**UNIVERSITETI I PRISHTINËS**

**Fakulteti i Inxhinierisë Elektrike dhe Kompjuterike**

**Departamenti i Kompjuterikës**



**Projekti 1**

**Lënda: Rrjetat kompjuterike**

**Profesor:** Blerim Rexha **Studentja: Vesa Ademi**  **Asistent:**  Haxhi Lajqi **Nr.Index:160714100042**

**Prishtinë, Prill 2018**

**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 varësisht se cfarë operacioni deshiron të kryej klienti të zgjedh numrin e tij dhe në baze të kësaj, kërkesa të dergohet në server , ku aty më pastaj janë kryer llogaritjet dhe është siguruar përgjigjja e cila përsëri është kthyer tek klienti .

Përfitimet nga ky projekt kanë qenë mjaftë të mëdha për arsye së ne cdo ditë jemi të rrethuara nga këto veprime , lidhje dhe komunikime, mirëpo në mënyrën se si janë arritur këto nuk kemi pasur njohuri sa kemi tani , andaj edhe pse nuk ishte ndonjë projekt I gjerë ishte shumë fitimprurëse për dijën tone .

**PERSHKRIMI**

**Socket Programming** paraqet një mënyrë për të lidhur dy nyje (nodes) në një rrjet për të komunikuar me njëri tjetrin.Ky programim fillon duke importuar librarian socket dhe duke bërë një socket të thjeshtë.

import socket

Njohuritë lidhur me sockets dhe protokolet ndihmojnë në një masë të madhe në realizimin e projektit. Shumë herë është përmendur që socket paraqesin një llojë porte në rrjet , andaj rëndësia e tyre është mjaftë e madhe. Nje socket ka numrin e portit(Port Number) dhe Ip adresen (IP Address) . Mund të përdorën protokolet TCP dhe UDP. E para ofron një kanal të komunikimit të sigurt , ndërsa ajo e dyta nuk ofron siguri në këtë kanal. TCP ka shumë arsye pse duhet të përdoret , mirëpo UDP ofron një shpejtësi më të madhe.

**TCP-KLIENTI**

E dimë që Ip adresa e identifikon një host mirëpo për të lejuar komunikimin nuk mjafton vetëm kjo për arsye se Sistemi Operativ pyet se kush duhet ta marre atë mesazh, pra duhet te dihet sakt edhe procesi , duke e ditur Port Number ose Socket Number.

Në bazë të kërkesës në projekt është përdorur “localhost” si host server dhe porti 11000 si port I nënkuptushëm.

Në pjesën e kodit fillimisht janë përfshirë

s=socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) paraqet socket instance

AF\_\* dhe SOCK\_\* paraqesin familjen(AddressFamily) dhe tipin e Socketit(SocketKind) .

AF\_INET paraqet opcionin më të sigurtë në internet dhe I referohet adresës së familjës ipv4. SOCK\_STREAM na lë të kuptohet që është një TCP socket.

Përmes input() është bërë e mundshme që klienti te ndryshoj emrin e serverit dhe portin sipas nevojës.

Shfrytëzuesi është në gjendje të thërrash kërkesën IPPADDR (duke shtypur 1), PORTNR (duke shtypur 2) ,ZANORE (duke shtypur 3) , PRINTO (duke shtypur 4), HOST (duke shtypur 5) , TIME (duke shtypur 6) , LOJA ( duke shtypur 7) , FIBONACCI (duke shtypur 8) , KOVERTO ( duke shtypur 9), INCHCM(duke shtypur 10) , BMIWEIGHT (duke shtypur 11).

Shfytëzuesi është në gjendje të përfundoj punën përmes:

var=input(" Deshironi te perfundoni, shtypni PO :")

if var=='PO':

break

Pas cdo kërkese të tij ai pyetet se a dëshiron të perfundon dhe nëse shtyp PO , programi përfundon.

sendall() – shfytëzohet për të dërguar të dhënat nga klienti në server dhe anasjelltas në rastin kur është TCP

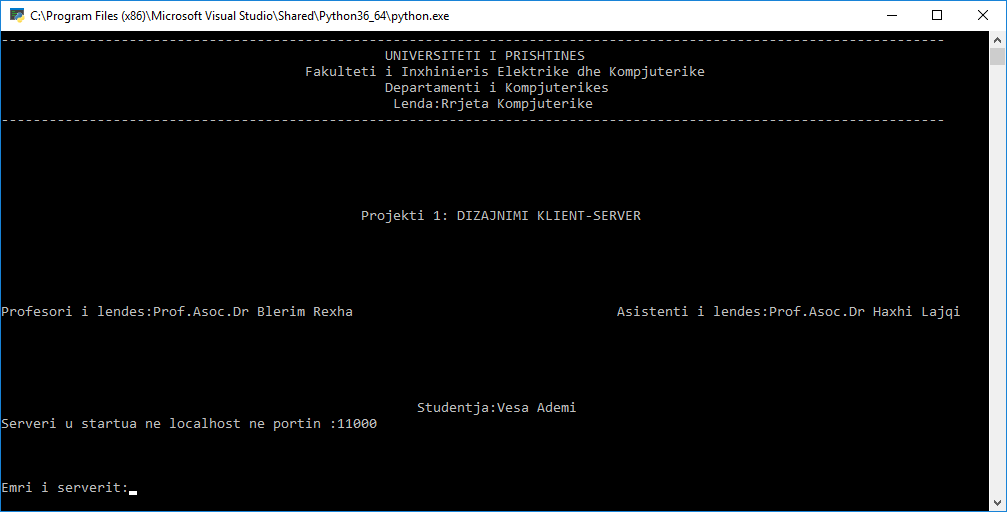
Në secilin rast pranohet një mesazh jo më I gjatë se 128 karaktere , në pjesën e kodit kjo është shprehur përmes :

data=s.recv(128)

Klientit I është ofruar mundësia që në bazë të numrit që ka operacioni që ai dëshiron ta kryej të shtyp dhe ajo kërkesë e tij të dërgohet te klienti , kjo nga degëzimet if , elif dhe else , për secilin funksion është vepruar njësoj. Jane përdor edhe encode dhe decode() për arsye të shënderrimeve që janë të detyrueshme nëse dëshirojmë që të dhënat tona të dërgohen apo pranohen , në projekt shendrime të tilla ka mjaftë shumë për shkak të numrit të funskionet dhe veprimeve që duhet të kryhen , p.sh për shëndrrime nga byte ne string apo anasjelltas etj. Pra encode dhe decode janë kërkesa per enkodim dhe dekodim.

Pasiqë klienti në rastin kur kërkon IP Addressën e tij dhe Port Number nuk dërgon dicka tjetër përvec numrit të kërkesës janë shprehur thjeshtë në anën e klientit, njësoj sikurse TIME(KOHA) ,HOST(EMRI I KLIENTIT) dhe LOJA.Në rastin 3 pra funksioni për ZANORE në klient është dhënë input(), për shkak që klienti duhet t’i dërgoj serverit fjalin ( e dergon permes s.sendall(str.encode(var))) apo fjalën në të cilën ai dëshiron të gjejë se sa është numri i zanoreve , pastaj në server përmes funksionit llogaritet se sa është numri i zanorev dhe kthehet tek klienti i cili i pranon permes data=s.recv(128)) ,konvertohen e shtypet numri i zanorev përmes print('Numrim i zanoreve ne fjali:',data.decode()) . Në mënyrë të njejtë veprohet edhe për PRINTO dhe tek FIBONACCI vetëm se tek kjo e fundit shfytëzuesi jep një numër. Tek KONVERTO klienti jep përmes var=input("Jepni numrin :") numrin dhe nga një inputi tjetër ai zgjedh llojin e konvertimit cilin dëshiron të marr var1=input('Zgjidhni njeren nga konvertimet: C-to-K , C-to-F , K-to-F , K-to-C , F-to-C , F-to-K , pound-to-kg , kg-to-pound: ') . Që të dyja dërgohen ashtu sic janë përshkruar edhe më lartë dhe pranohen. Njëlloj veprohet edhe për rastin e funksioneve shtesë INCH dhe BMIWEIGHT , tek e para shfytëzuesi jep numrin dhe llojin e kovertimin , tek e dyta jepen pesha dhe gjatësia në metër.

Po ashtu është cekur që nëse server nuk mund ta pranoj atë kerkesë në fund është perfshirë edhe kjo permes print("Kerkesa nuk mund te pranohet nga serveri!!!").



**TCP SERVERI**

Në pjesën e kodit fillimisht janë importuar disa librari dhe ato jane:

from socket import\*

import \_thread

import time

from socket import gethostname;

import random

import datetime

import math

Për të siguruar kërkesën e cekur në projekt që të sigurohet lidhja dhe komunikimi jo vetëm në një paisje dhe për të përdorur esencën kryesore të sockets është përfshirë libraria import\_thread . Libraria import time për funksionin TIME(KOHA) në mënyrë që të realizohet kërkesa që të merret koha aktuale e serverit, from socket import gethostname është shfrytëzuar në rastin kur është kerkuar emri I klientit , import random në rastin LOJA pra është importuar libraria random , import datetime në mënyrë që të merret edhe data , kjo është shfytezuar tek rasti TIME (KOHA ).

serverPort=11000 dhe serverSocket = socket(AF\_INET, SOCK\_STREAM) të cilat u pershkruan edhe më lartë.

serverSocket.bind(('',serverPort)) është një metodë që lidh adresën me socket dhe i ka dy parametra (hostname, port number pair) , dhe pasiqe përfshihen edhe listen() dhe accept() , dëgjon për kërkesa që dërgohen që të lidhen me adresën e vecant(particular address)

serverSocket.listen(5) tregon që është në modin e dëgjimit-TCP LISTENER

.accept() pranon “pasivisht” lidhjen e klientit TCP, duke pritur derisa lidhja te arrin .

connectionSocket,clientAddress=serverSocket.accept();

numri=numri.decode()

print('Operacioni i zgjedhur nga klienti: '+numri)

Në këtë pjesë kryhet konvertimi përmes decode() ,pranohet numri të cilin klienti e jep për të kryer më pastaj operacionin .

\_thread.start\_new\_thread(clientThread,(serverSocket,clientAddress,numri)) , Ndersa këtu sigurohet lidhja përmes threads , dhe startohet një thread i ri.

Një funksion në Python deklarohet me def , andaj në server të gjitha kërkesat që janë perfshirë në projekt janë bërë me funksione duke I emertuar ato dhe varesisht nga rasti , janë marrë me parametra apo pa parametra.

def clientThread(serverSocket, clientAddress, numri):

paraqet funksionin i cili ka tre parametra dhe në vete ka të gjitha funksionet e kërkuara .

Më poshtë shihen shqyrtimi i funksioneve.

def IPKLIENTI():

print("Ip Adresa e Klientit:",clientAddress[0])

connectionSocket.send(str(clientAddress[0]).encode("ASCII"))

if numri=='1':

IPKLIENTI()

IP Address-a e klientit është marr përmes clientAddress[0] për arsye se sic shihet edhe nga më lartë merret Ip-ja e klientit , pastaj kjo dergohet permes connectionSocket.send() , pjesa brenda kllapav paraqet enkodim dhe str() paraqet një nga shumë metodat e stringjev në python , encode() metoda kthen një version të enkoduar të strigut, Pythom by default encodimin e ka ASCII.

Shiko 1,2 më poshtë!

def PORTIKLIENTIT():

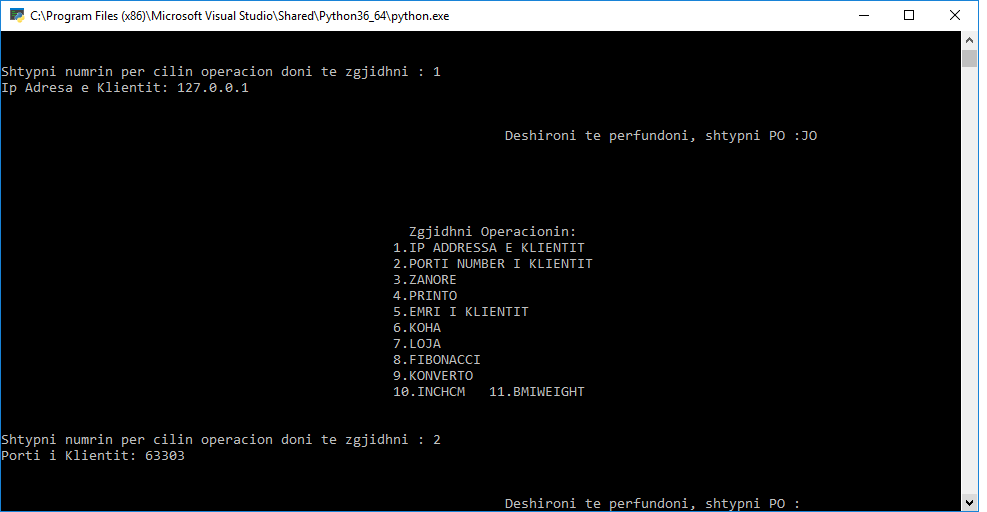
print("Port Number i Klientit :" ,str(clientAddress[1]))

connectionSocket.send(str(clientAddress[1]).encode("ASCII"))

elif numri=='2':

PORTIKLIENTIT()

Numri i portit te klietit mirret nga clientAddress[1] , dhe njësoj sikurse në rastin e IPKLIENTI () enkodohet dhe dergohet tek klienti .

**1,2**

def ZANORE(s):

i=0

counter=0

while i<len(s):

if s[i] in 'aeiouAEIOU':

counter=counter+1

i=i+1

print(counter)

connectionSocket.send(str.encode(str(counter)))

Fillimisht teksti i cili nga klienti është dhënë dërgohet tek serveri dhe pranohet nga ai permes:

elif numri=='3':

data1=connectionSocket.recv(128) 🡨

data1=data1.decode()

print(data1)

ZANORE(data1)

decodohet dhe pastaj shqyrtohet ne funksionin ZANORE , ku sic shihet është një funksion me nje parametër pasiqe e pranon tekstin. Ne while shihet i< len(s) , ku len(s) paraqet gjatësin e tekstit dhe themi se perderisa i është më e vogel se gjatësia e tekstit shqyrto pjesën e kodit të parë më poshtë ku permes if shqyrtohet ajo që nëse shkronja në tekstin e dhënë është ndonjëra nga zanoret shtohet counter për 1 , varësisht se sa është gjatësia e tekstit i-ja rritet dhe kështu shqyrtohen të gjitha shkronjat në tekst, dhe arrihet të grumbullohet numri i tërësishem i zanoreve ( nr i zanorev –counter) , pastaj dergohët permes connectionSocket.send(enkodimet e nevojshme të sqaruara edhe më lartë, përmes së cilave nr i zanoreve bëhet i kuptushëm edhe për makinën dhe dërgohen tek klienti) .

Shiko 3,4 më poshtë

def PRINTO():

data=connectionSocket.recv(128)

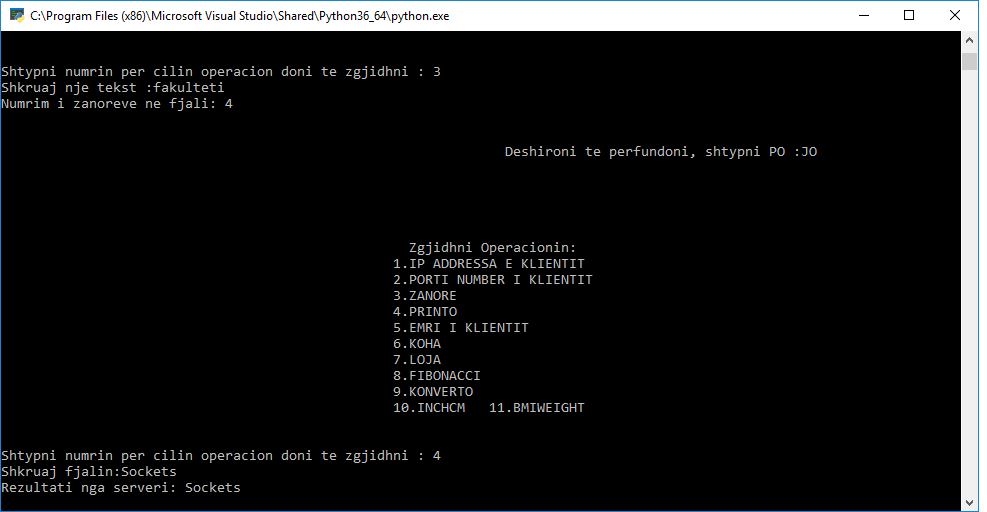
print(data)

connectionSocket.send(data)

elif numri=='4':

PRINTO()

Funksioni PRINTO nuk ka parametra dhe është një funksion i thjeshtë tek i cili pranohet teksti i dhënë nga shfrytëzuesi dhe pastaj i njejti dërgohet tek klienti.



**3,4**

def EMRIKLIENTIT():

emri=''

if gethostname()==None :

emri='\Emri i klientit nuk dihet !\n'

else:

emri='Emri i klientit eshte:'+gethostname()

print(emri)

connectionSocket.send(str.encode(emri))

elif numri=='5':

EMRIKLIENTIT()

Funksioni gethostname() kthen një string që përmban emrin e hostit të makinës ku interpreteri i Pythonit është duke u ekzekutuar. Keshtu dergohet pastaj tek klienti. Shiko 5,6

def KOHA():

currentTime=time.ctime(time.time())

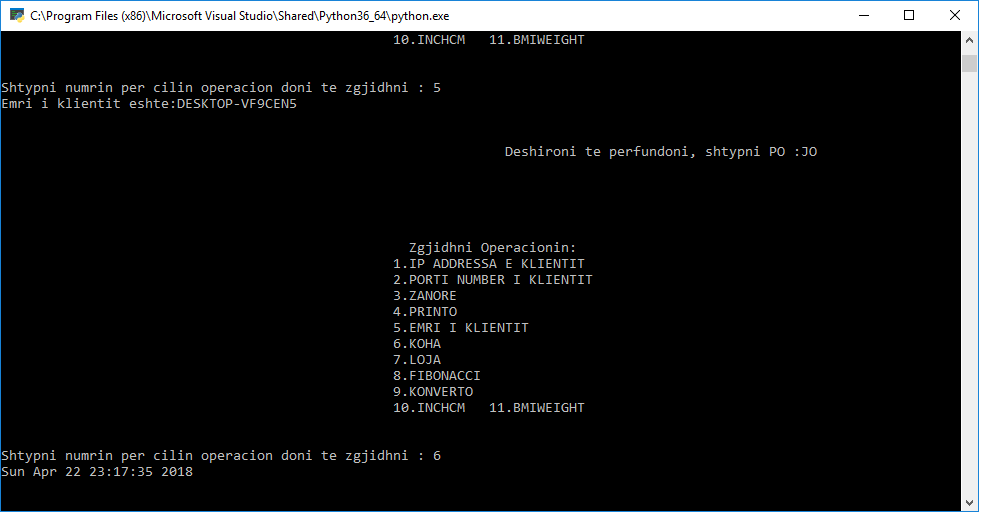
print(str(currentTime))

connectionSocket.send(str.encode(str(currentTime)))

elif numri=='6':

KOHA()

Koha reale është marrë përmes funksioneve të gatshme që I ofron python , duke pasur parasysh librarian e importuar arrihet që të kemi mundësine e shfytëzimit të tyre dhe pastaj sic shihet kjo kohë dërgohet tek shfytëzuesi,



**5,6**

def LOJA():

m=[]

for i in range (20):

m.append(random.randint(1,100))

print(m)

connectionSocket.send(str.encode(str(m)))

elif numri=='7':

LOJA()

Me [] deklarohet një listë e zbrazët në python , ndërsa me append kemi shtuar elementere të reja në listë , përmes libraris import random janë siguruar 20 numra të rastit dhe pastaj janë dërguar tek klienti.

Shiko 7,8.

def FIBONACCI(n):

first,second =0,1

for i in range (0,n):

first,second=second,first+second

print(first)

connectionSocket.send(str.encode(str(first)))

elif numri=='8':

data2=connectionSocket.recv(128)

data2=data2.decode()

print(data2)

FIBONACCI(int(data2))

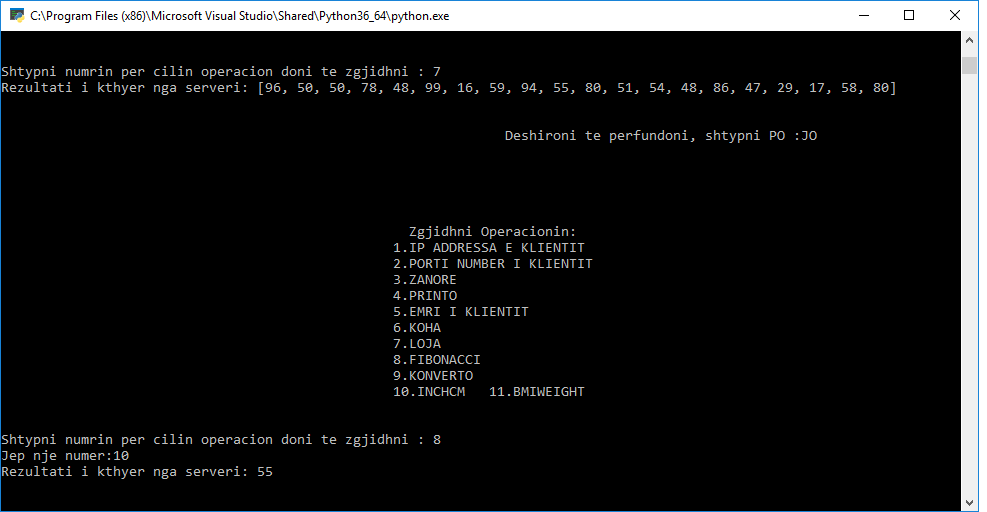
Sic është përshkruar edhe më lartë fillimisht pranohet në serveri numri të cilin shrytëzuesi e ka shtypur , pastaj përmes funksionit mirret numri fibonacci i numrit të kërkuar. Kështu kemi marrë dhe kemi 0 deri ne n permes një for loop, the pastaj ajo që gjendet brenda for loop mund të shkruhet edhe në formën

a=first

first=second

second+=a

kështu arrihet që të sigurohet numri dhe të dërgohet tek klienti!



**7,8**

def KONVERTO(temperatura,x):

if x=="C-to-K":

temperatura=str(float (float(temperatura)+273.00))

connectionSocket.send(str.encode(str(temperatura)))

elif x=="C-to-F":

temperatura= str(1.8 \* float(float(temperatura)+32))

connectionSocket.send(str.encode(str(temperatura)))

elif x=="K-to-F":

temperatura=str(1.8 \* float(float(temperatura)-273)+32)

connectionSocket.send(str.encode(str(temperatura)))

elif x=="K-to-C":

temperatura=str(float(float(temperatura)-273))

connectionSocket.send(str.encode(str(temperatura)))

elif x=="F-to-C":

temperatura=str(5/9\*float(float(temperatura))-32)

connectionSocket.send(str.encode(str(temperatura)))

elif x=="F-to-K":

temperatura=str(5/9\*float(float(temperatura)-32)+273)

connectionSocket.send(str.encode(str(temperatura)))

elif x=="pound-to-kg":

c=str(float(float(temperatura))/2.20462262)

connectionSocket.send(str.encode(str(c)))

elif x=="kg-to-pound":

c=str(float(float(temperatura))\*2.20462262)

print(c)

connectionSocket.send(str.encode(str(c)))

else:

print('Kerkesa juaj nuk mund te pranohet !')

connectionSocket.send(str.encode('Kerkesa juaj nuk mund te pranohet'))

elif numri=='9':

data2=connectionSocket.recv(128)

data2=data2.decode()

print(data2)

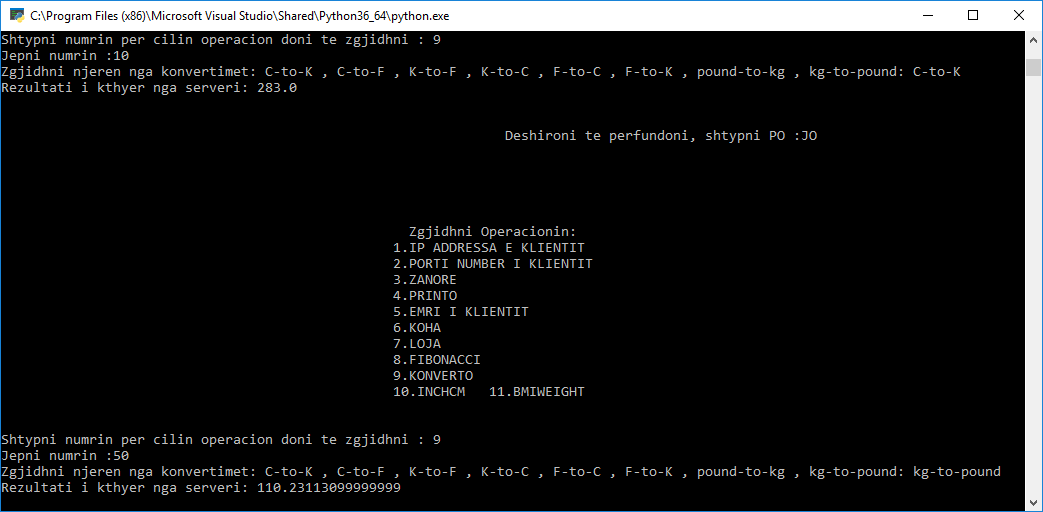
data3=connectionSocket.recv(128)

data3=data3.decode()

print(data3)

KONVERTO(data2,data3)

Klientit fillimisht i kërkohet që të shtyp numrin e pastaj llojin e konvertimit që ai dëshiron ta bëjë, pastaj në funksionin Konverto(funksion me dy parametra) varësisht nga lloji i konvertimit llogaritet dhe arrihet rezultati i deshiruar , është zgjedhur tipi float në rastë se klienti dëshiron të jap numër me presje dhjetore , andaj edhe rezultati mund të arrihet si i tille, pastaj ai rezultat dërgohet tek klienti, ashtu sic shihet më poshtë. (ngjashmeri me funksionet e meposhte , andaj ato janë shpjeguar me detajisht)

****

def INCH(numrii,x):

a=2.54

if x=="INCH-TO-CM":

numrii=str(float(float(numrii)\*2.54))

print(numrii)

connectionSocket.send(str.encode(numrii))

elif x=="CM-TO-INCH":

numrii=str(float(float(numrii)/a))

print(numrii)

connectionSocket.send(str.encode(numrii))

elif numri=='10':

data2=connectionSocket.recv(128)

data2=data2.decode()

print(data2)

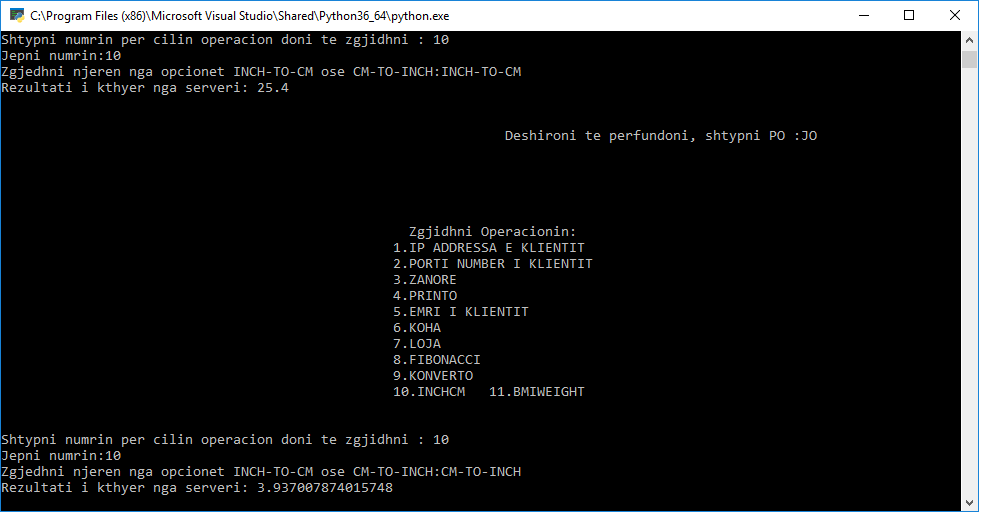
data3=connectionSocket.recv(128)

data3=data3.decode()

print(data3)

INCH(data2,data3)

Funksioni INCH njësoj sikurse i lartëcekuri është funksion me dy parametra , në të cilin shfytëzuesi duhet të jap numrin dhe llojin e konvertimit varësisht se cilin ai dëshiron , pastaj pranohen nga serveri dhe pas decodimeve , pranohen edhe nga funksioni në të cilin pastaj kryhen llogaritjet numrii paraqet numrin i cili është marrë kurse x paraqet llojin e konvertimit , pastaj përmes degëzimeve if dhe elif është arritur qe varesisht nga rasti te llogaritet numrri(rezultati) i shprehur në trajtë matematikore si numrii\*2.54 në qoftëse bëhet konvertimi nga inch në centimetër dhe /2.54 nëse konvertimi nga centimetër në inch . Numri që jepet nga shfrytëzuesi mund të jetë numër me presje dhjetore andaj është ruajtur edhe rezultati si i tillë dhe në fund që të bëhet enkodimi është shfytëzuar metoda str() , e pastaj përmes connectionSocket.send() eshte derguar duke bërë enkodimin str.encode(numrii).



def BMIWEIGHT(pesha,gjatesia):

bmi=float(float(pesha)/(float(gjatesia)\*float(gjatesia)))

if bmi <=18.5:

bmi=str(bmi)

c='Pesha juaj ne BMI , ju keni peshe me te vogel se pesha normale:'+bmi

connectionSocket.send(str.encode(str(c)))

elif bmi >=18.5 and bmi <25:

bmi=str(bmi)

c='Pesha juaj ne BMI, ju keni peshe normale:'+bmi

connectionSocket.send(str.encode(str(c)))

elif bmi >25 and bmi<30:

bmi=str(bmi)

c='Ju keni peshe , me te larte se pesha normale:'+bmi

connectionSocket.send(str.encode(str(c)))

else:

c='Keni peshe shume te ulet ose shume te larte!!!'

connectionSocket.send(str.encode(str(c)))

print(str(c))

elif numri=='11':

data2=connectionSocket.recv(128)

data2=data2.decode()

print(data2)

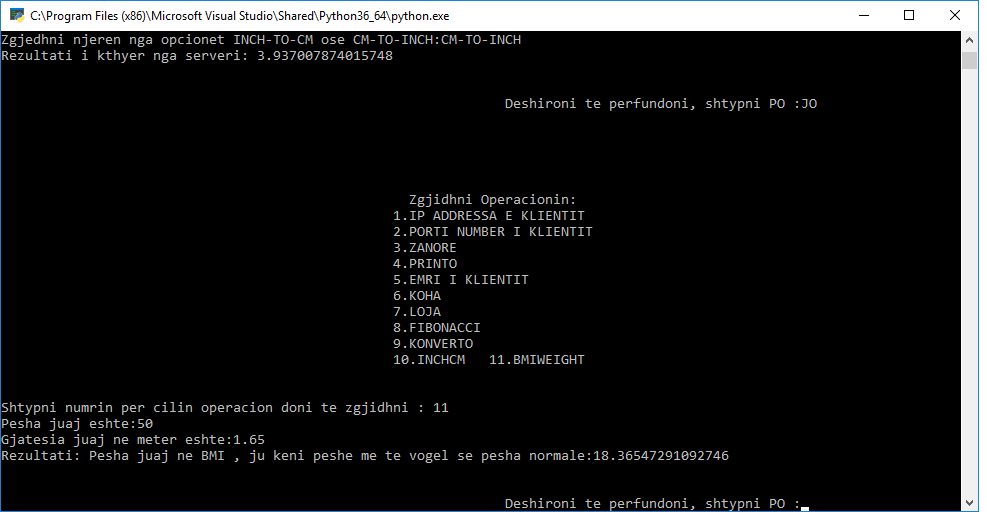
data3=connectionSocket.recv(128)

data3=data3.decode()

print(data3)

BMIWEIGHT(data2,data3)

Funksioni BMIWEIGHT është funksion me dy parametra dhe i ngjashem me funksionet e mësipermë , mënyra se si llogaritet është caktuar nga BMI standards . Fillimisht nga shfytëzuesi kërkohet që të jap peshën dhe gjatësinë e tij të shprehur në metër , pastaj permes formulës përkatese arrihet deri tek rezultati dhe nga degëzimet që janë bërë mund të kalkulohet ajo se shfytëzuesi a ka peshë normale apo jo se bashku me një mesazh të tillë.



dhe në fund mbyllet socketi :

connectionSocket.close()

**UDP KLIENTI**

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

Fillimisht sikurse edhe tek TCP klienti janë perfshirë libraritë e nevojshme

import socket

Te cilat u shtjelluan më në detaje edhe në pjesën me lartë.

s=socket(AF\_INET,SOCK\_DGRAM)

SOCK\_DGRAM nënkupton që është një UDP socket .

Tek pjesa e klientit kodi është i thjeshtë me print() janë dhënë disa sqarime së mënyrën se si duhet të trajtohet kërkesa , dhe pastaj permes nr=input(“kerkesa:”) është bërë e mundur që shfytëzuesi të jap numrin për cilin operacion dëshiron ta kryej. Me degëzime janë lënë disa mesazhe që nëse shfytëzuesi p.sh për operacionin numër 3 nuk shënon në trajtën e duhur kërkesen (3 fjalia) atëherë jepet mesazhi Shkruani 3 fjalia , e pastaj shfytëzuesi mund ta shqyrton kërkesën në trajtën e duhur.

try:

s.send(nr.encode("ASCII")

except:

print("Kerkesa nuk mund te pranohet nga serveri!")

Përmes kësaj pjese të kodit në try është përfshirë s.send() , nga e cila mund të dërgohet kërkesa nga klienti në server. Dhe brenda kllapav encodimi ASCII. E nese lidhja nuk mund të sigurohet përkatësisht e dhëna nuk mund të dërgohet lë mesazhin e parë më lartë!

Rezultati=s.recv(128)

try:

print("Rezultati:"+Rezultati.decode("ASCII"))

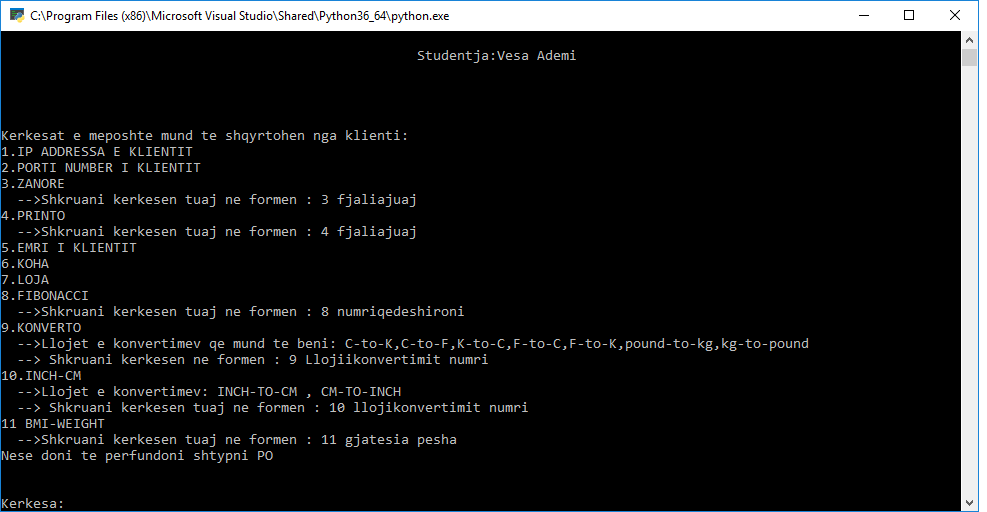
except Exception:

print("Rezultati:"+Rezultati.decode("utf8"))

Të dhënat që pranohen përmes recv , ashtu sic është përshkruar në kërkesë. Pastaj decode-hen në mënyrat e shprehura më larte , dhe të cilat janë cekur edhe në TCP.

s.close() – mbylle socketin

Pamja e parë e UDP Klientit shihet më poshtë



**UDP SERVERI**

Në fillim janë importuar librarit e nevojshme

from socket import \*

import time

import re

from socket import gethostname;

import random

import datetime

import math

import sys

import string

import \_thread

import re- importon librarin e Regular Expressions

import string- importon librarin string , ndersa te tjerat janë përshkruar edhe tek TCP Serveri.

Në mënyrë të ngjashme janë përdorur threads, përkatesisht është startuar një thread i ri pastaj funksioni është deklaruar i njejtë sikurse edhe tek TCP-ja.

Kërkesa nga klienti mirret , pranohet tek next dhe pastaj dekodohet dhe mund të vazhdojë në pjesën tjetër të kodit , tek funksioni .

next, Clientaddress=serverSocket.recvfrom(128)

print("Kerkesa:"+next.decode("ASCII"))

Funksionet e para pra për të siguruar IP adresen Port number dhe Numrin e zanorev janë njesoj sikurse tek TCP Serveri , vetem se tek kjo e fundit pëfshihet edhe:

elif re.match('3 .\*', next.decode("ASCII"),):

value=str(next) 🡪 next paraqet të dhënën që ka ardhur

ZANORE(value)

Dhe këtu dhe në pjesët e tjera të kodit shihet aryeja se pse janë përdour RegExp , një nga arsyet me të fora është që nuk mundësohet ajo qe na ofron TCP-ja pra në këtë rast shfytzuesi mund të jepte ndarazi kërkesën(numrin e kërkesës), dhe ndarazi fjalën apo fjalin . Re.match() e kerkon nje match vetem ne fillim te nje stringu(\*- universal quantifier), ndersa pervec kesaj ka edhe lloje te tjera por te cilat nuk jane perfshir ne pjesen e kodit.

Funksioni PRINTO nuk mund të trajtohet në mënyrë të thjeshtë sic është trajtuar në TCP andaj është shprehur ndryshe ( vlen aryeja e njejtë e cekur edhe më lartë).

def PRINTO(tex):

splitArray = tex

i=1

teksti = ""

s = len(next)

if(s <= 128):

while i<len(splitArray):

teksti = teksti + splitArray[i] + " "

i+=1

MESSAGE = teksti

serverSocket.sendto(MESSAGE.encode("ASCII"),Clientaddress)

elif re.match('4 .\*',next.decode(),):

\_string = str(next)

value = \_string.split(" ")

PRINTO(value)

Ky funksion ka një parameter dhe shqyrtohet si vijon njësoj si edhe rastet e tjera të shqyrtuara në TCP!

Funksionet LOJA HOST dhe KOHA janë shqytuar po ashtu në mënyrë të njejtë . Ndersa tek 4 funksionet e fundit ndërron vetëm pjesa qe shihet poshtë funksionit pra:

elif re.match('8 .\*', next.decode("ASCII"),):

request = next.decode("ASCII").split(" ")

if request[1].isdigit():

FIBONACCI(int(request[1]))

elif re.match('9 .\*[A-Za-z] .\*[0-9.]',next.decode("ASCII"),):

s = next.decode("ASCII").split(" ")

KONVERTO(s)

elif re.match('10 .\*[A-Za-z] .\*[0-9.]',next.decode("ASCII"),):

c = next.decode("ASCII").split(" ")

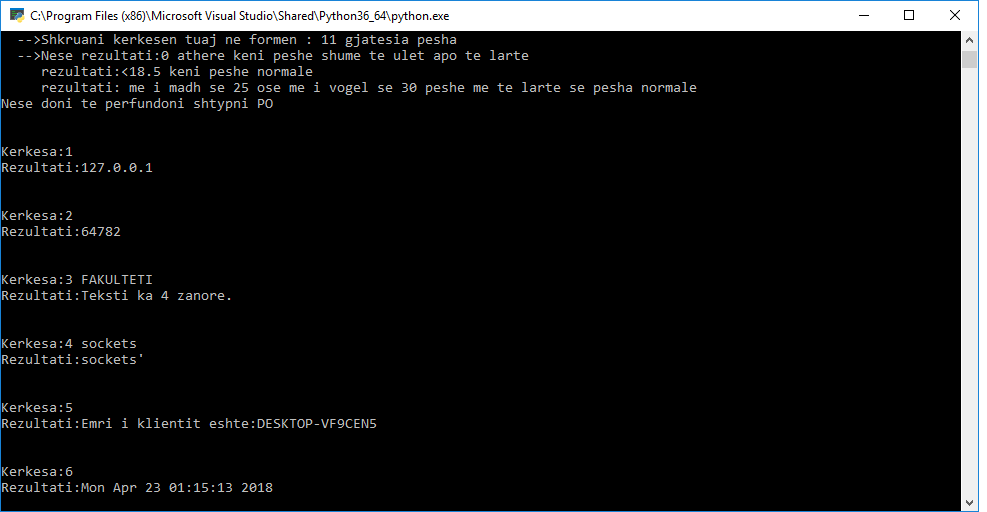
INCH(c)

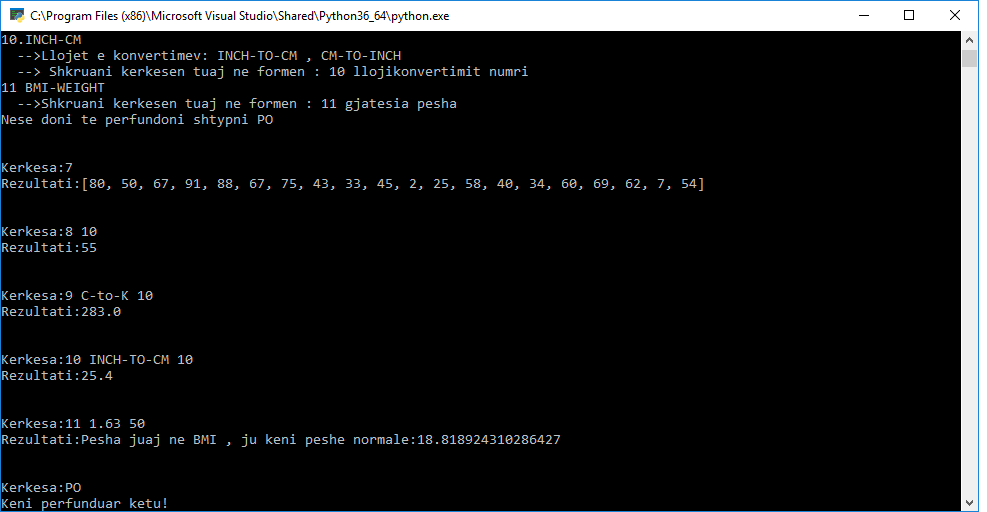
elif re.match('11 .\*[0-9.] .\*[0-9.]',next.decode("ASCII"),):

a = next.decode("ASCII").split(" ")

BMI(a)

Ku Fibonacci trajtuar njësoj sikurse funksioni ZANORE. Ndërsa ne rastet e mëposthme për arsye se është kërkuar që parametri i parë të jetë lloji i konvertimit (shkronja) , ndërsa rasti i dytë numri janë përdorur RegExp që shihen më lartë . Në rastin e fundit shfytëzuesi duhet të ketë dhënë peshën dhe gjatësite , janë përdorur sic shihen më lartë.





**PERFUNDIMET**

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, flow control, congestion control, timing , throughput garantee, security. E vetmja përparësi e tij është që është më e shpejtë.

**BIBLOGRAFIA**

N kete projekt jane shfrytuar burime te shumta per qellime te ndryshme te plotesimit te detyre qe ka projekti.

Computer Networks and Internets by Douglas E.COMER

Computer Networking a Top-Down Approach by Kursore and Ross

<https://www.pythoncentral.io/>

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

<https://www.codecademy.com/en/tracks/python>

<https://www.python.org/doc/>

<https://www.youtube.com/watch?v=F-_PKUUM-qY>

<https://docs.python.org/3/library/re.html>

<https://docs.python.org/3/howto/regex.html>

<https://www.tutorialspoint.com/python/python_reg_expressions.htm>

<https://www.khanacademy.org/computing/computer-programming>