

Universiteti i Prishtinës “Hasan Prishtina”

Fakulteti i Inxhinierisë Elektrike dhe Kompjuterike

Departamenti: Inxhinieri Kompjuterike



Raport

Lënda: Rrjeta Kompjuterike

Projekti 1: Dizajnimi Klient-Server

Studenti: **Edon Budakova**

ID: 170714100092

Profesori i lëndës: Blerim Rexha

Asistenti i lëndës: Haxhi Lajqi

Vegla e përdorur për zhvillim: Visual Studio 2017

Gjuha programuese e përdorur: Python

Sistemi operativ i përdorur: Windows 10 Pro

Data për dorëzim: 18 prill 2019

Përmbajtja

Hyrje	3
Metodat	4
Programi Klient – Server	8
Programimi me Sockets sipas TCP protokollit	8
FIEK-TCP Klienti.....	9
FIEK-TCP Serveri	11
Programimi me Sockets sipas UDP protokollit	16
Trajtimi i Gabimeve.....	17
Testimi	18
Përfundimi.....	24
Referencat	25

Hyrje

Raporti përshkruan punën e bërë gjatë zhvillimit të Protokollit FIEK, si projekti I pare në lëndën Rrjeta Kompjuterike. Ky protokoll i lejon klientit dhe serverit ti testoj lidhjet e tyre duke komunikuar nëpërmjet socket-ave. Vegla e përdorur për zhvillimin e software-it: Visual Studio 2017, gjuha programuese e përdorur: Python, sistemi operativ i përdorur: Windows 10 Pro (64 bit)

Mënyra se si funksionon ky protokoll është se programi i krijuar si server përmban disa metoda të cilat janë: IPADRESA, NUMRIIPORTIT, BASHKETINGELLORE, PRINTO, HOST, TIME, LOJA, KONVERTIMI, FIBONACCI, GJUJEZARIN, PRIME. Këto metoda mund të qasen nga klienti i cili dërgon një kërkesë në server duke shkruar emrin e metodës së caktuar dhe serveri i'u përgjigjet kërkesës së klientit duke kthyer rezultatin e asaj metode.

Projekti përmban 4 programe të ndryshme: FIEK-TCP Klienti, FIEK-TCP Serveri, FIEK-UDP Klienti dhe FIEK-UDP Serveri.

Metodat

Thirrja e metodave bëhet tek klienti duke shkruar emrin e metodës së përcaktuar. Nëse kërkohet të jepet një argument për zgjidhjen e metodës së caktuar, ajo jepet duke shkruar një hapësirë dhe pastaj argumentin e funksionit. Nuk duhet të jepen më shumë argumente sesa është e përcaktuar për atë metodë sepse serveri do të lajmëroj gabim (error) në shkrimin e kërkesës nga klienti.

Programet e krijuara përmbajnë këto metoda:

- **IPADRESA** – Përcakton dhe kthen IP adresën e klientit në formë dhjetore. IP adresa e klientit është marrë nga vlera e parë e fushës `address[]`, e cila përmban të dhënat e klientit i cili lidhet me serverin tonë. Pastaj është kthyer si format të lexueshëm për përdoruesin, me anë të metodës `str()`. Me thirrjen e kësaj metode nuk ka nevojë të shënohet ndonjë argument por vetëm emri `IPADRESA`.

```
def IPADRESA(address):  
    return str(address[0])
```

- **NUMRIIPORTIT** – Përcakton dhe kthen portin e klientit. Ngjashëm si te metoda `IPADRESA`, por këtu porti i klientit është marrë nga vlera e dytë e fushës `address[]`.

```
def NUMRIIPORTIT(address):  
    return str(address[1])
```

- **BASHKETINGELLORE** – Gjen numrin e bashketingellore në tekstin e dhënë dhe kthen si përgjigje numrin e tyre. Kjo metodë merr si parametër një tekst të çfarëdoshëm, pastaj krahason secilin karakter të atij stringu me fushën (array) tonë me emrin *bashketingelloret*, ku janë ruajtur të gjitha bashketingelloret e alfabetit shqip. Nëse karakteri i caktuar i tekstit përputhet me *bashketingelloret*, atëherë inkrementohet për një vlerë variabla që ruan numrin e bashketingellore. Me përshkrimin e gjithë tekstit të caktuar kthehet nga metoda numri i bashketingellore në atë tekst.

```
def BASHKETINGELLORE(teksti):  
    numriBashketingellore = 0  
    bashketingelloret = " b c ç d dh f g gj h j k l ll m n nj p q r rr s sh t th v  
x xh z zh B C Ç D Dh F G Gj H J K L Ll M N Nj P Q R Rr S Sh T Th V X Xh Z Zh"  
    for i in range(1, len(teksti)):  
        if teksti[i] in bashketingelloret:  
            numriBashketingellore +=1  
    return str(numriBashketingellore)
```

- **PRINTIMI** – Merr si parametër një tekst të caktuar, te i cili hiqen hapësirat në fillim dhe në fund të tekstit nëpërmes metodës `strip()`, dhe pastaj kthehet teksti i caktuar si i “pastërt”.

```
def PRINTIMI(teksti):
    teksti = str(teksti).strip()
    return teksti
```

- **HOST** – Kërkon dhe kthen emrin e hostit i cili mirret nga objekti `socket` nëpërmes metodës `gethostname()`, që ruhet në variablën e përcaktuar “hosti”. Nëse nuk mund të përcaktohet hosti i klientit, atëherë funksioni kthen një mesazh ku tregon se emri i hostit nuk mund të përcaktohet, kjo është bërë nëpërmjet një *try-except* të definuar si mëposhtë.

```
def HOST():
    try:
        hosti = socket.gethostname()
        pergjigja = "Emri i hostit eshte: " + hosti
        return str(pergjigja)
    except:
        pergjigja = "Emri i hostit nuk mund te percaktohet!"
        return str(pergjigja)
```

- **TIME** – Kthen kohën aktuale në server, si format të lexueshëm për përdoruesit. Kjo është përfituar nga libraria e importuar nga python `datetime`, ku është marrë koha aktuale në system përmes metodës `datetime.now()`.

```
def TIME():
    return str(datetime.datetime.now())
```

- **LOJA** – Kthen 7 numra nga rangi [1, 49]. Vargu i numrave është marrë nëpërmes një unaze që zhvillon 7 cikle, ku gjendet nga një numër random i rangut 1 deri 49 dhe ruhet në listën e përcaktuar më parë me emrin “listaNumrave”, e cila pastaj sortohet dhe kthehet si format të lexueshëm për klientin.

```
def LOJA():
    listaNumrave = []
    for i in range(7):
        listaNumrave.append(random.randint(1,49))
    listaNumrave.sort()
    return str(listaNumrave)
```

- **FIBONACCI** – Gjen numrin Fibonacci si rezultat i parametrin të dhënë hyrës.

```
def FIBONACCI(vlera):
    numri = 1
    numripr = 0
    numriDhene = 0
    numriDhene = int(vlera)
    for i in range(numriDhene-1):
        numri = numri + numripr
        numripr = numri - numripr
    return str(numri)
```

- **KONVERTIMI** – Kthen si rezultat konvertimin e parametrin të dhënë varësisht prej opcionit të zgjedhur. Opcionet e konvertimit janë: KilowattToHorsepower, HorsepowerToKilowatt, DegreesToRadians, RadiansToDegrees, GallonsToLiters, LitersToGallons. Metoda merr dy parametra nga klienti, së pari opcionin e konvertimit dhe pastaj vlerën që dëshiron të konvertoj. Funkcioni sipas tipit të dhënë përmes disa kushtëzimeve përcakton cili opcion është përzgjedhur dhe kthen rezultatin e caktuar, që paraqet vlerën e konvertuar. Nëse nuk jepet opcioni i cili është përcaktuar nga serveri metoda kthen një mesazh që e lajmëron klientin se ka gabuar në shkrimin e kërkesës ("ERROR, keni berim gabim gjate shenimit").

```
def KONVERTIMI(opcioni,vlera):
    vleraKonvertuar = 0
    vlera = float(vlera)

    if opcioni=="KILOWATTTOHORSEPOWER":
        return str(vlera/0.745699872)
    elif opcioni=="HORSEPOWERTOKILOWATT":
        return str(vlera*0.745699872)
    elif opcioni=="DEGREEESTORADIANS":
        return str(vlera*3.14/180)
    elif opcioni=="RADIANSTODEGREES":
        return str(vlera*180/3.14)
    elif opcioni=="GALLONSTOLITERS":
        return str(vlera/0.26417)
    elif opcioni=="LITERSTOGALLONS":
        return str(vlera*0.26417)
    else:
        return "ERROR, keni berim gabim gjate shenimit"
```

- **GJUZARET** – Kjo metodë simulon huthjen e dy zareve, pra kthen dy numra të plotë random nga 1 deri në 6.

```
def GJUZARET():
    min = 1
    max = 6
    kubi1 = str(random.randint(min, max))
    kubi2 = str(random.randint(min, max))
    pergjigja = str("Vlerat e zareve jane: " + kubi1 + " dhe " + kubi2)
    return pergjigja
```

- **PRIME** – Shikon nëse një numër është një numër i thjesht.

```
def PRIME(numri):
    x=True;
    for x in range(2,numri):
        if numri%x ==0:
            x=False;
            return x;
    return x;
```

Programi Klient – Server

Me ekzekutimin e këto dy programeve, ku së pari duhet të ekzekutohet programi server pastaj ai i klientit, krijohen dy procese ai i serverit dhe ai i klientit. Këto procese komunikojnë me njëri-tjetrin nëpërmjet socket-ave.

*TCP protokollit është një lidhje e orientuar dhe siguron një transfer të besueshëm të të dhënave.

* UDP protokollit nuk garanton dërgimin e të gjitha të dhënave nga një host në tjetrin dhe nuk ka nevojë të vendoset një lidhje që të shkëmbehen të dhënat.

Klienti dhe serveri gjatë zhvillimit të programeve të FIEK protokollit kanë përdorur portin 12000, kurse si host është përdorur 'localhost'-I (ose 127.0.0.1).

Programi Server, do të punoj vazhdimisht pa ndërprerje, siq është kërkuar në detyrë, vetëm nëse ndodh ndonjë gabim apo klienti kërkon daljen nga programi (EXIT). Serveri është programuar ashtu që të jetë në gjendje të pranoj një sekuencë të kërkesave nga i njëjti klient apo nga klientë të ndryshëm në vazhdimësi, sipas kërkesës në detyrë.

Programi Klient punon në atë mënyrë që të jetë në gjendje të shkruaj një kërkesë, e cila përmban emrin e metodës së caktuar dhe ta dërgoj atë te programi i serverit i cili e merr atë kërkesë dhe kthen një përgjigje të caktuar te klienti.

Programimi me Sockets sipas TCP protokollit

TCP protokollit gjatë komunikimit klienti-server garanton dërgimin e të gjitha të dhënave, pa asnjë humbje. Pra është një *protokoll i besueshëm* për shkëmbimin e të dhënave.

TCP protokollit është *connection-oriented*, pra para se të shkëmbehen të dhënat ndërmjet klientit dhe serverit fillimisht duhet të vendoset një lidhje TCP. Pasiqë që të vendoset lidhja, nëse njëra anë dëshiron të dërgoj të dhëna ajo vetëm vendos të dhënat në lidhjen TCP përmes socketit dhe dërgohen në anën tjetër.

Klienti ka për detyrë që të inicioj kontaktin me server. Në mënyrë që serveri të jetë në gjendje të reagoj ndaj kontaktit të klientit, ai duhet të jetë gati, pra fillimisht duhet të jetë duke punuar serveri. Te klienti krijohet socketi i klientit që punon sipas TCP protokollit, ku i bashkangjitet atij socketi adresa e socketit të serverit, që është IP adresa e hostit të serverit dhe portit të tij. Pasiqë që të krijohet socketi, klienti vendos një TCP lidhje me serverin. Me vendosjen e kësaj lidhjeje është e mundur të shkëmbehen të dhënat përmes socketave të cilët punojnë sipas parimit të TCP protokollit.

FIEK-TCP Klienti

Te programi TCP Klienti fillimisht janë importuar libraritë: socket dhe sys.

```
from socket import *  
import sys
```

Pastaj janë përcaktuar hosti i serverit dhe porti i tij në variablat e përshkruara:

Pastaj shfrytezuesi ka mundesine ta vandose emrin e serverin dhe portin sipas nevojës. Si host server i nënkuptueshëm përdoret “localhost” (ose 127.0.0.1), si port i nënkuptueshëm përdoret 12000.

```
serverName = input("Shenoni emrit e serverit: ");  
  
Port = input("Shenoni portin: ");  
serverPort = int(Port);
```

Kemi krijuar socketin e klientit me emrin e variablës si “s”, ku parametri i parë i socketit paraqet familjen e IP adresave të përdorura IPv4, kurse parametri i dytë tregon se tipi i socketit është SOCK_STREAM, që do të thotë se kemi të bëjmë me TCP socket.

```
s = socket(AF_INET, SOCK_STREAM)
```

Vendoset lidhja TCP ndërmjet klientit dhe serverit përmes metodës connect(), ku parametri i parë i saj duhet të jetë hosti i serverit, kurse parametri i dytë numri i portit. Në këtë mënyrë arrihet “three-way handshake” dhe vendoset lidhja TCP ndërmjet klientit dhe serverit.

```
s.connect((serverName,serverPort))
```

Me poshte krijohet nje pjese e programit qe i ndihmon klientit të zgjedh metodat:

```
print("""Jeni te lidhur me serverin!
```

```
    Cilat nga keto metoda deshironi ti perdorni:
```

*IPADRESA	- Percakton dhe kthen IP adresen e klientit
*NUMRIIPORTIT klientit(hostit)	- Percakton dhe kthen portin e
*BASHKETINGELLORE {hapesire} teksti numrin e bashketingelloreve ne ate tekst	- Merr si parameter nje tekst dhe kthen
*PRINTIMI {hapesire} teksti	- Kthen fjaline e shtypur ne tekst. Hapsirat
ne fillim dhe ne fund te fjalise nuk duhet te kthehen	
*HOST	- Kerkon emrin e kompjuterit dhe e kthen ate
*TIME	- Percakton kohen aktuale ne server

<p>*LOJA</p> <p>*FIBONACCI {hapesire} numer parametrin te dhene hyres</p> <p>*KONVERTIMI {hapesire} varesisht opcionit te zgjedhur: opcionit {hapesire} vlera DEGREESTORADIANS LITERSTOGALLONS</p> <p>*GJUJZARET numra të plotë random nga 1 deri në 6.</p> <p>*PRIME thjesht.</p>	<p>- Kthen 7 numra nga rangu [1,49]</p> <p>- Gjen numrin Fibonacci si rezultat i</p> <p>- Kthen si rezultat konvertimin e opcioneve KILOWATTTOHORSEPOWER, HORSEPOWERTOKILOWATT, RADIANSTODEGREES, GALLONSTOLITERS,</p> <p>- Kjo metodë simulon hudhjen e dy zareve, dy</p> <p>- Shikon nëse një numër është një numer i</p>
--	---

```

Sheno EXIT per te dalur nga programi!
=====
"""

```

Klienti është programuar në atë mënyrë që të punoj vazhdimisht derisa ai vet të dëshiroj ta mbyll lidhjen apo të mbyllet vetë nga ndonjë gabim eventual. Kjo është arritur përmes një unaze `while` 1, që punon deri në “pafundësi”, apo deri sa të ndërprehet nga klienti.

```

while 1:
    try:
        kerkesa = input('Shkruaj emrin e metodes dhe argumentin perkates: ')
        if kerkesa!=" " and kerkesa.upper()!="EXIT":
            # behet enkodimi i kerkeses dhe dergimi tek TCP serveri
            s.sendall(str(kerkesa).encode())
        else:
            break
        # ketu behet pranimi i pergjigjes nga serveri dhe dekodimi i saj
        # madhesia me e madhe qe mund te mirret eshte 128 byte
        data = s.recv(128)
        print('Te dhenat e pranuar nga serveri: ')
        print(str(data.decode()).strip())
        print("-----\n")
    except Exception as e:
        print(str(e))
        break

```

Në fund mbyllet lidhja e vendosur:

```
s.close()
```

FIEK-TCP Serveri

Fillimisht janë importuar libraritë që do të nevojiten për programimin e programit:

```
from socket import *
import random
import datetime
import sys
import threading
import math
```

Pastaj ndahet porti i serverit dhe Ip adresa:

```
serverPort = 12000
serverName = '127.0.0.1'
```

Është krijuar socketi i serverit sipas TCP protokollit si në rastin e klientit.

```
serverSocket = socket(AF_INET, SOCK_STREAM)
```

Vazhdon programimi me lidhjen apo bashkangjitjen e portit të serverit me socketin e krijuar sipas kodit në vijim:

```
serverSocket.bind((serverName, serverPort))
```

Pasiqë të krijohej socketi i serverit të TCP protokollit presim për një klient të lidhet me server.

```
serverSocket.listen(5)
```

Shfaqim një mesazh në server se: “Serveri është i gatshëm të pranoj kërkesa”.

```
print('Serveri eshte i gatshem te pranoj kerkesa.')
```

Dhe pastaj definojmë metodat sipas kodit të shfaqur më parë te Përshkrimi i metodave.

Definojmë një funksion shtesë që përdoret për shtypjen në server të të dhënave që dërgohen te klienti. Ky funksion do të thirret pas çdo dërgese të përgjigjeve nga serveri në klient dhe do të shtyp ato të dhëna në programin e serverit. Kodi për defimin e kësaj metode është:

```
def ShtypTeDhenat(teDhenat):
    print("\n-----")
    print("Te dhenat e derguara te klienti =====>> ", teDhenat)
    return
```

Bëhet definimi i një funksioni i cili thirret pas çdo krijimi të një procesi (thread) të ri, që bëhet me lidhjen e një klienti të ri me server. Ky funksion merr si parametra socketin i cili krijohet dhe i dedikohet klientit të lidhur me server, kurse parametri tjetër paraqet adresën e tij. Në këtë funksion fillimisht pranohet kërkesa nga klienti, bëhet dekodimi i saj, ajo kërkesë ndahet sipas hapësirave dhe ruhet në një Array (varg) me emrin kerkesaArray. Vlera e parë e kësaj fushe paraqet emrin e metodës e cila kërkohet nga klienti. Ky string kthehet ne upper case që të përputhet me kushtëzimet e përcaktuara nga zhvilluesi i kodit dhe pastaj bëhet krahasimi i këtij emri me emrin e metodave të parapërcaktuara nga ne. Në bazë të kërkesës përzgjedhet edhe metoda që duhet të kthehet nga serveri dhe dërgohet te klienti si në vijim:

```
connectionSocket.send( _____ .encode( ) ).
```

Nëse kërkesa nuk përputhet me asnjë nga metodat e përcaktuara nga ne, nga serveri dërgohet një mesazh që “Kërkesa juaj nuk është valide!”, me kod:

```
connectionSocket.send("Kerkesa juaj eshte invalide, ju lutem provoni perseri!".encode())
```

Kodi i shkruar për definimin e kësaj metode është:

```
def klienti_i_ri(connectionSocket,addr):

    kerkesa = (bytes)("empty".encode())
    try:
        while str(kerkesa.decode()).upper()!="EXIT" and str(kerkesa.decode())!="":
            # pranimi i kërkeses nga klienti
            kerkesa = connectionSocket.recv(128)

            # dekodimi i kërkeses
            kerkesaStr = str(kerkesa.decode()).strip()

            # behet ndarja e kërkeses sipas hapësirave dhe ruhet ne një Array
            kerkesaArray = kerkesaStr.split(' ')

            # kërkesa e dërguar nga klienti kthehet se pari ne upperCase:
            kerkesaArray[0] = kerkesaArray[0].upper()

            # metoda IPADRESA
            if kerkesaArray[0]=="IPADRESA":
                if len(kerkesaArray) == 1:
                    connectionSocket.send(("IP adresja juaj eshte:
"+IPADRESA(addr)).encode())
                    ShtypTeDhenat(("IP adresja e klientit: "+IPADRESA(addr)))
                else:
                    connectionSocket.send("Kerkesa juaj eshte invalide, ju lutem provoni
perseri!".encode())
                    ShtypTeDhenat("ERROR")

            # metoda NUMRIIPORTIT
            elif kerkesaArray[0]=="NUMRIIPORTIT":
                if len(kerkesaArray) == 1:
                    connectionSocket.send(("Numri i portit tuaj eshte:
"+NUMRIIPORTIT(addr)).encode())
```

```

        ShtypTeDhenat(("Numri i portit te klientit eshte:
"+NUMRIIPORTIT(addr)))
    else:
        connectionSocket.send("Kerkesa juaj eshte invalide, ju lutem provoni
perseri!".encode())
        ShtypTeDhenat("ERROR")

    # metoda BASHKETINGELLORE
    elif kerkesaArray[0]=="BASHKETINGELLORE":
        if len(kerkesaArray) == 2:
            rezultati = "Numri i bashketingelloreve ne fjalen e dhene eshte: " +
BASHKETINGELLORE(kerkesaArray[1])
            connectionSocket.send(rezultati.encode())
            ShtypTeDhenat(rezultati)
        else:
            connectionSocket.send("Kerkesa juaj eshte invalide, ju lutem provoni
perseri!".encode())
            ShtypTeDhenat("ERROR")

    # metoda PRINTIMI
    elif kerkesaArray[0]=="PRINTIMI":
        kerkesaStr2 = (str(kerkesaStr)).replace("PRINTIMI","")
        connectionSocket.send(("Fjalja e printuar: " +
PRINTIMI(kerkesaStr2)).encode())
        ShtypTeDhenat(("Fjalja e printuar: " + PRINTIMI(kerkesaStr2)))

    # metoda HOST
    elif kerkesaArray[0]=="HOST":
        if len(kerkesaArray) == 1:
            if HOST()=="Emri i hostit nuk mund te percaktohet!":
                connectionSocket.send(("Emri i hostit nuk mund te
percaktohet!").encode())
                ShtypTeDhenat("Emri i hostit nuk mund te percaktohet!")
            else:
                connectionSocket.send(("Emri i klientit eshte:
"+HOST()).encode())
                ShtypTeDhenat(("Emri i klientit eshte: "+HOST()))
        else:
            connectionSocket.send("Kerkesa juaj eshte invalide, ju lutem provoni
perseri!".encode())
            ShtypTeDhenat("ERROR")

    # metoda TIME
    elif kerkesaArray[0]=="TIME":
        if len(kerkesaArray) == 1:
            connectionSocket.send(("Koha e tanishme eshte: " + TIME()).encode())
            ShtypTeDhenat(("Koha e tanishme eshte: " + TIME()))
        else:
            connectionSocket.send("Kerkesa juaj eshte invalide, ju lutem provoni
perseri!".encode())
            ShtypTeDhenat("ERROR")

```

```

# metoda LOJA
elif kerkesaArray[0]=="LOJA":
    if len(kerkesaArray) == 1:
        connectionSocket.send(("Rezultati nga loja: " + LOJA()).encode())
        ShtypTeDhenat(("Rezultati nga loja: " + LOJA()))
    else:
        connectionSocket.send("Kerkesa juaj eshte invalide, ju lutem provoni
perseri!".encode())
        ShtypTeDhenat("ERROR")

# metoda FIBONACCI
elif kerkesaArray[0]=="FIBONACCI":
    for i in range(len(kerkesaArray)):
        if "" in kerkesaArray:
            kerkesaArray.remove("")
        if len(kerkesaArray)==1 or len(kerkesaArray)>2:
            connectionSocket.send(("Kerkesa juaj eshte invalide, ju lutem provoni
perseri!").encode())
            ShtypTeDhenat("ERROR")
        else:
            connectionSocket.send(("Numri i "+kerkesaArray[1]+" ne serine
fibonacci eshte: "+FIBONACCI(kerkesaArray[1])).encode())
            ShtypTeDhenat(("Numri i "+kerkesaArray[1]+" ne serine fibonacci
eshte: "+FIBONACCI(kerkesaArray[1])))

# metoda KONVERTIMI
elif kerkesaArray[0]=="KONVERTIMI":
    for i in range(len(kerkesaArray)):
        if "" in kerkesaArray:
            kerkesaArray.remove("")
        if len(kerkesaArray)>3 or len(kerkesaArray)<3:
            connectionSocket.send(("Kerkesa juaj eshte invalide, ju lutem provoni
perseri!").encode())
            ShtypTeDhenat("ERROR")
        else:
            konverto = str(kerkesaArray[1]).lower().split("to")
            pergjigja = kerkesaArray[2] + " " + str(konverto[0]) + " jane te
barabarte me " + KONVERTIMI(str(kerkesaArray[1]).upper(),kerkesaArray[2]) + " " +
str(konverto[1])
            connectionSocket.send(str(pergjigja).encode())
            ShtypTeDhenat(pergjigja)

# metoda shtese GjujZaret - kthen vlerat e dy zareve te gjuajtura.
elif kerkesaArray[0]=="GJUJZARET":
    if len(kerkesaArray) == 1:
        connectionSocket.send(str(GJUJZARET()).encode())
        ShtypTeDhenat(str(GJUJZARET()))
    else:
        connectionSocket.send("Kerkesa juaj eshte invalide, ju lutem provoni
perseri!".encode())

```

```

        ShtypTeDhenat("ERROR")                # nqs. nuk shkruhet asnjera nga
metodat e percaktuara
    else:
        connectionSocket.send("Kerkesa juaj eshte invalide, ju lutem provoni
perseri!".encode())
        ShtypTeDhenat("ERROR!")

# metoda shtese Prime
    elif kerkesaArray[0]=='PRIME':
        numri = int(kerkesaArray[1]);
        if PRIME(numri):
            connectionSocket.send(str.encode("Numri " + str(numri) + " eshte
numer i thjeshte!"))
            ShtypTeDhenat(str("Numri " + str(numri) + " eshte numer i
thjeshte!"))
        else:
            connectionSocket.send(str.encode("Numri " + str(numri) + " nuk
eshte numer i thjeshte!"))
            ShtypTeDhenat(str("Numri " + str(numri) + " nuk eshte numer i
thjeshte!"))

        # kur largohet klienti mbyllet lidhja me te, por serveri ende pret per lidhje
me klient te tjere
        connectionSocket.close()
    except Exception as e:
        print("\nERROR: ")
        print(str(e))
        connectionSocket.close()

```

Serveri është programuar në atë mënyrë që të jetë në gjendje të pranoj kërkesa të njëpasnjëshme dhe nga klienti të ndryshëm. Kjo është bërë përmes një unaze `while 1`, ku nga çdo klienti i ri që i qaset serverit krijohet një thread i ri për atë kërkesë. Pas kësaj kërkesë krijohen threads të reja për klienti të ri deri sa vetë klientët e ndërprejnë lidhjen. Por serveri gjithmonë është i gatshëm për pranimin e kërkesave përdërisa punon.

```

while 1:
    # ky rresht ben qe serveri te "degjoj" per kerkesa nga klienti permes lidhjes TCP
    connectionSocket, addr = serverSocket.accept()
    print('Klienti me IP adrese %s dhe me numrin e portit %s eshte lidhur me server'
%(addr))

    # krijimi i nje procesi te ri (threadi te ri), me lidhjen e nje klienti te ri
    threading._start_new_thread(klienti_i_ri,(connectionSocket,addr))

```

Programimi me Sockets sipas UDP protokollit

UDP protokollit paraqet një protokoll *jo të besueshëm*, pra nuk ka garancion se të dhënat do të mbërrijnë në cak gjithmonë dhe që do të ruhet rendi i dërgesës ashtu siç e kemi përcaktuar. Dallimi nga programimi sipas TCP protokollit qëndron se UDP protokollit është *connection-less*, pra nuk vendoset një lidhje ndërmjet klientit dhe serverit që ata të komunikojnë mes vete.

Gjatë krijimit të socketit si të klientit ashtu edhe te serveri, parametri i dytë i socketit do të ketë një tip tjetër të tij, pra do të jetë `SOCK_DGRAM`, që do të thotë se kemi të bëjmë me UDP socket.

```
s = socket(AF_INET, SOCK_DGRAM)
```

Si rrjedhim që nuk kemi nevojë të vendosim një lidhje në mes klientit dhe serverit si te TCP protokollit atëherë nuk na duhet metoda `connect()` te ana e klientit.

Dallimi tjetër qëndron në metodën e dërgimit të të dhënave që këtu përdoret metoda `sendto()`, ku si parametra iu bashkangjitet mesazhit adresa e plote e destinacionit, pra emri i hostit dhe numri i portit të serverit. Pasi që të dërgohet paketa, klienti pret të marr përgjigje nga serveri.

```
s.sendto(str(kerkesa).encode(), (serverName, serverPort))
```

Te ana e serverit dallimi qëndron në metodën e pranimit të kërkesës nga klienti që është: `recvfrom(128)`.

```
data, serverAddress = s.recvfrom(128)
```

Te UDP serveri nuk kemi nevojë të krijojmë threads të ri pas çdo kërkesë të klientit si te rasti i TCP, pra nuk kemi pasur nevojë të definojmë metodën `klienti_i_ri()`, por validimi i të dhënave, mënyra e qasjes ndaj kërkesës dhe kthimi i përgjigjes në bazë të asaj kërkesë është bërë përafërsisht në të njëjtën mënyrë. Njëjtë realizohet edhe UDP serveri përmes një unaze `while 1`, ku janë vendosur kushtëzimet për përzgjedhjen e metodës dhe dërgimin e përgjigjes të klienti. Serveri është i gatshëm të pranojë kërkesa të vazhdueshme dhe të mos ndërpres lidhjen me klientin.

Te UDP serveri kemi edhe një dallim se nuk duhet të mbyllin lidhjen me klientin pas një kërkesë, pra nuk duhet të kemi `s.close()`.

Trajtimi i Gabimeve

Programet FIEK-UDP Server dhe FIEK-TCP Server janë programuar në atë mënyrë që të trajtojnë gabimet që bëhen gjatë shënimit të kërkesës që përmban emrin e metodës.

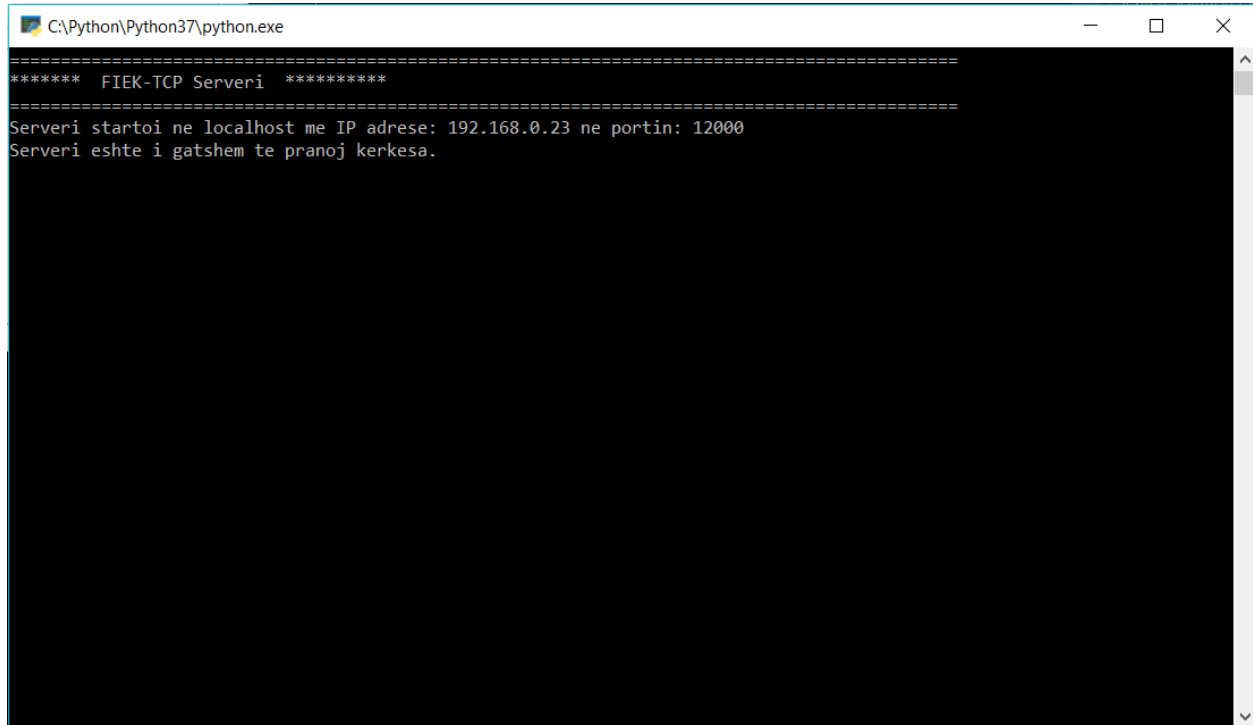
Nëse shënojmë emrin e metodës gabim do të paraqitet një mesazh nga serveri se kemi gabuar gjatë shënimit të kërkesës. Ky gabim është evituar duke bërë krahasimin e stringut të kërkesës dhe emrit të metodës, nëse nuk përputhet me asnjë nga metodat paraqitet mesazhi:

```
connectionSocket.send("Kerkesa juaj eshte invalide, ju lutem provoni perseri!".encode())
```

Tjetri gabim që mund të ndodhë është nëse klienti jep më shumë argumente për metodën e caktuar se sa është e nevojshme. Kjo është bërë përmes një kushtëzimi që kërkon një numër të caktuar të argumenteve. Pasiqë është bërë ndarja e kërkesës sipas hapësirave “ ”, ajo është ruajtur si array në variablën *kerkesaArray* që përmban emrin e metodës dhe argumentet përkatëse (ndodh që mos të kërkohet asnjë argument për ndonjë metodë të caktuar), atëherë bëhet pyetja se ashtë numri i argumenteve sa është kërkuar, nëse kemi dhënë më shumë apo më pak paraqitet një mesazh ku tregohet se kërkesa e klientit është invalide.

Testimi

Me hapjen e programit FIEK-TCP serverit do të shfaqet kjo dritare:

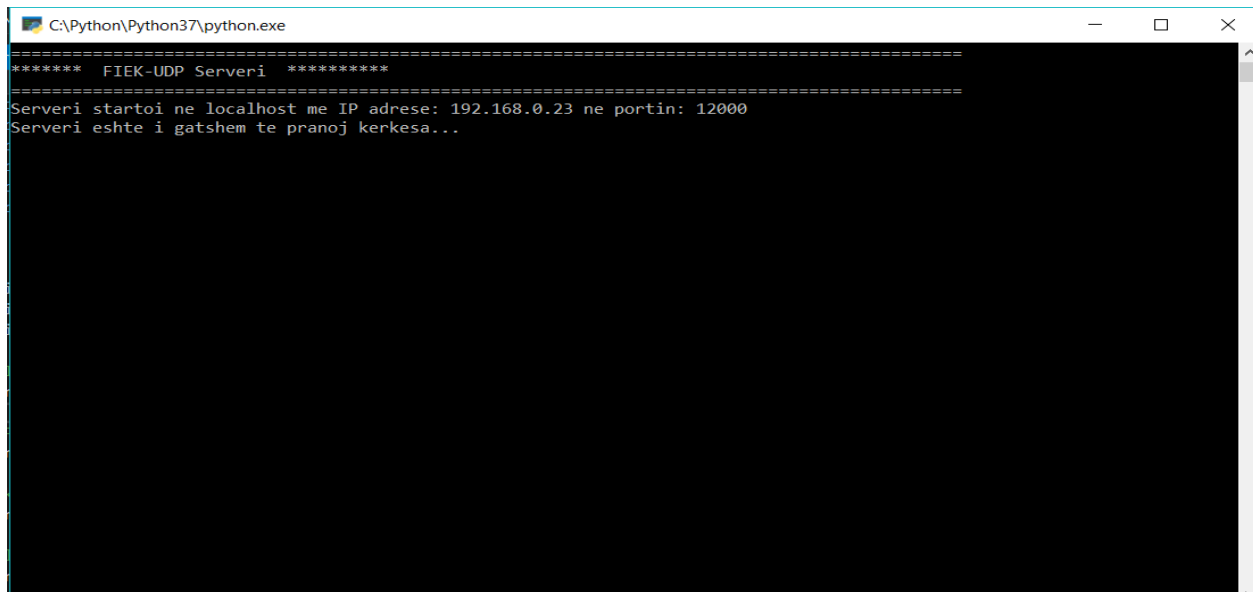


```
C:\Python\Python37\python.exe

***** FIEK-TCP Serveri *****

Serveri startoi ne localhost me IP adrese: 192.168.0.23 ne portin: 12000
Serveri eshte i gatshem te pranoj kerkesa.
```

Gjithashtu edhe tek UDP serveri:



```
C:\Python\Python37\python.exe

***** FIEK-UDP Serveri *****

Serveri startoi ne localhost me IP adrese: 192.168.0.23 ne portin: 12000
Serveri eshte i gatshem te pranoj kerkesa...
```

Menjëherë pas hapjes së programit të klientit edhe në TCP klientin edhe në UDP klientin do të jepet mundësia të shënohet emri i serverit dhe portit sipas nevojës

```
C:\Python\Python37\python.exe
=====
***** FIEK-TCP Klienti *****
=====
Shenoni emrit e serverit:
```

Përdoret “localhost” (ose 127.0.0.1) si host server i nënkuptueshëm, dhe portin 12000, si port të nënkuptueshëm, e pastaj do të shfaqet një pamje ndihmëse, ku tregohen metodat dhe përdorimi i tyre:

```
C:\Python\Python37\python.exe
=====
***** FIEK-TCP Klienti *****
=====
Shenoni emrit e serverit: localhost
Shenoni portin: 12000
Jeni lidhur ne serverin localhost ne portin 12000
=====
Jeni te lidhur me serverin!

Cilat nga keto metoda deshironi ti perdorni:

*IPADRESA                - Percakton dhe kthen IP adresen e klientit
*NUMRIIPORTIT            - Percakton dhe kthen portin e klientit(hostit)
*BASHKETINGELLORE {hapesire} teksti - Merr si parameter nje tekst dhe kthen numrin e bashketingelloreve ne ate tekst
*PRINTIMI {hapesire} teksti  - Kthen fjaline e shtypur ne tekst. Hapsirat ne fillim dhe ne fund te fjalise nuk duhet te kthehen
*HOST                    - Kerkon emrin e kompjuterit dhe e kthen ate
*TIME                    - Percakton kohen aktuale ne server
*LOJA                    - Kthen 7 numra nga rangu [1,49]
*FIBONACCI {hapesire} numer  - Gjen numrin Fibonacci si rezultat i parametrit te dhene hyres
*KONVERTIMI {hapesire}  - Kthen si rezultat konvertimin e opcioneve varesisht opcionit te zgjedhur:
  opcioni {hapesire} vlera    KILOWATTTOHORSEPOWER, HORSEPOWERTOKILOWATT, DEGREESTORADIANS
                                RADIANSTODEGREES, GALLONSTOLITERS, LITERSTOGALLONS
*GJUJZARET              - Kjo metode simulon hudhjen e dy zareve, dy numra te plotë random nga 1 deri në 6.
*PRIME                  - Shikon nëse një numër është një numer i thjesht.

=====
Sheno EXIT per te dalur nga programi!
=====
Shkruaj emrin e metodes dhe argumentin perkates:
```

E njëjta edhe për UDP klientin:

```
C:\Python\Python37\python.exe
=====
***** FIEK-UDP Klienti *****
=====
Shenoni emrit e serverit: localhost
Shenoni portin: 12000
Deni lidhur ne serverin localhost ne portin 12000
=====
Deni te lidhur me serverin!

Cilat nga keto metoda deshironi ti perdorni:

*IPADRESA                - Percakton dhe kthen IP adresen e klientit

*NUMRIIPORTIT            - Percakton dhe kthen portin e klientit(hostit)

*BASHKETINGELLORE {hapesire} teksti - Merr si parameter nje tekst dhe kthen numrin e bashketingelloreve ne ate tekst

*PRINTIMI {hapesire} teksti - Kthen fjaline e shtypur ne tekst. Hapsirat ne fillim dhe ne fund te fjalise nuk duhet te kthehen

*HOST                    - Kerkon emrin e kompjuterit dhe e kthen ate

*TIME                    - Percakton kohen aktuale ne server

*LOJA                    - Kthen 7 numra nga rangu [1,49]

*FIBONACCI {hapesire} numer - Gjen numrin Fibonacci si rezultat i parametrut te dhene hyres

*KONVERTIMI {hapesire}
opcion {hapesire} vlera - Kthen si rezultat konvertimin e opcioneve varesisht opcionit te zgjedhur:
KILOWATTTOHORSEPOWER, HORSEPOWERTOKILOWATT, DEGREESTORADIANS
RADIANSSTODEGREES, GALLONSTOLITERS, LITERSTOGALLONS

*GJUJZARET              - Kjo metodë simulon hudhjen e dy zareve, dy numra të plotë random nga 1 deri në 6.

*PRIME                   - Shikon nëse një numër është një numer i thjesht.

Sheno EXIT per te dalur nga programi!
=====
```

=>Këto janë pamjet (screenshots) të bëra gjatë testimit të metodave në programet e klientit:

FIEK-UDP Klienti

```
C:\Python\Python37\python.exe
***** FIEK-UDP Klienti *****
Shenoni emrit e serverit: localhost
Shenoni portin: 12000
Jeni lidhur ne serverin localhost ne portin 12000
Jeni te lidhur me serverin!

Cilat nga keto metoda deshironi ti perdorni:

*IPADRESA                - Percakton dhe kthen IP adresen e klientit
*NUMRIIPORTIT            - Percakton dhe kthen portin e klientit(hostit)
*BASHKETINGELLORE (hapesire) teksti - Merr si parameter nje tekst dhe kthen numrin e bashketingelloreve ne ate tekst
*PRINTIMI (hapesire) teksti - Kthen fjaline e shtypur ne tekst. Hapsirat ne fillim dhe ne fund te fjalisë nuk duhet te kthehen
*HOST                    - Kerkon emrin e kompjuterit dhe e kthen ate
*TIME                    - Percakton kohen aktuale ne server
*LOJA                    - Kthen 7 numra nga rangu [1,49]
*FIBONACCI (hapesire) numer - Gjen numrin Fibonacci si rezultat i parametrit te dhene hyres
*KONVERTIMI (hapesire) - Kthen si rezultat konvertimin e opsioneve varesisht opcionit te zgjedhur:
  opcion (hapesire) vlera      KILOWATTTOHORSEPOWER, HORSEPOWERTOKILOWATT, DEGREESTORADIANS
                                RADIANS TODIGREES, GALLONSTOLITERS, LITERSTOGALLONS
*GJUZARET                - Kjo metode simulon hudhjen e dy zareve, dy numra të plotë random nga 1 deri në 6.
*PRIME                    - Shikon nëse një numër është një numër i thjeshtë.

Sheno EXIT per te dalur nga programi!

Shkruaj emrin e metodës dhe argumentin perkates: ipadresa
Te dhenat nga serveri:
IP adresa juaj eshte: 127.0.0.1

Shkruaj emrin e metodës dhe argumentin perkates: numriiporit
Te dhenat nga serveri:
Numri i portit tuaj eshte: 52251

Shkruaj emrin e metodës dhe argumentin perkates: bashketingellore edonbudakova
Te dhenat nga serveri:
Numri i bashketingelloreve ne fjaline e dhene eshte: 6

Shkruaj emrin e metodës dhe argumentin perkates: printimi hello
Te dhenat nga serveri:
Fjalja e printuar: printimi hello

Shkruaj emrin e metodës dhe argumentin perkates: host
Te dhenat nga serveri:
Emri i hostit nuk mund te percaktohet!

Shkruaj emrin e metodës dhe argumentin perkates: loja
Te dhenat nga serveri:
Rezultati nga loja: [2, 17, 17, 20, 30, 37, 38]

Shkruaj emrin e metodës dhe argumentin perkates: time
Te dhenat nga serveri:
Koha e tani shë eshte: 2019-04-17 23:48:04.194768

Shkruaj emrin e metodës dhe argumentin perkates: fibonacci 9
Te dhenat nga serveri:
Numri i 9 ne serine fibonacci eshte: 34

Shkruaj emrin e metodës dhe argumentin perkates: konvertimi kilowatttohorsepower 130
Te dhenat nga serveri:
130 kilowatt jane te barabarte me 174.332871549695 horsepower

Shkruaj emrin e metodës dhe argumentin perkates: gjuzaret
Te dhenat nga serveri:
Vlerat e zareve jane: 2 dhe 3

Shkruaj emrin e metodës dhe argumentin perkates: gjuzaret
Te dhenat nga serveri:
Vlerat e zareve jane: 2 dhe 4

Shkruaj emrin e metodës dhe argumentin perkates: prime 17
Te dhenat nga serveri:
Numri 17 eshte numer i thjeshte!

Shkruaj emrin e metodës dhe argumentin perkates: exit
Press any key to continue . . .
```

FIEK-TCP Klienti

```
C:\Python\Python37\python.exe
***** FIEK-TCP Klienti *****
=====
Shenoni emrit e serverit: localhost
Shenoni portin: 12000
Jeni lidhur ne serverin localhost ne portin 12000
=====
Jeni te lidhur me serverin!

Cilat nga keto metoda deshironi ti perdorni:

*IPADRESA                - Percakton dhe kthen IP adresen e klientit
*NUMRIIPORTIT            - Percakton dhe kthen portin e klientit(hostit)
*BASKETINGELLORE (hapesire) teksti - Merr si parameter nje tekst dhe kthen numrin e bashketingelloreve ne ate tekst
*PRINTIMI (hapesire) teksti - Kthen fjaline e shtypur ne tekst. Hapsirat ne fillim dhe ne fund te fjalise nuk duhet te kthehen
*HOST                    - Kerkon emrin e kompjuterit dhe e kthen ate
*TIME                    - Percakton kohen aktuale ne server
*LOJA                    - Kthen 7 numra nga rangu [1,49]
*FIBONACCI (hapesire) numer - Gjen numrin Fibonacci si rezultat i parametrit te dhene hyres
*KONVERTIMI (hapesire) opioni (hapesire) vlera - Kthen si rezultat konvertimin e opioneve varesisht opcionit te zgjedhur:
KILOWATTTOHORSEPOWER, HORSEPOWERTOKILOWATT, DEGREESTORADIANS
RADIANTSTODEGREES, GALLONSTOLITERS, LITERSTOGALLONS
*GJUZARET                - Kjo metode simulon hudhjen e dy zareve, dy numra te plotë random nga 1 deri në 6.
*PRIME                   - Shikon nëse një numër është një numer i thjeshtë.

=====
Sheno EXIT per te dalur nga programi!
=====

Shkruaj emrin e metodës dhe argumentin perkates: ipadresa
Te dhenat nga serveri:
IP adresa juaj eshte: 127.0.0.1
=====

Shkruaj emrin e metodës dhe argumentin perkates: numriiporit
Te dhenat nga serveri:
Numri i portit tuaj eshte: 53317
=====

Shkruaj emrin e metodës dhe argumentin perkates: bashketingellore edonbudakova
Te dhenat nga serveri:
Numri i bashketingelloreve ne fjalen e dhene eshte: 6
=====

Shkruaj emrin e metodës dhe argumentin perkates: printimi hello
Te dhenat nga serveri:
Fjalia e printuar: printimi hello
=====

Shkruaj emrin e metodës dhe argumentin perkates: host
Te dhenat nga serveri:
Emri i hostit nuk mund te percaktohet!
=====

Shkruaj emrin e metodës dhe argumentin perkates: time
Te dhenat nga serveri:
Koha e tanishme eshte: 2019-04-15 23:22:52.890097
=====

Shkruaj emrin e metodës dhe argumentin perkates: loja
Te dhenat nga serveri:
Rezultati nga loja: [4, 23, 24, 36, 40, 44, 49]
=====

Shkruaj emrin e metodës dhe argumentin perkates: fibonacci 25
Te dhenat nga serveri:
Numri i 25 ne serine fibonacci eshte: 75025
=====

Shkruaj emrin e metodës dhe argumentin perkates: konvertimi gallonstoliters 20
Te dhenat nga serveri:
20 gallons jane te barabarte me 75.70802386342127 liters
=====

Shkruaj emrin e metodës dhe argumentin perkates: gjuzaret
Te dhenat nga serveri:
Vlerat e zareve jane: 5 dhe 2
=====

Shkruaj emrin e metodës dhe argumentin perkates: prime 10
Te dhenat nga serveri:
Numri 10 nuk eshte numer i thjeshte!
=====

Shkruaj emrin e metodës dhe argumentin perkates: exit
Press any key to continue . . .
```

Gjatë komunikimit klient-server apo gjatë dërgimit të përgjigjeve nga serveri tek klienti, në anën e serverit vazhdimisht do të shtypen të gjitha të dhënat që dërgohen te klienti. Këto janë pamjet që shfaqen te programi i serverit (tek UDP serveri dhe TCP serveri).

```
C:\Python\Python37\python.exe
***** FIEK-UDP Serveri *****
Serveri startoi ne localhost me IP adrese: 192.168.0.23 ne portin: 12000
Serveri eshte i gatshem te pranoj kerkesa...

Te dhenat e derguara te klienti ===== >> IP adresa e klientit: 127.0.0.1

Te dhenat e derguara te klienti ===== >> Numri i portit te klientit eshte: 52251

Te dhenat e derguara te klienti ===== >> Numri i bashketingelloreve ne fjaline e dhene eshte: 6

Te dhenat e derguara te klienti ===== >> Fjalja e printuar: printimi hello

Te dhenat e derguara te klienti ===== >> Emri i hostit nuk mund te percaktohet!

Te dhenat e derguara te klienti ===== >> Rezultati nga loja: [6, 15, 17, 31, 33, 42, 49]

Te dhenat e derguara te klienti ===== >> Koha e tanishme eshte: 2019-04-17 23:48:04.289701

Te dhenat e derguara te klienti ===== >> Numri i 9 ne serine fibonacci eshte: 34

Te dhenat e derguara te klienti ===== >> 130 kilowatt jane te barabarte me 174.332871549695 horsepower

Te dhenat e derguara te klienti ===== >> Vlerat e zareve jane: 5 dhe 6

Te dhenat e derguara te klienti ===== >> Vlerat e zareve jane: 5 dhe 2

Te dhenat e derguara te klienti ===== >> Numri 17 eshte numer i thjeshte!
```

```
C:\Python\Python37\python.exe
***** FIEK-TCP Serveri *****
Serveri startoi ne localhost me IP adrese: 192.168.0.23 ne portin: 12000
Serveri eshte i gatshem te pranoj kerkesa.
Klienti me IP adrese 127.0.0.1 dhe me numrin e portit 53317 eshte lidhur me server

Te dhenat e derguara te klienti ===== >> IP adresa e klientit: 127.0.0.1

Te dhenat e derguara te klienti ===== >> Numri i portit te klientit eshte: 53317

Te dhenat e derguara te klienti ===== >> Numri i bashketingelloreve ne fjalen e dhene eshte: 6

Te dhenat e derguara te klienti ===== >> Fjalja e printuar: printimi hello

Te dhenat e derguara te klienti ===== >> Emri i hostit nuk mund te percaktohet!

Te dhenat e derguara te klienti ===== >> Koha e tanishme eshte: 2019-04-15 23:22:52.800097

Te dhenat e derguara te klienti ===== >> Rezultati nga loja: [7, 9, 14, 34, 37, 47, 49]

Te dhenat e derguara te klienti ===== >> Numri i 25 ne serine fibonacci eshte: 75025

Te dhenat e derguara te klienti ===== >> 20 gallons jane te barabarte me 75.70882386342127 liters

Te dhenat e derguara te klienti ===== >> Vlerat e zareve jane: 4 dhe 1

Te dhenat e derguara te klienti ===== >> Numri 10 nuk eshte numer i thjeshte!
```

Përfundimi

Janë dizajnuar, implementuar dhe testuar programet klient dhe server, sipas FIEK-TCP protokollit dhe FIEK-UDP protokollit. Komunikimi i programit të klientit dhe serverit është bërë përmes socket programming. Komunikimi i këtyre dy programeve ka qenë i mundur të bëhet sipas dy protokolleve: TCP dhe UDP. Ku kanë ekzistuar dallime në mënyrën e qasjes në programim dhe në vendosjen e lidhjes ndërmjet socketit të klientit dhe atij të serverit.

Janë krijuar metoda të ndryshme në ndryshme në server të cilat janë të çasshme nga klienti. Klienti dërgon “mesazhe” në server që përmbajnë emrin e metodave, dhe serveri në anën tjetër, në bazë të atij “mesazhi” kthen përgjigjen tek klienti.

Në këtë mënyrë funksionojnë nëntë metodat e parapërcaktuara, plus dy metoda sipas dëshirës, me kushtëzimet e tyre specifike.

Referencat

- Ushtrimet Numerike.
- Computer Networking – A Top-Down Approach, by Jim Kurose and Keith Ross.
- <https://www.geeksforgeeks.org/socket-programming-python/>
- <https://realpython.com/python-sockets/>
- https://www.tutorialspoint.com/python/python_networking.htm
- <http://www.cs.dartmouth.edu/~campbell/cs60/socketprogramming.html>
- https://www.tutorialspoint.com/python/python_networking.htm
- https://www.differen.com/difference/TCP_vs_UDP
- YouTube