**UNIVERSITETI I PRISHTINËS**

**Fakulteti i Inxhinierisë Elektrike dhe Kompjuterike**

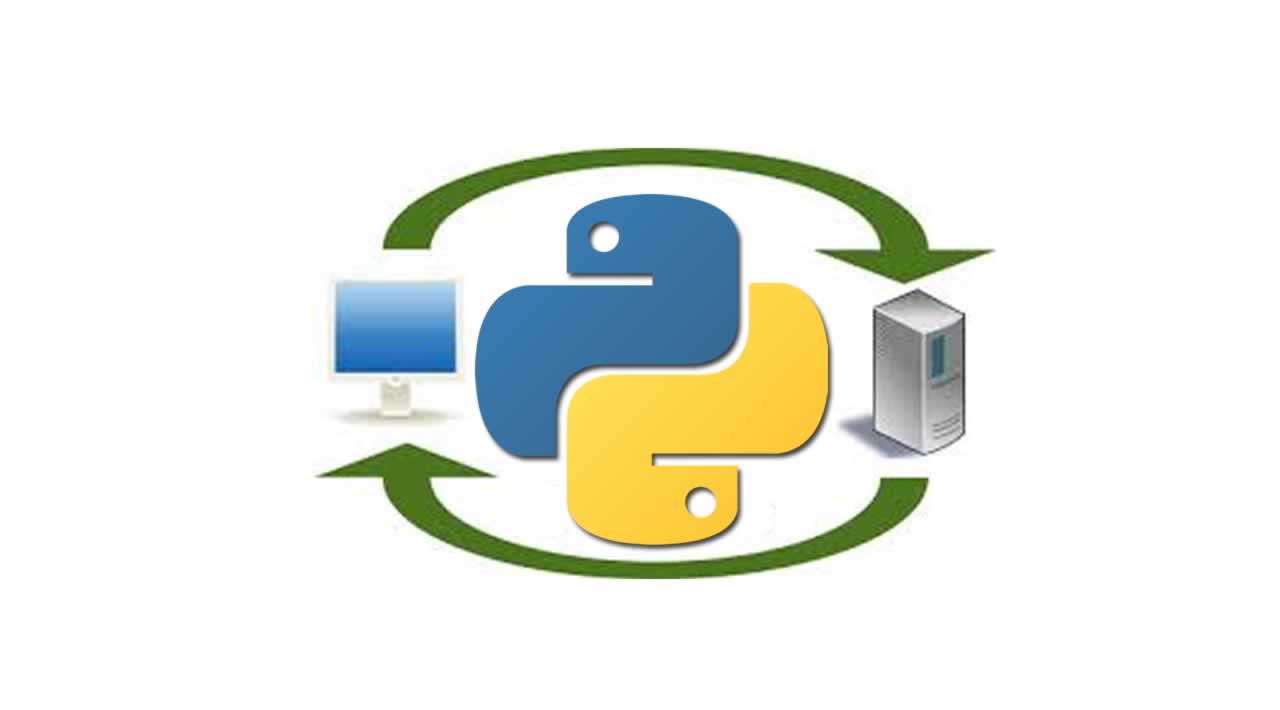
**Departamenti i Kompjuterikës**



**Projekti 1**

**Lënda: Rrjetat kompjuterike**

Programimi me soketa

****

**Profesor:** Prof. Dr.Blerim Rexha **Studenti:** Betim Thaçi **Asistent:** Msc. Haxhi Lajqi **Nr.Index:** 180716100041

**Prill 2020**

PЁRBAJTJA..................................................................................

1. Abstrakti..................................................................................................................................................3

2. Hyrja........................................................................................................................................................4

2.1 Protokolli TCP...........................................................................................................................4

2.2. Protokolli UDP.........................................................................................................................4

3. Përshkrimi kodit......................................................................................................................................5

3.1. TCP-Serveri..............................................................................................................................5

3.2. TCP-Klienti...............................................................................................................................7

3.3. UDP-Serveri & Klienti...............................................................................................................8

4. Metodat dhe pëshkrimi i tyre..................................................................................................................8

4.1. Ipaddress.................................................................................................................................8

4.2. Port..........................................................................................................................................9

4.3. Count.......................................................................................................................................9

4.4. Reverse....................................................................................................................................9

4.5. Palindrome.............................................................................................................................10

4.6. Time........................................................................................................................................10

4.7. Game......................................................................................................................................11

4.8. GCF.........................................................................................................................................11

4.9. Convert...................................................................................................................................11

4.10. Pow.......................................................................................................................................12

4.11. Tabela...................................................................................................................................13

5. Veglat e përdorura.................................................................................................................................14

6. Përfundimi..............................................................................................................................................14

7. Biblografia..............................................................................................................................................14

**1. ABSTRAKTI**

Fokusi i këtij projekti është në të kuptuarit e mënyrës se si bëhet komunikimi në mes serverit dhe klientit permës sockets duke marrë për bazë TCP-n dhe UDP-n . E tërë kjo është bërë me veglën Visual Studio në gjuhën e lartë programuese Python andaj I tërë problem I sigurimit të lidhjes dhe mënyrës se si pergjigjja vie deri tek klienti është shtruar në të lartëcekurat , duke pasur parasysh të gjitha rregullat dhe konditat e parashtruara në projekt.

Pasi është siguruar lidhja në mes klientit dhe serverit për kushtet që janë paraparë në projekt , është bërë zgjidhja e atillë që klienti dërgon te serveri kërkesë se varësisht se cfarë metode deshiron të thirrë, ku aty më pastaj kryhen llogaritjet dhe sigurohet përgjigjja e cila përsëri është kthehet tek klienti .

**2. HYRJA**

Programimi me soketa nuk është aq i vështirë e as aq i lehtë sa paragjykohet në fillim. Mjafton të mendohen klienti dhe serveri si dy programe të ndara por shumë të ngjashme që dallojnë kryesisht nga qëllimi i tyre. Ajo çka dërgohet nga klienti quhet kërkesë ndërsa çka dërgohet nga serveri tek klienti do quhet përgjigje.

**2.1. Protokolli TCP**

Në protokollin TCP tek klienti dhe serveri krijohen soketat përkatëse, ku klienti konektohet në port ndërsa serveri "ngjitet" (bind). Dallimi mes *connect* dhe *bind* është se me *connect* lirohet porti për përdorim nga ana e klientit posa të dërgohet kërkesa, ndërsa me *bind* nga ana e serverit ai port është i zënë gjersa serveri të jetë aktiv. Hapi i ardhshëm i serverit është që të ndëgjojë *listen*(*nr*) ku *nr* paraqet se sa klienta qëndrojnë në pritje tek ai port nëse ai port është tashmë i zënë me ndonjë klient tjetër. E më pas serveri duhet të bëhet gati per pranimin e kërkesave me *socket.accept()*.

Tanimë klienti e dërgon kërkesen *me socket.sendall*(*kerkesa*) e cila kërkesë duhet të shëndrrohet në bajta paraprakisht. Me connectionSocket.recv(*length*) (parametri i parë i socket.accept() ) serveri e pranon kërkesën në formë të bajtave ku *length* paraqet numrin e bajtave. Pas përpunimit të kërkesës nga ana e serverit, ai e kthen përgjigjen me anë të komandës *connectionSocket.send*(*përgjigjeja*) ku *përgjigjeja* duhet të konvertohet në bajta. Më pas lidhja mbyllet me *connectionSocket.close()* nëse klienti veçse nuk ka ndonjë kërkesë të re, dhe gjithashtu nëse dëshirojmë ta ndalojmë punën e serverit, atë e bëjmë me socket.close(). Klienti më pas e pranon përgjigjen nga serveri me komanden *socket.recv*(*length*) e cila përgjigje është e gatshmë të printohet. Pra thënë më shkurtë, fillimisht bëhet një lidhjë përshëndetjeje (handshake) në portin e caktuar me komandat *socket.connect* nga ana e klientit dhe *socket.bind* nga ana e serverit, e më pas përdoren portet tjera të lira të gjeneruara me radhë të cilat nuk janë nën kontrollën e asnjërit përdorues

**2.2. Protokolli UDP**

Në protokollin UDP është shumë më thjeshtë, nga ana e klientit vetëm krijohet soketa dhe dërgohet kërkesa me komandën *socket.sendto*(*kërkesa,* (*IPadresa* , *porti*)) ku kërkesa duhet të jetë e tipit bytes, pra duhet konvertuar paraprakisht. Nga ana e serverit krijohet soketa dhe lidhet serveri për portin dhe hostin përmes komandës *socket.bind*((*serverIP*, *serverPort*)). Më pas serveri e pranon kërkesën me komandën *socket.recvfrom*(*length*) ku *length* është gjatësia e bajtave të kërkesës. Pas përpunimit të kërkesës, dërgohet përgjigjeja (e konvertuar paraprakisht ne bajta) me komandën *socket.sendto*(*përgjigjeja*, *addr*) ku *addr* është ajo lista në të cilën ruhet IP-adresa e klientit dhe numri i portit. Klienti e pranon më pas përgjigjen përmes komandës *socket.recvfrom*(*length*) e cila është një listë ku anëtari i parë është përgjigjeja në bajta ndërsa anëtari i dytë është një listë e përbërë nga IP-adresa dhe numri i portit të caktuar në fillim.

TCP ka shumë arsye pse duhet të përdoret , mirëpo UDP ofron një shpejtësi më të madhe.

**3. PERSHKRIMI PROGRAMIT**

**3.1. TCP-SERVERI**

Fillimisht janë të importuara libraritë që na nevojiten

from socket import \*

import random

import datetime

import sys

import threading

import math

Permes serverPort = 13000 kam definuar portin e serverit

krijohet soketi i serverit, ngjashem me klientin perdoren IPv4 dhe TCP

serverSocket = socket(AF\_INET, SOCK\_STREAM)

soketit te serverit i rezervohet porti 13000

serverSocket.bind(('',serverPort))

dhe server tani është I gatshëm të pranojë kërkesa

serverPort = 13000

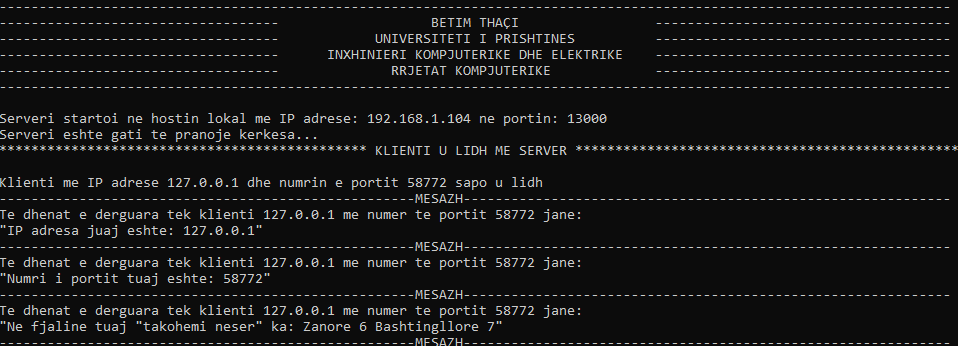
serverSocket = socket(AF\_INET, SOCK\_STREAM)

serverSocket.bind(('',serverPort))

Pastaj kemi metodat qe mund të lexoni për secilën më shumë në sesionin më poshtë (4. Metodat dhe përshkrimi I tyre)

Gjithashtu kemi edhe metodën DERGESAKLIENTIT që ndihmon të tregojë se cka është derguar tek cdo klient

def DERGESAKLIENT(IPKlientit,portiKlientit,mesazhi):{}



E kemi dekoduar kërkesën e marrë nga klienti e kthejmë në string dhe i largojme whitespace-t nga fillimi dhe fundi

kerkesaStr = str(kerkesa.decode()).strip()

Ndajme kerkesen me split për të parë se cka ka shenuar klienti

kerkesaArray = kerkesaStr.split(' ')

Kemi krijuar variablën ndihmëse fjalapar që ka fjalën e parë të shënuar nga klienti

fjalapar = kerkesaArray[0]

Për të mos krijuar krahasime të kota shëndrrojmë kërkesën në fjalë me shkronja të mëdha dhe krahasojmë vetëm me shkronja të mëdha

kerkesaArray[0] = kerkesaArray[0].upper()

Në kod kam tri lloje të funksioneve të thirrura, funksionet me vetem një argument, me dy argumente dhe me tre argumente. Me poshtë janë shembujt se si kam vepruar për te mos lejuar më shume argumentë seq duhet

Nëse kërkesa është me vetëm një argument .

elif kerkesaArray[0]=="PORT":

if(len(kerkesaArray)>1):

error=True

else:

DERGESAKLIENT(addr[0],addr[1],"Numri i portit tuaj eshte: "+PORT(addr))

connectionSocket.send(("Numri i portit tuaj eshte: "+PORT(addr)).encode())

Funksionet me tre argumente

elif kerkesaArray[0]=="TABELA":

for i in range(len(kerkesaArray)):

if "" in kerkesaArray:

kerkesaArray.remove("")

if len(kerkesaArray)>3 or len(kerkesaArray)<3:

error=True

else:

if str(TABELA(str(kerkesaArray[1]),kerkesaArray[2]))=="Error":

error = True

else:

rezultati = "Tabela e gjeneruar sipas kerkesave tuaja eshte: " + TABELA(str(kerkesaArray[1]),str(kerkesaArray[2]))

DERGESAKLIENT(addr[0],addr[1],str(rezultati))

connectionSocket.send(str(rezultati).encode())

Dhe funksionet me dy argumente, ku nuk kemi pasur nevoj të bëjmë handle errorat pasi që argumenti i dytë është fjali dhe mund të jetë i gjatë sa të dojë.

elif kerkesaArray[0]=="COUNT":

fjaliaShenuar = kerkesaStr.replace(fjalapar,"",1)

rezultati = "Ne fjaline tuaj \"" + fjaliaShenuar.strip() + "\" ka: " + COUNT(fjaliaShenuar.strip())

DERGESAKLIENT(addr[0],addr[1],rezultati)

connectionSocket.send(rezultati.encode())

Nëse nuk thirret asnjera nga metodat, dmth argumenti I parë është gabim atëherë kthehet mesazhi: “Kerkesa juaj eshte invalide, ju lutem provoni perseri!”

|  |
| --- |
| else: |
|  | connectionSocket.send("Kerkesa juaj eshte invalide, ju lutem provoni perseri!".encode()) |

Arsyeja pse në fund të çdo metode e kam konvertuar variablen kryesore në string është sepse ajo me pas duhet te enkodohet në bytes, dhe metoda e gatshme encode mund të enkodoje vetëm stringje.

Tani bëjmë pranimin e kerkesave, unaza është e pafundme pasi serveri përderisa është funksional duhet te jetë në gjendje të pranoj kërkesa tërë kohën.

Pranohet lidhja nga cilido klient që i adresohet serverit permes localhost dhe portit 13000

while 1:

connectionSocket, addr = serverSocket.accept()

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* KLIENTI U LIDH ME SERVER \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

print("Klienti me IP adrese %s dhe numrin e portit %s sapo u lidh" %(addr))

Startohet një thread për tu marr me kërkesat e klientit përkates

threading.\_start\_new\_thread(newClient,(connectionSocket,addr))

**3.2. TCP-KLIENTI**

Kemi bere import libraritë e nevojshme sikurse të TCP-Serveri

Te TCP-Klienti kam krijuar një funksion main, ku është futur i gjithë kodi, dhe në qofse klienti shënon CHANGE thirret përsëri funksioni main dhe mundet klienti të lidhet me një server tjetër.

Pastaj kemi bërë deklarimin e dy variablave

serverName = ""

serverPort = 0

Deklarojmë variablën ndihmëse për të shikuar a mund të sigurojmë lidhjen me serverin e caktuar në portin e caktuar

notConnected = True

while notConnected:

notConnected = False

Me variablën komandaInput marrim si string se qka ka shenuar klienti

komandaInput = "empty"

Unaza perseritet gjithmone, perveq nese ka ndonje break brenda saj, me ane te kesaj unaze sigurohet qe klienti t'i dergoje shume kerkesa serverit ne nje lidhje TCP te vetme

while 1: {...}

Mbrenda unazes kemi kushtzimet if për:

Ne rast se nuk shenon klienti asgje, shkon kerkesa invalide

if komandaInput == "":

soketiKlientit.sendall(str("ERROR").encode())

Ne rast se klienti shenon EXIT duhet qe te mbyllet lidhja me server, perndryshe dergohet kerkesa e enkoduar me UTF-8

elif komandaInput.upper()!="EXIT":

soketiKlientit.sendall(str(komandaInput).encode())

else:

break

Me këtë marrim të dhënat nga serveri

data = soketiKlientit.recv(128)

Mbyllim lidhjen me server

soketiKlientit.close()

**3.2. UDP-KLIENTI & SERVERI**

Pasiqë UDP nuk pranon shumë kërkesa por vetëm një kërkesë për një herë , në pjesën e kodit janë bërë disa ndryshime .

Nuk e kemi unazën while 1 e cila përsëritet gjithmonë dhe këtu nuk e kam bërë të mundur rastin CHANGE pasiqë UDP pranon vetëm një kërkesë dhe kjo nuk është e nevojshme.

SOCK\_DGRAM nënkupton që është një UDP socket .

Te pranimi i kërkesave nuk e kemi më unazën while 1 pasi që duhët të pranohet vetëm një kërkesë.

Kërkesat i marrim përmes,

kerkesa, addr = serverSocket.recvfrom(128).

Pastaj dekodimi, largimi I ëhitespace, krijimi I nje variable të re bëhet njejtë sikur te TCP-SERVERI

Me dallim nga TCP këtu të dhënat kthehen pëmes serverSocket.sendto(("IPADDRESS(addr)).encode(),addr)

Programi im ka të mundur që një server të lidhet me shumë klienta njëkohësisht dhe një klient të ndrroj serverin të lidhet nga një server me një portë të caktuar me një server me port tjetër.

**4. METODAT DHE PERSHKIRMI I TYRE**

Programi jonë përmban 11 metoda:

1. IPPADRESS,

2. PORT,

3. COUNT,

4. REVERSE,

5. PALINDROME,

6. TIME,

7. GAME,

8. GCF,

9. CONVERT,

10. TABELA,

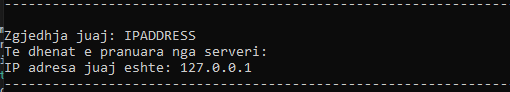
11. POW.

**4.1. IP ADDRESSA**

Kjo metodë është pa parametra dhe qofse e thirrim këtë metodë kjo përcakton dhe kthen IP adresën e klientit p.sh. (10.10.7.251)

def IPADDRESS(address):

return str(address[0])

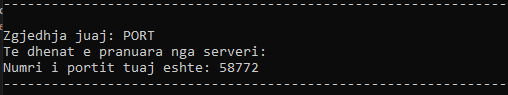


**4.2. PORT**

Edhe kjo metodë nuk ka asnjë parametër, kjo metodë përcakton dhe kthen portin e klientit (dmth portin e klientit jo portin e serverit).

def PORT(address):

return str(address[1])



**4.3. COUNT**

Për të thirrur këtë metodë duhet që të shenojmë COUNT dhe nje fjali, kjo metodë kthen numrin e zanoreve dhe bashtinglloreve që janë në atë fjali.

def COUNT(fjalia):

zan = 0;

basht = 0;

ZANORET = ['A','E','Ë','I','O','U','Y','a','e','ë','i','o','u','y']

BASHT = ['Q', 'R', 'T', 'P','S','D','F','G','H','J','K','L','Z','X','C','V','B','N','M','q','r','t','p','s','d','f','g','h','j','k','l','z','x','c','v','b','n','m']

for i in str(fjalia):

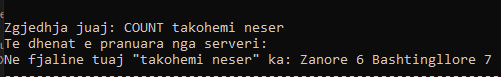
if i in ZANORET:

zan += 1

elif i in BASHT:

basht+= 1

return "Zanore " + str(zan) + " Bashtingllore " + str(basht)



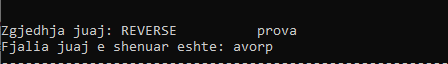
**4.4. REVERSE**

Gjithashtu edhe kjo metodë thirret duke shenuaeë REVERSE dhe një fjali, kjo metodë kthen fjalinë e shtypur ne tekst . Hapësirat në fillim dhe në fund te fjalisë nuk kthehen.

def REVERSE(fjalia):

fjalia1 = fjalia[::-1].strip()

return fjalia1



**4.5. PALINDROME**

Si metodat e lartëcekura edhe kjo metodë thirret njejtë, kjo metodë kthen nëse fjalia juaj është palindromë ose jo. Ne mbrendësi të saj ajo krahason fjalin me fjalinë revers dhe qofse të dyja janë të barabarta rezultati del që është palindrom.

def PALINDROME(fjalia):

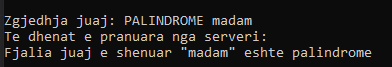
fjalia1 = fjalia.replace(" ","")

if(fjalia1[::1] == fjalia1[::-1]):

return str("palindrome")

else:

return str("jopalindrome")



**4.6. TIME**

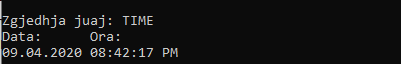
Kjo metodë është pa parametra dhe për këtë metodë duhet importuar librarinë *datetime* dhe përdoret funksioni i gatshëm *datetime.datetime.noë()* i cili kthen vitin, muajin, diten, orën, minutat, sekondat, pastaj e kemi formatuar sipas kërkesës në detyrë DD.MM.YY H:M:S perms *koha.strftime("%d.%m.%Y %H:%M:%S")*. Pra përcakton kohen aktuale ne server dhe e dërgon atë tek klient në format të lexueshëm për njerzit.

def TIME():

koha = datetime.datetime.noë()

rezultati = "Data: Ora: \n" + koha.strftime("%d.%m.%Y %I:%M:%S %p")

return str(rezultati)



**4.7. GAME**

Edhe për këtë metodë për ta implementuar duhet të importohet libraria *random* dhe përmes funksionit të gatshëm *random.randint(1,35*) gjenerojmë numra të randomtë. Pra kjo metodë kthen 5 numra nga rangu [1, 35]. Numrat jane gjithashtu të sortuar.

def GAME():

numrat = []

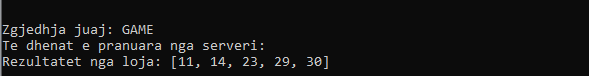
for x in range(5):

numrat.append(random.randint(1,35))

numrat.sort()

numratstring = str(numrat)

return numratstring



**4.8. GCF**

Për këtë metodë duhet të importohet libraria *math* dhe përmes metodës së gatshmë math.gcd() kthehet faktori më I madh I përbashkët ndërmjet dy numrave të plotë.

def GCF(nr1, nr2):

try:

x = int(nr1)

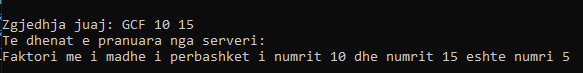
y = int(nr2)

except:

return "Error"

gcd = math.gcd(x,y)

return str(gcd)



**4.9 CONVERT**

Për të thirrur këtë metodë duhet te jepen tre parametra CONVERT, pastaj se varësisht qfarë dëshirojmë të konvertojmë, CmToFeet, FeetToCm, KmToMiles dhe MilesToKm, dhe vlera qe deshirjmë te konvertojme. Rezultati që kthehet duhet të jetë me dy shifra pas vleres decimale.

def CONVERT(opsioni,vleraHyrese):

try:

vlera = float(vleraHyrese)

except:

return "Error"

if opsioni=="CmToFeet":

rezultati = (vlera/30.40)

elif opsioni=="FeetToCm":

rezultati = (vlera\*30.40)

elif opsioni=="KmToMiles":

rezultati = (vlera/1.60)

elif opsioni=="MileToKm":

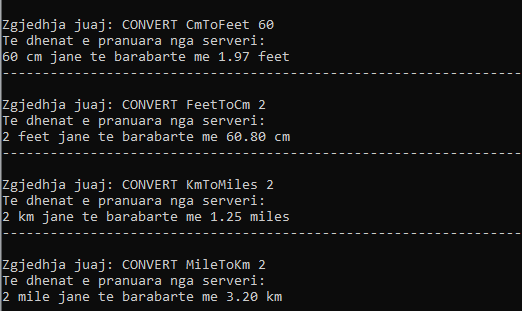
rezultati = (vlera\*1.60)

else:

return "Error"

rezultati = "{:.2f}".format(rezultati)

return str(rezultati)



**4.10. POW**

Gjithashtu për këtë metodë duhet të importohet libraria *math* (tani më veqse e kemi të inportuar pasi na është nevojitur tek GCF) dhe përmes metodës së gatshmëm math.pow() kthehet numri I parë në fuqi të numrit të dytë.

def POW(nr1, nr2):

try:

x = int(nr1)

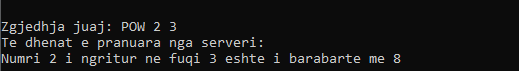
y = int(nr2)

except:

return "Error"

powr = pow(x,y)

return str(powr)



**4.11. TABELA**

Përmes kësaj metodë gjenerohen tabelat e një makine, ju duhet të tregoni nese makina juaj është private apo për institucion dhe nga cili qytet është, qe metoda teë gjenerojë një tabelë random.

def TABELA(opsioni,vlera):

if vlera=="PRISHTINA":

qyteti = "01"

elif vlera=="MITROVICA":

qyteti = "02"

elif vlera=="PEJA":

qyteti = "03"

elif vlera=="PRIZRENI":

qyteti = "04"

elif vlera=="FERIZAJI":

qyteti = "05"

elif vlera=="GJILANI":

qyteti = "06"

elif vlera=="GJAKOVA":

qyteti = "07"

else:

return "Error"

randomnumber = random.randint(100,999)

alfabeti = 'QERTYUIOPASDFGHJKLZXCVBNM'

shkronjat = ''.join(random.sample(alfabeti,2))

if opsioni=="PRIVAT":

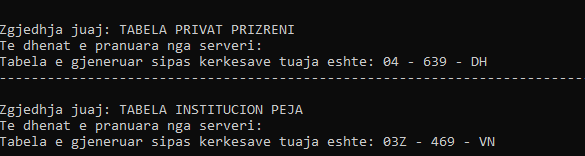
return str(qyteti + " - " + str(randomnumber) + " - " + shkronjat)

elif opsioni=="INSTITUCION":

return str(qyteti + "Z" + " - " + str(randomnumber) + " - " + shkronjat)

else:

return "Error"



**5. VEGLAT E PЁRDORURA**

Vegla: Microsoft Visual Studio(2019)

Gjuha Programuese: Python

Sistemi operativ: Windows 10

**6. PЁRFUNDIMI**

Nga ato që pame , u arritën shumë përfundime lidhur me projketin dhe TCP e UDP sockets , punimi I këtij projekti na ofroj të njihemi më shumë me “parapvijat “ që ndodhin pas cdo përdoruesi që përdor internetin . Gjatë shqyrtimit të projektit përkatesisht definimit të funksionev është marrë vetëm nga një zgjidhje e mundshme për secilin nga funksionet , ofrohen edhe shumë zgjidhje të tjera mirepo normalisht është marrë ajo që për ne si student duket më e përshtatshme .Përvec të tjerash u mësuan përparësite që kanë dy protokolet. TCP ofron Reliable Data Transfer Services- proceset e komunikimit mund të mbështetën në TCP për të ofruar të gjitha të dhënat e dërguara pa gabime dhe nën renditjen e duhur . UDP siguron shërbime minimale , e cila është edhe connectionless që do të thotë që nuk ka handshaking përpara se proceset të komunikojnë , siguron një kanal të komunikimit jo të besueshëm , që do të thotë që kur dërgohet një mesazh në socketin e UDP-së , UDP nuk sihuron asnjë garanci që mesazhi do te arrijë ne procesin e pranimit , gjëra që TCP-ja i bënë .Pra UDP nuk siguron connection setup , reliability, floë control, congestion control, timing , throughput garantee, security. E vetmja përparësi e tij është që është më e shpejtë.

**7. BIBLOGRAFIA**

Më poshtë keni disa burimet që janë përdorur për këtë projektë.

<https://www.geeksforgeeks.org/python-programming-language/>

<http://zetcode.com/python/socket/>

<https://www.tutorialspoint.com/index.htm>

<https://stackoverflow.com/questions/44662178/client-server-communication-in-python>

<https://pymotw.com/2/socket/tcp.html>

<https://www.journaldev.com/15906/python-socket-programming-server-client>

<https://notimi.uni-pr.edu/DokumentetProfesori/09f5a9b4-2bd6-4e87-881a-f85550aff776.pdf>

<https://www.youtube.com/watch?v=Lbfe3-v7yE0>

<https://www.youtube.com/playlist?list=PLhTjy8cBISErYuLZUvVOYsR1giva2payF>