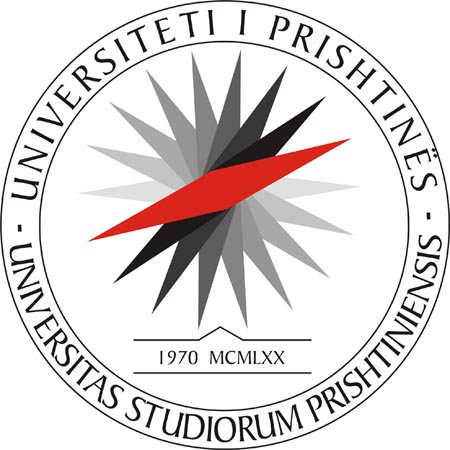
**UNIVERSITETI I PRISHTINËS**

Fakulteti i Inxhinierisë Elektrike dhe Kompjuterike



**Lënda:** Rrjetat Kompjuterike - 2019

**Projekti 1:** Dizajnimi Klient-Server

**Studenti:** **Asistenti:**

Jetmir Avdullahu Msc. Haxhi Lajqi

170714100102

Prill, 2019 – Prishtine

Permbajtja

[Permbajtja 1](#_Toc7378910)

[1. Veglat e perdorura, versionet e tyre dhe lista e metodave te implementuara 2](#_Toc7378911)

[1.1 Sistemet operative 2](#_Toc7378912)

[1.2 Veglat e perdorura dhe versionet e tyre 2](#_Toc7378913)

[1.3 Modulet e perdorura 2](#_Toc7378914)

[1.4 Lista e metodave te implementuara 2](#_Toc7378915)

[2. Hyrja 3](#_Toc7378916)

[2.1 Shpjegim I shkurter per projektin 3](#_Toc7378917)

[2.2 Skenari I klientit dhe serverit 3](#_Toc7378918)

[3. Pershkrim I kodit te serverit 4](#_Toc7378919)

[3.1 Pershkrimi I serverit TCP 4](#_Toc7378920)

[3.2 Pershkrimi i serverit UDP 6](#_Toc7378921)

[4. Pershkrimi I kodit te aplikacionit te clientit 8](#_Toc7378922)

[4.1 Logjika e aplikacionit TCP-Klienti 8](#_Toc7378923)

[4.2 Realizimi permes kodit 8](#_Toc7378924)

[4.3 Logjika e aplikacionit UDP-Klienti 11](#_Toc7378925)

[4.4 Realizimi permes kodit 11](#_Toc7378926)

[5. Pershkrimi I metodave 11](#_Toc7378927)

[5.1 IPADRESA 11](#_Toc7378928)

[5.2 NUMRIIPORTIT 11](#_Toc7378929)

[5.3 EMRIIKOMPJUTERIT 12](#_Toc7378930)

[5.4 BASHKETINGELLORE 12](#_Toc7378931)

[5.5 PRINTIMI 13](#_Toc7378932)

[5.6 KOHA 13](#_Toc7378933)

[5.7 LOJA 13](#_Toc7378934)

[5.8 FIBONACCI 14](#_Toc7378935)

[5.9 KONVERTIMI 15](#_Toc7378936)

[5.10 PASSWORDGEN 16](#_Toc7378937)

[5.11 KONTROLLOPORTIN 17](#_Toc7378938)

[6. Pershkrimi I FIEK Chat 18](#_Toc7378939)

[6.1 Logjika e aplikacionit 18](#_Toc7378940)

[6.2 Realizimi permes kodit 18](#_Toc7378941)

[7. Testimi 23](#_Toc7378942)

[7.1 Testimi ne lokalhost 23](#_Toc7378943)

[7.2 Testimi ne LAN 28](#_Toc7378944)

## 1. Veglat e perdorura, versionet e tyre dhe lista e metodave te implementuara

### 1.1 Sistemet operative

* Windows 10
* Ubuntu 18.04
* Mac OS Mojave 10.14.4

### 1.2 Veglat e perdorura dhe versionet e tyre

* Visual Studio Code 1.31
* Python 3

### 1.3 Modulet e perdorura

* socket
* sys
* threading
* socketserver
* datetime
* random
* tkinter
* textwrap

### 1.4 Lista e metodave te implementuara

* IPADRESA
* NUMRIIPORTIT
* EMRIIKOMPJUTERIT
* BASHKETINGELLORE
* FIBONACCI
* KOHA
* LOJA
* PRINTIMI
* KONVERTIMI
* PASSWORDGEN
* KONTROLLOPORTIN
* CHECK\_PORT
* VALIDATE\_IP

## 2. Hyrja

### 2.1 Shpjegim I shkurter per projektin

Projekti ka per qellim krijimi e nje lidhjeje klient-server ne te protokollet TCP dhe UDP me ane te socket-ave.

Komunikimi ndermjet klientit dhe serverit behen permes socket-ave te cilat mundesojne komunikim e tyre ne portin default 12000 ose nje port te vecante te cilin mund ta caktoje klienti.

FIEK Protokolli eshte nje protokoll I thjeshte I komunikimit I cili per baze ka fjalen kyqe ne fillim te kerkesa e cila percakton pergjigjen te cilen do ta kthej serveri. Jan dy versione te ketij protokolli:

1. FIEK-TCP
2. FIEK-UDP

Ky protokoll ju pergjigjet vetem ketyre kerkesave: IPADRESA, NUMRIIPORTIT, BASHKETINGELLORE, PRINTIMI, EMRIIKOMPJUTERIT, KOHA, LOJA, FIBONACCI, PASSWORDGEN, KONTROLLOPORTIN, KONVERTIMI te cilat dergohen nga klienti ne server dhe per te cilat serveri kthen nga nje pergjigje te vecante, (ne protokollin TCP serveri pasi pranon kerkesen, varesisht nga metoda, pret edhe ndonje kerkese te dyte nga klienti), klienti se bashku me kerkesen mund te dergoje edhe input varesisht nga kerkesa.

Gjithashtu eshte realizuar edhe nje aplikacion FIEK Chat I cili mundeson komunikim mes klienteve sipas protokollit TCP dhe ne te njejten kohe pranon kerkesat per metodat te cilat jane implementuar. Ne kete aplikacion kemi pjesen e serverit e cila eshte nje console application dhe pjesa e klientit e cila pjesa me e madhe eshte nje GUI aplikacion

Projekti startohet duke hyre ne njerin nga folderet TCP apo UDP me ane te ndonje console ose terminali dhe me ane te komandave python server.py ku starton serveri dhe ne terminalin tjeter me ane te python client.py starton klienti (ne disa console ose terminale duhet te perdoret komanda python3)

### 2.2 Skenari I klientit dhe serverit

FIEK-TCP:

Se pari vendoset nje lidhje(segment) ne mes te klientit dhe serverit permes socket-ave ne portin e caktuar(default 12000). Pastaj klienti permes command line e dergon kerkesen tek server, i cili neser kerkesa eshte valide I pergjigjet kerkeses specifike te klientit . Serveri eshte ne gjendje te lidhet me me shume kliente ne te njejten kohe dhe ti kthej pergjigje klienteve ne menyre paralele.

FIEK-UDP:

Ne versionin UDP te prokollit FIEK komunikimi realizohet vetem permes UDP datagram-ve, ku me c’rast nuk krijohet nje lidhje(segment) mes klientit dhe serverit por komunikimi fillon drejt. Klienti dergon kerkesen tek server permes nje UDP datagram. Pasi qe kerkesa te validohet ne server ajo poashtu kthehet tek klienti permes UDP datagram-it. Protokolli FIEK limiton qe klienti te dergon vetem nje kerkese per datagram.

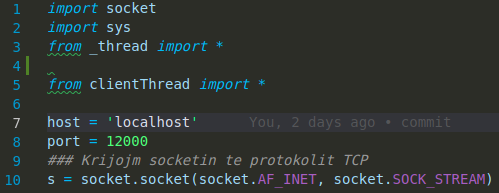
## 3. Pershkrim I kodit te serverit

### 3.1 Pershkrimi I serverit TCP

3.1.1 Logjika e aplikacionit

Serveri është console aplikacion që lidhet me klient përmes socketave dhe për secilen lidhje me klient krijon një thread të ri. Serveri dallon llojet e kërkesave që vijnë nga klienti dhe për secilin lloj ka përgjigje të veqantë. Serveri është aplikacion që gjendet në unazë të pafundme gjë që nuk e lejon të ndalet, pra gjat tërë kohës është aktiv, perpos rasteve kur ndalet nga perdoruesi me komanda te veqanta. Socketi mbete I hapur deri sa klienti te kerkoje te ckycet.

3.1.2 Realizimi permes kodit

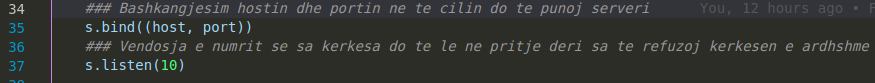


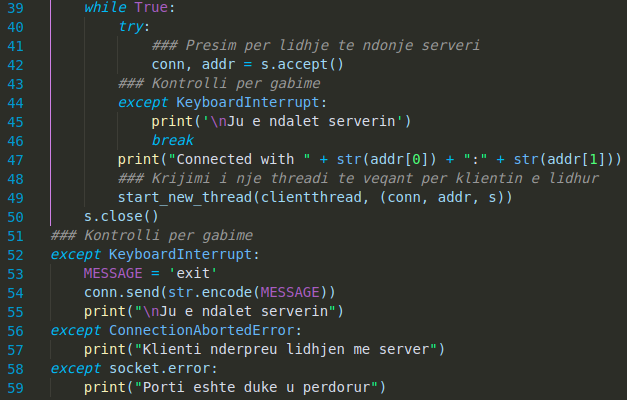
Se pari

Moduli socket eshte moduli kryesor qe perdoret pergjat gjith projektit, module te tjera te cilat do te perdoren jan edhe sys dhe thread I cili na lejon te startojm nje thread te ri per cdo klient.

Me ane te variablave host dhe port kemi caktuar se ne cilen ip dhe port do te punoj serveri, pra ne cilen addres do te mund te kycen klientet.

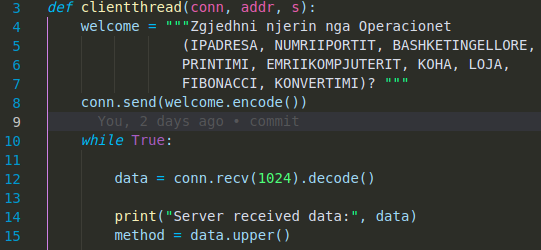
Me pas kemi inicializuar variablen **s** me te cilen krijojme socket-in me ane te formatit IPv4 dhe protokollit TCP.

Ketu pastaj I kemi dhene hostin dhe portin ne te cilin do te punoj serveri. Dhe me ane te metodes **listen** I kemi percaktuar se sa kerkesa mund te lejoj ne pritje deri sa te refuzoj kerkesen e ardhshme

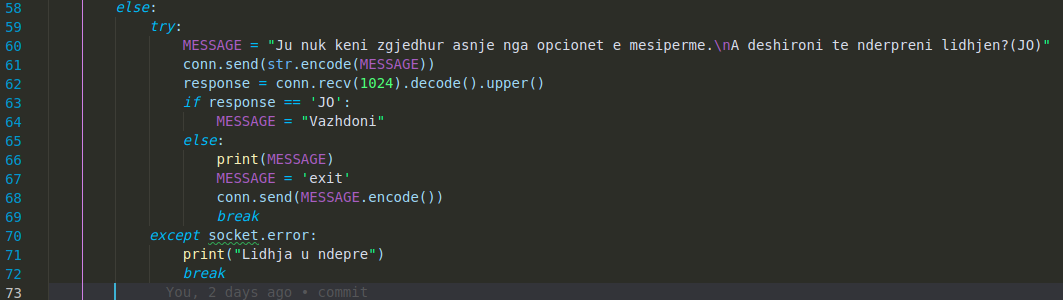


Me ane te unazes while kemi krijuar nje loop te pafundme me te cilen e mbajme serverin hapur tere kohen. Brenda unazes presim tere kohen per ndonje klient eventual dhe ne rastin qe fitohet lidhja krijohet threadi I ri per ate klient me ane te komandes start\_new\_thread e cila theret metoden clientthread e cila per lexim me te lehte te kodit eshte ndare ne nje file tjeter me emrin clientThread.py.

Ne rast se loop-a prishet atehere mbyllet socketi. Ne rast te ndonje gabimi jane vendosur disa prites per gabimet qe mund te ndodhin dhe japin nga nje log mesazh mbi arsyen e tyre.



Ne metoden clientthread inicializojm nje variabel qe do e kete tekstin me opcione I cili do I paraqitet klientit. Dhe me ane te komandes ne rreshtin 8 ia dergojm dhe me pas fillojm loop- en e pafundme deri ne momentin kur klienti shkeput lidhjen ose serveri ndalet. Ne reshtin 12 presim per kerkesen nga klienti dhe ate kerkese ne kodin me poshte e krahasojme me ndonjeren nga metodat qe I kemi ne server.



Nese kerkesa pershatet me ndonjeren nga ato qe jan ne server atehere dergohet pergjigja perkatese, perndryshe dergohet mesazhi qe nuk perputhen me asnje dhe dergohet pyetja se a deshiron te vazhdoj lidhje apo jo.

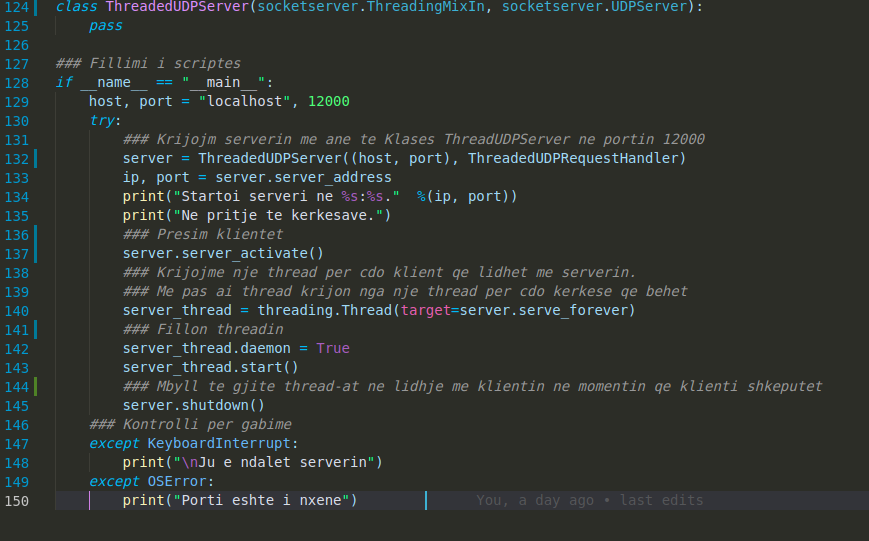
### 3.2 Pershkrimi i serverit UDP

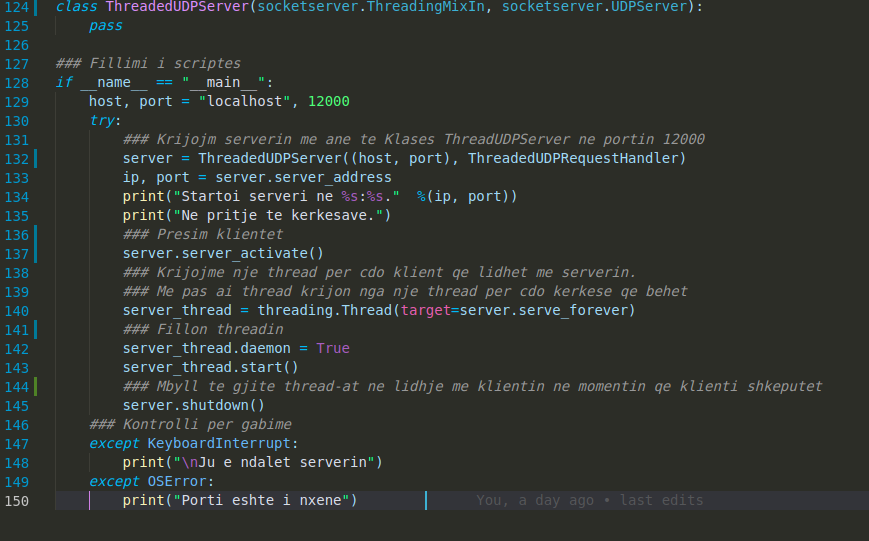
3.1.1 Logjika e aplikacionit

Serveri është console aplikacion që lidhet me klient përmes socketave dhe komunikom me datagram-a. Serveri dallon llojet e kërkesave që vijnë nga klienti dhe për secilin lloj ka përgjigje të veqantë. Serveri është aplikacion që gjendet në unazë të pafundme gjë që nuk e lejon të ndalet, pra gjat tërë kohës është aktiv. Për një datagram serveri mund të dërgoj vetëm një datagram.

3.1.2 Realizimi permes kodit

UDP serveri eshte realizuar ne dy forma. Ne njeren forme eshte perdorur e njejta logjik sikurse tek TCP sercveri ndersa ne tjetren eshte perdorur nje librari tjeter e gatshme e cila eshte socketserver. E cila e kryen pjesen e serverit vetvetiu. Ketu do te dokumentohet vetem UDP serveri I realizuar permes metodes socketserver e cila ne projekt do jet ne folderin UDPv2

Ketu kemi krijuar klasen ThreadedUDPServer e cila do te jete pjesa e pare qe ekzekutohet. Krijojme serverin ne variablen server ku japim hostin dhe portin ne te cilin punon dhe gjithashtu klasen e dyte e cila do te miret me krijim e threadeve per cdo klient dhe kerkese te tyre. Pjesa tjeter e kodit verehet ne foto dhe eshte e komentuar.

Ne rast te krijimit te ndonje lidhjeje me socketin scripta vazhdon ne klasen ThreadedUDPServer e cila e pret klientin ia dergon mesazhin me metodat qe I posedone serveri dhe pret per kerkese. Me pas kerkesa procesohet deh kthehet pergjigja per kerkesen perkatese ose instruksione per perdorim.

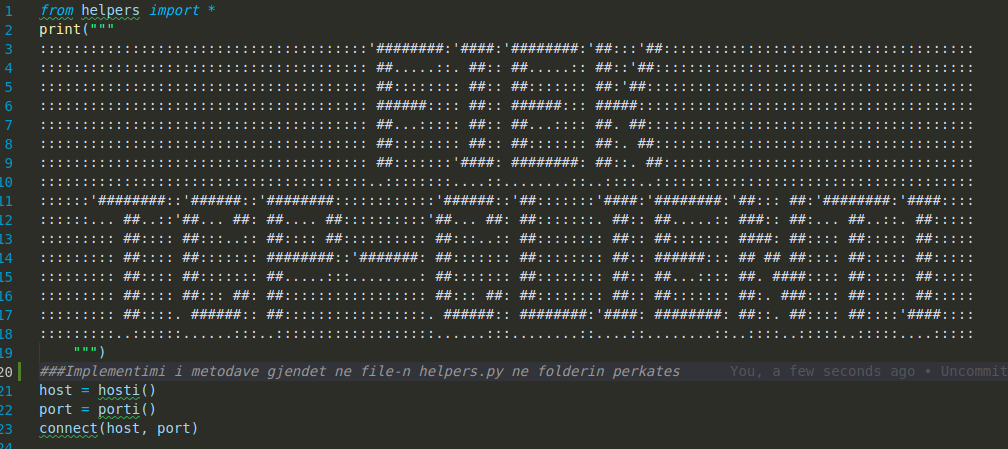
## 4. Pershkrimi I kodit te aplikacionit te clientit

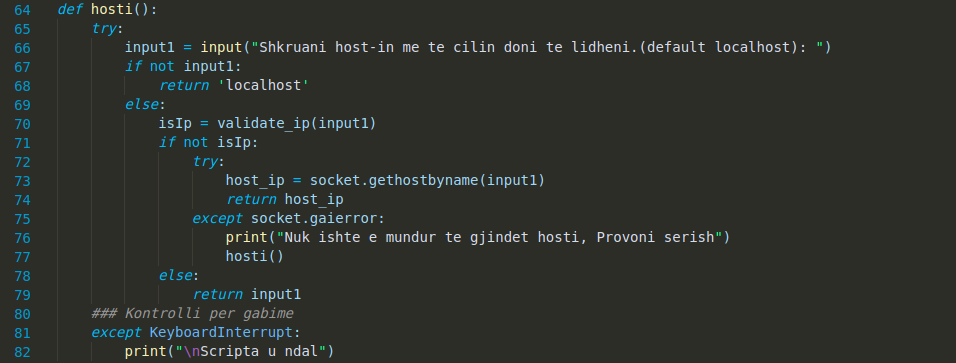
### 4.1 Logjika e aplikacionit TCP-Klienti

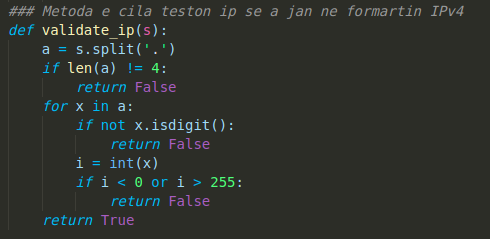
Klienti eshte nje command line e cila pas startimit pyet klientin se a deshiron te konnektohet ne serverin dhe portin default (127.0.0.1:12000) apo ne ndonje server tjeter te cilen e bene duke dhene hostin dhe portin ne momentin kur kerkohet.

Perndryshe klienti konnektohet ne serverin default me c’rast klientit I mundesohet dergimi I kommandave tek serveri.

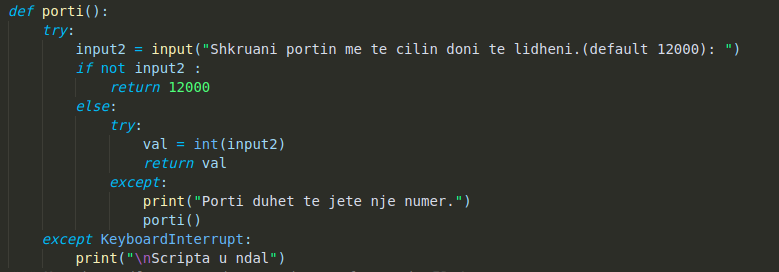
### 4.2 Realizimi permes kodit

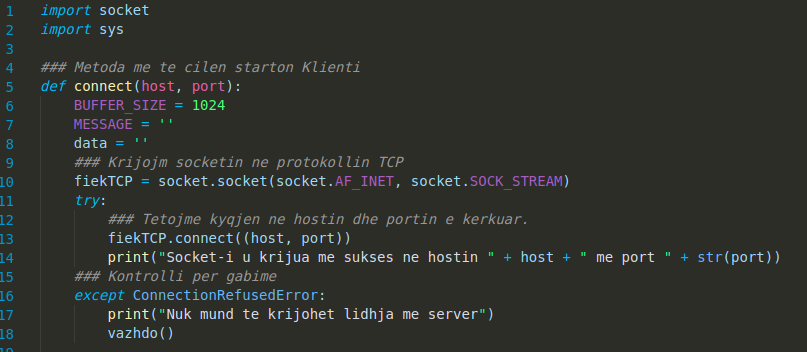
Ne file-n kryesore te klientit TCP kemi vetem header-in dhe thirjen per metodadt hosti porti dhe connnect. Pjesa e klientit eshte bere ne modulin helpers per arsye te leximit sa me te lehte te kodit.

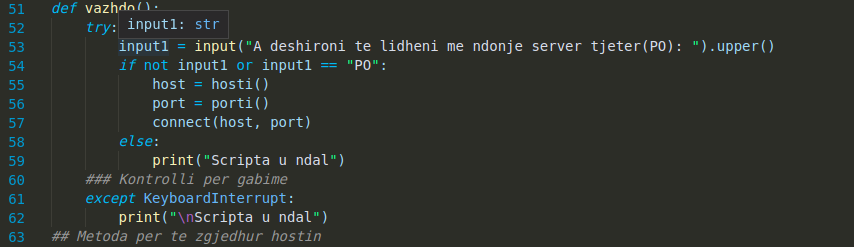
Ne metoden hosti kerkohet hosti se ku do te konektohet klienti. Pasi te mer inputin nga perdoruesi e teston se a eshte ne format te ip apo ne forme te url. Kete e ben me ane te metodes validate\_ip, e cila eshte e implementuar me poshte. Nese nuk eshte ip atehere me ane te metodes gethostbyname te klases socket.

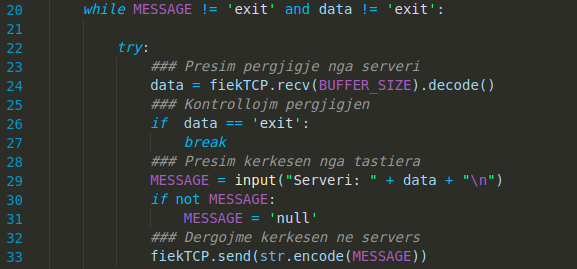


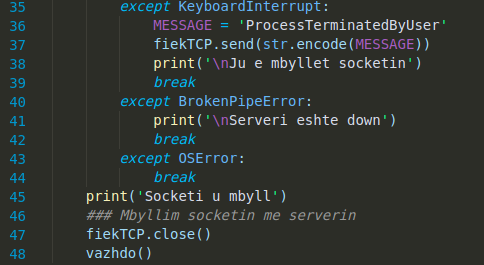
Metoda per validimin e IP.

Metoda e cila kerkon portin nga klienti dhe ben nje validim se a eshte numer inputi.

Me pas kemi metoden connnect e cila kryen punen e lidhjes me server. Kemi krijuar nje socket te tipit TCP me IPv4 dhe me pas kemi tentuar te lidhemi me server. Ne rast suksesi kemi vazhduar ne te kunderten kemi paraqitur nje mesazh dhe kemi thirur metoden vazhdo e cila pyet klientin se a do te vazhdoje te lidhet me ndonje host tjeter apo do te perfundoje.

Implementimi I metodes vazhdo.

Me ane te nje while loop-e ku eshte e sakt deri ne momentin kur mer ose nga klienti os nga server nje input me tekst exit. Me crast e perfundon lidhjen me server. Ne te kunderten pret nje kerkese nga serveri dhe me pas mund te dergoj ndonje kerkese ne server me te cilen serveri mund ti kthej pergjigje nese e ka zgjidhjen e saj.



Me pas jan implementuar disa kontrolle per gabime eventuale me te cilat e ndalin serverin dhe japin nje arsye se pse eshte ndalur. Mbyllet socketi dhe thirret metoda vazhdo.

### 4.3 Logjika e aplikacionit UDP-Klienti

Klienti eshte nje command line e cila pas startimit pyet klientin se a deshiron te konnektohet ne serverin dhe portin default (127.0.0.1:12000) apo ne ndonje server tjeter te cilen e bene duke dhene hostin dhe portin ne momentin kur kerkohet.

Perndryshe klienti konnektohet ne serverin default me c’rast klientit I mundesohet dergimi I kommandave tek serveri.

### 4.4 Realizimi permes kodit

Realizimi I aplikacionit UDP-Klienti eshte I njejte sikur ai I TCP-Klienti vetem se ketu klienti mund te beje vetem nje kerkese dhe me pas lidhja perfundon.

## 5. Pershkrimi I metodave

Ne metodat ku kerkohet nje hyrje P.sh fibonacci dhe me pas numri ne protokollin TCP serveri kerkon ne fillim metoden e me pas edhe inputa te tjere qe I nevoiten per metoden, nderssa ne UDP kerkon nje cift metode dhe kerkesa.

### 5.1 IPADRESA

Aksioni:

* Përcakton dhe kthen IP adresën të klientit në formë dhjetore me pike në një tekst mesazh p.sh. (10.10.7.251)

Pergjigja:

* Një mesazh jo me i gjate se 128 karaktere qe përmban IP adresën e klientit.
* Mesazhi nuk guxon te përmbaj NULL karakterin

Realizimi permes kodit:



Shpjegimi I kodit:

* Permes variables addr e cila mirret nga socket-i i klientit mirret IP

### 5.2 NUMRIIPORTIT

Aksioni:

* Përcakton dhe kthen portin e klientit. (Ky duhet te jete porti I klientit dhe jo porti I serverit.)Pergjigja:

Pergjigja:

* Një mesazh jo me i gjate se 128 karaktere qe përmban portin e klientit.
* Mesazhi nuk guxon te përmbaj NULL karakterin



Realizimi permes kodit:

* Permes variables addr e cila mirret nga socket-i i klientit mirret IP

### 5.3 EMRIIKOMPJUTERIT

Aksioni:

* Kërkon emrin e kompjuterit/hostit dhe e kthen atë. Nëse emri i hostit nuk mund te përcaktohet (gjendet) atëherë duhet te kthehet një mesazh dhe te tregohet se emri i hostit nuk mund te gjendet

Pergjigja:

* Një mesazh jo me i gjate se 128 karaktere qe përmban emrin e hostit ose te tregohet se emri i hostit nuk mund te përcaktohet. Mesazhi nuk guxon te përmbaj NULL karakterin



Realizimi permes kodit:

* Si parameter mer instancen e socket-it te krijuar dhe me ane te metodes gethostname() jep emrin e hostit

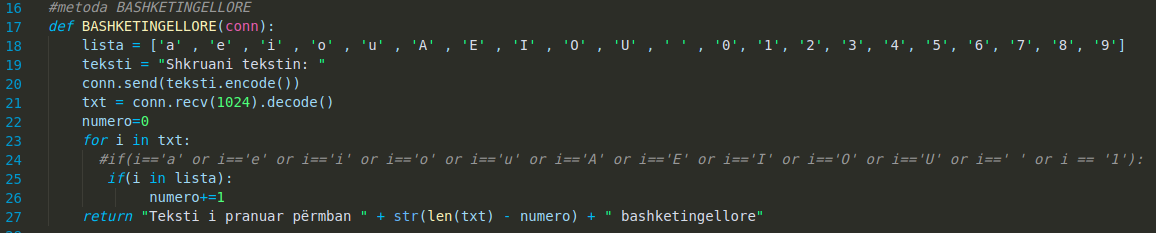
### 5.4 BASHKETINGELLORE

Aksioni:

* Gjen numrin e bashkëtingëlloreve ne tekst dhe kthen përgjigjen.

Pergjigja:

* Një mesazh jo me i gjate se 128 karaktere.
* Mesazhi nuk guxon te përmbaj NULL karakterin



Realizimi permes kodit:

Eshte inicializuar lista me zanore, hapsire, dhe numra dhe eshte kontrolluar cdo karakter se a ben pjes ne ate liste dhe me pas I ka numeruar karakteret qe jan ne liste dhe ia ka zbritur gjatesise totale.

### 5.5 PRINTIMI

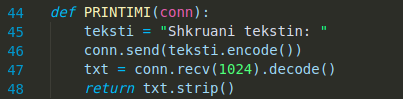
Aksioni:

* Kthen fjalinë e shtypur ne tekst

Pergjigja:

* Tekst

Realizimi permes kodit:



Eshte pranuar teksti dhe me pas eshte kthyer po I njejti vetem se pa hapira ne fillim dhe fund, e cila eshte aritur me metoden strip.

### 5.6 KOHA

Aksioni:

* Përcakton kohen aktuale ne server dhe e dërgon atë tek klienti si format te lexueshme për njerëzit

Pergjigja:

* Një mesazh jo me i gjate se 128 karaktere.
* Mesazhi nuk guxon te përmbaj NULL karakterin

Realizimi permes kodit:



Eshte perdorur libraria datetime.

### 5.7 LOJA

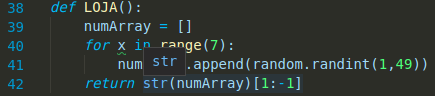
Aksioni:

* Kthen 7 numra nga rangu [1,49].

Pergjigja:

* Tekst

Realizimi permes kodit:



Eshte perdorur libraria random

### 5.8 FIBONACCI

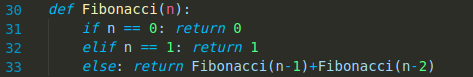
Aksioni:

* Gjënë numrin FIBONACCI si rezultat i parametrit të dhënë hyrës.

Pergjigja:

* Numër i plotë

Realizimi permes kodit:



Eshte perdorur nje funksion rekursiv.

### 

### 5.9 KONVERTIMI

Aksioni:

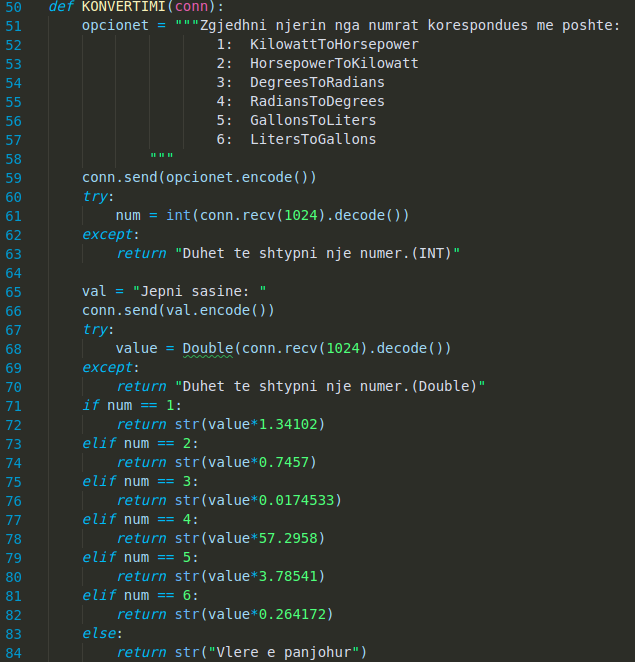
Kthen si rezultat konvertimin e opcioneve varësisht opcionit të zgjedhur.

* Lista e parametrave opcioni janë:
* KilowattToHorsepower
* HorsepowerToKilowatt
* DegreesToRadians
* RadiansToDegrees
* GallonsToLiters
* LitersToGallons

Pergjigja:

* Numër float

Realizimi permes kodit:



Eshte pranuar kerkesa dhe eshte derguar pergjigja varesisht ng aajo qe eshte kerkuar.

### 

### 5.10 PASSWORDGEN

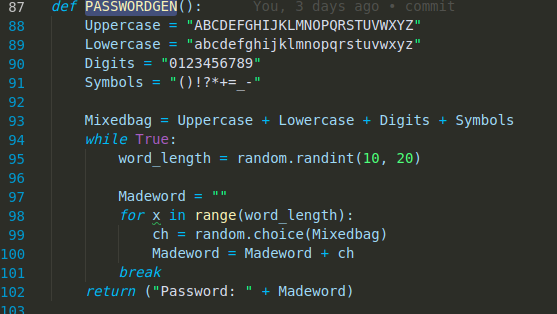
Aksioni:

* Gjeneron nje password te fuqishem

Pergjigja:

* Tekst

Realizimi permes kodit:



Jan inicializuar 4 stringje ku permbajne shkronja te medha, te vegjel, numra dhe simbole speciale dhe me pas me ane te llupes kemi gjeneruar nje gjatesi te rendomt nga 10 deri 20 karaktere dhe me ane te nje for loop-e tjeter kemi zgjedhur nga nje karakter te rastesishem.

### 5.11 KONTROLLOPORTIN

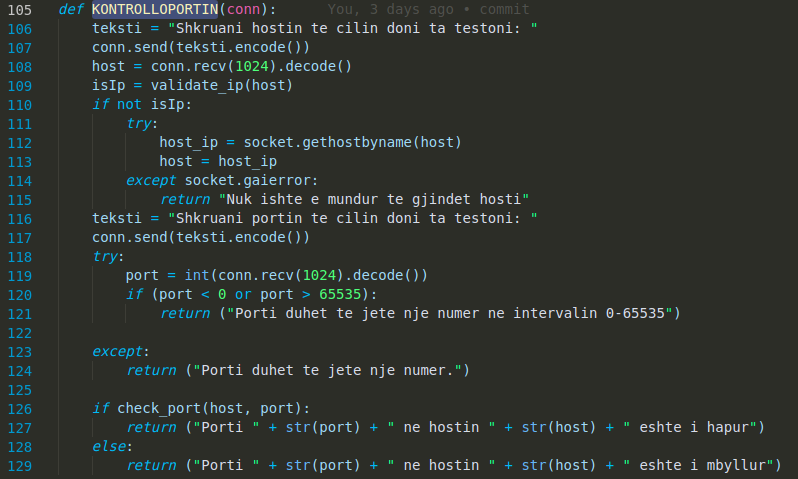
Aksioni:

* Kontrollon hostin se a eshte I hapur porti te cilin e kerkon klienti

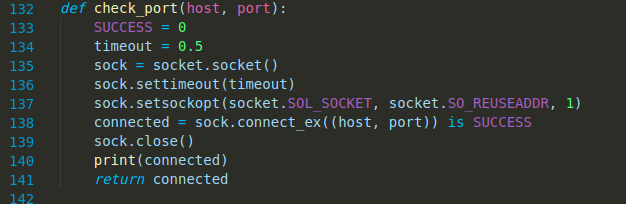
Pergjigja:

* Tekst

Realizimi permes kodit:



Eshte kerkuar hosti dhe gjetur ip e tij dhe me pas edhe porti (njejte sikur te metoda hosti dhe porti per server) dhe me pas serveri eshte munduar te krijoj nje lidhje me hostin e kerkuar ne portin e kerkuar. Nese ka aritur ka kthyer rezultat pozitiv ne te kundert negativ. Kete e ka bere me ane te metodes check port



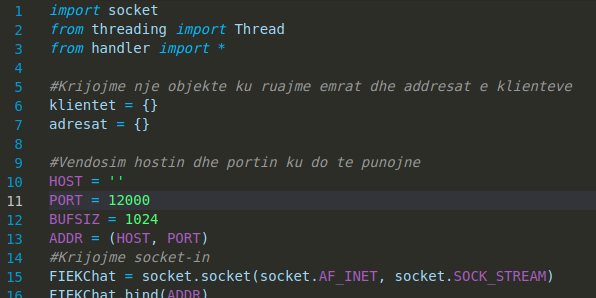
## 6. Pershkrimi I FIEK Chat

### 6.1 Logjika e aplikacionit

Applikacioni do te mundesoje qe disa kliente te lidhen ne nje server dhe te komunikojne me njeri tjetrin dhe ne te njejten kohe te therasin metodat te cilat serveri I ka te implementuara.

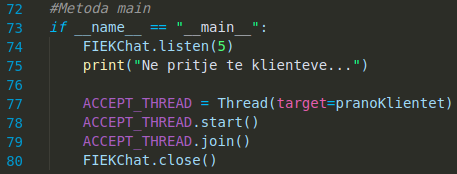
### 6.2 Realizimi permes kodit

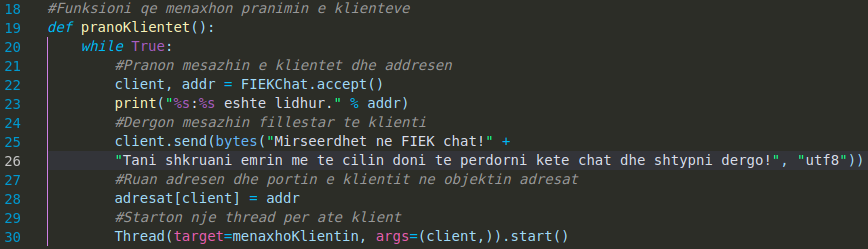
6.2.1 Realizimi I serverit

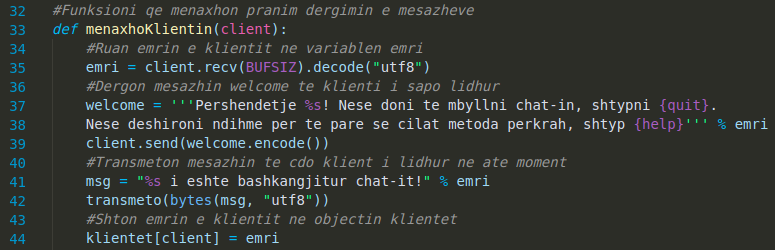
Eshte bere importi I moduleve te nevojshme, socket, threading dhe handler i cili eshte modul I krijuar per te menaxhuar kerkesat e klientit.

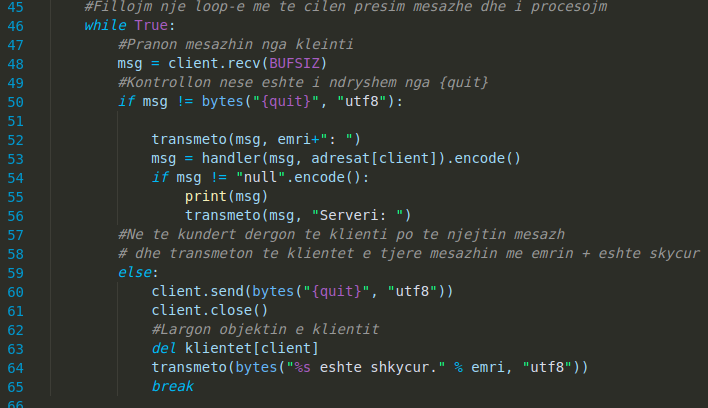
Kemi krijuar dy objekte klientet dhe adresat ku do te ruhen klientet dhe adresat

Kemi krijuar variabla per hostin dhe portin ne te cilin do te punoje serveri dhe me pas kemi krijuar socketin dhe ia kemi bashkangjitur adresen.

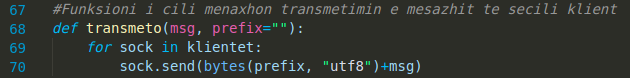
Ne metoden main kemi vendosur vleren se sa kliente do te presim dhe me pas kemi krijuar nga nje thread per cdo klient qe konektohet. Threadi do te theras funksioni pranoKlientet.

Ne kete metode menaxhojm pranimin e cdo klienti ne chat ku ne rreshtin 22 marim addresen dhe lidhjen me klientin. Me pas I dergohet nje mesazh mirseardhes dhe adresa e tij shtohet ne objektin e krijuar per adresa. Me pas krijojm nje thread te vecant per te e cila thheret metoden menaxhoKlientin qe per argumente mer lidhjen me clientin.

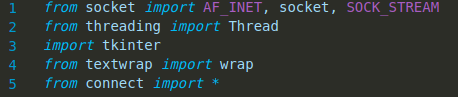
Te kjo metod presim qe klienti te dergoj nje emer per identifikim dhe me pas I dergojm nje mesazh qe eshte kycur me sukses dhe disa komanda ndihmese qe mund ti perdor. Dhe transmeton nje mesazh te te gjithe te pranishmit ne chat se klienti eshte bashkuar chatit. Dhe me pas ruajme emrin e klientit ne objektin e krijuar per emra.

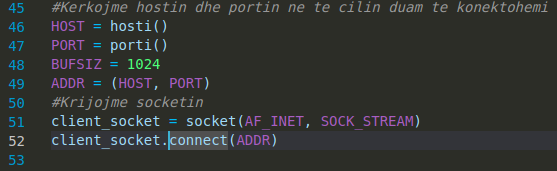


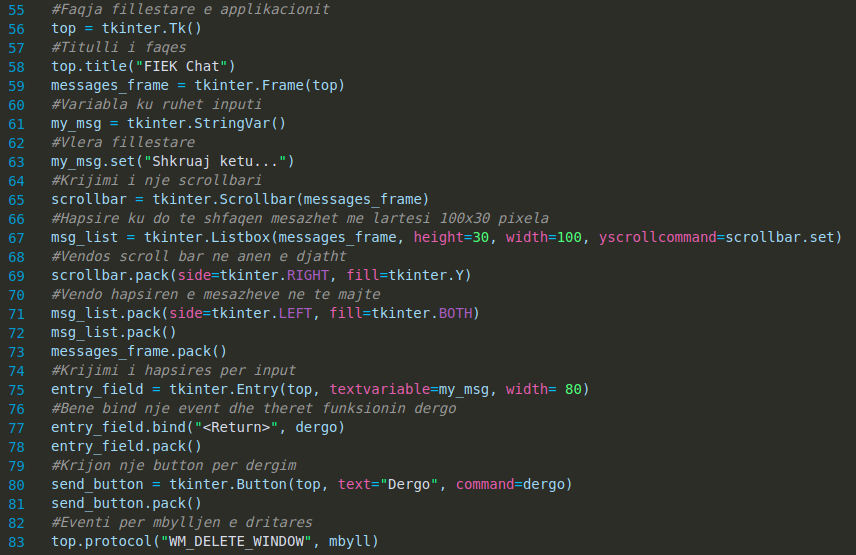
Me pas eshte krijuar nje loop e pafundme e cila pret per mesazhet qe do I dergoje klienti dhe me pas I shikon a perputhen me ndonje nga metodat qe I kemi zhvilluar. Nese po transmetohet kerkesa e klientit dhe pergjigja nga serveri. Ne te kundert dergohet vetem mesazhi I klientit. Nese klienti kerkon shkycjen nga serveri atehere transmetohet mesazhi te klientet e tjere se eshte shkycur dhe dergohet komanda te klienti qe te mbyllet lidhja.

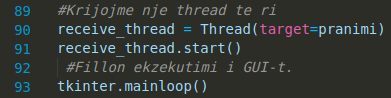
 Funksion transmeto I cili bene dergimin e mesazhit te cdo klient qe eshte I kycur ne server.

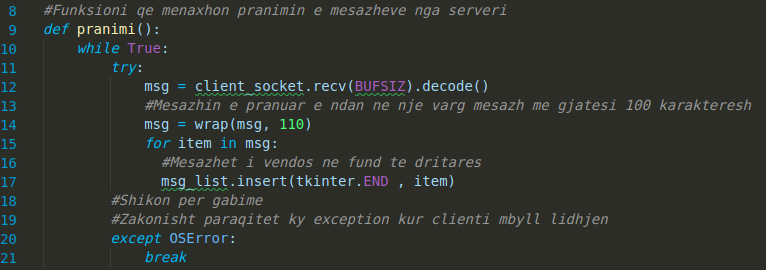
5.2.2 Realizimi I klietit

Si fillim kemi bere importet e nevojshme per te krijuar socket-in thread-at, tkinter I cili do te perdoret per te krijuar GUI-n dhe textwrap I cili do perdoret per te shfaqur mesazhet ne nje gjatesi te caktuar. Dhe skripta connect qe eshte e veteshkruar e cila do permbaj metodat per tu kycur ne server.

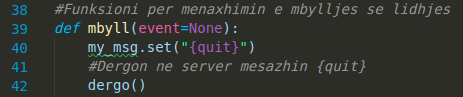
Me ane te metodave hosti dhe porti kerkojme hostin dhe portin se ku do te lidhet klienti. Me pas krijojme socketin ku dhe formohet lidhja.

Pasi te formohet lidhja hapet GUI aplikacioni I cili eshte realizuar me ane te libraris se gatshme tkinter. Jane dhene instruksionet e nevojshme dhe jan krijuar dritaret e nevojshme per shfaqjen dhe dergimin e mesazheve. Ne rreshtin 67 eshte variabla qe permbane hapsiren se ku shfaqen mesazhet. Ne 75 eshte krijuar hapsira per te shkruar mesazhin dhe eshte bere bind eventi return qe theret metoden dergo. Ne 83 eshte krijuar edhe eventi qe nese klikohet ne button x te mbyllet dritarja, e cila theret metoden mbyll.

Krijohet nje thread I ri I cili theret metoden pranimi dhe me pas fillon applikacioni GUI.

Te metoda pranimi menaxhohet pranimi I mesazheve dhe vendosen ne hapsiren per mesazhe. Pasi qe tkinter.Listbox nuk perkrah nje rresht te ri eshte perdorur libraria textwrap e cila e ndan stringun e pranuar ne nje array stringjesh dhe me pas I shton ne dritaren e percaktuar.

Me pas kemi metoden dergo e cila eshte bere bind me ane te eventit return dhe ne momentin kur klikohet buttoni dergo. Kjo metod mer gjithcka cka eshte shkruar ne tektbox-in per input dhe e dergon neserver.

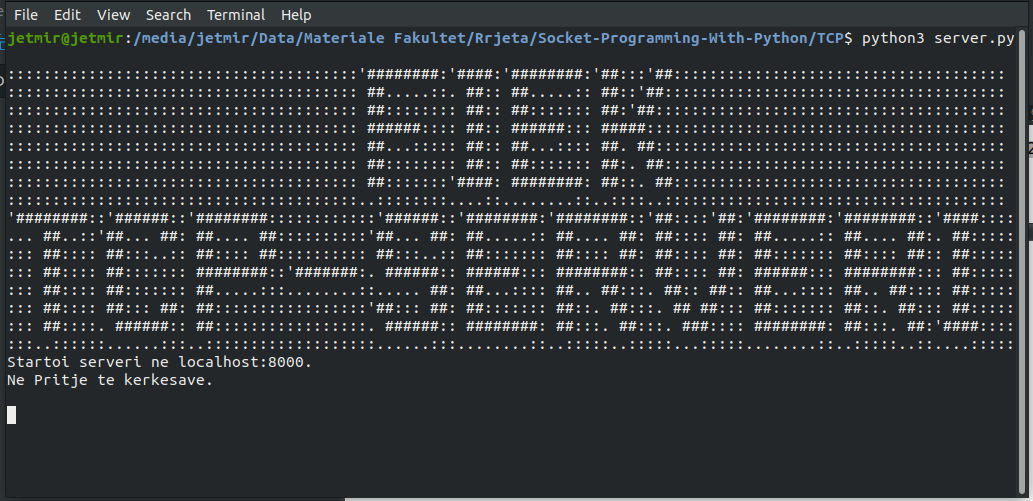
Ne fund kemi edhe metoden mbyll e cila thirret kur klikohet buttoni x e clia e mbyll dritaren e klientit.

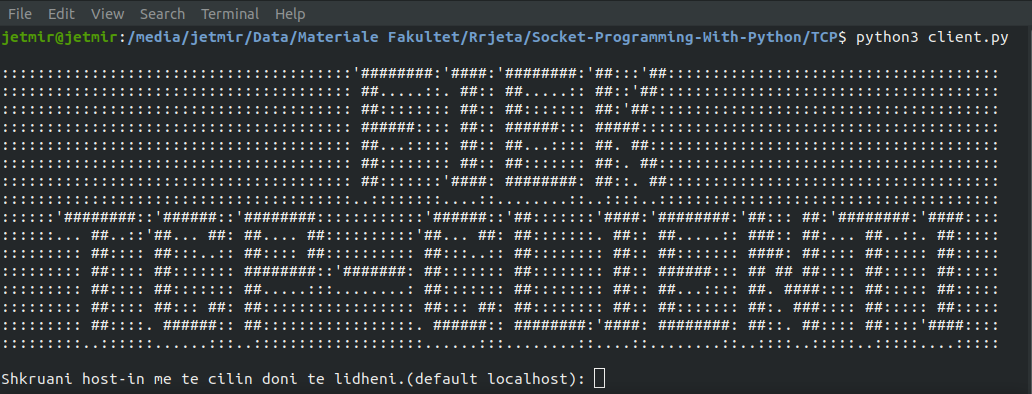
## 7. Testimi

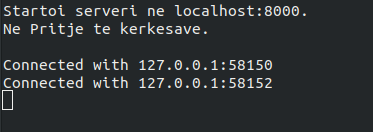
### 7.1 Testimi ne lokalhost

7.1.1 Testimi i protokollit TCP

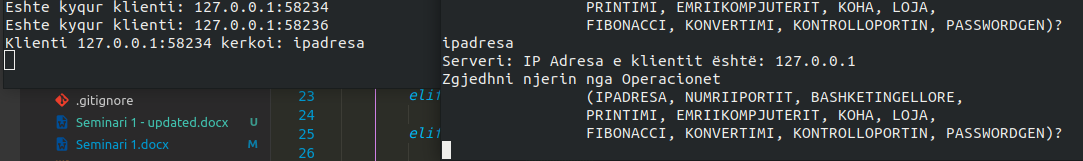
Startimi I serverit:

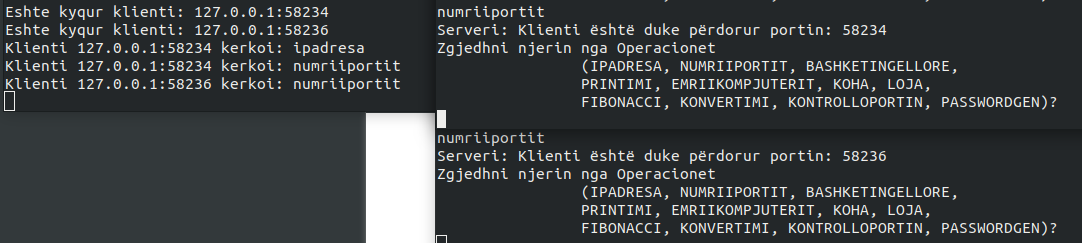
Startimi I klientit:

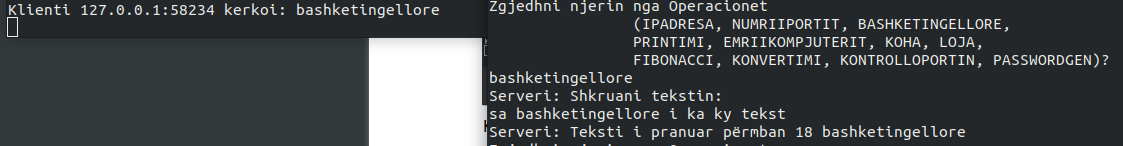
Kycja e dy klienteve njekohesisht:

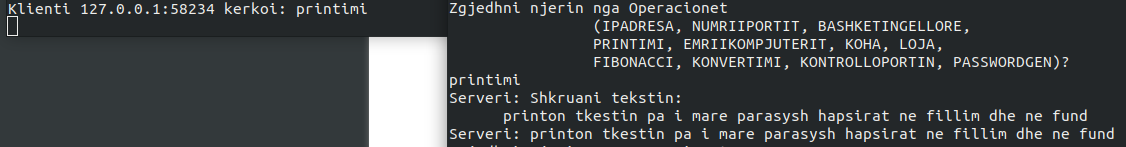


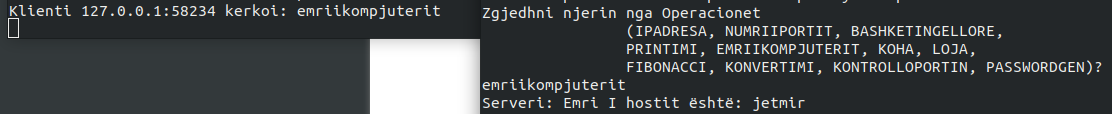
Kerkesa ipadresa:

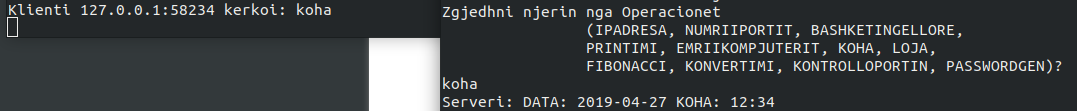
Kerkesa numriiportit:

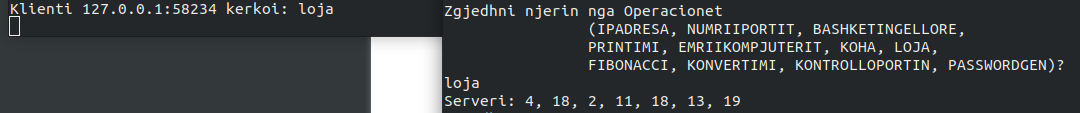
kerkesa bashketingellore:

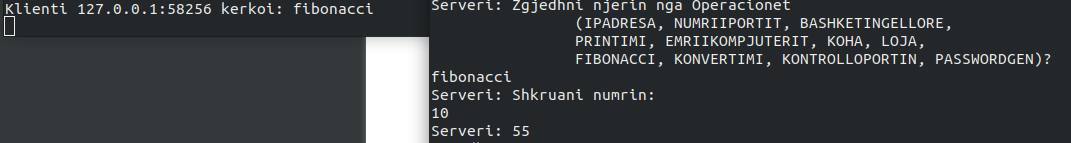
Kerkesa printimi:

Kerkesa emriikompjuterit:

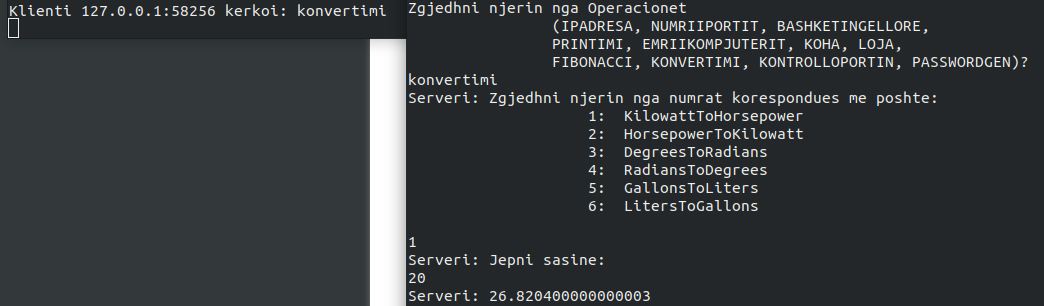
Kerkesa koha:

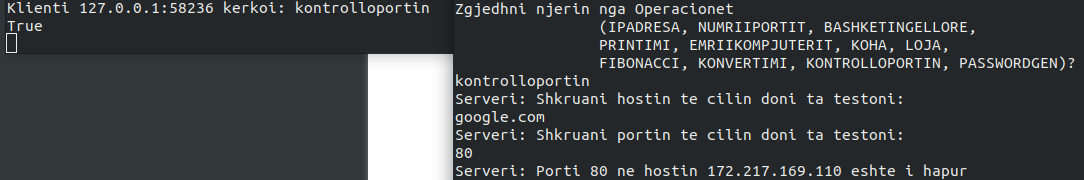
Kerkesa loja:

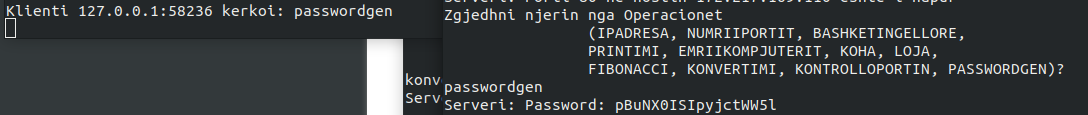
Kerkesa fibonacci:

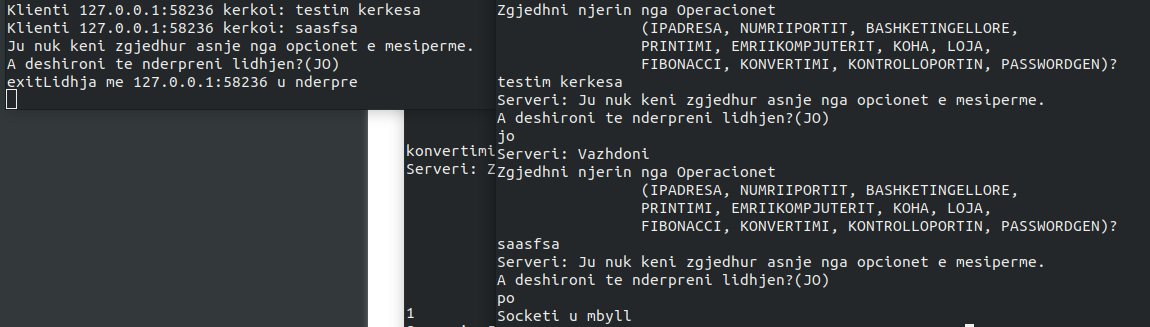


Kekesa konvertimi:

Kerkesa kontrolloportin:

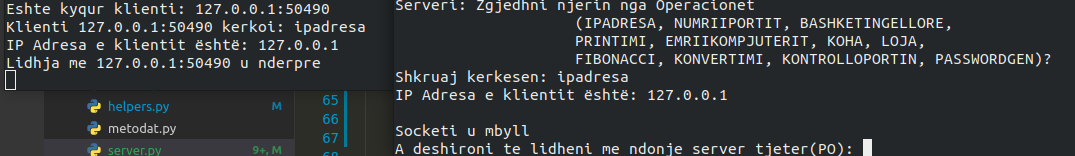
Kerkesa passwordgen:

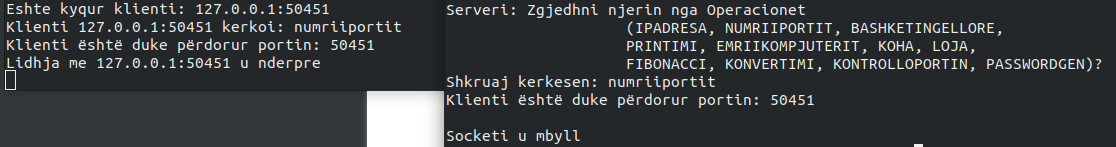
Kerkesat te cilat nuk mund te pergjigjen nga serveri:

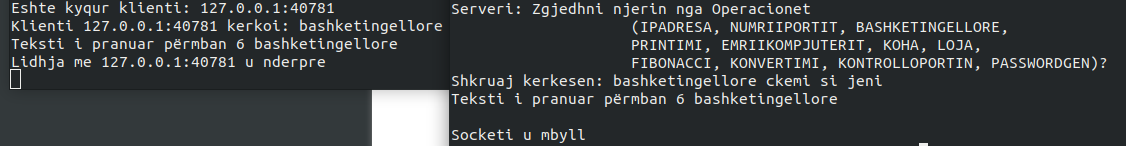


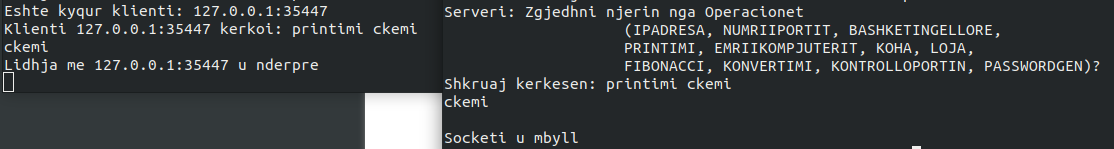
7.1.2 Testimi i protokollit UDP

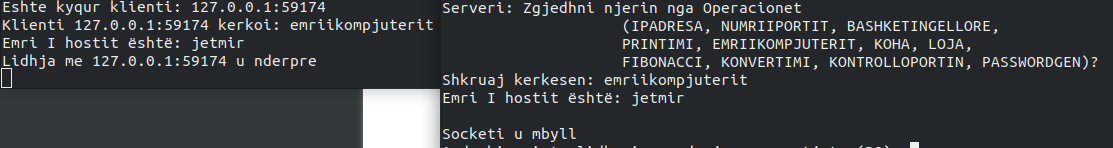
Kerkesa ipadresa:

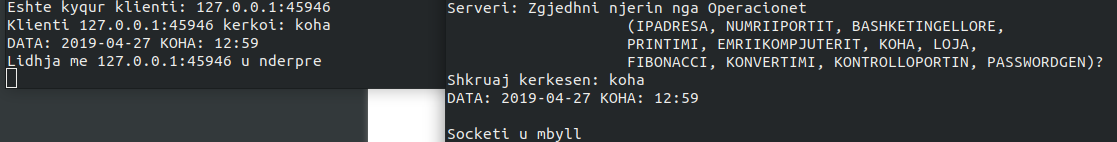
Kerkesa numriiportit:

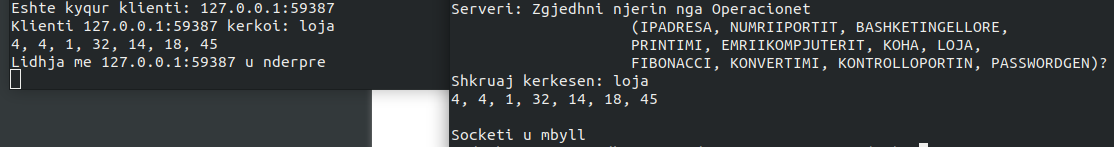
Kerkesa bashketingellore:

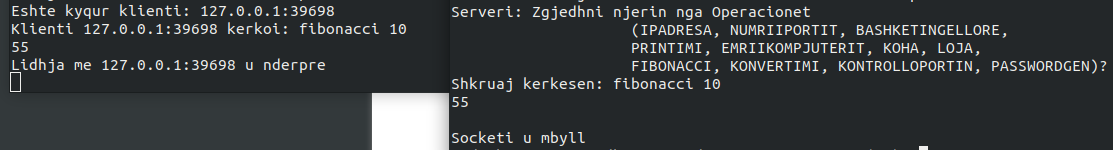
Kerkesa printimi:

Kerkesa emriikompjuterit:

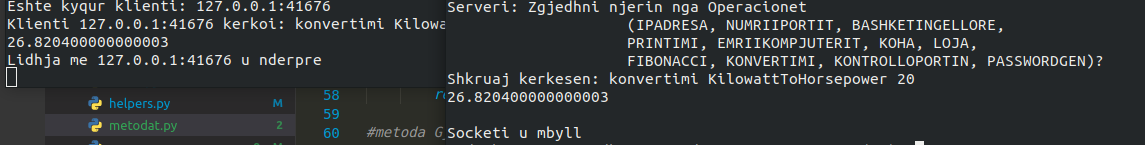
Kerkesa koha:

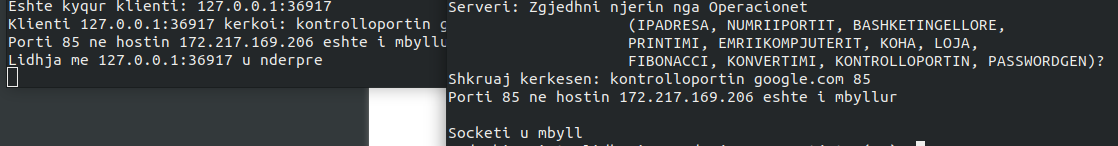
Kerkesa loja:

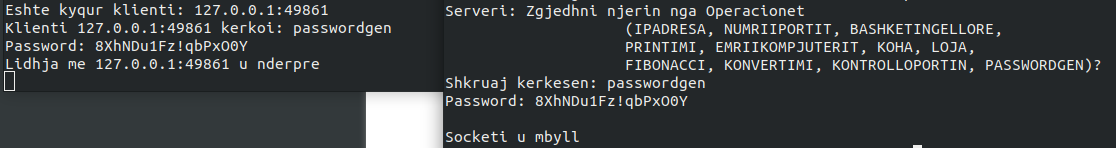
Kerkesa fibonacci:

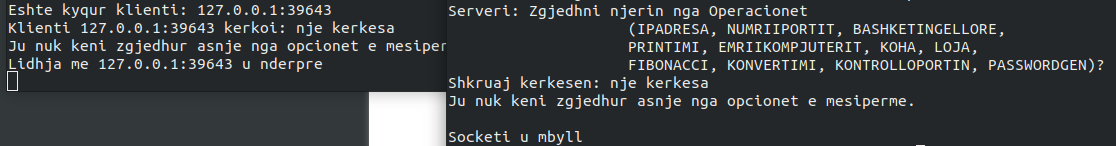


Kerkesa konvertimi:

Kerkesa kontrolloportin:

Kerkesa passwordgen:

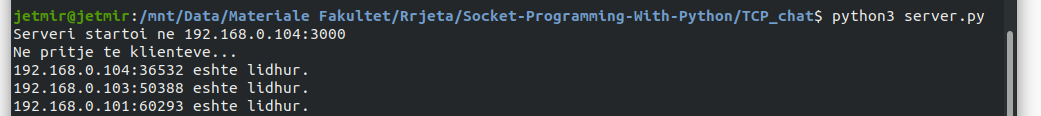
Kerkesa e cila nuk mund te pergjiget nga serveri:



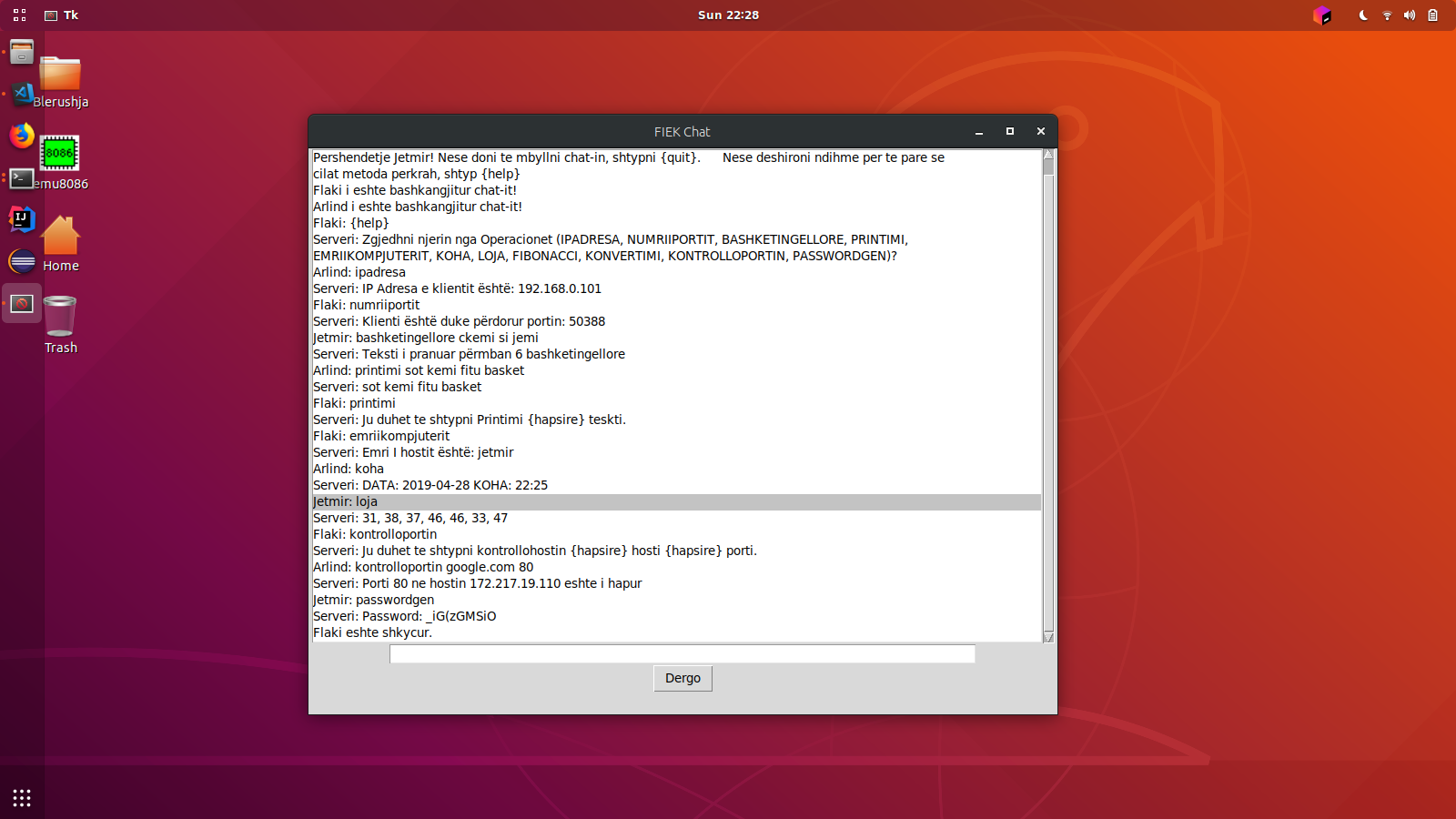
### 

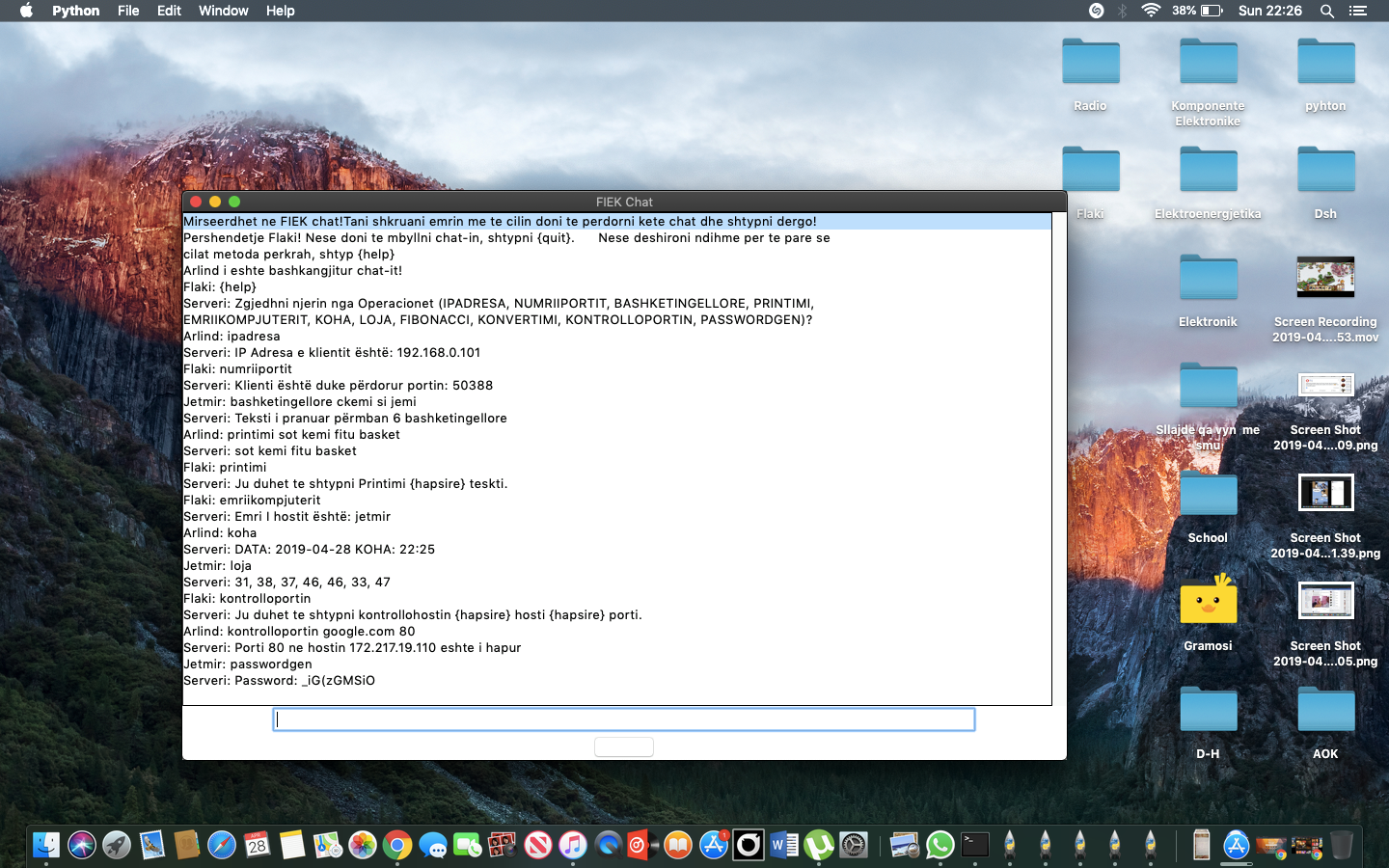
### 7.2 Testimi ne LAN

7.2.1 Serveri:

Klienti:

Ubuntu:

Mac:



Windows:

