UNIVERSITETI I PRISHTINËS

FAKULTETI I INXHINIERISË ELEKTRIKE DHE KOMPJUTERIKE

DEPARTAMENTI: KOMPJUTERIKË

LËNDA: RRJETAT KOMPJUTERIKE

PROJEKTI I: PROGRAMIMI KLIENT-SERVER

Mentorët:

Prof. Dr. Blerim Rexha

Msc. Haxhi Lajqi

Studenti:

Fortesa Mujaj

PRISHTINË, 2021

PËRMBAJTJA

HYRJE

Projekti i realizuar në lendën “Rrjeta Kompjuterike” kishte për qellim të krijonte lidhjen klient-server. Kjo lidhje klient-server u realizua përmes *sockets* (në gjuhën shqipe: *soketat*), ku si gjuhë programuese për realizimin e kesaj lidhjeje unë përdora gjuhën Python.
Soketat, në rrjeta kompjuterike paraqesin pikat fundore apo “endpoints” të një lidhjeje komunikimi dykahëshe ndërmjet dy programeve që ekzekutohen në një rrjet kompjuterik. Një “endpoint” apo pikë fundore është një kombinim i një IP adrese (*IP address*) dhe një porti (*Port Number*), ku cdo lidhje TCP mund të identifikohet në menyrë unike përmes këtyre dy pikave fundore. Pra, kështu mund të krijojmë lidhje të shumta ndërmjet host-it dhe serverit.

Detyra jonë, në këtë projekt ishte krijimi i një TCP Protokolli dhe një UDP Protokolli të emëruar FIEK, ky protokoll është një protokoll shumë i thjeshtë i cili e lejon klientin dhe serverin ti testoj lidhjet e tyre. TCP versioni i tij quhet FIEK-TCP dhe UDP versioni i tij quhet FIEK-UDP.
Më poshtë janë shënuar detaje të tjera në lidhje me projektin, përkatësisht: veglat e përdorura për zhvillim dhe verzioni i tyre, sistemet operative ku është kryer testimi i programeve si dhe versioni i tyre dhe lista e metodave të implementuara.

GJUHA E PËRDORUR:

- *Python Programming Language*

VERSIONI I GJUHËS SË PËRDORUR:

- *Version: Python 3.9.3 – April 4, 2021*

VEGLA E PËRDORUR:

- *Microsoft Visual Studio Community 2019*

SISTEMI OPERATIV:

- *Microsoft Windows 10 Home 64-bit*

LISTA E METODAVE TE IMPLEMENTUARA:

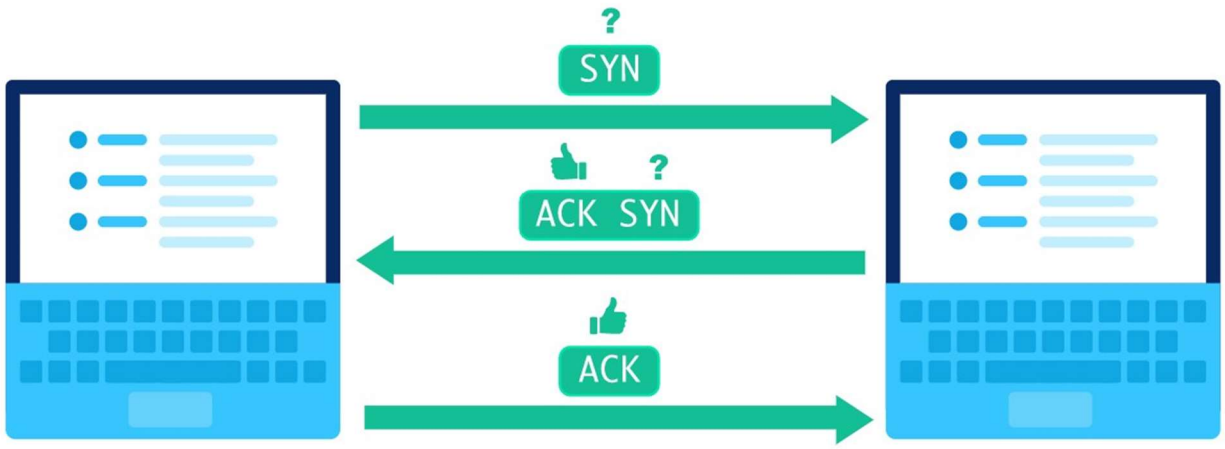
- *IP*
- *NRPORTIT*
- *NUMERO*
- *ANASJELLTAS*
- *KOHA*
- *LOJA*
- *GCF*
- *KONVERTO*
- *ASTRO*
- *DUPLIKAT*

PËRSHKRIMI

Protokolli FIEK përmban këto kërkesa (metoda): *IP, NRPORT, NUMERO, ANASJELLTAS, PALINDROM, KOHA, LOJA, KONVERTO, GFC* si dhe dy metoda tjera të krijuara shtesë që është metoda *ASTRO* dhe metoda *DUPLIKAT* te cilat dërgohen nga klienti tek serveri. Serveri, pastaj përgjigjet me një mesazh i cili është specifik për secilën kërkesë (metodë), dhe ai injoron kërkesat jovalide duke mos dështuar në rast se pranon një kërkesë te tillë.

FIEK-TCP PROTOCOL

TCP apo Transmission Control Protocol është një protokoll komunikimi i orientuar drejt lidhjës që lehtëson shkëmbimin e mesazheve ndërmjet pajisjeve kompjuterike në nje rrjet. Ky protokoll është shumë i zakonshëm në rrjetet që përdorin IP apo Internet Protocol-in dhe së bashku referohen si TCP/IP.



Kur dy kompjuterë dëshirojnë të transmetojnë data apo të dhena njëri-tjetrit përmes TCP, atëherë ata në fillim duhet të krijojnë një lidhje që quhet ***three-way handshake***.

Në kohët e sotme TCP dhe IP janë rregullat kryesore qe e përcaktojnë internetin. Programimi me sockets fillon atëherë kur importohet libraria socket dhe krijohet objekti.

FIEK-TCP CLIENT

Ashtu sic është cekur edhe më lartë lidhja në mes klientit dhe serverit arrihet përmes programimit me sockets, kështu tek FIEK-TCP Client në fillim është importuar libraria socket përmes komandës `import socket`. Pastaj është krijuar funksioni apo metoda `client_program()` me anë të cilit mundësojmë krijimin e socket-it në anën e klientit dhe lidhjen e tij me serverin përkatës. Te kodi i paraqitur më poshtë kemi dhënë IP adresen e klientit, portin default ku ai është i qasshëm dhe pastaj përmes `socket.socket(socket.AF_INET, socket.SOCK_STREAM)` kemi krijuar një socket object. `socket.AF_INET` paraqet familjen e adresës kurse `socket.SOCK_STREAM` paraqet tipin e socket-it. Kur bëjmë këtë, protokolli default që do të përdoret do të jetë protokolli TCP apo Transmission Control Protocol. Përmes kodit të shkruar `client_socket.connect((host, port))` është formuar lidhja ne mes klientit dhe serverit.

```
def client_program():
    print("FIEK-TCP Protocol - Client \n")
    print(
      "-----------------------------------------------------------------\n"
      "1. IP\n"
      "2. NRPORTIT\n"
      "3. NUMERO\n"
      "4. ANASJELLTAS\n"
      "5. PALINDROM\n"
      "6. KOHA\n"
      "7. LOJA\n"
      "8. GCF numri1 numri2\n"
      "9. KONVERTO cmneinch/inchnecm/kmnemiles/milenekm numri\n"
      "10. ASTRO\n"
      "11. DUPLIKAT\n")

    host = socket.gethostname()
    port = 14000
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect((host, port))
```

Sipas kërkesave pastaj është dhënë edhe mundësia që nëse dëshironi të nderroni adresën dhe portin atëherë shtyp numrin 1, përndryshe në rast se refuzoni shtyp cdo numër tjetër përvec numrit 1. Kur klienti shtyp numrin 1, atij i kërkohet që të vendosi IP adresën dhe portin e ri. Në rast të paraqitjes së ndonje gabimi me krijimin e socket-it atëherë paraqitet një mesazh dhe raportohet gabimi që ka shkaktuar punën jo të rregullt të klientit, po ashtu prap i ipet mundësia shfytëzuesit të ndërroj adresën dhe portin. Mirëpo, në rast të mosparaqitjes së cfardo gabimi, shfrytezuesit i paraqitet mesazhi për zgjedhjen e një operacioni nga disa nga operacionet të listuara me lartë.

```
ndrr = int(input("Shtypni 1 nese deshironi te nderroni adresen dhe portin:
").strip())

    while ndrr == 1:
        host_changed = input("IP adresa: ")
        port_changed = int(input("Porti: "))
        while port > 14000:
            port = int(input("Shkruaj nje numrer porti tjeter:"))
        try:
            client_socket.connect((host_changed, port_changed))
            break
        except (socket.error, OverflowError, ValueError) as error:
            print("Gabime gjate krijimit te socket-it: " + str(error))
            ndrr = int(input("Shtypni 1 nese deshironi te nderroni adresen
dhe portin: ").strip())

    message = input("Operacioni: ")
```

Nëse shfrytëzuesi apo klienti nuk e jep mesazhin “exit” atëherë mesazhi i dhënë nga shfrytëzuesi kalon nepër disa “verifikime”. Në rastin e parë nëse mesazhi i dhënë nga shfrytezuesi nuk është valid atëherë në konzollën e tij do ti kërkohet që të provoj përseri. Ky validim është kryer përmes një funksioni apo metode të krijuar qe quhet $isValid()$, ku si parameter do të ketë mesazhin e dhënë nga klienti. Përndryshe nese ai mesazh është valid atëherë përmes një funksioni të quajtur $operation()$ varësisht të kërkesës së klientit do të kthehet mesazhi. Nëse mesazhi është “Provo prapë!” klientit i tregohet që komanda nuk është shkruar si duhet dhe duhet të provoj operacionin përseri, përndryshe nëse mesazhi është njera nga opsionet e dhëna në fillim atëherë ky opsion i dërgohet serverit përmes komandës client_socket.send(op.encode()). Mesazhi që vjen nga serveri pastaj ruhet në variablen data dhe ky mesazh printohet. Ne rast se opsioni i dhënë nga klienti është “exit” kjo lidhje në mes klientit dhe serverit ndërpritet.

```
    while message.lower().strip() != 'exit':
        if not isValid(message.upper()):
            message = input(" Komanda nuk eshte valide. Provoni operacionin
perseri: ")
            continue
        else:
            op = operation(message.lower())
            if op == 'Provo Prape!':
                message = input(" Komanda nuk eshte shkruar si duhet! Provoni
operacionin perseri: ")
                continue
            client_socket.send(op.encode())
            data = client_socket.recv(128).decode()
            print('Pergjigja nga serveri: ' + data + "\n\n")
        message = input("Operacioni: ")

    client_socket.close()
```

Pra, përmes metodës $isValid()$, ne shikojmë nëse mesazhi i dhënë nga klienti është valid. Nëse mesazhi i parë nga “shumë mesazhet” e dhëna prej klientit nuk gjendet në array-in commands atëherë kjo tregon që mesazhi nuk është valid përndryshe ai është valid.

```
def isValid(message):
    if message.split(' ')[0] not in commands:
        return False
    else:
        return True
```

Metoda $concat()$ është një metodë tjetër e cila është përdorur sidomos tek metoda operation. Kjo metodë është përdorur për bashkimin e opsionit të zgjedhur nga klienti dhe një mesazhi tjetër te dytë, sic është rasti te ‘numero’, ‘anasjelltas’, ‘palindrom’, ku klientit i kërkohet vendosja edhe një mesazhi tjetër për realizimin e funksionit.

```
def concat(text, message):
    if len(text) <= 0:
        return 'Provo Prape!'
    else:
        message = message + ' ' + text
        return message
```

```
def operation(message):
    cmd_message = message.split(' ')[0].lower().strip()

    if cmd_message == 'ip':
        return message

    elif cmd_message == 'nrportit':
        return message

    elif cmd_message == 'numero':
        if len(message.split(' ')) == 1:
            text = input("Teksti? ")
            text = concat(text, cmd_message)
        else:
            text = 'Provo Prape!'
        return text

    elif cmd_message == 'anasjelltas' and  len(message.split(' ')) == 1:
        text = input("Teksti? ")
        return concat(text, cmd_message)

    elif cmd_message == 'palindrom' and  len(message.split(' ')) == 1:
        text = input("Teksti? ")
        return concat(text, cmd_message)

    elif cmd_message == 'koha':
        return message

    elif cmd_message == 'loja':
        return message

    elif cmd_message == 'konverto':
        message_arr = message.split(' ')
        if not len(message_arr) == 3:
            return 'Provo Prape!'
        else:
            if message_arr[1] not in convertions:
                return 'Provo Prape!'
            elif not message_arr[2].isnumeric():
                return 'Provo Prape!'
            else:
                return message

    elif cmd_message == 'gcf':
        message_arr = message.split(' ')
        if not len(message_arr) == 3:
            return 'Provo Prape!'
        elif message_arr[1].isnumeric() and message_arr[2].isnumeric():
            return message
        else:
            return 'Provo Prape!'

    elif cmd_message == 'astro':
```

```
        if len(message.split(' ')) == 1:
            text = int(input(" Dita e lindjes? "))
            if text < 1 or text > 32:
                print("Dita duhet te shkruhet si numer ndermjet vlerave 1 dhe
32")
                return 'Provo Prape!'
            else:
                text_concat = concat(str(text), cmd_message)
            text = int(input("Muaji i lindjes? "))
            if text < 1 or text > 12:
                print("Muaji duhet te shkruhet si numer ndermjet vlerave 1
dhe 12")
                return 'Provo Prape!'
            else:
                text_concat =  concat(str(text), text_concat)
            return text_concat
        else:
            return 'Provo Prape!'

    elif cmd_message == 'duplikat':
        if len(message.split(' ')) == 1:
            text = input("Teksti? ")
            return concat(text, cmd_message)
        else:
            return 'Provo Prape!'
    else:
        return 'Provo Prape!'
```

FIEK-TCP SERVER

Programi server i degjon kërkesat të cilat vijnë nga klienti dhe i kthen përgjigje klientit. Serveri është në gjendje që të pranojë kërkesa nga disa klient. Arsyeja pse ndodh kjo është sepse gjatë krijimit të programit FIEK-TCP Server, ne kemi përdorur dukurinë e quajtur multithreading. Në vijim është paraqitur kodi ku tregohet nje funksion i quajtur server_program, në këtë metodë përmes $host = socket.gethostname()$ kam caktuar IP adresën e serverit, kurse përmes $port = 14000$ kam caktuar portin.

```
def server_program():
    print("FIEK-TCP Protocol - Server \n")

    host = socket.gethostname()
    port = 14000
    server_socket = socket.socket()
    server_socket.bind((host, port))
    server_socket.listen(2)

    while True:
        conn, address = server_socket.accept()
        print("Lidhja nga: " + str(address))
        thread = threading.Thread(target=server_threaded, args=(conn,
address))
        thread.start()
        print("Lidhjet aktive: ",  threading.activeCount() - 1)
    conn.close()
```

Është zgjedhur porti 14000 për shkak se ashtu na janë dhënë edhe kërkesat për projektimin dhe ekzekutimin e programit FIEK-TCP Server. Pastaj është krijuar objekti server_socket, njejtë sic është krijuar edhe te client_socket. Te $server\_socket.bind((host, port))$ kemi metodën bind e cila është metodë e klasës socket të Python-it dhe cakton një IP adresë dhe një numër porti në një instancë socket. Përvec kesaj kemi edhe $server\_socket.listen(2)$, ku metoda listen bën gati një socket për pranimin e lidhjeve. Kjo metodë duhet të thirret para se të thirret metoda $accept()$ dhe ajo pranon një numer si parametër, ky numër tregon numrin maksimal të lidhjeve që mund të rreshtohen për një socket nga sistemi operativ.

Në funksionin e mëposhtëm $server\_threaded$ është bërë pranimi, shqyrtimi dhe validimi i kërkesave të ardhura nga klienti. Ky funksion thirret për cdo thread të ri që krijohet për secilin klient unik. Në rast të ndonjë gabimi atehërë paraqitet mesazhi “E dhëna është gabim!”.

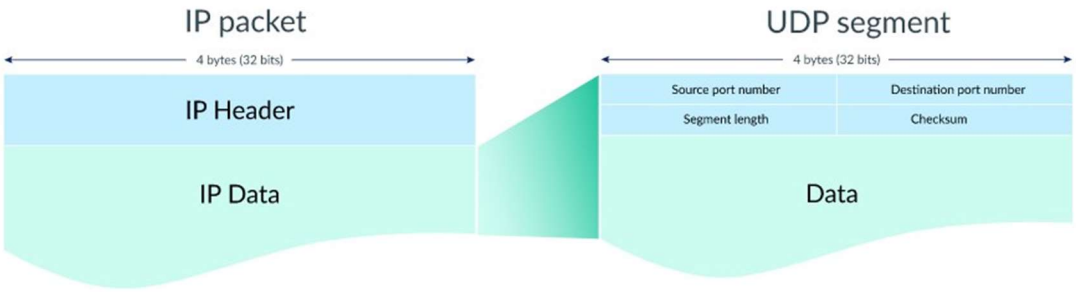
```
def server_threaded(conn, address):
    while True:
        data = conn.recv(128).decode()
        if not data:
            break
        data = data.split(' ')

        if data[0].upper() == 'IP':
            text = "IP address is: " + address[0]
            send(conn, str(text))
        elif data[0].upper() == 'NRPORTIT':
            text = "Port number is: " + str(address[1])
            send(conn, str(text))
        elif data[0].upper() == 'NUMERO':
            send(conn, numero(data))
        elif data[0].upper() == 'ANASJELLTAS':
            send(conn, anasjelltas(data))
        elif data[0].upper() == 'PALINDROM':
            send(conn, palindrom(data))
        elif data[0].upper() == 'KOHA':
            send(conn, str(koha(data)))
        elif data[0].upper() == 'LOJA':
            send(conn, loja(data))
        elif data[0].upper() == 'GCF':
            send(conn, gcf(data))
        elif data[0].upper() == 'KONVERTO':
            send(conn, konverto(data))
        elif data[0].upper() == 'ASTRO':
            send(conn, astro(data))
        elif data[0].upper() == 'DUPLIKAT':
            send(conn, duplikat(data))
        else:
            send(conn, "E dhena eshte gabim! ")
        print("Kerkesa nga klienti: " + str(data[0]))
```

FIEK-UDP PROTOCOL

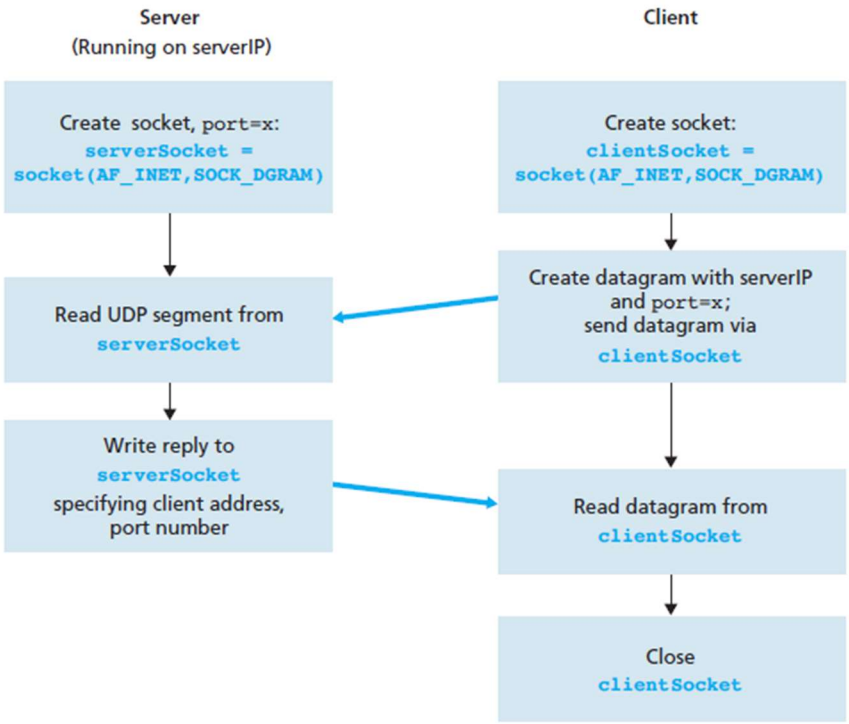
User Datagram Protocol apo UDP – paraqet një protokoll komunikimi që lehtëson shkëmbimin e mesazheve midis pajisjeve kompjuterike në një rrjet.

Është një alternativë për Transmission Control Protocol apo (TCP). Në një rrjet që përdor Internet Protocoll-in apo (IP), nganjëherë është referuar si UDP/IP.



Kur dërgoni pako duke përdorur UDP mbi IP, pjesa e të dhënave për secilën paketë IP formatohet si një segment UDP. Secili segment UDP përmban një header i cili eshte 8 bajt dhe të dhëna me gjatësi të ndryshueshme.

Gjithsesi, edhe User Datagram Protocol është po ashtu një protokoll i cili përcakton mënyrën e komunikimit mes dy pikave në rrjet. Për dallim nga TCP, ky i transmeton të dhënat mes dy pikave të ndryshme të cilat nuk kanë nevojë të lidhen paraprakisht. Nuk e bën kontrollimin e paketave të të dhënave të cilat shkëmbehen mes klientit dhe serverit, pra në rast të humbjes së paketave marrësit nuk i dërgohen përsëri.



FIEK-UDP CLIENT

Tek FIEK-UDP Client, përmes rreshtit `UDPclient = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)` ne krijojmë socket-in e klientit të quajtur UDPclient. Parametri i parë, sic përmendëm edhe tek FIEK-TCP Client paraqet familjen e adresës; kështu AF_INET do të thotë rrjeti kompjuterik përdor famljen e adresave IPv4. Parametri i dytë tregon qe tipi i socket-ave është SOCK_DGRM, që do të thotë që nuk kemi të bëjmë me një TCP socket, por me nje UDP socket.

```
try:
        UDPclient = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    except (socket.error, ValueError, OverflowError) as error:
        print("Gabim gjate krijimit te socket-it: ", error)
```

Përmes rreshtit të parë të paraqitur më poshtë, ne i japim mundesinë klientit të ndërroj adresën edhe portin sipas deshirës. Nëse klienti apo shfrytëzuesi shtyp 1 atëherë ai ka mundësinë e ndryshimit, në rast se ai nuk shtyp numrin 1 por numrat tjerë atehere ai pajtohet me portin dhe IP adresën e tij të dhënë në fillim. Pra, te kjo pjesë e kodit tentohet të krijohet lidhja në mes klientit dhe serverit, mirëpo në rast të dështimit të krijimit të kësaj lidhjeje për arsyje të ndryshme na kthehet një error ose exception dhe paraqitet mesazhi te klienti.

```
ndrr = int(input("Shtypni 1 nese deshironi te nderroni adresen dhe portin:
").strip())
    while ndrr == 1:
        host = input("IP adresa: ")
        port = int(input("Porti: "))
        while port > 14000:
            port = int(input("Shkruaj nje numer porti tjeter: "))
        try:
            UDPclient = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
            break
```

Edhe këtu njejtë sikurse tek TCP, nëse mesazhi i klientit nuk është “exit” atëherë programi do të verifikoj se a është ky mesazh valid apo jo, ashtu sic u sqarua edhe më parë tek FIEK-TCP Client.

```
while message.lower().strip() != 'exit':
        if not isValid(message.upper()):
            message = input("Komanda nuk eshte valide. Provoni operacionin
perseri: ")
            continue
        else:
            op = operation(message.lower())
            if op == 'Provo Prape!':
                message = input("Komanda nuk eshte shkruar si duhet! Provoni
operacionin perseri: ")
                continue
            UDPclient.sendto(op.encode(), (host, port))
            received_data, address = UDPclient.recvfrom(128)
            data = received_data.decode()
            data = str(data)
            print("\nPergjigjja nga serveri: " + data + "\n\n")
        message = input("Operacioni: ")
```

FIEK-UDP SERVER

Serveri tek UDP protokolli nuk pret që të krijohet lidhja, thjeshtë pret që të arrijë datagrami. Datagram i cili arrin tek klienti, përmban edhe adresën e dërguesit dhe kjo ia mundëson serverit që të dijë se ku ta dërgojë përgjigjen saktësisht.

```
def server_program():

    print("FIEK-UDP Protocol - Server \n")

    host = 'localhost'
    port = 14000

    print(
        "-------------------------------------------------------------\n")

    try:
        UDPserver = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    except (socket.error, ValueError, OverflowError) as error:
        print("Gabime gjate krijimit te socket-it! ", error)
    UDPserver.bind((host, port))

    print("Serveri eshte startuar ne localhost me portin: " + str(port))
```

Përmes `send(UDPserver, str(text), address)` dërgohet mesazhi në parametrin e dytë me anë të lidhjes UDPserver e cila është krijuar paraprakisht.

```
while True:
        received_data, address = UDPserver.recvfrom(128)
        data = received_data.decode()
        if not data:
            break
        data = data.split(' ')
        if data[0].upper() == 'IP':
            text = "IP address is: " + address[0]
            send(UDPserver, str(text), address)
        elif data[0].upper() == 'NRPORTIT':
            text = "Port number is: " + str(address[1])
            send(UDPserver, str(text), address)
        elif data[0].upper() == 'NUMERO':
            send(UDPserver, numero(data), address)
        elif data[0].upper() == 'ANASJELLTAS':
            send(UDPserver, anasjelltas(data), address)
        elif data[0].upper() == 'PALINDROM':
            send(UDPserver, palindrom(data), address)
        elif data[0].upper() == 'KOHA':
            send(UDPserver, str(koha(data)), address)
        elif data[0].upper() == 'LOJA':
            send(UDPserver, loja(data), address)
        elif data[0].upper() == 'GCF':
            send(UDPserver, gcf(data), address)
        elif data[0].upper() == 'KONVERTO':
            send(UDPserver, konverto(data), address)
        elif data[0].upper() == 'ASTRO':
            send(UDPserver, astro(data), address)
        elif data[0].upper() == 'DUPLIKAT':
            send(UDPserver, duplikat(data), address)
        else:
            send(UDPserver, "E dhena eshte gabim! ", address)
        print("Kerkesa nga klienti: " + str(data[0]))
```

Interneti pastaj do të dorëzojë paketën në adresën e klientit. Pasi serveri dërgon paketën, ai mbetet në loop duke pritur për një paketë tjetër UDP për të arritur.

PERSHKRIMI I METODAVE

Janë përdorur gjithsej 11 metoda, 9 prej të cilave janë zbatuar sipas përshkrimit kurse 2 prej të cilave janë zbatuar nga unë.

- "Metoda "*IP*"

*Metoda **IP*** – faktikisht nuk është një metodë apo funksion për shkak sepse nuk më është dukur e nevojshme krijimi i një metode. Mënyra qe une kam zgjedhur per realizimin dhe dërgimin e adresës së IP-së në rast se klienti e kërkon opsionin IP është përmes if-it. If shikon nëse opsioni i kërkuar i klientit është IP, nëse është atëherë mirret nga tuple value address, e cila përmban në vete IP adresën dhe portin e klientit të lidhur dhe ja dërgon klientit adresën e tij të kërkuar.

```
if data[0].upper() == 'IP':
     text = "IP address is: " + address[0]
     send(conn, str(text))
```

- Metoda *NRPORTIT*

*Metoda **NRPORTIT*** – gjithashtu nuk është një metode apo funksion për shkak sepse nuk më është dukur e nevojshme krijimi i nje metode. Mënyra që une kam zgjedhur për realizimin dhe dërgimin e portit është e njejtë me atë të IP adresës. Me anë të kësaj mënyre ne kthejmë portin e klientit përkates në të njejtën mënyrë sikur të metoda IP, përvec se këtu indeksi 1 tek variabla address e përmban portin.

```
elif data[0].upper() == 'NRPORTIT':
     text = "Port number is: " + str(address[1])
     send(conn, str(text))
```

- Metoda *NUMERO*

*Metoda **NUMERO** –* është një metodë apo funksion i cili gjen numrin e zanoreve dhe bashkëtingëlloreve në tekst dhe kthen përgjigjen. Ecuria e kontrollimit, pra është bërë duke kontrolluar për cdo shkronjë për zanore, kurse sa i perket bashktingëllorëve nëpermjet vlerave në ASCII code.

```
def numero(data):
    print(data[1])
    shuma_zanore = 0
    shuma_bashktng = 0
    for shkronja in data[1]:
        if shkronja not in zanore:
            shuma_bashktng += 1
        else:
            shuma_zanore += 1
    return "Teksti i pranuar permban " + str(shuma_zanore) + " zanore dhe " +
str(shuma_bashktng) + " bashketingellore"
```

- Metoda *ANASJELLTAS*

*Metoda **ANASJELLTAS*** – paraqet një funksion i cili kthen fjalinë e shtypur në tekst, gjithashtu hapësirat në fillim dhe në fund të fjalisë nuk kthehen.
Për realizimin e kësaj metode kemi iteruar me një for loop të nje rangu sa gjatësia e tekstit hyrës. Për cdo iterim shtohet shkronja përkatëse e tekstit hyrës tek indeksi [gjatesia e tekstit – 1] nje stringu bosh te krijuar. Ne fund, e kthejmë si rezultat atë string, të cilit i largohen hapësirat në skaje.

```
def anasjelltas(data):
    rev_text = ""
    text = data[1]
    data_len = len(text)
    for letter in range(data_len):
        rev_text += text[data_len - 1]
        data_len -= 1
    return rev_text.strip()
```

- Metoda *PALINDROM*

Metoda ***PALINDROM*** – paraqet një funksion apo metodë e cili kërkon një fjali dhe tregon se a është fjalia palindrome apo jo. Nëse fjalia është palindrome atëherë na kthen tekstin: “Teksti i dhënë është palindrom”, kurse në rastet kur fjalia nuk është palindrome atëherë na kthen tekstin: “Teksti i dhënë nuk është palindrom”.

```
def palindrom(data):
    rev_text = ""
    text = data[1]
    data_len = len(text)
    for letter in range(data_len):
        rev_text += text[data_len - 1]
        data_len -= 1
    if text == rev_text:
        return "Teksti i dhene eshte palindrom"
    else:
        return "Teksti i dhene nuk eshte palindrom"
```

- Metoda *KOHA*

Metoda ***KOHA*** – paraqet një funksion i cili përcakton kohën aktuale në server dhe e dergon atë tek klienti si format të lexueshëm për njerëzit. Për realizimin e kësaj metode janë përdorur funksionet noë, strftime nga moduli time ne Python.

```
def koha(data):
    return datetime.now().strftime("%d/%m/%Y %H:%M:%S")
```

- Metoda *LOJA*

*Metoda **LOJA***– paraqet një funksion i cili kthen 5 numra nga rangu [1,35]. Kështu p.sh “tekst”, rezultati i saj duhet të jetë i sortuar, perkatesisht (1, 14, 21, 27, 34). Pra, metoda Loja na kthen një listë të kthyer në string, e cila përmban 5 numra te gjënëruar në menyre të rëndomtë, të renditur sipas madhsisë dhe unike nga njëri-tjetri. Metoda është realizuar përmes një ëhile loop, e cila përseritet derisa nuk bëhet 5 numri i anëtarëve në listë. Për gjenerimin e numrave në mënyrë të rëndomtë është përdorur funksioni random, ku si parameter ka rangun 1-36 dhe numri i fundit nuk përfshihet. Pastaj per ti ikur duplikateve ne listë atëherë kam perdorur if-statements.

```
def loja(data):
    arr = []
    while len(arr) != 5:
        y = random.randint(1, 36)
        if y not in arr:
            arr.append(y)
    arr.sort()
    arrToStr = ', '.join([str(elem) for elem in arr])
    return arrToStr
```

- Metoda *GFC*

*Metoda **GFC*** – paraqet një funksion i cili gjen faktorin më të madh të përbashkët në mes dy numrave.

```
def gcf(data):
    return str(math.gcd(int(data[1]), int(data[2])))
```

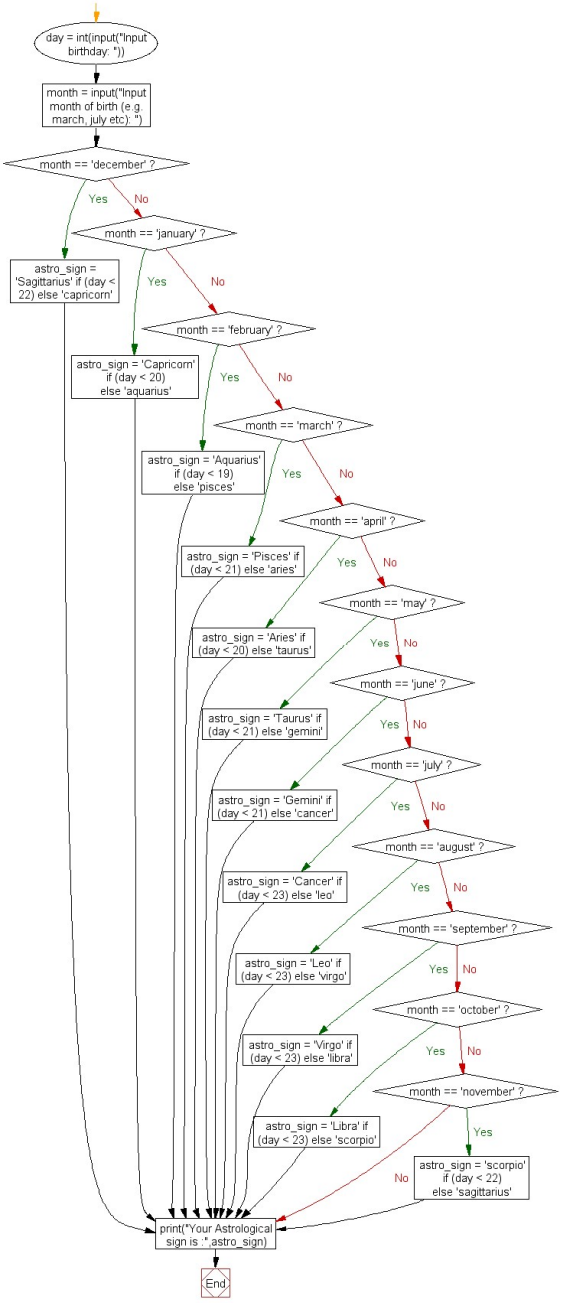
- Metoda *KONVERTO*

*Metoda **KONVERTO*** – paraqet një funksion i cili kthen si rezultat konvertimin e opsioneve varësisht prej opsionit të zgjedhur. Kështu lista e parametrave opsioni janë: *cmneinch, inchnecm, kmnemiles, milenekm.*

```
def konverto(data):
    if data[1].lower() == "cmneinch":
        return str(round((float(data[2]) * 0.393701), 2)) + "in"
    elif data[1].lower() == "inchnecm":
        return str(round((float(data[2]) / 0.393701), 2)) + "cm"
    elif data[1].lower() == "kmnemiles":
        return str(round((float(data[2]) * 0.621371), 2)) + "miles"
    elif data[1].lower() == "milenekm":
        return str(round((float(data[2]) / 0.621371), 2)) + "km"
    else:
        return "Nuk mund te konvertohet"
    return True
```

- Metoda *ASTRO*

*Metoda **ASTRO*** – paraqet një funksion i cili përcakton dhe jep shenjën e astrologjisë bazuar në ditën dhe muajin qe klienti jep përmes tastierës. Realizimi i kësaj metode është bazuar në një algoritëm që ndihmon në përcaktimin e shenjës së astrologjisë. Ky algoritëm është dhënë më poshtë:



```
def astro(data):
    day = int(data[1])
    month = int(data[2])

    if month == 12:
        if day < 22:
            return "Shigjetari"
        else:
            return "Bricjapi"
    elif month == 1:
        if day < 20:
            return "Bricjapi"
        else:
            return "Ujori"
    elif month == 2:
        if day < 19:
            return "Ujori"
        else:
            return "Peshqit"
    elif month == 3:
        if day < 21:
            return "Peshqit"
        else:
            return "Dashi"
    elif month == 4:
        if day < 20:
            return "Dashi"
        else:
            return "Demi"
    elif month == 5:
        if day < 21:
            return "Demi"
        else:
            return "Binjaket"
    elif month == 6:
        if day < 21:
            return "Binjaket"
        else:
            return "Gaforrja"
    elif month == 7:
        if day < 23:
            return "Gaforrja"
        else:
            return "Luani"
    elif month == 8:
        if day < 23:
            return "Luani"
        else:
            return "Virgjeresha"
    elif month == 9:
        if day < 23:
            return "Virgjeresha"
        else:
```

```
            return "Peshorja"
    elif month == 10:
        if day < 23:
            return "Peshorja"
        else:
            return "Akrepi"
    elif month == 11:
        if day < 22:
            return "Akrepi"
        else:
            return "Shigjetari"
    else:
        return "Nuk ka shenje"
```

- Metoda *DUPLIKAT*

*Metoda **DUPLIKAT*** – paraqet një funksion apo metodë me c'rast klienti jep një tekst dhe pastaj ky funksion verifikon se ky tekst përmban karaktere duplikate apo ajo. Në rast se përmban atëherë kthehët teksti "Teksti ka duplikate", në të kundërten kthehët teksti "Teksti nuk ka duplikate".

```
def duplikat(data):
    chars = "abcdefghijklmnopqrstuvwxyz"
    string = data[1]

    for char in chars:
        count = string.count(char)
        if count > 1:
            return "Teksti ka duplikate"
        else:
            continue
    return "Teksti nuk ka duplikate"
```

TESTIMI DHE PERFUNDIMI

Duke krahasuar kërkesat e projektit dhe ekzekutimin e programeve, mendoj që të gjitha kërkesat janë plotësuar ashtu edhe sic është kërkuar. Poshtë do të gjeni ekzekutimin e programit për cdo metode:

- Metoda *IP*

```
Shtypni 1 nese deshironi te nderroni adresen dhe portin: 2
Operacioni: IP
Pergjigja nga serveri: IP address is: 192.168.1.7
```

- Metoda *NRPORTIT*

```
Shtypni 1 nese deshironi te nderroni adresen dhe portin: 2
Operacioni: NRPORTIT
Pergjigja nga serveri: Port number is: 51659
```

- Metoda *NUMERO*

```
Operacioni: NUMERO
Teksti? Fakulteti i Inxhinierise Elektrike dhe Kompjuterike
Pergjigja nga serveri: Teksti i pranuar permban 3 zanore dhe 6 bashketingellore
```

- Metoda *ANASJELLTAS*

```
Operacioni: ANASJELLTAS
Teksti? FIEK
Pergjigja nga serveri: KEIF
```

- Metoda *PALINDROM*

```
Operacioni: PALINDROM
Teksti? FORTESA
Pergjigja nga serveri: Teksti i dhene nuk eshte palindrom


Operacioni: PALINDROM
Teksti? ANA
Pergjigja nga serveri: Teksti i dhene eshte palindrom
```

- Metoda *KOHA*

```
Operacioni: KOHA
Pergjigja nga serveri: 04/05/2021 03:26:25
```

- Metoda *LOJA*

```
Operacioni: LOJA
Pergjigja nga serveri: 2, 6, 11, 17, 19
```

- Metoda *GCF*

```
Operacioni: GCF 12 30
Pergjigja nga serveri: 6
```

- Metoda *KONVERTO*

```
Operacioni: KONVERTO cmneinch 12
Pergjigja nga serveri: 4.72in
```

```
Operacioni: KONVERTO inchnecm 12
Pergjigja nga serveri: 30.48cm
```

```
Shtypni 1 nese deshironi te nderroni adresen dhe portin: 2
Operacioni: KONVERTO kmnemiles 200
Pergjigja nga serveri: 124.27miles
```

```
Operacioni: KONVERTO milenekm 23
Pergjigja nga serveri: 37.01km
```

- Metoda *ASTRO*

```
Operacioni: ASTRO
 Dita e lindjes? 14
Muaji i lindjes? 11
Pergjigja nga serveri: Akrepi
```

- Metoda *DUPLIKAT*

```
Shtypni 1 nese deshironi te nderroni adresen dhe portin: 2
Operacioni: DUPLIKAT
Teksti? fakulteti
Pergjigja nga serveri: Teksti ka duplikate
```

```
Operacioni: DUPLIKAT
Teksti? Haxhi
Pergjigja nga serveri: Teksti nuk ka duplikate
```

Pra, të gjitha komandat ekzekutohet dhe kthejnë rezultat, në rastin kur klienti jep ndonjë kërkesë që nuk ekziston atëherë programi merr kërkesën si jovalide dhe vazhdon ekzekutimin.