­­­­ UNIVERSITETI I PRISHTINËS

FAKULTETI I INXHINIERISË ELEKTRIKE DHE KOMPJUTERIKE

**RRJETA KOMPJUTERIKE**

PROJEKTI 1:  
**Dizajnimi Klient-Server**

­­­­­­

Mentori: Punoi:

*MSc. Haxhi Lajqi Vigan Dika*

Prishtinë,

Prill, 2019

­­

Table of Contents

[1. Veglat e perdorura dhe metodat e implementuara 3](#_Toc6692067)

[2. Hyrje e shkurtër për projektin 3](#_Toc6692068)

[2.1 FIEK – TCP (Transmission Control Protocol) 3](#_Toc6692069)

[2.2 FIEK – UDP (User Datagram Protocol) 3](#_Toc6692070)

[3. FIEK - TCP 4](#_Toc6692071)

[3.1 TCP Serveri - Përshkrimi i kodit 4](#_Toc6692072)

[3.2 TCP Klienit – Përshrkimi i kodit 5](#_Toc6692073)

[3.3 Trajtimi i gabimeve 5](#_Toc6692074)

[4. FIEK - UDP 6](#_Toc6692075)

[4.1 UDP Serveri – Përshkrimi i kodit 6](#_Toc6692076)

[4.2 UDP Klienti – Përshkrimi i kodit 6](#_Toc6692077)

[5. Metodat 7](#_Toc6692078)

[5.1 IPADRESA & NUMRIIPORTIT 8](#_Toc6692079)

[5.2 BASHKETINGELLORE 8](#_Toc6692080)

[5.3 PRINTIMI 8](#_Toc6692081)

[5.4 EMRIIKOMPJUTERIT 9](#_Toc6692082)

[5.5 KOHA 9](#_Toc6692083)

[5.6 LOJA 9](#_Toc6692084)

[5.7 FIBONACCI 10](#_Toc6692085)

[5.8 KONVERTIMI 10](#_Toc6692086)

[5.9 PRIM 11](#_Toc6692087)

[5.10 ROCKPAPERSCISSORS 12](#_Toc6692088)

[6. Monitorimi nga ana e Serverit 13](#_Toc6692089)

# Veglat e perdorura dhe metodat e implementuara

Projekti është implementuar ne gjuhen programuese Python, versioni 3.7 kurse si IDE është përdorur JetBrains PyCharm versioni 2019.1.

Libraritë e përdorura për zhvillimin e projektit janë: *socket, sys, datetime, random, re, math, \_thread*.

Lista e metodave të implementuara: *IPADRESA, NUMRIIPORTIT, BASHKETINGELLORE, PRINTIMI, EMRIIKOMPJUTERIT, KOHA, LOJA, FIBONACCI, ROCKPAPERSCISSORS, PRIM, KONVERTMI*.

# Hyrje e shkurtër për projektin

Projekti ka si fokus komunikimin klient-server i cili është i mundur me anë të programimit me soketa (ang. sockets). Programimi me socketa është një mënyrë e lidhjes së dy pikave (soketave) që komunikojnë me njëra tjetrën. Njeri socket është socketi dëgjues (listener socket), që shërben si pikë referuese për socketin tjetër i cili tenton të lidhet me të. Serveri është socketi dëgjues që pret kërkesa dhe e servon klientin me të dhëna, kurse klienti është socketi që lidhet me server dhe merr të dhënat që i kërkon.

Serveri dëgjon në një adresë të caktuar, një kombinim unik (IP adresë, port), adresë kjo që targetohet nga klienti që kërkon të lidhet dhe ta shfrytëzojë atë.

Pasi vendoset lidhja e tyre, klienti do ta ketë të mundur të thërrasë metodat e ofruara nga serveri dhe ta shoh përgjigjen të cilën e proceson serveri, ndërsa serveri përpos që ofron përgjigje për kërkesat e klientit, është në gjendje të evidentojë çdo komunikim të ndërsjelltë me klientin.

Ky lloj komunikimi i shfaqur në projekt, nuk është protokoll standard por është një një protokoll i veçantë, të cilin do ta quajmë protokolli FIEK. Ky protokol ka versionin e tij TCP që quhet FIEK-TCP, dhe UDP versionin, FIEK-UDP.

## 2.1 FIEK – TCP (Transmission Control Protocol)

Është protokoll i besueshëm. Komunikimi TCP nuk është komunikim një kahorë, andaj bazohet në komunikimin e vazhdueshëm klient-server. Ky protokoll detekton paketat që kanë dështuar të transmetohen (dropped packages) dhe provon t’i ritransmetojë ato, njëkohësisht duke dhënë edhe arsyen e dështimit të transmetitmit. Ky protokoll dërgon të dhënat në rendin e njejtë siç i ka dërguar dërguesi. Serveri ka mundësi të komunikojë me më shumë se një klient duke përdorur ‘multi-threading’, teknikë kjo që lejon më shumë klient të kenë qasje në kodin e njejtë.

## 2.2 FIEK – UDP (User Datagram Protocol)

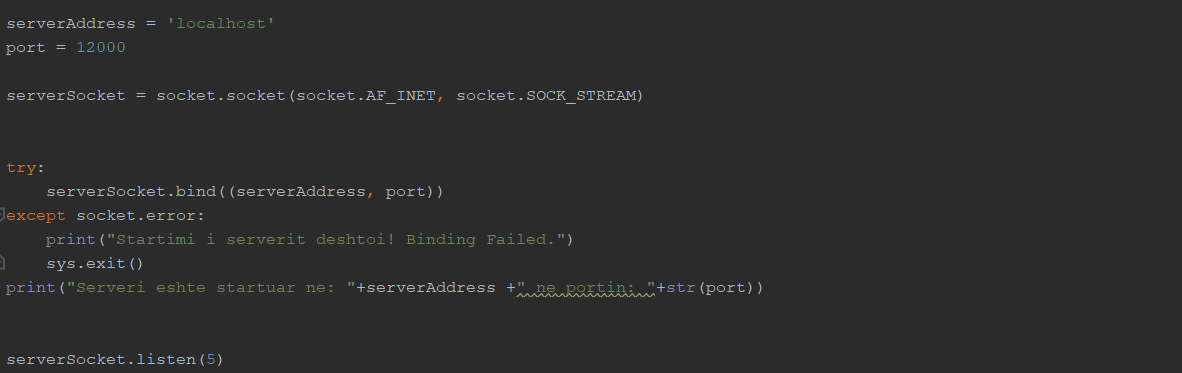
Vendosja e paketave në një sekuencë, vërtetimi i dërgesave dhe kërkesa e ritransmetimit të paketave që kanë dështuar humbin shumë kohë dhe ngadalëson komunikimin, andaj protokolli UDP i anashkalon të gjithë këto procese. Gjatë komunikimit UDP, dërguesi është i interesuar që të dërgojë sa më shumë informata, kurse nuk sigurohet se i kanë mbërritur të gjitha paketat, pra nuk i detekton paketat e humbura dhe rrjedhimisht nuk është i besueshëm.

# FIEK - TCP

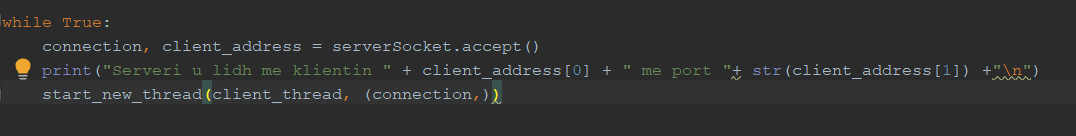
## 3.1 TCP Serveri - Përshkrimi i kodit

Synimi është të realizohet një aplikacioni server i cili do të jetë gjithmonë aktiv dhe në pritje të interaksionit me klientin.

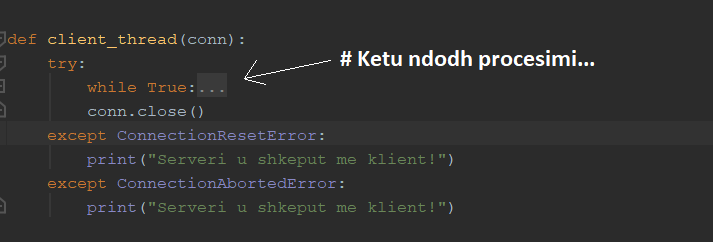
Së pari serverit i caktohet një adresë dhe nje port tek i cili ai do të dëgjojë dhe presë komanda. Në vijim krijohet socketi i serverit apo socketi dëgjues me metodën .socket(), duke i dhënë dy parametra: familjen e IP adresave, në rastin tonë AF\_INET, ku i referohemi familjes së IP adresave IPv4 dhe protokollin që do ta ndjekë për komunikim, në rastin tonë SOCK\_STREAM që i referohet protokollit TCP. Socketin e krijuar e asociojmë me metodën .bind() me IP adresën dhe portin tani më të dhënë. Me metodën .listen(), e vendosim serverin në gadishmëri të pranojë komanda kurse parametri (në rastin tone 5) tregon se sa klient mund të lidhen me serverin në të njejtën kohë.



Me anën e një unaze që nuk përfundon asnjëhere e mbajmë serverin githnjë aktiv, përveç në rastin kur ndodh ndonjë gabim që shpie në mbylljen e serverit. Me metodën .accept() marrim dy argumente siç janë lidhja me klient dhe adresa e tij. Pasi ne duam që serveri të manovrojë me disa kërkesa nga klient të ndyrshëm njëkohësisht ne përdorim teknikën ‘multi-thread’, të cilën e vëmë në veprim me thërritjen e funksionit start\_neë\_thread().



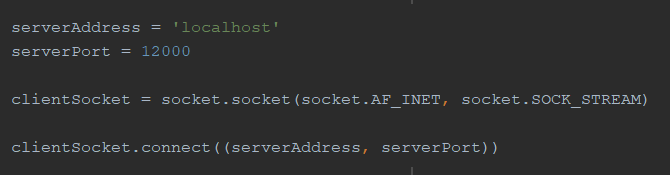
I gjithë procesimi tani kryhet në metodën e krijuar client\_thread(). Serveri dhe klienti komunikojnë mes vete me funksionet .recv() për të marrë infromacion dhe .send() për ta dërguar atë te klienti. Pas interaksionit server-klient pastrohet lidhja me anë të funksionit .close(), që e bën serverin të gatshëm për klientat e ardhshëm.



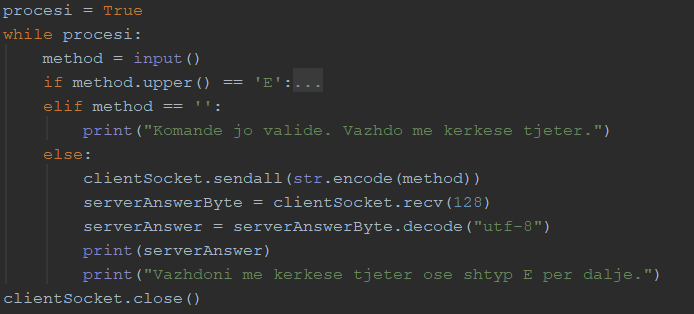
Metodat .socket(), .bind(), .listen(), .accept() merren nga libraria *socket*, kurse metoda start\_neë\_thread merret nga libraria *\_thread*.

## 3.2 TCP Klienit – Përshrkimi i kodit

Ngjashëm sikurse tek serveri, do të krijojmë edhe socketin e klientit me funksionin .socket() nga libraria socket. Këtë socket e lidhim me socketin e serverit përmes metodës .connect() ku i japim dy parametra, IP adresën dhe portin e serverit.



Unaza përdoret për t’i dhënë lirinë klientit që të përfundojë lidhjen me server kur ai dëshiron vetë, e jo pas një komande të vetme. Me metodat .send() dhe .recv() bëhet shkëmbimi i informatave mes dy socketave, kurse pas përfundimit të punës me metodën .close() klienti liron portin për tu përdorur nga ndonjë klient tjetër.



## 3.3 Trajtimi i gabimeve

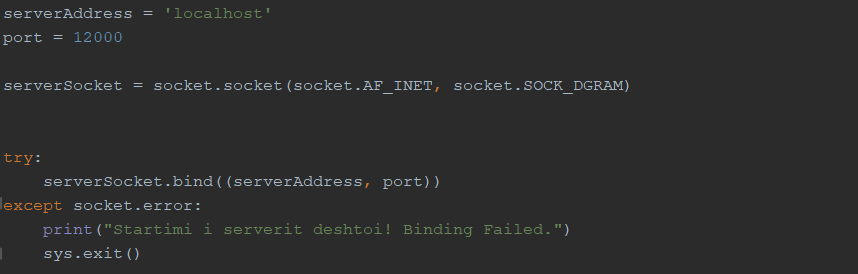
Siç edhe është shfaqur në fotografitë deri më tash, janë përdorur komandat try: dhe except: për t’i trajtuar gabimet eventuale në ekzekutimin e programeve. Që serveri të mos dështojë por të qëndroj aktiv përgjatë kodit janë përdorur mjaft komanda try: dhe except: ku është synuar të parashikohen gabimet e mundshme siç janë TimeOutError, socket.error, MemoryError, ConnectionAbortError.. dhe të shmangen ato.

# FIEK - UDP

## 4.1 UDP Serveri – Përshkrimi i kodit

Serveri që përdorë protokollin UDP tani nuk i nevojitet që të konfirmojë lidhjen me klient, por menjëherë pasi të vendoset lidhja ai do të fillojë të komunikojë me të duke përdorur datagrama, dërgimin e së cilës gjithashtu nuk ka pse ta konfirmojë.

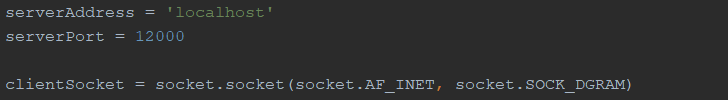
Për krijimin e socketit veprohet ngjashëm si në rastin e protokollit TCP, por tani në funksionin .socket() si parametër i dytë i ipet SOCK\_DGRAM, gjë që nënkupton se ky program do të zbatojë protokollin UDP.



Edhe komunikimi është i menjëhershëm nga socketi në socket pa ndonjë parametër në mes, që do të siguronte se nuk do të kishte paketa që dështojnë të transmetohen.

## 4.2 UDP Klienti – Përshkrimi i kodit

Ngjashëm si tëk aplikacioni Server, dallimi në krijimin e socketit ndodhet në parametrin e dytë që i ipet funksionit që krijon socketin, ku përcaktohet protokolli UDP.



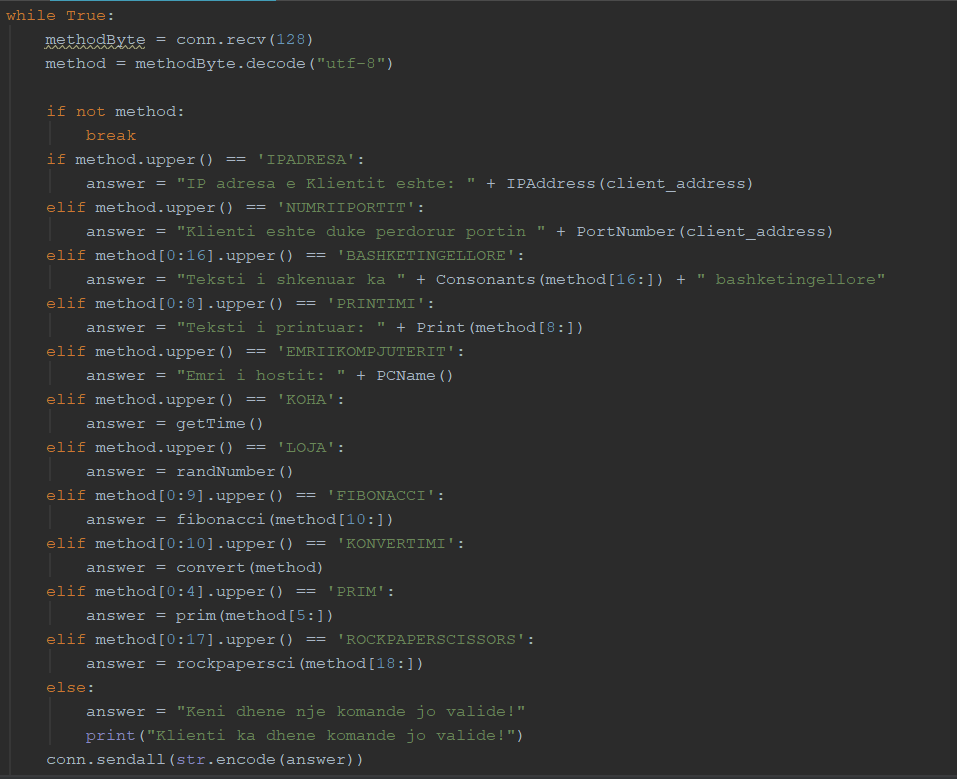
Forma e komunikimit gjithashtu është direkte nga socketi në socket, ku edhe caktohet adresa se ku duhet ta dërgojë kërkesën, apo adresa e serverit.



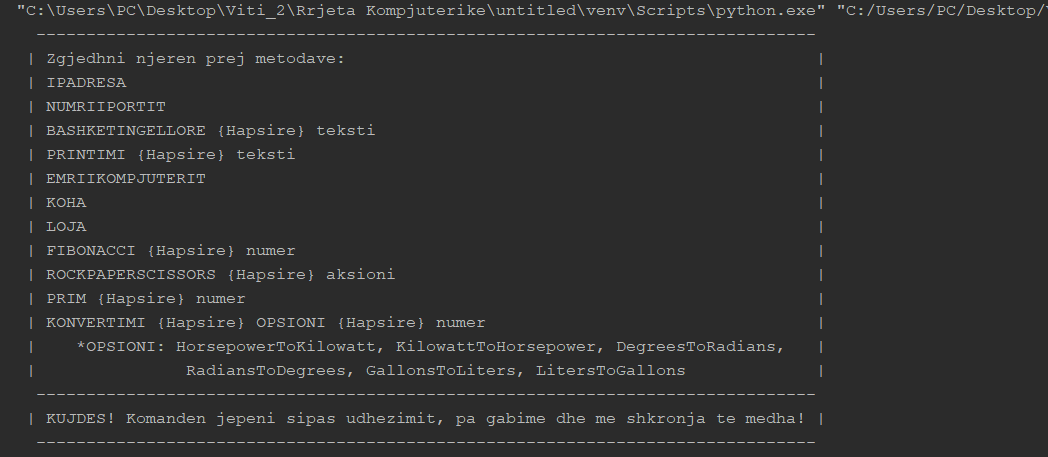
# Metodat

Në kod është paraqitur unaza kryesore sipas së cilës operon serveri. Serveri merr kërkesën nga klienti me funksionin .recv(), e dekodon dhe varësisht nga opsioni i zgjedhur i klientit e ri-drejton kërkesën në funksionin përkatës. Ky filtrim bëhet përmes *if statements*. Varësisht nga kërkesa e klientit serveri merr pergjigjen dhe ia kthen atë klientit i cili e pranon dhe e shfaq në ekran (pjesa e serverit). Serveri me anë të *else statement,* kontrollon komandëm jo valide dhe kthen pergjigje te klienti që të provojë përsëri me komandë tjetër, për arsye që të mos dështojë aplikacioni Server.

\*Kjo është foto e funksionimit të serverit që do t’i referohemi si foto\_1 për shpjegime të mëtejme.



Paraqitja e opsioneve në klient:



\*Edhe pse e udhëzoj klienti emrin e kërkesës ta shkruaj me shkronja të mëdha, në server korrigjoj ndonjë gabim eventual me anë të metodës .upper() duke i bërë të gjitha shkronjat shkronja të mëdha.

## 5.1 IPADRESA & NUMRIIPORTIT

Metodë e thjeshtë e cila merr si parametër një *tuple*, që në python paraqet një sekuencë të objekteve të pandashëm. Argumenti i dytë që merret me metodën .accept() që është adresa e klientit *client\_address*, paraqet një tuple, anëtari i parë i së cilës është IP adresa e i dyti porti. Kështu, në foto\_1, thërritet funksioni IPAddress e i ipet si parametër client\_address (tuple i marrë nga funksioni accept). Ngjashëm me IP adresën gjehet edhe porti të cilin është duke e përdorur klienti, i cili përdor anëtarin e dytë të tuple-it client\_address. Në anën e djathtë shihet përgjigja në klient.

|  |  |
| --- | --- |
| Trupi i metodave | Përgjigja në klient: |
|  |  |

## 5.2 BASHKETINGELLORE

Metoda që i gjen bashkëtingëlloret në një tekst është krijuar me një unazë. Unaza lëviz nëpër të gjithë shkronjat e tekstit të dhënë. Teksti i dhën me input i shkon funksionit si parametër. Mirëpo nëse shohim *foto\_1*, shohim që si parametër funksionit i është dhën vetëm një pjesë e ndarë e stringut të marrë (me kllapa të mesme ‘[ ]’), gjë që na lejon të marrim vetëm pjesën e stringut pas komandës BASHKETINGELLORE, pra tekstin të cilin duhet kontrolluar. Marrim një variabël e cila rritet për çdo herë që unaza has në një bashkëtingëllore dhe në fund vlera përfundimtare e kësaj variable dërgohet në klient.

|  |  |
| --- | --- |
|  |  |
|  |  |

## 5.3 PRINTIMI

Metodë që merr nje tekst nga shfrytëzuesi dhe e kthen tekstin e njejtë por duke eleminuar hapësirat e panevojshme në fillim dhe fund të fjalisë. Ngjashëm si tek metoda BASHKETINGELLORE, nje pjesë e inputit (nga shkronja [0:8], fjala printimi) shërben për filtrimin në *if statement* – in kryesor, i cili më pas e thërret funksionin për printim duke i dërguar pjesën tjetër te inputit si parametër (pra, pjesën e tekstit të dhënë nga shfrytëzuesi). Funksioni e merrë këtë tekst, dhe me anë të një built-in funksioni e pastron tekstin nga hapësirat e panevojshme para dhe pas stringut.

|  |  |
| --- | --- |
| Trupi i metodës | Përgjigja në klient |
|  |  |

## 5.4 EMRIIKOMPJUTERIT

Metoda në fjalë jep si rezultat emrin e hostit nën emrin e të cilit lexohet dhe ekzekutohet kodi i dhënë. Për këtë qëllim përdorim metodën *gethostbyname(),* një metodë built-in në librarinë socket.

|  |  |
| --- | --- |
| Trupi i metodës | Përgjigja në klient |
|  |  |

## 5.5 KOHA

Kur klienti zgjedh metodën KOHA nga foto\_1, serverin vë në veprim metodën getTime() e cila ka për qëllim të dërgojë si përgjigje kohën ekzakte të serverit. Kjo është e mundur duke përdorur librarinë *datetime,* nga e cila përdorim funksionin noë() i cili kthen datën dhe orën aktuale. Formati i shfaqjes së tyre mund të caktohet sipas detyrës.

|  |  |
| --- | --- |
| Trupi i metodës | Përgjigja në klient: |
|  |  |

\*Karakteri ‘ \ ’ është vetëm instruksion sintaksor në python që implikon që kodi vazhdon në rreshtin tjetër, pra nuk ka kurrfarë ndikimi në funksionalitetin e kodit.

## 5.6 LOJA

Metoda randNumber() përgjigjet kur klienti bën kërkesën LOJA. Ky funksion gjeneron 7 numra të rastësishëm në rangun 1-49 përmes librarisë *random* që importohet për aplikim. Unaza ka si detyrë që numrat e gjeneruar t’i rreshtojë në një array përmes funksionint .append(). Për arsye sintaksore, kthejmë array-in në string dhe përdorim funksionin replace() për të zëvëndësuar kllapat e mesme të array-it në kllapa të vogla të rëndomta të cilat i dërgojmë në ekran si përgjigje.

|  |  |
| --- | --- |
| Trupi i metodës | Përgjigja në klient: |
|  |  |

## 5.7 FIBONACCI

Metoda fibonacci ka për detyrë të kthejë një vlerën një anëtari të vargut të njohur fibonacci (0, 1, 1, 2, 3, 5, 8, 13...) varësisht kërkesa e shfrytëzuesit. Ngjashëm si në funksionin BSHKETINGELLORE dhe PRINTIMI, nga inputi i shfytëzuesit (p.sh FIBONACCI 13), pjesa ‘FIBONACCI’ përdoret në if-statement kryesor foto\_1, i cili me të marrë këtë string e vë në funksion metodën fibonacci duke i dhënë atij një parametër (në rastin tonë numri 13 që reprezenton anëtarin e 13 të kërkuar).

Pasi që ky parametër na vie si string, përdorim funksionin isdigit() për të parë nëse argumenti është numër dhe nëse e përmbush këtë kusht atëherë e kthejmë stringun në integer me funksionin int(). Pasi i caktojmë dy anëtarët e parë të vargut, 0 dhe 1, krijojmë tërë vargun përmes një unaze që fillon duke e inicializuar anëtarin e trete si shumë të dy anëtarëve paraardhës. Anëtari i dhënë nga shfrytëzuesi do të përdoret si kufi se deri sa cikle duhet të bëjë unaza (nese kërkojmë anëtarin e 3-të, 3 cikle). Pas përfundimit të unazës do të marrim anëtarin e fundit në array i cili do të jetë vlera e anëtarit të kërkuar nga klienti -1, pasi që anëtari i parë është 0 dhe e dërgojmë si përgjigje në klient.

Në rastin e marrjes të një numri shumë të madh si anëtar i kërkuar i vargut, serveri eviton dështimin me një   
try: except: i cili detekton gabimin MemoryError dhe kthen një përgjigje tek klienti dhe serveri se pse funksioni dështoi.

|  |  |
| --- | --- |
| Trupi  i  metodës |  |
| Pergjigja  në klient: |  |

## 5.8 KONVERTIMI

Metoda convert() aplikohet në rastin kur klienti zgjedh opsionin KONVERTIMI në menynë e metodave, të përcjellur nga një string tjetër i cili do të jetë opsioni, varësisht se çka ai/ajo dëshrion të konvertojë (shih opsionet tek menyja e klientit, poshtë foto\_1), e e gjithë kjo e përcjellur nga vlera që duhet të konvertohet në njërin nga opsionet e cekura më lart. Nëse klienti jep përgjigjen: ‘KONVERTIMI RADIANSTODEGREES 30’ , një pjesë e tërë këtij stringu do të merret nga if-statement kryesor (KONVERTIMI) kurse pjesa e mbetur (RADIANSTODEGREES) do të hyjë në një if-statement tjetër brenda funksionit convert(), për të kontrolluar se cilin opsion ka zgjedhur klienti.

Hapi i rradhës është që në stringun e marrë të merret vlera numerike, pra në rastin tonë vlera 30. Për këtë qëllim është përdorur libraria e python *re* (regular expressions). Pasi që është paraparë që vlera e dhënë nga klienti mund edhe të mos jetë e plotë, pra vlerë *float,* përdorim një kod të huazuar nga interneti që sipas formatit që e kemi caktuar gjen numrat me të gjithë numrat, me dhe pa pikë, dhe i kthen si listë të stringjeve. Stringun e marrë tani e pastrojmë nga kllapat, thojëzat dhe presjet dhe e kthejmë ne tipin *float*.

Tani gjithçka është gati dhe mund të vazhdojmë me llogaritjet për secilin opsion dhe kthimin e përgjigjes në klient. Gjithashtu, në rast të ndonjë gabimi eventual të komandës nga klienti situata zgjidhet me një else-statement.

|  |  |
| --- | --- |
| Trupi  i  metodës |  |
| Pergjigja  në klient: |  |

## 5.9 PRIM

Funksioni i zgjedhur prim është një funksion i thjeshtë i cili thërritet kur klienti do që të dijë se a është një numër i caktuar numër prim apo jo, përmes komandës PRIM. Funksioni merr stringun e dërguar nga klienti dhe e kontrollon nëse ai është numër, nëse kushti plotësohet stringu konvertohet në numër. Numri do të hyjë në unazë duke u pjesëtuar me secilin numër deri tek vetvetja (number+1, kufiri i cili nuk përfshihet). Për sa herë që numri plotpjesëtohëet me ndonjërin nga numrat në unazë, variabla që mban numrin e pjestuesve rritet për një. Pas përfundimit të unazës shihet nëse numri i pjestuesve është dy, nëse po atëherë vërtetohet që numri i dhënë është prim.

|  |  |
| --- | --- |
| Trupi i metodës | Përgjigja në klient: |
|  |  |

## 5.10 ROCKPAPERSCISSORS

Kjo metodë është një lojë ndërmjet kompjuterit dhe klientin nëse ai vendos të luajë ‘gur-letër-gërshërë’ duke dhënë komandën ROCKPAPERSCISSORS. Përgjigja apo aksioni i klientit (gur, letër ose gërshërë) i ipet funksionit rockpapersci() si parametër, aksion ky i cili bëhet çift me një aksion nga kompjuteri që zgjidhet në mënyrë të rastësishme nga lista *aksioni* me funksionin .randint(). Këto dy aksione pastaj krahasohen me njëri-tjetrin dhe varësisht nga rasti i kthehet përgjigja klientit se a ka fituar, ka barazuar apo ka humbur, se bashku edhe me aksionin e kompjuterit.

|  |  |
| --- | --- |
| Trupi  i  metodës |  |
| Pergjigja  në klient: |  |

# Monitorimi nga ana e Serverit

Gjatë gjithë kohës sa klienti vazhdon dhe dërgon komanda tek serveri, ky i fundit monitoron dhe regjistron veprimet dhe përgjigjet që i dërgohen klientit.

