**UNIVERSITETI I PRISHTINËS**

**Fakulteti i Inxhinierisë Elektrike dhe Kompjuterike**



Lenda: Rrjetat Kompjuterike

Emri i projektit: Programimi me Soketa

Studenti: Rinor Ahmeti

Data e dorezimit: 10 Prill 2020

Mesimdhenesit e lendes:

Prof.Dr.Blerim Rexha & Msc.Haxhi Lajqi PhD Cand

Veglat e perdorura: Python for Visual Studio Code

Sistemi Operativ: Windows-10-10.0.18362-SP0

Funksionet e implementuara:

IPADDRESS

PORT

COUNT

REVERSE

PALINDROME

TIME

GAME

CONVERT

GCF

SERVERINFO

CHECKEMAIL

ISDIGIT

SHUMA

ARMSTRONG

Tabela e përmbajtjes

[HYRJE 4](#_Toc37168449)

[TCP Serveri dhe Klienti 5](#_Toc37168450)

[UDP SERVERI DHE KLIENTI 9](#_Toc37168451)

[Përshkrimi i metodave 11](#_Toc37168452)

[Metoda IPADDRESS 11](#_Toc37168453)

[Metoda PORT 11](#_Toc37168454)

[Metoda COUNT 11](#_Toc37168455)

[Metoda REVERSE 12](#_Toc37168456)

[Metoda PALINDROME 12](#_Toc37168457)

[Metoda TIME 12](#_Toc37168458)

[Metoda GAME 13](#_Toc37168459)

[Metoda CONVERT 13](#_Toc37168460)

[Metoda GCF 14](#_Toc37168461)

[Metoda CheckEmail 14](#_Toc37168462)

[Metoda ISDIGIT 15](#_Toc37168463)

[Metoda SHUMA 15](#_Toc37168464)

[Metoda SERVERINFO 15](#_Toc37168465)

[Metoda ARMSTRONG 16](#_Toc37168466)

[Rezultatet e testimit 16](#_Toc37168468)

[Përmbledhje 19](#_Toc37168469)

[Referencat 20](#_Toc37168470)

# HYRJE

Qëllimi i projektit të zhvilluar është krijimi i një TCP Protokolli dhe një UDP Protokolli ku klienti (apo klientët) mund të komunikojnë me serverin tonë dhe t’i dërgojnë kërkesa të ndryshme, te cilave iu pergjigjet ky server.

Projekti është shkruajtur në gjuhën programuese Python, përmes veglës Visual Studio Code. Per punimin e projektit janë shfrytëzuar libraritë si: ***sockets, numpy, platform, os, reg, datetime, threading*** etj, por libraria kryesore e cila e mundëson lidhjen server-klient është ***sockets***.

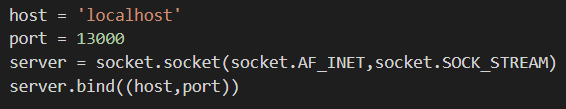
Programi gjithsej permban 4 fajlla, pra dy çifte server-klient per protokollin TCP dhe UDP respektivisht.

Ne faqet e tjera do të shohim në detaje se si është realizuar & organizuar programi, do të shohim funksionet e implementuara si dhe rezultatet e testimit të metodave.

# TCP Serveri dhe Klienti

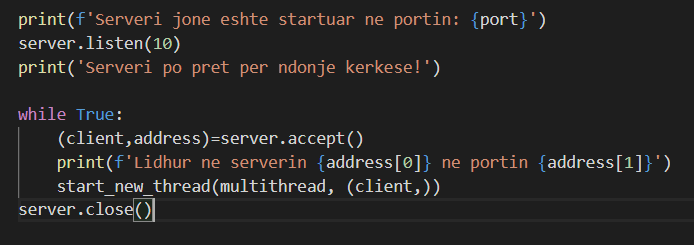
Fillimisht në variablat host dhe port i ruajmë IP adresën e serverit dhe portin respektivisht, pastaj krijohet socket session me protokolin TCP përmes atributit SOCK\_STREAM. Argumenti i parë AF\_INET paraqet address family kurse SOCK\_STREAM paraqet protokollin, në rastin tonë TCP.

Pastaj, përmes komandës server.bind krijohet tuple host&port apo ipaddress dhe port me ç’rast klientët tanë mund t’i çasen serverit tonë.



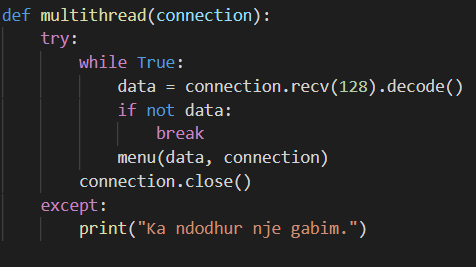
Përmes komandës server.listen() degjojmë kërkesat e klientëve.

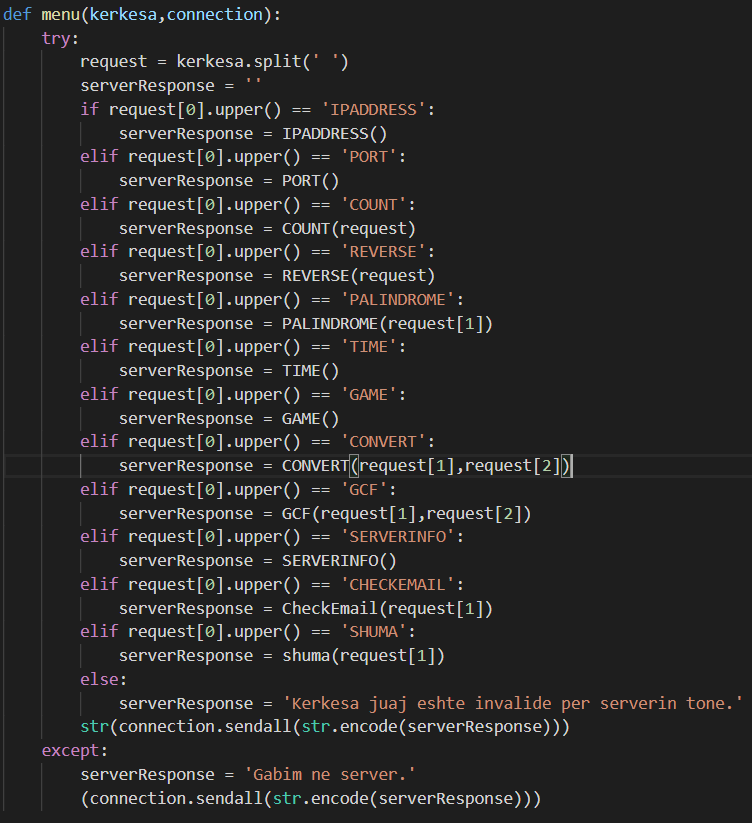
Përmes unazës se pafundme pranojmë lidhjën e klientit përmes komandës server.accept, me ç’rast në ekran shfaqet klienti në serverin dhe në portin që është lidhur.



Në thread pranohet kërkesa e dërguar nga klienti e cila në rast se nuk dështon, i dërgohet si argument metodës **menu** e cila i përmban të gjitha funksionet tjera që i ofron serveri, pastaj serveri i përgjigjet kërkesës së klientit.

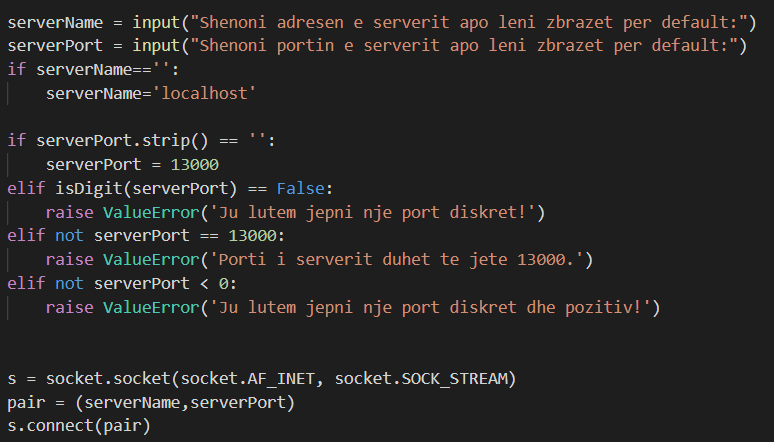
Serveri mund të punojë me shumë klientë në të njejtën kohë, përmes multithreading.





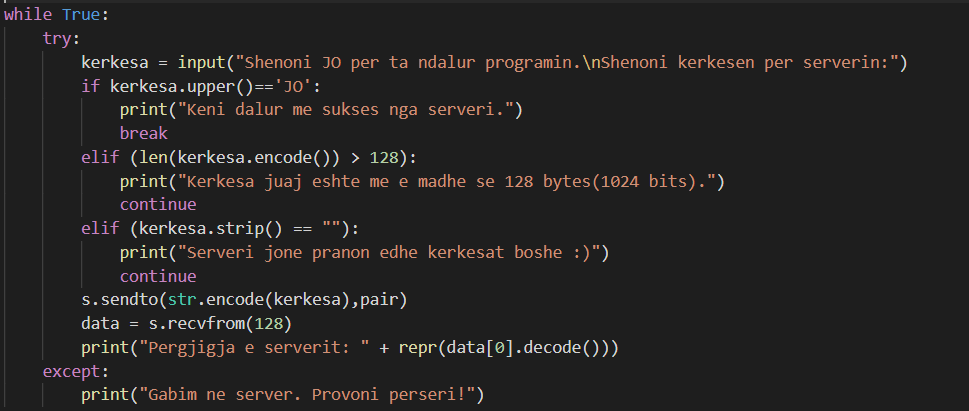
Nëse serveri mund ta procesojë kërkesën e shfrytëzuesit, e kthen rezultatin e kërkesës, nëse kërkesa e shfrytëzuesit është invalide atëherë serveri e informon shfrytëzuesin, ndërsa nëse në server ndodhë ndonje gabim, atëherë përmes bllokut try & except serveri e informon klientin se ka ndodhur një gabim por nuk përfundon sesioni server-klient dhe serveri vazhdon të punojë pa u ndalur.

Në anën tjeter, shfrytëzuesit fillimisht i ipet mundësia qe të zgjedhë IP Address e serverit dhe portin qe dëshiron të lidhet. Në rast se klienti nuk jep ndonjë address dhe port, ato automatikisht vendosen si vlera default. Mirëpo, nëse klienti jep vlera të pakuptimta për serverin, atëherë e lajmëron shfrytëzuesin se ku ka ndodhur gabimi përmes kushtëzimeve dhe error mesazheve që shfaqen në ekran.



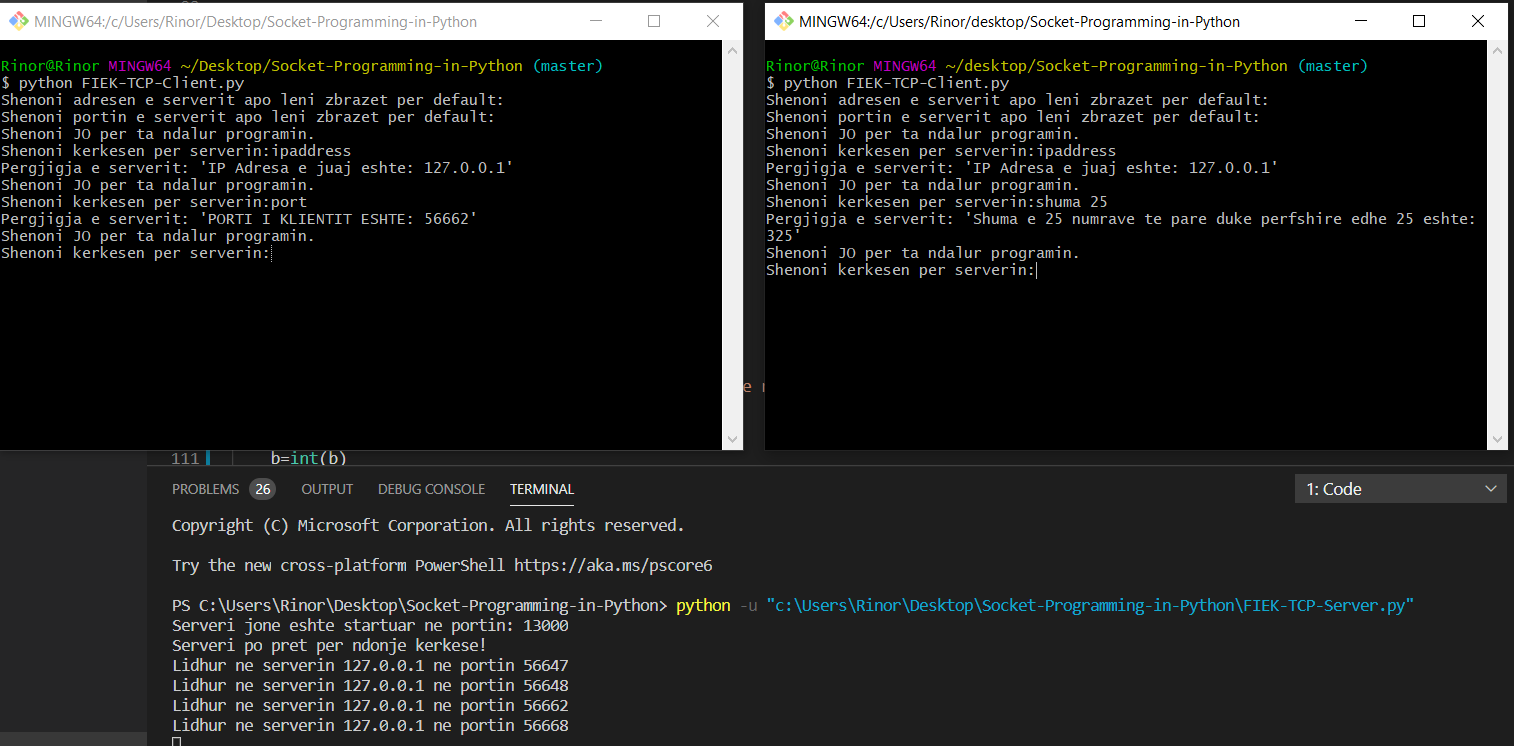
Sikur tek serveri ashtu edhe tek klienti krijohet socket sessioni. Pra tek ky protokoll, bëhet handshake, pra krijohet lidhja mes klientit dhe serverit, pastaj pasojnë kërkesat e klientit dhe shërbimet qe mund t’i kryejë serveri. Përmes komandës s.connect lidhet klienti me serverin, ku pair paraqet nje tuple e cila përmban adresën e serverit dhe portin e tij.

Përmes unazës së pafundme while True, klienti e ka mundësinë që t’i dergojë kërkesa të ndryshme serverit dhe lidhja të mos dështojë deri sa klienti të ketë dëshirë.



Klienti e ka mundësinë që ta ndalojë lidhjen e tij me serverin në çdo kohë. Edhe atëherë kur klienti nuk e dërgon asnjë kërkesë, serveri vazhdon të punojë. Mirëpo nëse ndodhë ndonjë gabim eventual gjatë sesionit, përmes bllokut try & except e kapim gabimin dhe serveri vazhdon punën pa ndaluar fare.

Serveri punon me shumë klientë dhe kërkesa në të njejtën kohë.



# UDP SERVERI DHE KLIENTI

Përveç TCP protokollit, e kemi të implementuar edhe UDP protokollin ne server. Këto protokolle janë shumë te ngjashme përveç se këtu nuk ka nevojë për handshake, pra nuk bëhet lidhja klient-server mirëpo vetem dërgohet kërkesa në server e cila pastaj trajtohet nga serveri dhe kthehet përgjigja.

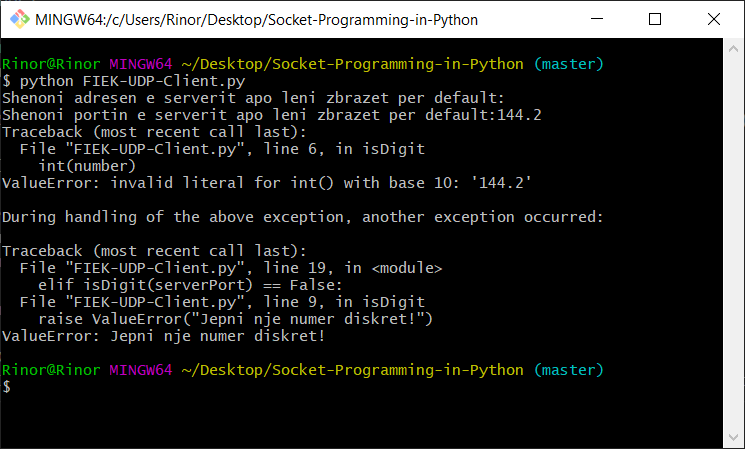
UDP protokolli krijohet përmes komandes SOCK\_DGRAM e cila qëndron për DATAGRAM, prej nga vjen edhe emri i këtij protokolli, pra User Datagram Protocol.



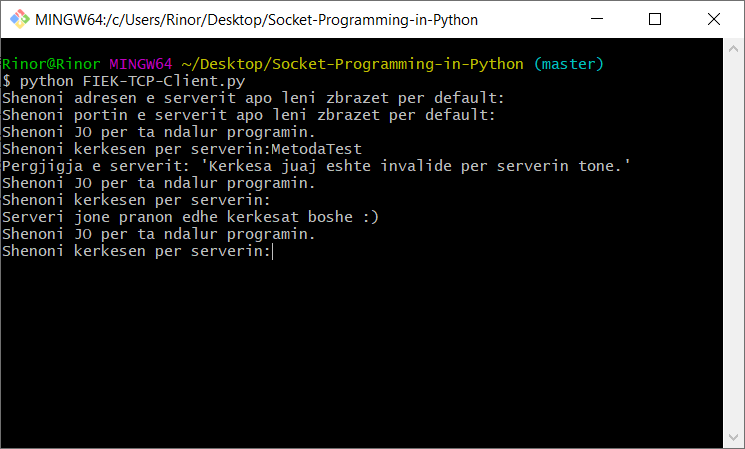
Kerkesat trajtohen ngjashëm sikurse tek TCP serveri, përveç se këtu nuk krijohen threads për klientë por vetem pranohen kërkesat. Poashtu, në rast se klienti jep port të gabuar apo të formatit të gabuar, serveri e lajmëron për gabimin se ku ka ndodhur dhe për çfarë arsye. Të gjitha metodat që janë të implementuara në TCP server janë të implementuara edhe ne UDP server kështu qe provat dhe imazhet në vijim vlejnë për të dy protokollet.

Në vijim do e testojmë se çfarë ndodhë kur japim një port të gabuar, dërgojmë një kërkesë të gabuar apo ndodhë ndonjë gabim në server.

Në rast se japim port të gabuar, atëherë serveri na përgjigjet si në vijim:



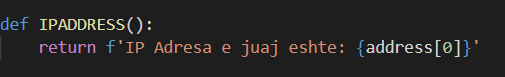
Në rast se dërgojmë ndonjë kërkesë qe nuk mund ta kryejë serveri, atëherë serveri përgjigjet si në vijim:



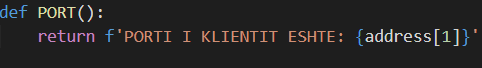
# Përshkrimi i metodave

Ne vijim do t’i përshkruajmë metodat bazë dhe metodat shtesë.

**Metoda IPADDRESS** e kthen tek klienti ip adresën e tij e cila merret nga antari i parë i dyshës address.

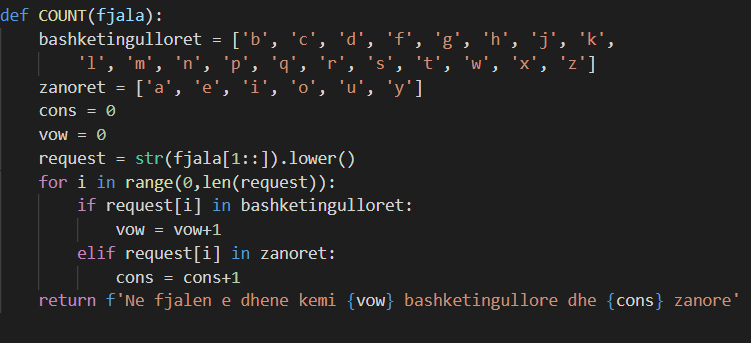


**Metoda PORT** e kthen tek klienti portin në të cilin është lidhur, i cili merret nga antari i dytë i dyshës address.

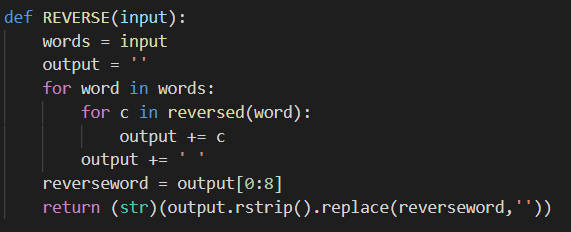


**Metoda COUNT** e kthen numrin e zanoreve dhe bashkëtingulloreve. Permes vetive të stringut dhe slicing funksioneve, e anashkalojmë argumentin e parë e cila në këtë rast paraqet emrin e funksionit COUNT sepse metodes “menu” si argumente i shkojnë **emri i funksionit** dhe argumentet e funksionit të dhënë.

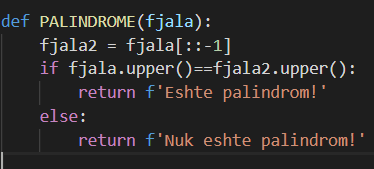
Pastaj i kemi të deklaruara dy lista ku njëra i përmban zanoret e alfabetit e tjetra bashkëtingulloret dhe përmes unazës dhe kushtëzimeve shohim se nëse germa ne pozitën e dhënë të argumentit tonë i takon bashkëtingulloreve ose zanoreve.



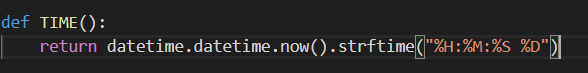
**Metoda REVERSE** pranon një apo më shumë fjalë të cilat i kthen mbrapsht shkronjë për shkronjë dhe pastaj e kthen rezultatin. Për shkak se funksioni kryesor, **menu** i pranon si argumente emrin e funksionit dhe pastaj argumentet e tij, në këtë rast p.sh REVERSE argument1 argumenti2, atëherë për ta anashkaluar emrin e funksionit në printim, kemi përdorë vetitë e slicing në string ku pra fjala “REVERSE(apo sido qe i jepet emri p.sh rEVerSe) zëvendësohet me ‘’ përmes funksionit replace me ç’rast funksioni ynë është shumë dinamik.



**Metoda PALINDROME** e merr si argument fjalën e dhënë dhe pastaj e krahason me reversin e saj dhe pastaj shfaqë rezultatin në ekran.

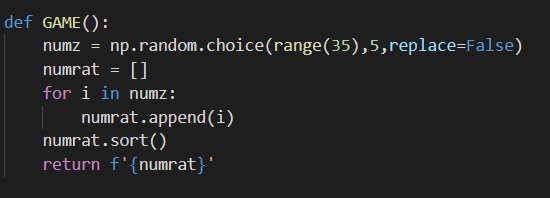


**Metoda TIME** e kthen kohen ne nje format te pershtatshem per t’u lexuar nga njeriu.

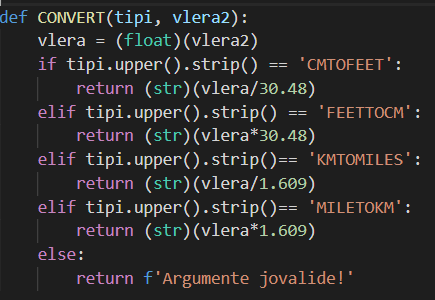


**Metoda GAME** – përmes kësaj metode e kemi bërë mbështjelljën e metodës së gatshme random.choice e cila i takon librarisë ***numpy***. Kjo metodë pranon gjithsej 3 argumente. Argumenti i parë është range, pra në çfarë range dëshirojmë qe t’i gjenerojmë numrat tanë, argumenti i dytë e tregon se sa numra dëshirojme t’i gjenerojmë dhe argumenti i tretë na mundëson qe mos të gjenerojmë numra duplikatë, pra argumenti replace=False.

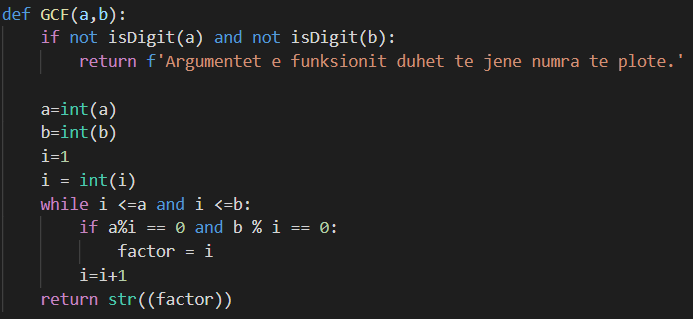
Përmes unazës iterojmë në cdo numër të gjeneruar të cilët pastaj i ruan në listën tonë, sortohet e pastaj i kthehet klientit si përgjigje.



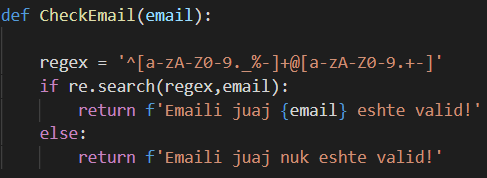
**Metoda CONVERT** i pranon dy argumente: tipin dhe vlerën. Përmes kësaj metode mund të kryjmë konvertimet më të zakonshme p.sh prej cm në feet, apo prej miles në km. Në rast se argumentet janë jo valide na lajmëron funksioni apo shfaqet gabimi në server i cili mbahet nga blloku try&except.



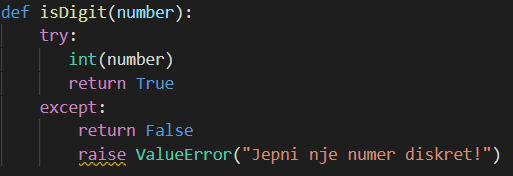
**Metoda GCF** i pranon dy numra të plotë dhe e kthen faktorin e tyre më të madh të përbashkët. Nëse argumentet e funksionit nuk janë numra të plotë, atëherë shfaqet mesazhi që argumentet duhet të jenë numra të plotë dhe vazhdon serveri punën pa u terminuar. Shohim se përmes nje unaze while testohet se nëse te dy argumentet plotëpjestohen me numrin e njejtë. Nëse po atëherë numri i dhënë zgjedhet si faktor dhe shikohet se a gjendet ndonjë faktor tjeter më i madh.



**Metoda CheckEmail** është metodë shtesë e cila pranon si argument një email dhe e validon atë, pra tregon nese ajo email adresë e dhënë është valide apo jo. Kjo metodë e përmban metodën search e cila i takon librarisë ***re*** e cila shërben për Regular Expressions. Kjo metodë pranon dy argumente: patternin dhe argumentin i cili do të testohet.

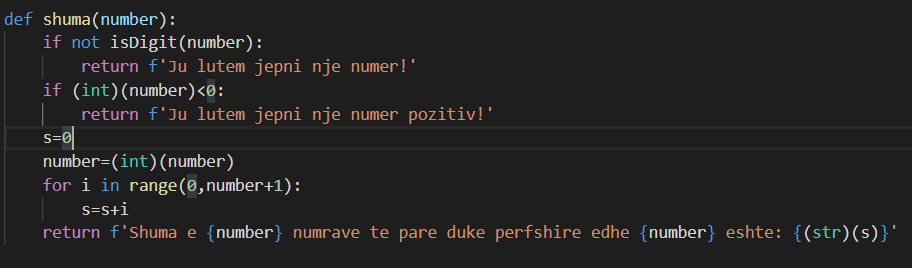


**Metoda ISDIGIT** është metodë ndihmëse e cila na ndihmon qe t’i validojmë inputet ne metodat tjera në server. Kjo metodë pranon një argument dhe shikon nëse argumenti i dhënë është numer i plotë apo jo. Nëse numri i dhënë është i tipit int atëherë metoda kthen True, në të kundërtën ajo kthen False dhe gjuan një error të klases **ValueError** me mesazhin se klienti duhet të japë numër të plotë. Këtë metodë e kemi përdorë në disa metoda tjera, p.sh tek GCF por edhe tek inputi i portit të cilin e jep klienti.



**Metoda SHUMA** është një metodë e cila e pranon si argument një numër të plotë dhe më të madh se zero dhe e kthen shumën e numrave deri tek numri i dhënë duke e përfshirë edhe atë numer, pra kufiri i unazës (number+1).

Në rast se numri i dhënë nuk është int apo nuk është më i madh se zero, atëherë serveri na tregon se ka ndodhur gabim në input.



**Metoda SERVERINFO** është një metodë relativisht e thjeshtë e cila i printon disa nga të dhënat e serverit si në vijim duke përdorur librarinë ***platform***:

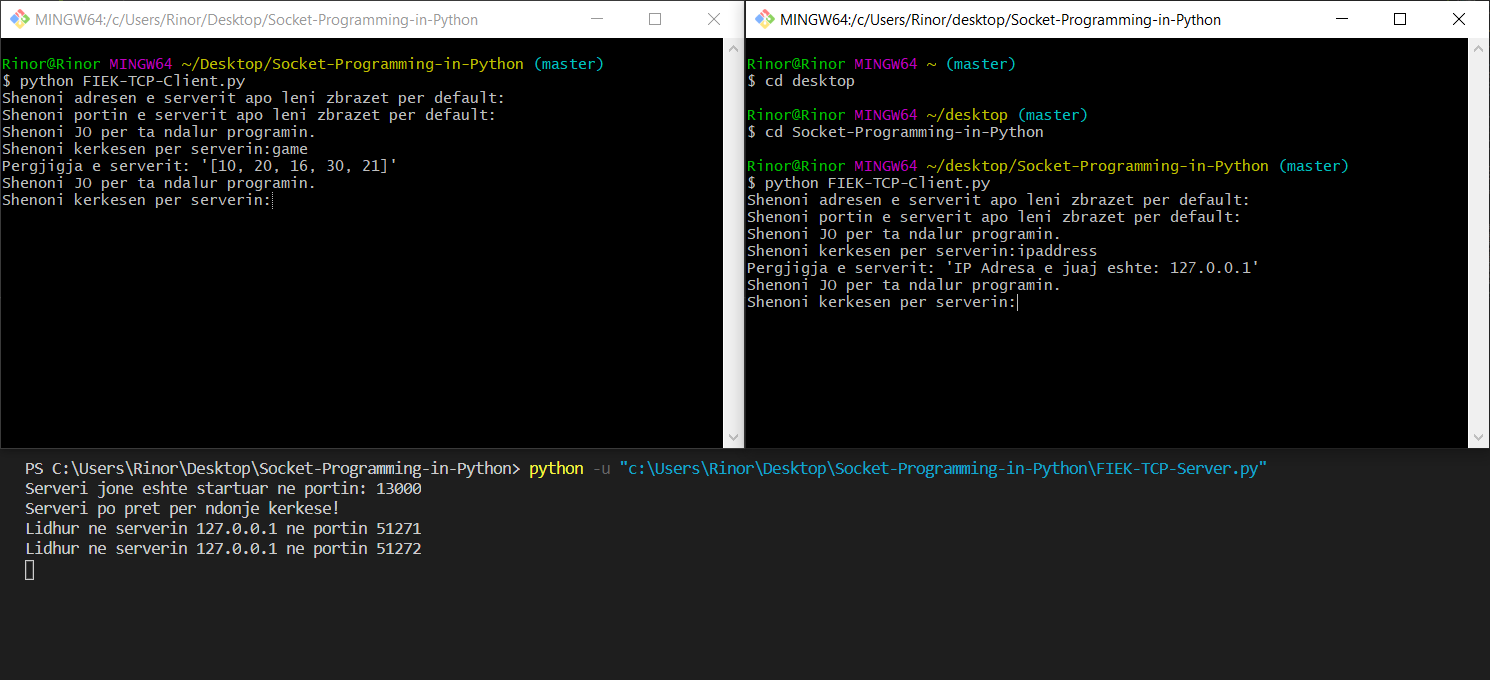


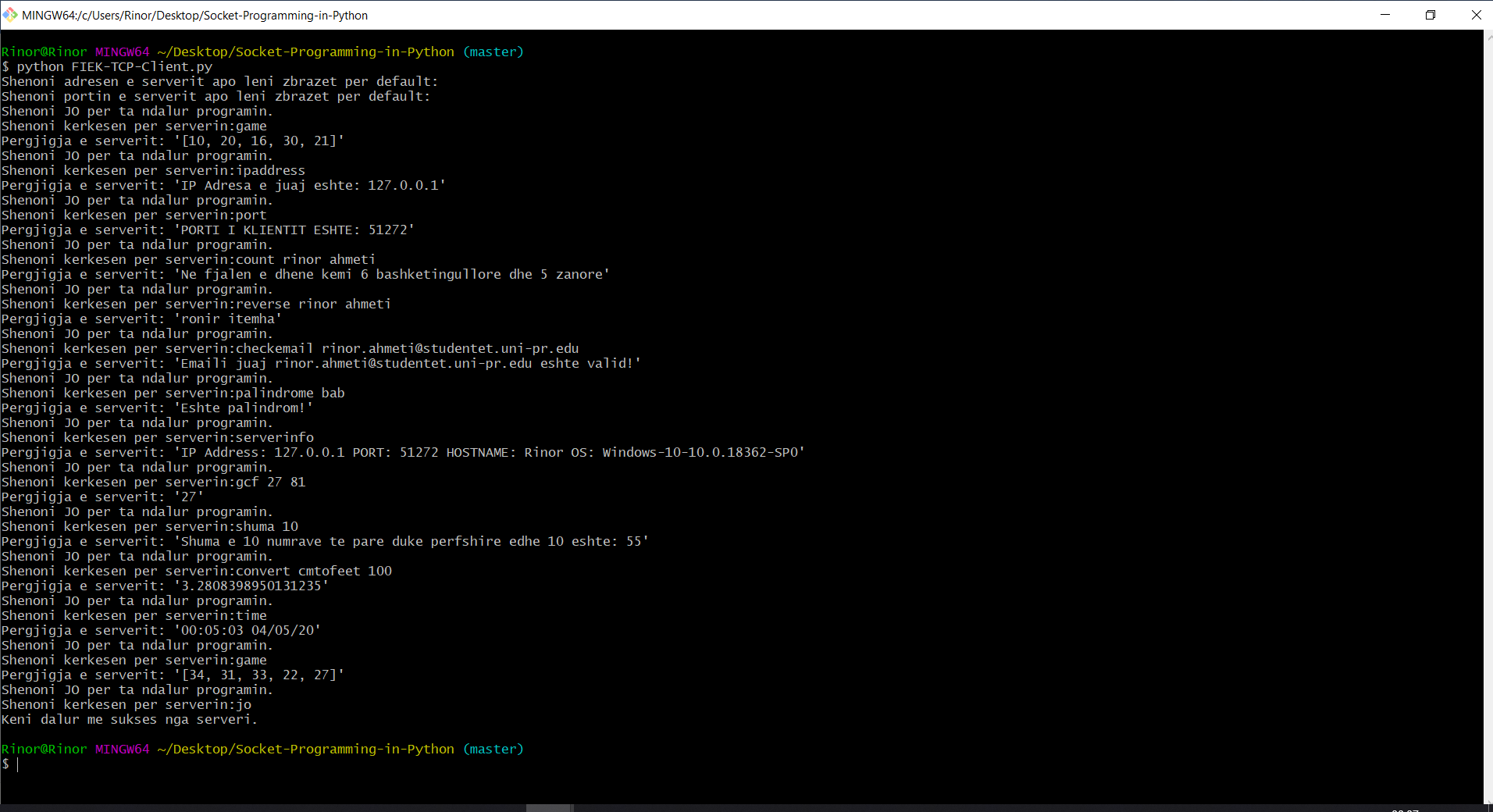
**Metoda ARMSTRONG** është një metodë e cila pranon ne hyrje nje number të plotë dhe në dalje e informon klientin se a eshte numer i Armstrongut numri i dhënë apo jo. Numri i Armstrongut është ai numer i cili punon sipas parimit: xyz = x^n + y^n + z^n ku n paraqet numrin e shifrave qe ka numri. Pra numri treshifror eshte numer i Armstrongut atehere dhe vetem atehere kur shuma e kubeve te seciles shifer te atij numri na jep vet numrin e dhene.Ne kete rast, kjo metode funksionon vetem per numra 3 shifror. Disa nga numrat treshifror te Armstorngut jane 153 dhe 407.

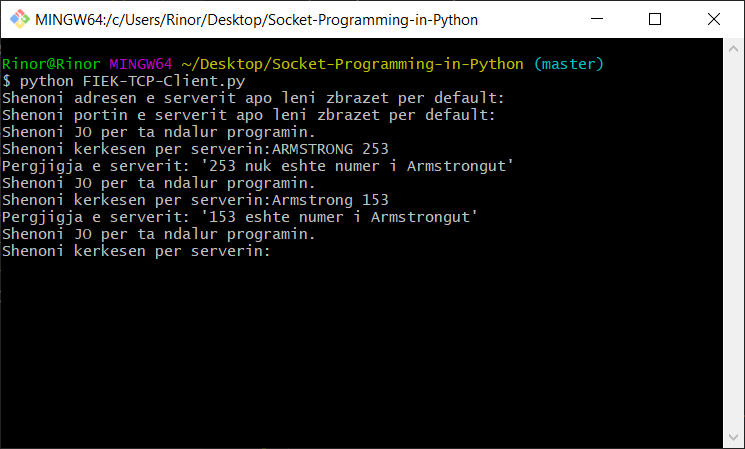
# 

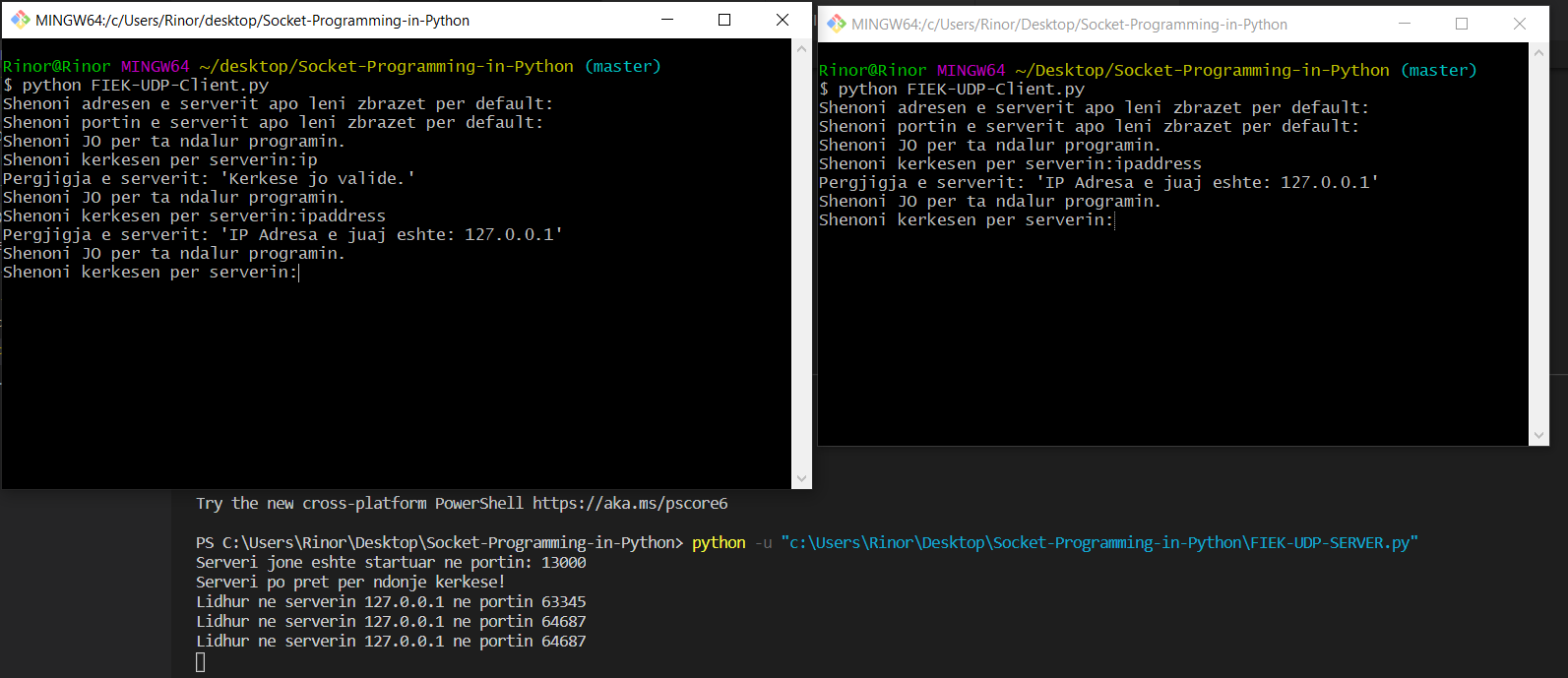
# **Rezultatet e testimit**

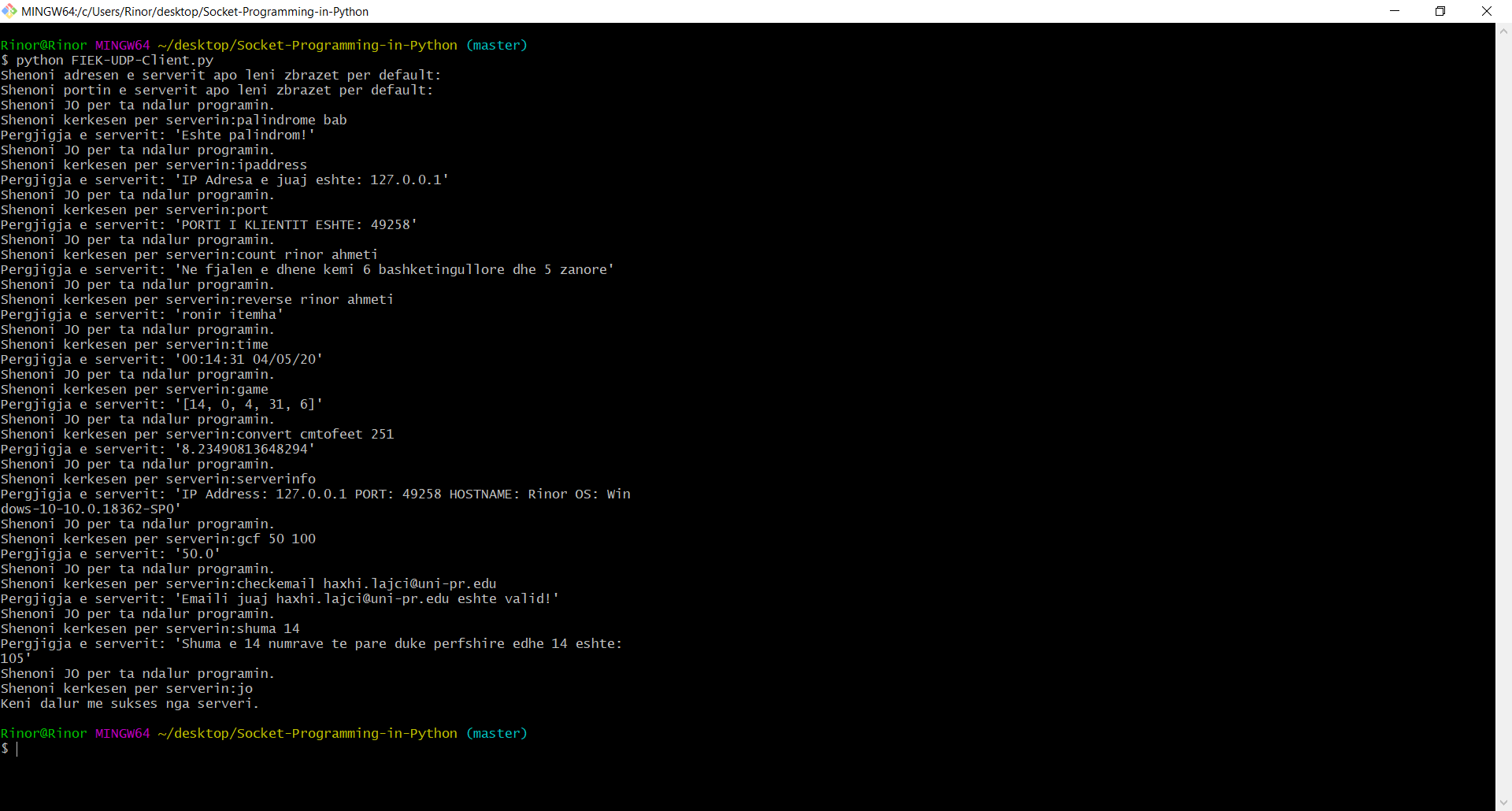
Lidhja në TCP server shohim se mund të bëhet me dy e më shumë klientë dhe mund të pranojë disa kërkesa njëkohësisht. Në vijim do t’i testojmë edhe të gjitha funksionet.







Rezultatet e testimit ne UDP: 



# **Përmbledhje**

Të gjitha funksionet ne të dy protokollet jane identike dhe të gjitha janë testuar. Çdo funksion është i validuar sipas nevojës, ashtu që nëse pranon ndonje argument jo-valid atëherë të reagojë dhe të mos terminohet puna e serverit, pra edhe nëse ipet ndonjë argument jo-valid në server, serveri vazhdon punën pa u ndalur deri sa të deshiron klienti.

Përveç validimeve, në server dhe në klient kam përdorur try&except blloqe të cilat e mundësojne vazhdimin e punës së serverit pavarësisht gabimeve qe mund të ndodhin përderisa serveri është online. Funksionet janë testuar të gjitha dhe të gjitha jane në gjendje të rregulltë. Funksionet jam munduar që t’i bëj shumë dinamike dhe t’i përshtatem formatit dhe kërkesave.

# **Referencat**

Artikujt dhe linqet tek të cilët jam bazuar gjatë punimit të këtij projekti janë:

1. <https://docs.python.org/3/library/socket.html>
2. <https://realpython.com/python-sockets/>
3. <https://www.geeksforgeeks.org/socket-programming-multi-threading-python/>
4. Ushtrimet e klases

Prishtine, Prill 2020