

Encrypted Client-Server Chat

For my program I decided to incorporate a one time pad encryption option for extra credit. I don't know if that will count but it was fun!

To run my server program in **plaintext mode** run:

```
python3 server.py 20202
```

where "20202" is any free port.

To run my client program in **plaintext mode** run to connect to a server running on **localhost**:

```
python3 client.py 20202
```

where the port number is the same as the server.

There are several flag options to add functionality.

The **-i** "IPADDR" flag is an option for the client.py program that lets you pass in an ip address of the server you want to connect to. (I can only get this to work when the client and server are on the same network)

```
python3 client.py 20202 -i 10.197.138.235
```

?? QUESTION

I could actually only get this to work on osu's access.engr server. I had my roommate fork my program and connect to the vpn and it worked but I could not get it to work on our local wifi.

?? What would we need to do to make this work on our home network?

The **-k** (key) flag takes in a file containing a key for the one time pad and puts the program in **encrypted mode**.

```
python3 client.py 20202 -k test
```

In this example "test" is a file in the same directory that was created by running the **otp.py** like this:

```
python3 otp.py test
```

When you run **otp.py** it will generate a key, create and write to the filename you passed in, then it will enter a demo mode where you can type test and watch it encrypt and decrypt it.

Here is a demo of the client and server running side-by-side on localhost:

```
[jcz:~/Desktop/w21/CS372/chat] $ ls
README  __pycache__ client.py*  otp.py  server.py*  test
[jcz:~/Desktop/w21/CS372/chat] $ python3 client.py -h
usage: client.py [-h] [-l IPADDR] [-k KEY] port

positional arguments:
  port                Port number to listen on

optional arguments:
  -h, --help            show this help message and exit
  -l IPADDR, --ipaddr IPADDR
                        ip address of the host (defaults to localhost)
  -k KEY, --key KEY     path to the encryption key file
[jcz:~/Desktop/w21/CS372/chat] $ python3 client.py 20202
Client connecting...
Address : 10.197.138.235
Port : 20202
connected to 10.197.138.235
> testing 123
sending: "testing 123"
Waiting for message...
received message: "it worked!!"
> okay bye bye
sending: "okay bye bye"
Waiting for message...
chat ended.
[jcz:~/Desktop/w21/CS372/chat] $ python3 client.py 20202 -k test
Client connecting...
Address : 10.197.138.235
Port : 20202
connected to 10.197.138.235
> this will be encrypted
sending: "p8Yj:ls
      %4a.\vvclfWw"
Waiting for message...
received message: "successfully decrypted!"
> sweet bye bye
sending: "aQve8lb%4a8"
Waiting for message...
received message: "bye"
> /q
sending: "K"
Waiting for message...
chat ended.
[jcz:~/Desktop/w21/CS372/chat] $
```

```
[jcz:~/Desktop/w21/CS372/chat] $ ls
README  __pycache__ client.py*  otp.py  server.py*  test
[jcz:~/Desktop/w21/CS372/chat] $ python3 server.py -h
usage: server.py [-h] [-k KEY] port

positional arguments:
  port                Port number to listen on

optional arguments:
  -h, --help            show this help message and exit
  -k KEY, --key KEY     path to the ciphertext file
[jcz:~/Desktop/w21/CS372/chat] $ python3 server.py 20202
Server listing...
Address: 10.197.138.235
Port : 20202
Connected by ('10.197.138.235', 53366)
Waiting for message...
received message: testing 123
> it worked!!
sending it worked!!
Waiting for message...
received message : okay bye bye
> /q
sending /q
[jcz:~/Desktop/w21/CS372/chat] $ python3 server.py 20202 -k test
Server listing...
Address: 10.197.138.235
Port : 20202
Connected by ('10.197.138.235', 53369)
Waiting for message...
received message : this will be encrypted
> successfully decrypted!
hCj7-eu.8/vvC2fWwN
Waiting for message...
received message : sweet bye bye
> bye
sending 75u
Waiting for message...
chat ended.
[jcz:~/Desktop/w21/CS372/chat] $
```

Here is a demo of otp.py

```
[jcz:~/Desktop/w21/CS372/chat] $ python3 otp.py asdfg
type some text: asdf
coded: [l
      }q]
decoded: [asdf]
type some text: █
```

Useful links I used along the way:

Argparse: <https://docs.python.org/3/library/argparse.html#required>

Server-Client stuff: <https://realpython.com/python-sockets/>

Sockets: <https://docs.python.org/3.4/howto/sockets.html>

Files: <https://www.geeksforgeeks.org/reading-writing-text-files-python/>

Mapping a lambda function and producing a string in python:

<https://www.geeksforgeeks.org/python-map-function/>

<https://stackoverflow.com/a/50492545>

Find index of element in list:

<https://www.geeksforgeeks.org/python-list-index/>