UNIVERSITETI I PRISHTINËS

FAKULTETI I INXHINIERISË ELEKTRIKE DHE KOMPJUTERIKE

Tema: Programimi me Soketa

Lënda: Rrjeta Kompjuterike

Studenti: Festim Braha **Mentorët: Prof. Dr. Blerim Rexha**

Nr. ID 190714100099 **MSc. Haxhi Lajqi Phd cand.**

Data: 02/05/2021

Përmbajtja

Hyrja

Tools e përdorura

Si Tools që jane përdorur për realizimin e projektit janë:

- **Gjuha Programuese:** Python 3.9.4
- **Sistemi Operativë:** Microsoft Windows 10 64-bit latest version 20H2
- **Vendi ku është koduar (IDE):** Microsoft Visual Studio Community 2019

Përshkrimi i Projektit

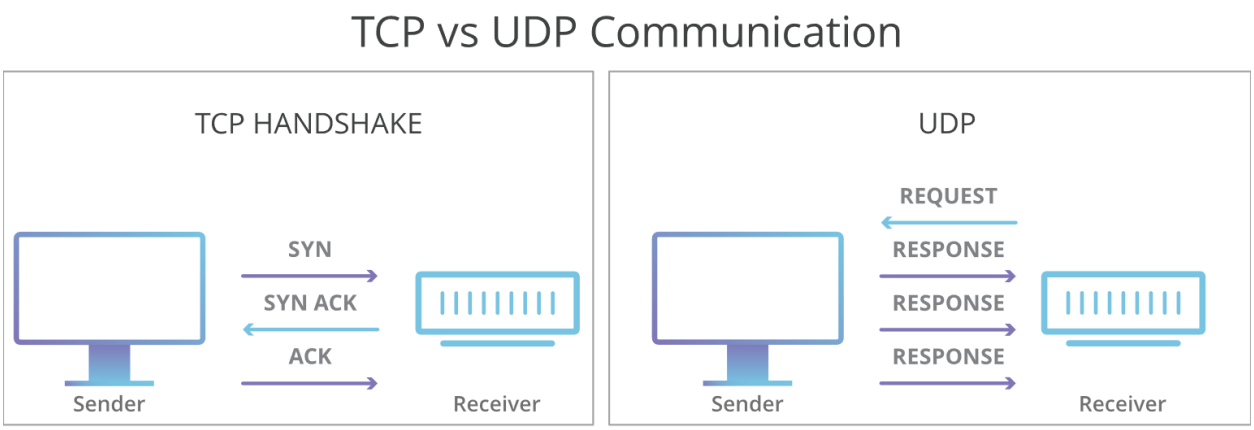
Ky projekt përshkruan dizajnimin, implementimin dhe testimin e komunikimit klient-server përmes socket programming në gjuhën programuese **Python.** Protokolli i përdorur do të jetë ***FIEK*** protokolli i cili u mundëson serverëve dhe klientëve komunikimin përmes dy versioneve të tij: *TCP-protocol* dhe *UDP-protocol*. Në kuadër të këtij projekti përfshihen dy palë Socket-a, ku njëra palë realizohet me anë të protokolit TCP, ndërsa tjetra duke e shfrytëzuar protokolin UDP.

Pra, Socket Programming apo i njohur edhe si Programimi me Soketa, është një mënyrë për t'i lidhur dy nyje në rrjet që ato të komunikojnë mes vete. Socketa identifikohet nga IP adresa si dhe numri i portit, keto dy kompnente jane ato që e mundësojnë komunikimin ndërmejt socketave.

TCP Protokolli

TCP (Transmission Control Protocol) është një protokoll komunikimi i cili e përcakton se si të krijohet e të menaxhohet një lidhje në rrjet, me anë të së cilës, programe të ndryshme mund të shpërndajnë të dhëna ndërmjet vete. TCP konsiderohet si connection-oriented dhe te dhenat i dërgon të renditura sipas një rendi të specifikuar, pra pa e humbur integritetin e të dhënave.

TCP përdoret nga protokolet tjera si HTTP, HTTPs, FTP, SMTP dhe Telnet. Shpejtësia e dërgimit të të dhënave nëpërmjet TCP është më e vogel se ajo e UDP, për shkak të kontrollimeve shtesë që bëhen nga TCP.



TCP protokoli e krijon lidhjen sipas “three way handshake” (SYN, SYN-ACK, ACK).

UDP Protokolli

UDP (User Datagram Protocol) protokolli është gjithashtu protokoll, i cili ka përdorim më të vogël se TCP për arsye se është më pak i besueshëm, sepse ai nuk krijon lidhje fillimisht, mirëpo dërgon më shumë të dhëna në formë të paketave dhe është më i shpejtë

Përdoret kryesisht për në services të ashtuquajtura real time, ku sic dihet në atë rast duhet transmetim i shpejtë i të dhënave (Multiplayer Gaming, Streaming etj). Rreziqet e UDP, si humbja e paketave nuk janë një problem serioz në shumicën e rasteve të përdorimit. Sidoqoftë, UDP mund të shfrytëzohet për qëllime dashakeqe. Meqenëse UDP nuk kërkon një shtrëngim duarsh(handshake), sulmuesit mund të ‘përmbysin’ një server të synuar me trafik UDP pa marrë më parë lejen e atij serveri për të filluar komunikimin

UDP përdoret nga protokole tjera si: DNS, DHCP, TFTP, SNMP, RIP, VOIP. UDP, për dallim nga TCP protokoli, nuk brengoset për renditjen e paketave të të dhënave. Kjo përgjegjesi i mbetet nivelit të aplikacionit.

FIEK-TCP:

Së pari vendoset një lidhje në mes të klientit dhe serverit përmes socket-ave në portin e caktuar(default 14000). Pastaj klienti përmes command line e dërgon kërkesen tek server, i cili nëse kërkesa është valide, i përgjigjet kërkesës specifike të klientit. Serveri është në gjendje të lidhet me më shumë kliente në të njëjtën kohë dhe ti kthej përgjigje klientëve në menyrë paralele

FIEK-UDP:

Në versionin UDP të prokollit FIEK komunikimi realizohet vetëm përmes UDP datagram-ve, ku me ç'rast nuk krijohet një lidhje mes klientit dhe serverit, por komunikimi fillon drejt. Klienti dërgon kërkesën tek serveri përmes një UDP datagram. Pasi që kërkesa të validohet në server ajo poashtu kthehet tek klienti përmes UDP datagram-it. Protokolli FIEK limiton që klienti të dërgon vetëm një kërkesë për datagram.

Metodat dhe përshkrimi i tyre

Lista e Metodave të kërkuara

- ✓ IP
- ✓ NRPORTIT
- ✓ NUMERO
- ✓ ANASJELLTAS
- ✓ PALINDROM
- ✓ KOHA
- ✓ LOJA
- ✓ GCF
- ✓ KONVERTO

Metodat shtese

- ✓ UPPLOWCHAR
- ✓ ROCK_PAPER_SCISSORS
- ✓ RRITE

Metoda IP

Kjo metodë ka për detyrë që të përcaktoje dhe të ktheje IP adresën e klientit

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

METODA NRPORTIT

Kjo metodë kthen portin e klientit,rast i ngjashëm sikur tek IP, përveç se ketu indeksi 1 tek variabla address e përmban portin.

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

METODA NUMËRO

Në këte metodë ka qenë qëllimi që të numërohen zanoret dhe bashkëtingëlloret të dhënë nga përdoruesi, pra teksti jepet si argument në këtë metodë. Metoda është e punuar në dy forma. Së pari me .**lower**() e kthejmë atë tekst në shkronja të vogla për t'a pasur më lehte tek kontrollimi i zanoreve dhe bashkëtingëlloreve (pra varësisht se si e shkruan tekstin përdoruesi). Pra, në metodë kemi dy counter-a, njëri per zanore e tjetri për bashkëtingëllore e ku pastaj në bazë të kushtëzimeve gjendet sa zanore dhe sa bashkëtingëllore janë.

```
def NUMERO(str):

    Znum = 0  #counteri i zanoreve
    Bnum = 0    #counteri i bashketingelloreve


    str = str.lower()
    for i in range(0, len(str)):
        if str[i] in ('a', "e", "i", "o", "u"):
            Znum +=  1;

        elif (str[i] >= 'a' and str[i] <= 'z'):
            Bnum +=  1;
    return Znum, Bnum


def NUMERO(teksti):
    zan = 0
    bashk = 0
    for z in teksti:
        if z in 'aeëiouyAEËIOUY':
            zan = zan + 1
        elif z in "bcdfghjklmnpqrstvxzBCDFGHJKLMNPQRSTVXZ":
            bashk = bashk + 1
    x = "Zanore :" + str(zan)
    y = "Bashketingellore : " + str(bashk)
    return [x, y]
```

METODA ANASJELLTAS

Kjo metodë e kthen në renditje të kundërt ose reverse, tekstin e dhënë nga përdoruesi. Si kërkesë këtu ka qenë që "***Hapësirat në fillim dhe në fund te fjalisë nuk duhet të kthehen***"[1]. E këtë e kemi bërë përmes metodës strip që mundëson largimin e hapësirave(space-ave). Pastaj është përdorur një unazë while, ku ka shkuar karakter për karakter ti kthejë reverse (tek pjesa reverse += s[indeksi - 1] që dmth shkon nga karakteri i fundit e pastaj parafundit e me radhë, deri kur të kthehet mbrapsht teksti)

```
def ANASJELLTAS(teksti):
    s = teksti.strip()
    reverse = ""
    indeksi = len(s)
    while indeksi > 0:
        reverse += s[indeksi - 1]
        indeksi = indeksi - 1
    return reverse
```

METODA PALINDROM

Kjo metodë ka qenë për te kontrolluar nëse teksti i dhënë nga përdoruesi lexohet njejtë edhe nga fillimi edhe nga fundi, pra a është fjalë palindrome. Në kushtëzimin if kemi thenë se a është teksti i dhënë i njejtë sikur të kthehej mbrapsht, pra (`if (string == string[::-1]):` "Pjesa e kodit te string[::-1] eshte teksti i kthyer mbrapsh(reverse)"

```
def PALINDROM(string):
    if (string == string[::-1]):
        return "Eshte Palindrom"
    else:
        return "Nuk eshte Palindrom"
```

METODA KOHA

Kjo metodë na e kthen kohën aktuale në momentin që thirret. Për këtë metode kam shfrytezuar funskionet localtime, strftime nga moduli **time** në Python
(`import string`
`from time import localtime, strftime`).

```
def KOHA():
    return strftime("%Y-%m-%d %H:%M:%S", localtime())
```

METODA LOJA

Tek kjo metodë kemi pasur për detyrë që të bëjmë një listë me numra të rëndomtë dhe ata pastaj të jenë të sortuar nga më i vogli deri te më i madhi. Për gjenerimin e numrave në mënyrë të rëndomtë është përdorur moduli random, përkatësisht **funksioni randint(),** ku si parameter i ka numrat nga 1 deri ne 35. E pastaj që të mos kemi vlera të njejta në listë apo duplicate, e kemi bërë kushtëzimin që nëse numri është në listë, të vazhdohet me një numër tjetër (kushti `if nr in numrat: continue)`

```
def LOJA():
        numrat = []
        while len(numrat) <= 5:
            nr = randint(1,35)
            if nr in numrat:
                continue
            else:
                numrat.append(nr)
        numrat.sort()
        return ' '.join(str(v) for v in numrat)
```

METODA GCF

Kjo metodë na kthen faktorin(pjestuesin) më të madh të përbashkët ndërmjet dy numrave të dhënë si parametra. Pastaj bëhen kushtëzimet përkatëse se a kanë numrat e dhënë ndonjë pjestues të përbashkët, e nëse jo përgjigjja kthehet 1. Krijo një sekuencë numrash nga numri1 deri në 1 , por rritje(zbritje) me -1

```
def GCF(num1, num2):
    num1 = int(num1)
    num2 = int(num2)
    if num1 > num2:
        num1, num2 = num2, num1
    for x in range(num1, 0, -1):
        if num1 % x == 0 and num2 % x == 0:
            return str(x)
```

METODA KOVERTO

Metoda Konverto fillimisht merr si paramentra operacionin(ne cilën njësi do të kthehet) si dhe rezultati që fitohet nga njësitë përktatëse nga kthimi nga një njësi në tjetrën. Pra varësisht nga operacioni do të marrim rezultatet përkatëse. Konvertimi është bërë sipas formulave përkatëse të sistemit të njësive SI.

```python
def KONVERTO(operacioni, rezultati):

    rezultati = float(rezultati)

    if operacioni == "cmNeInch":
        return str(rezultati * 0.393701)
    elif operacioni == "inchNeCm":
        return str(rezultati * 2.52)
    elif operacioni == "mileNeKm":
        return str(rezultati * 1.609)
    elif operacioni == "kmNeMiles":
        return str(rezultati / 1.609)

    else:
        return "Keni gabuar operacionin"
```

METODAT SHTESE

METODA UPPLOWCHAR

Në metodën e parë shtesë kam paraqitur funksionin i cili tregon se sa shkronja janë të mëdha dhe sa janë të vogla, në momentin kur përdoruesi jep nje tekst të caktuar. Këtu kam përdorur metodat e Python-it **isupper()** dhe **islower()** për të testuar se sa karaktere nga teksti janë të mëdha ose të vogla. Janë 2 counter-a, njëri për numrimin e shkronjave të mëdha (variabla e inicializuar, u = 0) dhe njëri për numërimin e shkronjave të vogla (l = 0)

```python
def UPPLOWCHAR(s):
    u = 0;
    l = 0;
    for c in s:
        if c.isupper():
            u += 1
        elif c.islower():
            l += 1
    return u, l
```

METODA ROCK_PAPER_SCISSORS

Kjo metodë e ka logjikën e lojës ROCK PAPER SCISSORS, ku varësisht nga lëvizja e përdoruesit cila është dhe vlerës së rastësishme nga PC zgjedhet fituesi përkatës nga rregullat e lojës që janë. Pra, në bazë të kushtëzimeve të dhëna zgjidhet fituesi.

```python
def ROCK_PAPER_SCISSORS(Zgj_jone):
    Zgj_jone = Zgj_jone.lower()
    if Zgj_jone == 'rock':
        zgjedhja = 1
    elif Zgj_jone == 'paper':
        zgjedhja = 2
    elif Zgj_jone == 'scissors':
        zgjedhja = 3
    else:
        return "Keni gabuar gjate shtypjes se opsionit! Vendos rock, paper apo scissors nese deshironi te vazhdoni

    Pc_zgj = random.randint(1, 3)

    if Pc_zgj == 1:
        comp_Zgj_jone = 'Rock'
    elif Pc_zgj == 2:
        comp_Zgj_jone = 'paper'
    else:
        comp_Zgj_jone = 'scissor'

    if (zgjedhja == 1 and Pc_zgj == 3) or (zgjedhja == 2 and Pc_zgj == 1) or (zgjedhja == 3 and Pc_zgj == 2):
        return "Ju keni fituar. Kompjuteri zgjodhi " + comp_Zgj_jone
    elif (zgjedhja == Pc_zgj):
        return "Barazim. Edhe kompjuteri zgjodhi " + comp_Zgj_jone
    else:
        return "Fitoi kompjuteri . Kompjuteri zgjodhi " + comp_Zgj_jone
```

METODA RRITE

Kjo metodë e ka për detyrë që tekstin i cili jepet nga user-i në shkronja të vogla, të kthehet i gjithe teksti në shkronja të mëdha. Fillimisht e kam përdorur .lower() për shkak se nëse brenda tekstit ka një ose 2 shkronja të mëdha e tjerat të vogla, të kthehen njëherë të gjitha në të vogla, e pastaj me **.upper()** i kthejmë në shkronja të mëdha.

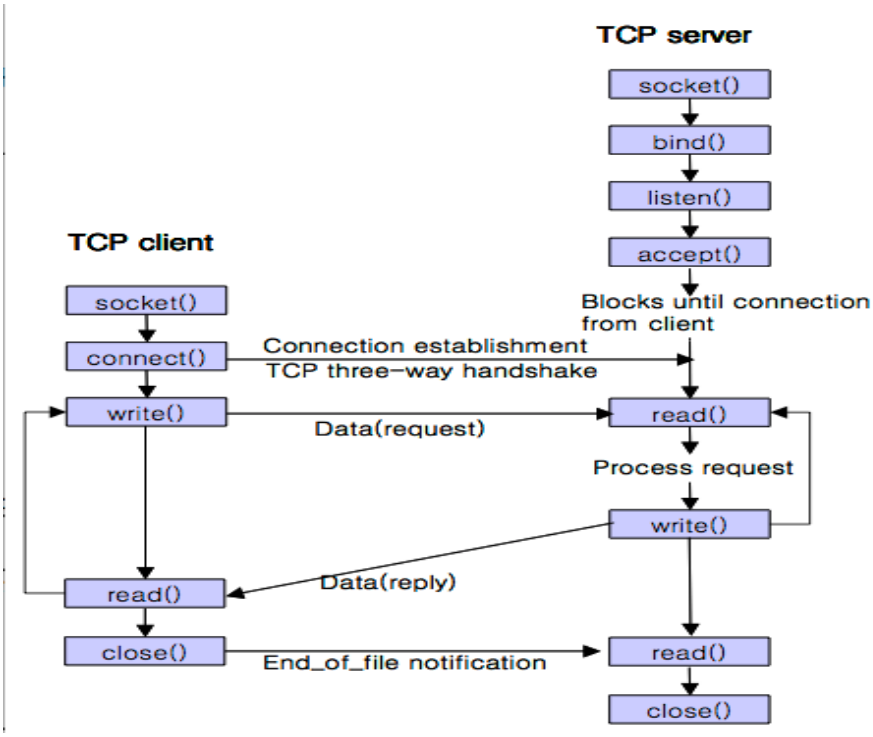
```python
def RRITE(teksti):
    type(teksti)
    return teksti.lower().upper()
```

FIEK TCP SERVER

Serveri është console aplication i cili lidhet me klientin përmes soketave dhe për secilen lidhje të re në server krijohet një thread i ri. Serveri dallon llojet e kërkesave të cilat vijnë nga klienti dhe për secilin lloj të kërkesave të cilat i parashtron klienti serveri kthen një përgjigje të veqantë. Serveri është aplikacioni i cili gjendet në një unazë të përhershme e cila nuk e lejon të ndalet edhe në qoftë se kërkesat e klientit janë jovalide, pra është aktiv gjatë gjithë kohës, përpos disa rasteve kur ndalet nga përdoruesi me komanda të veqanta. Soketi mbetet i hapur derisa klienti të kërkoj të çkyçet.

```
from socket import *
import random
import datetime
import sys
import threading
import string
from time import localtime, strftime
```

Moduli socket është moduli kyç i cili përdoret përgjatë gjithë projektit, modulet e tjera të cilat përdoren janë datetime, random të cilat na ndihmojnë gjatë aplikimit të metodave si koha e cila duhet të kthejë kohën në server dhe loja e cila na kthen 5 numra të papërsëritura dhe të sortuar, po ashtu moduli tjetër është edhe thread i cili na lejon të startojmë nga një thread të ri per cdo klient të ri i cili kyçet në server.



Së pari i kemi shkruar metodat, pastaj më poshtë është krijuar socket-i përkatës i këtij serveri, i cili do të na mundësojë lidhjen me klientët. Krijimi i socket-it që përkrah protokolin TCP është bërë në këtë

mënyrë (**SOCK_STREAM** tregon që kemi të bëjmë me protokolin TCP, ndërsa **AF_INET** tregon që kemi të bëjmë me tipin e IP adresave IPv4):

```
SRVPORT = 14000
serverSocket = socket(AF_INET, SOCK_STREAM)
```

Pastaj, pas krijimit të socket-it, ne duhet t'a bëjmë të gatshëm të pranojë nga jashtë me anë të funksionit bind(), sipas Hostit dhe Portit përkatës:

```
serverSocket.bind(('', SRVPORT))
```

E pastaj kodojmë metodën **listen()** e që e lejon serverin të dëgjojë kërkesat që i vijnë nga klientët. Në metodën **listen()** vendoset si parameter numri maksimal i klientëve që dëshirojme të i dëgjojë Serveri në kohën e njejtë. Në rastin tone 5.

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

E pastaj e kemi edhe metoden **accept**(), me të cilën i pranojmë kërkesat nga Klienti

```
while 1==1:

    connectionSocket, addr = serverSocket.accept()
    print('Klienti me IP-ne %s dhe me numer te portit %s eshte lidhur me server' % (addr))

    threading._start_new_thread(newClient, (connectionSocket, addr))
```

Trajtimi i disa klientëve në të njejtën kohë na mundësohet nëpërmjet **modulit threading**, i cili secilin klient të lidhur në server e ndan si një thread në vete, duke e bërë këtë proces shumë më të shpejtë.

Definimi i Klientëve e kemi sipas funksionit më poshtë, si dhe është bërë edhe trajtimi i gabimeve. Në fund e kemi funksionin **close(),** që të përfundojë. Me anë të recv() ne i marrim të dhënat që dërgohen, ku si parametër vendoset buffersize i lidhjes. Të dhënat e pranuara më tutje dekodohen dhe përdoren për validimin e kërkesës.

```
def newClient(connectionSocket, addr):
    kerkesa = (bytes)("empty".encode())
    try:
        while str(kerkesa.decode()).upper() != "EXIT" and str(kerkesa.decode()) != "":

            kerkesa = connectionSocket.recv(128)
            kerkesaStr = str(kerkesa.decode()).strip()
            kerkesaArray = kerkesaStr.split(' ')
            kerkesaArray[0] = kerkesaArray[0].upper()
```

```
  except Exception as mesazhi:
      print("\n Keni gabuar, shiko edhe njehere kerkesen: ")
      print(str(mesazhi))
      connectionSocket.close()
```

VALIDIMI I METODAVE NE FIEK TCP SERVER

Më poshtë është bërë validimi i metodave të kërkuara, ku nëse klienti gabon ose ndonje gjë tjeter i lejohet mundësia për përmirësime si rrjedhojë e kushtëzimeve specifike.

```
        #Metoda IP
        if kerkesaArray[0] == "IP":
            if len(kerkesaArray) == 1:
                connectionSocket.send(("IP adresa juaj si klient eshte: " + IP(addr)).encode())
                print(("Klienti ka kerkuar metoden IP dhe IP e tij eshte: " + IP(addr)))
                print("--------------------------------------------------------------------------------")
            else:
                connectionSocket.send("Keni gabuar metoden apo ajo nuk ekziston. Provoni Perseri!".encode())
                print("Keni gabuar, shiko perseri ...")
        #Metoda NRPORTI
        elif kerkesaArray[0] == "NRPORTIT":
            if len(kerkesaArray) == 1:
                connectionSocket.send(("Numri i portit tuaj eshte: " + NRPORTIT(addr)).encode())
                print(("Klienti ka kerkuar metoden NRPORTI dhe PORTI i tij eshte: " + NRPORTIT(addr)))
                print("--------------------------------------------------------------------------------")
            else:
                connectionSocket.send("Keni gabuar metoden apo ajo nuk ekziston. Provoni Perseri!".encode())
                print("Keni gabuar, shiko perseri ...")
```

```
    # Metoda NUMERO
elif kerkesaArray[0] == "NUMERO":
    if len(kerkesaArray) == 2:
        var1, var2 = NUMERO(kerkesaArray[1])
        var1 = str(var1)
        var2 = str(var2)
        connectionSocket.send(
            ("Numri i zanoreve eshte: " + var1 + " Numri i bashtingllorеve " + var2).encode())
        print(("Numri i zanoreve dhe i bashtingllorеve eshte : " + var1 + " " + var2))
        print("--------------------------------------------------------------------------------")
    else:
        connectionSocket.send("Keni gabuar metoden apo ajo nuk ekziston. Provoni Perseri!".encode())
        print("Keni gabuar, shiko perseri ...")

 # metoda ANASJELLTAS
elif kerkesaArray[0] == "ANASJELLTAS":
    if len(kerkesaArray) == 2:
        connectionSocket.send(("Fjala e kthyer mbrapsht eshte : " + ANASJELLTAS(kerkesaArray[1])).encode())
        print(("Klienti ka kerkuar metoden Anasjelltas dhe fjala e kthyer mbrapsht eshte: " + ANASJELLTAS(kerkesaArray[1])))
        print("--------------------------------------------------------------------------------")
    else:
        connectionSocket.send("Keni gabuar metoden apo ajo nuk ekziston. Provoni Perseri!".encode())
        print("Keni gabuar, shiko perseri ...")
```

```
        # metoda PALINDROM
        elif kerkesaArray[0] == "PALINDROM":
            if len(kerkesaArray) == 2:
                connectionSocket.send(("Fjala e dhene: " + PALINDROM(kerkesaArray[1])).encode())
                print(("Klienti ka kerkuar metoden Palindrom dhe fjala e dhene: " + PALINDROM(kerkesaArray[1])))
                print("--------------------------------------------------------------------------------")
            else:
                connectionSocket.send("Keni gabuar metoden apo ajo nuk ekziston. Provoni Perseri!".encode())
                print("Keni gabuar, shiko perseri ...")
        #Metoda KOHA
        elif kerkesaArray[0] == "KOHA":
            if len(kerkesaArray) == 1:
                connectionSocket.send(("Koha tani per momentin eshte: " + KOHA()).encode())
                print(("Klienti ka kerkuar metoden KOHA dhe koha momentale eshte:" + KOHA()))
                print("--------------------------------------------------------------------------------")
            else:
                connectionSocket.send("Keni gabuar metoden apo ajo nuk ekziston. Provoni Perseri!".encode())
                print("Keni gabuar, shiko perseri ...")
       #Metoda LOJA
        elif kerkesaArray[0] == "LOJA":
            if len(kerkesaArray) == 1:
                connectionSocket.send(("Rezultatet nga loja: " + LOJA()).encode())
                print(("Klienti ka kerkuar metoden LOJA dhe rezultati eshte:" + LOJA()))
                print("--------------------------------------------------------------------------------")
            else:
                connectionSocket.send("Keni gabuar metoden apo ajo nuk ekziston. Provoni Perseri!".encode())
                print("Keni gabuar, shiko perseri ...")
```

```
    # metoda GCF
    elif kerkesaArray[0] == "GCF":
        if len(kerkesaArray) == 3:
            connectionSocket.send(("GCF eshte : " + GCF(kerkesaArray[1], kerkesaArray[2])).encode())
            print(("Klienti ka kerkuar metoden GCF dhe GCF e dy numrave eshte:" + GCF(kerkesaArray[1], kerkesaArray[2])))
            print("--------------------------------------------------------------------------------")
        else:
            connectionSocket.send("Keni gabuar metoden apo ajo nuk ekziston. Provoni Perseri!".encode(clientAddress),
                clientAddress)
            print("Keni gabuar, shiko perseri ...")
    #Metoda KONVERTO
    elif kerkesaArray[0] == "KONVERTO":
        for i in range(len(kerkesaArray)):
            if "" in kerkesaArray:
                kerkesaArray.remove("")
        if len(kerkesaArray) > 3 or len(kerkesaArray) < 3:
            connectionSocket.send( ("Keni gabuar metoden apo ajo nuk ekziston. Provoni Perseri!").encode())
            print("Keni gabuar, shiko perseri ...")
        else:
            Konvertimi = str(kerkesaArray[1]).lower().split("ne")
            pergjigja = kerkesaArray[2] + " " + str(Konvertimi[0]) + " jane te barabarte me " + KONVERTO(
                str(kerkesaArray[1]), kerkesaArray[2]) + " " + str(Konvertimi[1])
            connectionSocket.send(str(pergjigja).encode())
            print(pergjigja)
            print("--------------------------------------------------------------------------------")
```

```
        # metodat shtese

        elif kerkesaArray[0] == "UPPLOWCHAR":
            if len(kerkesaArray) == 2:
                var1,var2 = UPPLOWCHAR(kerkesaArray[1])
                var1 = str(var1)
                var2 = str(var2)
                connectionSocket.send(("Jane "+ var1 +" shkronja te medha dhe "+ var2 +" shkronja te vogla").encode())
                print(("Numri i shkronjave te medha dhe i shkronjave te vogla eshte : " + var1 + " " + var2))
                print("--------------------------------------------------------------------------")
            else:
                connectionSocket.send("Keni gabuar metoden apo ajo nuk ekziston. Provoni Perseri!".encode())
                print("Keni gabuar, shiko perseri ...!")



         # metoda ROCK_PAPER_SCISSORS
        elif kerkesaArray[0] == "ROCK_PAPER_SCISSORS":
            if len(kerkesaArray) == 2:
                connectionSocket.send((ROCK_PAPER_SCISSORS(kerkesaArray[1])).encode())
                print(("Klienti ka kerkuar metoden ROCK_PAPER_SCISSORS dhe fituesi eshte " + ROCK_PAPER_SCISSORS(kerkesaArray[1])))
                print("--------------------------------------------------------------------------")
            else:
                connectionSocket.send("Keni gabuar metoden  apo ajo nuk ekziston. Provoni Perseri!".encode())
                print("Keni gabuar, shiko perseri ...")


        # metoda RRITE
        elif kerkesaArray[0] == "RRITE":
            if len(kerkesaArray) == 2:
                connectionSocket.send(("Fjala e kthyer ne Shkronja te medha eshte: " + RRITE(kerkesaArray[1])).encode())
                print(("Klienti ka kerkuar metoden RRITE dhe fjala e kthyer ne shkrnja te medha eshte eshte: " + RRITE(kerkesaArray[1])))
                print("--------------------------------------------------------------------------")
            else:
                connectionSocket.send("Keni gabuar metoden apo ajo nuk ekziston. Provoni Perseri!".encode())
                print("Keni gabuar, shiko perseri ...")
```

FIEK TCP KLIENT

Klienti është një command line e cila pas startimit e pyet klientin se a dëshiron të konektohet në server me anë të vlerave default ('localhost',13000) apo të tjera vlera. Klientit i mundësohet shkëmbim i informatave në rrugën server – klient varësisht se çfarë kërkesa i parashtron klienti tek serveri dhe varësisht shërbimeve të cilat i permban serveri.

Tek ana e klientit nuk mund të mungojë moduli i soketave, pa të cilin dë të ishte i pamundur lidhja ndëmjet këtyre. Në rast se klienti dëshiron që t'a mbyll programin duhet që të shtyp **EXIT**.

```
from socket import *
import sys

print("------------------------------FIEK_TCP_KLIENT-------------------------------------------")
print("------------------------------Lenda Rrjeta Kompjuteriek - Profesoret: Blerim Rexha dhe Haxhi Lajqi")
serverName = '127.0.0.1'  #127.0.0.1 ose mund te shkruanii localhost
SRVPORT = 14000


s = socket(AF_INET, SOCK_STREAM)

s.connect((serverName,SRVPORT))
kerkesa = ""
print("\n")
print("Jeni lidhur ne serverin ",serverName," ne portin ",SRVPORT)
# MENYJA e programit
print("\nZgjedhni ndonjeren nga metodat: ")
print("-IP\n-NRPORTIT\n-NUMERO [hapsire] teksti]\n-ANASJELLTAS [hapsire} teksti \n-PALINDROM [hapsire] teksti]\n-KOHA\n-LOJ

print("\n")

print("Nese deshironi te ndalni programin, shkruani ne console EXIT")
```

Try except është shtuar për ti parandaluar gabimet eventuale gjatë lidhjes së klientit me server, ku përdoren Host dhe Port, që nëse janë dhënë gabim nga klienti, dështon procesi i krijimit të socketit, me ç'rast thirret edhe një herë funksioni në vetëvete.

```
while 1 == 1: #ketu e kemi bere nje while loop dmth while true(qe i bie 1==1 eshte true)

    try:
        kerkesa = input('Shkruani kerkesen dhe opsionet nese ajo metode ka: ')
        if kerkesa!="" and kerkesa.upper()!="EXIT":

            s.sendall(str(kerkesa).encode())
        else:
            break

        data = s.recv(128)
        print(str(data.decode()).strip())
        print("-------------------------------------------------------------------------------------\n")
    except Exception as mesazhi:
        print(str(mesazhi))
        break

s.close()
```

Gjithashtu mund të shihet se ne e shikojmë gjatësinë sepse nuk duhet të këtë më shume se 128 karaktere në rast se nuk vlen ky kusht, atëherë vazhdojmë pjesën tjeter duke e ruajtur inputin apo kërkesën e klientit që duam të dërgojmë në server, por para se të dërgohet kërkesa e klientit në server ne për siguri duhet ta validojmë atë edhe nga ana e serverit në mënyrë që të dërgojmë kërkesa valide dhe që të marrim përgjigje nga ana e serverit. . Në rast se kërkesa e klientit nuk gjendet në shërbimet te cilat na i ofron serveri, atëherë na kthehet një mesazh i cili na tregon që kerkesa nuk është valide. Në rastin më lartë shihet se si klienti i mirepret dhe i dekodon përgjigjet e serverit dhe na i shfaq ato në ekran.

Kur dalim nga while loop, thirret metoda për mbylljen e lidhjes së klientit me server: **close()**

FIEK UDP SERVER

Për dallim nga TCP, tek UDP nuk është përdorur **multithreading** për t'i trajtar secilin klient me thread të veçantë mirëpo, pasi që UDP konsiderohet si "connectionless", nuk ka rëndësi se sa lidhje ekzistojnë apo se cili klienti po shërbehet. Prandaj, kodi i UDP serverit është paksa më i thjeshtë se ai tek TCP.

Realizimi i klientit këtu të protokoli UDP është i njëjtë si tek TCP, përveç dallimeve esenciale si në vend të **SOCK_STREAM** kemi **SOCK_DGRAM**.

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

Për dallim nga nga TCP, UDP nuk përdor funksioni **listen()**. Pastaj, pas krijimit të socket-it, ne duhet t'a bëjmë të gatshëm të pranojë nga jashtë me anë të funksionit bind(), sipas Hostit dhe Portit përkatës njejtë sikurse te TCP.

```
serverSocket.bind(('', SRVPORT))
```

Vërejtje: Në UDP nuk janë paraqitur metodat pasi që janë të njejtat sikur ne TCP (shih më lartë)

```
while 1==1:
    kerkesa = (bytes)("empty".encode())
    try:
        while str(kerkesa.decode()).upper()!="EXIT" and str(kerkesa.decode())!="":


            kerkesa, clientAddress = serverSocket.recvfrom(128)
            kerkesaStr = str(kerkesa.decode()).strip()
            kerkesaArray = kerkesaStr.split(' ')
            kerkesaArray[0] = kerkesaArray[0].upper()
```

Ngjashëm sikur te TCP vetem se në këtë pjesë e shtojmë clientAddressen, pasi që në UDP nuk e kemi multithreading. Dallim mes TCP dhe UDP është se në vend të **send** kemi komanden **sendto** e cila merr 2 parametra mesazhin të cilin dëshrijomë ta dërgojmë dhe adresën, si dhe tek TCP përdoret **recv** ndërsa në UDP sic shihet është **recvfrom.** Dallim tjetër që vlen të theksohet është se kur pranohet kërkesa me recvfrom(), variabla e parë këtu e përmban atë që e ka dërguar klienti, ndërsa e dyta e përmban adresën e klientit. Kjo pasi që tek UDP nuk kemi **accept().**

```
    # metoda IP
    if kerkesaArray[0]=="IP":
        if len(kerkesaArray) == 1:
            serverSocket.sendto(("IP adresa juaj si klient eshte: "+IP(clientAddress)).encode(), clientAddress)
            print(("Klienti ka kerkuar metoden IP dhe IP e tij eshte: "+IP(clientAddress)))
            print("----------------------------------------------------------------------------------------")
        else:
            serverSocket.sendto("Kerkesa juaj eshte gabim ose nuk ekziston, ju lutem provoni perseri!".encode(), clientAddress)
            print(" Keni gabuar, shiko perseri ...")

     # metoda NRPORTIT
    elif kerkesaArray[0]=="NRPORTIT":
        if len(kerkesaArray) == 1:
            serverSocket.sendto(("Numri i portit tuaj eshte: "+NRPORTIT(clientAddress)).encode(), clientAddress)
            print(("Klienti ka kerkuar metoden NRPORTI dhe PORTI i tij eshte: "+NRPORTIT(clientAddress)))
            print("----------------------------------------------------------------------------------------")

        else:
            serverSocket.sendto("Kerkesa juaj eshte gabim ose nuk ekziston, ju lutem provoni perseri!".encode(), clientAddress)
            print("Keni gabuar, shiko perseri ...")

       #metoda NUMERO
    elif kerkesaArray[0]=="NUMERO":
        if len(kerkesaArray) == 2:
            var1, var2 = NUMERO(kerkesaArray[1])
            var1 = str(var1)
            var2 = str(var2)
            serverSocket.sendto(("Numri i zanoreve eshte : " + var1 + " Numri i bashtinglloreve " + var2).encode(),clientAddress)
            print(("Numri i zanoreve dhe i bashtinglloreve eshte : " + var1 + " zanore dhe " + var2 + " bashketingellore"))
            print("----------------------------------------------------------------------------------------")
        else:
            connectionSocket.sendto("Kerkesa juaj eshte gabim apo nuk ekziston, ju lutem provoni perseri!".encode(), clientAddress)
            print("Keni gabuar, shiko perseri ...")

            #metoda ANASJELLTAS
    elif kerkesaArray[0]=="ANASJELLTAS":
        if len(kerkesaArray) == 2:
            serverSocket.sendto(("Fjala e kthyer mbrapsht eshte : " + ANASJELLTAS(kerkesaArray[1])).encode(),clientAddress)
            print(("Klienti ka kerkuar metoden Anasjelltas dhe fjala e kthyer mbrapsht eshte: " + ANASJELLTAS(kerkesaArray[1])))
            print("----------------------------------------------------------------------------------------")
        else:
            connectionSocket.sendto("Kerkesa juaj eshte gabim apo nuk ekziston, ju lutem provoni perseri!".encode(clientAddress), clientAddress)
            print("Keni gabuar, shiko perseri ...")
```

```
            #metoda PALINDROM
    elif kerkesaArray[0]=="PALINDROM":
        if len(kerkesaArray) == 2:
            serverSocket.sendto(("Fjala e dhene eshte : " + PALINDROM(kerkesaArray[1])).encode(), clientAddress)
            print(("Klienti ka kerkuar metoden Palindrom dhe fjala e dhene: " + PALINDROM(kerkesaArray[1])))
            print("----------------------------------------------------------------------------------------")
        else:
            connectionSocket.sendto("Kerkesa juaj eshte gabim apo nuk ekziston, ju lutem provoni perseri!".encode(), clientAddress)
            print("Keni gabuar, shiko perseri ...")

         # metoda KOHA
    elif kerkesaArray[0]=="KOHA":
        if len(kerkesaArray) == 1:
            serverSocket.sendto(("Koha e tanishme eshte: " + KOHA()).encode(), clientAddress)
            print(("Koha e tanishme eshte: " + KOHA()))
        else:
            serverSocket.sendto("Kerkesa juaj eshte gabim ose nuk ekziston, ju lutem provoni perseri!".encode(), clientAddress)
            print("Keni gabuar, shiko perseri ...")
    # metoda LOJA
    elif kerkesaArray[0]=="LOJA":
        if len(kerkesaArray) == 1:
            serverSocket.sendto(("Rezultati nga loja: " + LOJA()).encode(), clientAddress)
            print(("Rezultati nga loja: " + LOJA()))
        else:
            serverSocket.sendto("Kerkesa juaj eshte gabim ose nuk ekziston, ju lutem provoni perseri!".encode(), clientAddress)
            print("Keni gabuar, shiko perseri ...")

     #metoda GCF
    elif kerkesaArray[0]=="GCF":
        if len(kerkesaArray) == 3:
            serverSocket.sendto(("GCF eshte : " + GCF(kerkesaArray[1],kerkesaArray[2])).encode(), clientAddress)
            print(("GCF eshte : " + GCF(kerkesaArray[1],kerkesaArray[2])))
        else:
            connectionSocket.sendto("Kerkesa juaj eshte gabim apo nuk ekziston, ju lutem provoni perseri!".encode(clientAddress), clientAddress)
            print("Keni gabuar, shiko perseri ...")
```

```
    # metoda KONVERTO
    elif kerkesaArray[0] == "KONVERTO":
        for i in range(len(kerkesaArray)):
            if "" in kerkesaArray:
                kerkesaArray.remove("")
        if len(kerkesaArray) > 3 or len(kerkesaArray) < 3:
            serverSocket.sendto( ("Kerkesa juaj eshte gabim apo nuk ekziston, ju lutem provoni perseri!").encode(), clientAddress)
            print("GABIM / ERROR")
        else:
            Konvertimi = str(kerkesaArray[1]).lower().split("ne")
            pergjigja = kerkesaArray[2] + " " + str(Konvertimi[0]) + " jane te barabarte me " + KONVERTO(
                str(kerkesaArray[1]), kerkesaArray[2]) + " " + str(Konvertimi[1])
            serverSocket.sendto(str(pergjigja).encode(),clientAddress)
            print(pergjigja)
            print("--------------------------------------------------------------------------")

    # metodat shtese
    # metoda UPLOWCHAR
    elif kerkesaArray[0] == "UPPLOWCHAR":
        if len(kerkesaArray) == 2:
            var1,var2 = UPPLOWCHAR(kerkesaArray[1])
            var1 = str(var1)
            var2 = str(var2)
            serverSocket.sendto(("Jane: "+ var1 +" shkronja te medha dhe "+ var2 +" shkronja te vogla").encode(),clientAddress)
            print(("Numri i shkronjave te medha dhe te vogla eshte:"+ var1 + " te medha dhe " + var2 + " te vogla"))


    elif kerkesaArray[0]=="ROCK_PAPER_SCISSORS":
        if len(kerkesaArray) == 2:
            serverSocket.sendto((ROCK_PAPER_SCISSORS(kerkesaArray[1])).encode(), clientAddress)
            print(("Klienti ka kerkuar metoden ROCK_PAPER_SCISSORS dhe fituesi eshte: " + ROCK_PAPER_SCISSORS(kerkesaArray[1])))
            print("----------------------------------------------------------------------------------------")
        else:
            connectionSocket.sendto("Kerkesa juaj eshte gabim apo nuk ekziston, ju lutem provoni perseri!".encode(), clientAddress)
            print("Keni gabuar, shiko perseri ...")
```

```
    elif kerkesaArray[0]=="RRITE":
        if len(kerkesaArray) == 2:
            serverSocket.sendto(("Fjala e kthyer ne Shkronja te medha eshte: "  + RRITE(kerkesaArray[1])).encode(), clientAddress)
            print(("Klienti ka kerkuar metoden RRITE dhe fjala e kthyer ne shkronja te medha eshte eshte: " + RRITE(kerkesaArray[1])))
            print("----------------------------------------------------------------------------------------")
        else:
            connectionSocket.sendto("Kerkesa juaj eshte gabim apo nuk ekziston, ju lutem provoni perseri!".encode(), clientAddress)
            print("Keni gabuar, shiko perseri ...")
```

Edhe në këtë pjesë është bërë trajtimi i gabimeve.

```
except Exception as mesazhi:
    print("\n Keni gabuar, shiko perseri ...: ")
    print(str(mesazhi))
```

FIEK UDP KLIENT

Fillimisht krijohet soketi si gjithmonë sipas një funksioni të krijuar, njejtë sikur tek TCP klienti vetë se këtu kemi `s = socket(AF_INET, SOCK_DGRAM),` pra ne vend te **SOCK_STREAM** kemi **SOCK_DGRAM**.

Në UDP nuk e përdorim funksionin listen(), për dallim nga TCP ku vendoset si parameter numri maksimal i klientëve që dëshirojme të i dëgjojë Serveri në kohën e njejtë.

```
from socket import *

print("-------------------------------FIEK_UDP_KLIENT-----------------------------------------")
print("--------------------------------Lenda Rrjeta KompjuteriKE - Profesoret: Blerim Rexha dhe Haxhi Lajqi")
serverName = '127.0.0.1'  #127.0.0.1 ose mund te shkruanii localhost
SRVPORT = 14000

s = socket(AF_INET, SOCK_DGRAM)

print("\n")

print("Jeni lidhur ne serverin ",serverName," ne portin ",SRVPORT)


print("\nZgjedhni ndonjeren nga metodat: ")
print("-IP\n-NRPORTIT\n-NUMERO [hapsire] teksti\n-ANASJELLTAS [hapsire] teksti \n-PALINDROM [hapsire] teksti\n-KOHA\n-LOJA\

print("Nese deshironi te ndalni programin, shkruani ne console EXIT")
```

Try except është shtuar për ti parandaluar gabimet eventuale gjatë lidhjes së klientit me server, ku përdoren Host dhe Port, që nëse janë dhënë gabim nga klienti, dështon procesi i krijimit të socketit.

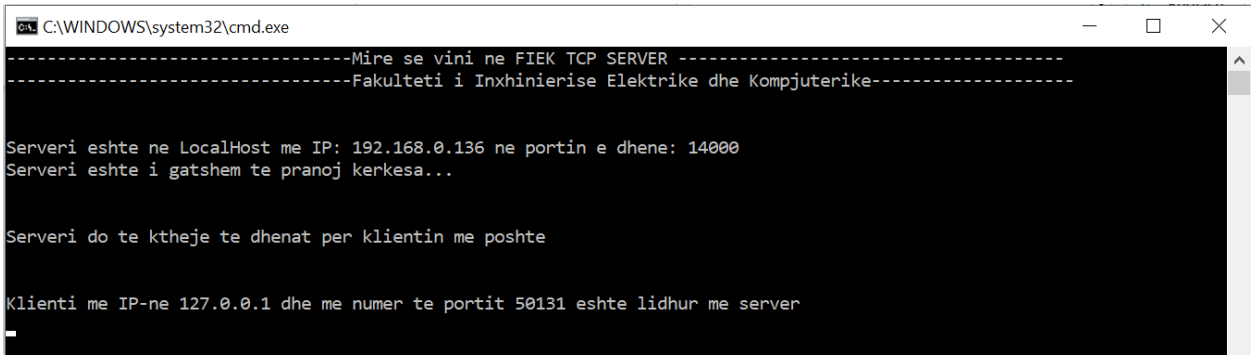
```
while 1==1: #ketu e kemi bere nje while loop dmth while true(qe i bie 1==1 eshte true)
    try:
        kerkesa = input('Shkruani kerkesen dhe opsionet nese ajo metode ka: ')
        if kerkesa!="" and kerkesa.upper()!="EXIT":

            s.sendto(str(kerkesa).encode(), (serverName, SRVPORT))
        else:
            break

        data, serverAddress = s.recvfrom(128)
        print('Te dhenat e pranuara nga serveri: ')
        print(str(data.decode()).strip())
        print("-----------------------------------------------------------------------------------------\n")
    except Exception as mesazhi:
        print(str(mesazhi))
        break

s.close()
```

Për dallim nga TCP ku unë e kam shkruajtur `data = s.recv(128),` ne UDP është dashur të shkruajme `data, serverAddress = s.recvfrom(128),` për arsye se në UDP nuk kemi përdorur multithreading pasi

që UDP konsiderohet si “connectionless” dhe nuk ka rëndësi se sa lidhje ekzistojnë apo se cili klienti po shërbehet. Kemi thënë që nëse kërkesa është zbrazët(pra nëse shtypet direct **ENTER**) dhe nëse në console shkruhet **EXIT**, atëherë lidhja në mes klientit dhe Serverit dështon.Vërehet se në UDP pëdoret **sendto** për dallim nga TCP klienti, ku **sendto** merr 2 parametra mesazhin(kërkesën) të cilin dëshrijomë ta dërgojmë dhe adresën përkatëse.

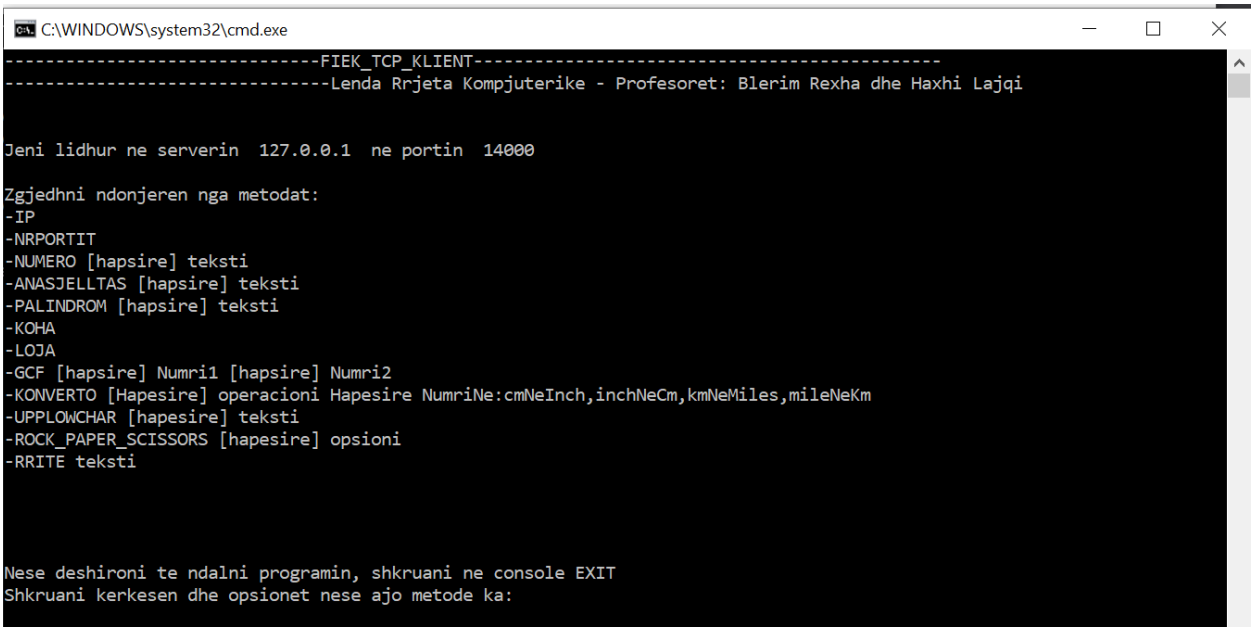
TESTIMET

Në momentin e ekzekutimit të kodit për **Serverin** na shfaqet kjo pjesë në Console



Pra, në atë moment që klienti të kyqet në Server do të paraqitet se klienti me IP dhe me numër të portit është lidhur.

Në anën e klientit paraqitet fillimisht kjo pjesë në Console



Serveri pret tash ndonjë kërkesë nga klienti e që duhet t'a shkruajmë emrin e kërkesës si më poshtë:

```
Nese deshironi te ndalni programin, shkruani ne console EXIT
Shkruani kerkesen dhe opsionet nese ajo metode ka: ip
IP adresa juaj si klient eshte: 127.0.0.1
--------------------------------------------------------------------------------

Shkruani kerkesen dhe opsionet nese ajo metode ka: nrportit
Numri i portit tuaj eshte: 50131
--------------------------------------------------------------------------------

Shkruani kerkesen dhe opsionet nese ajo metode ka: numero FestimBraha
Numri i zanoreve eshte: Zanore :4 Numri i bashtinglloreve Bashketingellore : 7
--------------------------------------------------------------------------------

Shkruani kerkesen dhe opsionet nese ajo metode ka: anasjelltas FestimBraha
Fjala e kthyer mbrapsht eshte : aharBmitseF
--------------------------------------------------------------------------------

Shkruani kerkesen dhe opsionet nese ajo metode ka: palindrom kimik
Fjala e dhene: Eshte Palindrom
--------------------------------------------------------------------------------

Shkruani kerkesen dhe opsionet nese ajo metode ka: koha
Koha tani per momentin eshte: 2021-05-02 18:10:37
--------------------------------------------------------------------------------

Shkruani kerkesen dhe opsionet nese ajo metode ka: loja
Rezultatet nga loja: 2 7 21 23 28 34
```

```
Nese deshironi te ndalni programin, shkruani ne console EXIT
Shkruani kerkesen dhe opsionet nese ajo metode ka: gcf 12 4
GCF eshte : 4
--------------------------------------------------------------------------------

Shkruani kerkesen dhe opsionet nese ajo metode ka: konverto cmNeInch 500
500 cm jane te barabarte me 196.8505 inch
--------------------------------------------------------------------------------

Shkruani kerkesen dhe opsionet nese ajo metode ka: upplowchar FESTIMbraha
Jane 6 shkronja te medha dhe 5 shkronja te vogla
--------------------------------------------------------------------------------

Shkruani kerkesen dhe opsionet nese ajo metode ka: ROCK_PAPER_SCISSORS rock
Fitoi kompjuteri . Kompjuteri zgjodhi paper
--------------------------------------------------------------------------------

Shkruani kerkesen dhe opsionet nese ajo metode ka: rrite festim
Fjala e kthyer ne Shkronja te medha eshte: FESTIM
--------------------------------------------------------------------------------

Shkruani kerkesen dhe opsionet nese ajo metode ka:
```

Ndërsa pas kërkesve nga klienti në server është paraqitur kjo pjesë:

```
Serveri eshte ne LocalHost me IP: 192.168.0.136 ne portin e dhene: 14000
Serveri eshte i gatshem te pranoj kerkesa...


Serveri do te ktheje te dhenat per klientin me poshte


Klienti me IP-ne 127.0.0.1 dhe me numer te portit 50131 eshte lidhur me server
Klienti ka kerkuar metoden IP dhe IP e tij eshte: 127.0.0.1
--------------------------------------------------------------------------
Klienti ka kerkuar metoden NRPORTI dhe PORTI i tij eshte: 50131
--------------------------------------------------------------------------
Numri i zanoreve dhe i bashtinglloreve eshte : Zanore :4 Bashketingellore : 7
--------------------------------------------------------------------------
Klienti ka kerkuar metoden Anasjelltas dhe fjala e kthyer mbrapsht eshte: aharBmitseF
--------------------------------------------------------------------------
Klienti ka kerkuar metoden Palindrom dhe fjala e dhene: Eshte Palindrom
--------------------------------------------------------------------------
Klienti ka kerkuar metoden KOHA dhe koha momentale eshte:2021-05-02 18:10:37
--------------------------------------------------------------------------
Klienti ka kerkuar metoden LOJA dhe rezultati eshte:3 4 22 26 27 29
--------------------------------------------------------------------------
Klienti ka kerkuar metoden GCF dhe GCF e dy numrave eshte:4
--------------------------------------------------------------------------
500 cm jane te barabarte me 196.8505 inch
--------------------------------------------------------------------------
Numri i shkronjave te medha dhe i shkronjave te vogla eshte : 6 5
--------------------------------------------------------------------------
```

```
Klienti me IP-ne 127.0.0.1 dhe me numer te portit 52870 eshte lidhur me server
Fitoi kompjuteri . Kompjuteri zgjodhi paper
--------------------------------------------------------------------------
Klienti ka kerkuar metoden RRITE dhe fjala e kthyer ne shkrnja te medha eshte eshte: FESTIM
--------------------------------------------------------------------------
```

KONKLUZIONET

Përmes këtij projekti ne kemi bërë dizajnimin Klient – Server duke e zhvilluar programin me anë të soketave.

Qëllimi i këtij projektit është arritur, pasi është krijimi i protokoleve FIEK-TCP dhe FIEK-UDP që lejojnë krijimin e një lidhje ndërmjet klientit dhe serverit, në atë mënyrë që klienti të parashtroj kërkesa tek serveri dhe ai të i kthejë përgjigje varësisht nga kërkesa e parashtruar.

Në rastin kur arrihet lidhja e klientit me server, komunikimi i tyre vazhdon deri në atë moment kur klienti dëshiron t'a ndërprej këtë lidhje.

Çdo kërkesë që nuk i përputhet rregullave të përcaktuara nëpërmjet validimit injorohen dhe i tregohet klientit se kërkesa nuk është valide. Kërkesat jovalide nuk dërgohen në server, kjo pasi që kam mbajtur serverin sa më pak të ngarkuar, e sa më të gatshëm për të gjithë klientët aktiv. Anës së serverit i mbetet vetëm ta kontrollojë kërkesën e pranuar nga klienti dhe të thërrasë metodën përkatëse në përputhje me të. Po ashtu, serveri është siguruar me try except (ashtu edhe klienti), të cilat i parandalojnë, ose thënë më mirë, i trajtojnë dhe shumëkund edhe lajmërojnë, për gabimet (errors) që mund të na shfaqen eventualisht në program.

Shërbimet të cilat i ofron serveri ose metodat të cilat i përmban serveri dhe nëse i kërkojmë ato serveri është i gatshëm të na kthejë përgjigje janë: IP, NRPORTIT, NUMERO, ANASJELLTAS, PALINDROM, KOHA, LOJA, GFC, KONVERTO (cmNeInch, inchNeCm, kmNeMiles, mileNeKm). përveç meotdave të lartëcekura serveri përmban edhe 3 metoda shtesë të implementuara nga unë e të cilat janë: UPPLOWCHAR, ROCK_PAPER_SCISSORS, RRITE. Në anën e TCP serverit gjithashtu është e implementuar edhe MULTITHREADING, e cila lejon të pranojë kërkesa dhe të kthej përgjigje dy e më shumë klientëve ne të njëjtën kohë. Dallimi kryesor në këtë projekt ndërmjet TCP dhe UDP është se TCP për këtë shfrytëzon modulin threading, për t'i trajtuar të gjithë klientët njëkohësisht, ndërsa për ta arritur këtë gjë tek UDP nuk është shfrytëzuar moduli threading, kjo pasi që UDP, duke e marrë parasysh që konsiderohet si "connectionless", nuk ka nevojë për një gjë të tillë, pasi që të gjithë klientët e kyçur në të trajtohen pa problem njëkohësisht.

Sa i përket gjërave të reja që fitova gjatë këtijë projekti ishte një njohuri më e madhe rreth Programimit me Soketa, si dhe një Gjuha Programuese Python, e cila ishte mjaft gjuhë e përshtatshme dhe tërheqëse për të punuar. Si përvojë ishte mjaftë e mire, pasi që edhe ato konceptet teorike që i kishim, i zbatonim menjëherë në praktikë dhe kuptohej më lehte lënda.

Nga kërkesat e parashtruara nga Profesori: **MSc Haxhi Lajqi, Phd cand** dhe nga testimet e bëra mund ë them se i kam përmbushur kërkesat e këtij projekti.

REFERENCAT

[SP01]: GeekForGeeks – Socket Programming with Python

https://www.geeksforgeeks.org/socket-programming-python/ *[Qasur: 20/04/2021]*

[PY02]: Documentation of Socet Programming from Python - Low-level networking interface

https://docs.python.org/3/library/socket.html [Qasur: 25/04/2021]

[*PS03]:* Python Socket Programming Examples - Loyola Marymount University

https://cs.lmu.edu/~ray/notes/pythonnetexamples/ *[Qasur me 28/04/2021]*

https://cse.lmu.edu/department/computerscience/ *[Qasur me 28/04/2021]*

[*WL04]:* W3SCHOOLS – Learn Python

https://www.w3schools.com/python/default.asp *[Qasur me 25/04/2021]*