-shut
- https://pythontic.com/modules/socket/udp-client-server-example
- https://pymotw.com/2/socket/udp.html
- https://searchnetworking.techtarget.com/definition/TCP