

**SOCKET PROGRAMMING**

Lënda: Rrjetet Kompjuterike

Punoi: Arbër Hyseni

Ideja e këtij projekti është programimi me soketa (*angl. Sockets*) ne gjuhën Python, të krijohet një kuptim I mirëfilltë se çfarë pune kryejnë soketat, përkatësisht çfarë roli kanë në rrjetet kompjuterike, si barten informatat ndërmjet serverit dhe klientit apo anasjelltas. Ne mund të punojmë me soketa duke e importuar modulin **socket**. Në pika të shkurtra soketi paraqet një pikë fundore (Angl. *Endpoint)*, e cila lejon komunikimin dykahor të dy programeve të cilat janë në të njejtin rrjet dhe me të është e “lidhur” numri specifik I portit, me anë të të cilit TCP layer e identifikon adresën e aplikacionit në të cilin është qëllimi që të dërgohen të dhëna.

Detyra që na është shtruar është dizajnimi I një arkitekture Klient-Server me protokollet TCP dhe UDP(*më shumë rreth tyre më vonë)*. Diçka e re që na është paraqitur është se si mundet Serveri të punoj me disa klienta njëkohësisht, pra të pranoj kërkesa dhe të dërgon kërkesa me shumë klienta njëkohësisht. E gjitha kjo është realizuar nëpërmjet parimit të punës së threads-ave (*angl. Thread)*.

Threads paraqesin një sekuencë të instruksioneve që mund të ekzekutohen pavarësisht kodit tjetër ne atë program, apo sikur literaturës kryesore angleze që I përmbahemi në ligjerata, threadi mund ta paramendojmë si një sub-proces I lehtë I programit. Një multi-thread program d.m.th ka dy ose më shumë pjesë të cilat mund të punojnë njëkohësisht. Sa herë që një klient lidhet me Serverin duhet të krijohet një thread I ri për Server. Për të punuar me Threads së pari duhet importuar modulin **threading** dhe pastaj klasën [**threading.Thread**](http://faculty.salina.k-state.edu/tim/NPstudy_guide/servers/threads.html#threading.Thread). Në qoftë se dojmë ta ndajmë serverin në pjesë për ta kuptuar më mirë, ndarja do të ishte kështu:

* Ndëgjo dhe prano lidhje me soketa
* Krijo dhe starto threads-at “fëmijë”
* Krijo një infinit loop

Serveri me protokollin TCP dhe serveri me protokollin UDP duhet që të ketë metodat e definuara për zgjidhjen e problemeve të dhëna, dhe të pranoj një input nga Klienti, të përpunoj të dhënat dhe të dërgoj mesazhin e duhur te Klienti. Si p.sh. kështu më poshtë:

Operacioni (IPADDRESS, PORT, COUNT, REVERSE, PALINDROME, TIME, GAME, GCF, CONVERT, PASSWORDSTRENGTH, IPCLASS)? TIMEPërgjigjja: 10.04.2020 11:00:00

Që të kuptojmë se si kjo arkitekturë e serverit funksionon më shumë duhet kuptuar konceptet e TCP-së dhe UDP-së, si dhe te soketave.

Në menyrë që të hyjmë në detaje, së pari t’I kemi të qarta disa koncepte:

* **Transmission Control Protocol** apo shkurt TCP është një nga protokollet kryesore të suitës së protokolleve të Internetit. Ky protokoll buron së pari nga zbatimet fillestare të rrjetit në të cilin zhvilloi dukshëm Protokollin e Internetit (IP). Prandaj, e gjithë suita zakonisht quhet TCP/IP. TCP siguron shpërndarje të besueshme, të renditur dhe të kontrolluar nga gabimet e një rrjedhe (*Angl. Stream)* oktetesh (bajtësh) midis aplikacioneve që funksionojnë në hostet që komunikojnë përmes një rrjeti IP. Aplikacione të mëdha të internetit të tilla si World Wide Web, email, administrimi i largët dhe transferimi i file-ve mbështeten në TCP, e cila është pjesë e shtresës së transportit(Angl. Transport Layer) të kompletit TCP/IP. TCP është I orientuar në konektivitet, një lidhje në mes të serverit dhe klientit duhet realizuar para së të dërgohen të dhënat.
* **User Datagram Protocol** (UDP) është një nga anëtarët kryesorë të grupit të protokollit të Internetit. Aplikacionet kompjuterike mund të dërgojnë mesazhe, në këtë rast të referuara si datagrame, te hostet e tjera në një rrjet të Protokollit të Internetit (IP). Komunikimet paraprake nuk kërkohen për të vendosur kanale komunikimi ose shtigje të të dhënave. UDP përdoret më së shumti për të krijuar lidhje të cilat karakterizohen me “low latency” dhe “loss tolerating” ndërmjet aplikacioneve në internet. UDP konsiderohet protokoll “connectionless” për arsye se nuk ka nevojë të krijoj një qark virtual të realizohet para se të dergohen të dhëna.

Dallimet fundamentale ndërmjet TCP dhe UDP janë të paraqitura në tabelën e mëposhtme:

|  |  |  |
| --- | --- | --- |
| Karakteristikat | UDP | TCP |
| Analogjia | Dergimi me postë | Telefonatë |
| Faza e komunikimit | Nuk ekziston | Ekziston |
| Informatat | Individuale | Sekuenciale |
| Informata të humbura | Klienti duhet t’I pret humbjet | Protokolli nuk lejon *(Angl. Reliable)* |
| Radhitja e informatave | Nuk mbahet radhë | Mbahet radhë |
| Faza e shkyçjës | Nuk ekziston | Ekziston |
| Kompleksiteti | I ultë | I lartë |

*Burimi: University of Glamorgan*

**Funksionet:**

* Para se të vazhdojmë me shpjegimin e funksioneve, vlen të ceket se çdo funksion I mëposhtëm mund të zhvillohet në mënyra të ndryshme, varësisht se cila I duket studentit më e përshtatshme, disa funksione specifike sikur ajo re.sub, [::-1] apo slicing, I kam hulumtuar dhe I kam gjetur në faqe të ndryshme të internetit që do mundohem të I inkuadroj në sektorin e bibliografisë në fund të raportit

**Funksioni regex()**

import re

#client input validation

def regex(string):

    string = string.lower()

    string = re.sub(' +', ' ', string)

    string = string.lstrip() #Left strip to remove whitespaces

    return string.split(' ') #Split for splitting input in spaces

Për përpunimin e inputit te klientit shërben funksioni i mësipërm, I cili sipas meje është një funksion shumë kreativ dhe I qëlluar për arsyjen që përdoret. Përpos komenteve në gjuhën angleze ky funksion ka një parametër hyrës d.m.th (inputin e klientit), I cili pastaj përpunohet më tutje me funksionet përkatese siç është ajo sub dhe split, të cilat I përkasin modulit re, metoda **sub()** kthen një string në baze të kushtit më të majtë, pastaj bëhet **lstrip()** në rast se stringu I kthyer ka një whitespace në anën e majtë të stringut përkatës, duke vazhduar me funksionin **split()** e cila krijon një listë në bazë të kushtit brenda kllapave.

**Funksioni COUNT()**

def COUNT(value):

    if(value.isalpha()): #A eshte shkronje alfabetike

        vow = 0

        con = 0

        list1 = ["a", "e", "i", "o", "u"] #krijimi i nje liste

        for char in value:

            if char in list1: #nese char gjendet ne liste

                vow+=1

            elif char not in list1 and char>="a" and char<="z":

                con+=1

    else:

        raise Exception("Lejohen vetem shkronja te alfabetit.")

    result = f'Consonant {con} Vowel {vow}' #return permes f string

    return result

Siç po shihet metoda e mësipërme, shërben për njehsimin e zanoreve dhe bashkëtingëlloreve të stringut që është në hyrje të funksionit. Brenda if statement kam përdorur një funksion **isalpha()** I cili vërteton se a është çdo karakter I stringut shkronjë alfabetike, dhe nëse jo atëherë ndodh një **Exception** me mesazhin e caktuar dhe rezultati përmes metodës së **f-stringjeve** e cila është një risi ne Python 3, kam krijuar një “rezultat” I cili pasohet mandej në program main dhe te klienti.

**Funksioni REVERSE()**

def REVERSE(s):

    if(s.isalpha()):

        revres = s[::-1]

    elif not (s.isalpha()):

        #Show exception on terminal for invalid input

        raise Exception("Only words from the alphabet are allowed")

Ky funksion është një funksion I shkurtër dhe poashtu shumë efikas, pasi që edhe ka error handling por edhe përdor një metodë e cila quhet **slicing** dhe rasti konkret ne rreshtin e tretë të kodit të mësipërm, përdoret slicing për të kthyer stringun hyrës të funksionit mbrapsht.

**Funksioni PALINDROME()**

def Palindrome(s):

    true = "True"

    false = "False"

    rev = s[::-1] #slicing method to reverse given string

    if(s.isalpha()):

        if (s == rev):

            return true

        else:

            return false

    else:

        result = "Only numbers allowed"

        return result

Funksioni palindrome ka për detyrë që kur klienti të jep komandën palindrome *stringicfaredoshem,* të marr një përgjigjje se a është stringu që e ka dhënë një palindrom, d.m.th.: se a është I njejti edhe kur kthehet mbrapsht, sikur psh fjala *kimik*. Kjo është arritur duke përdorur përsëri slicing, dhe është trajtuar paraqitja eventuale e gabimeve, do të ishte mirë të implementonim raise Exception etj., mirëpo pasi që arkitektura e këtij programi është delikat ajo ndërpren workflowin e duhur të programit.

**Funksioni GAME()**

from random import seed, randint

def GAME():

    lst = []

    forseed = randint(1, 100) #random number generator

    seed(forseed)

    for \_ in range(5):

        rvalue = randint(1, 35) #generate random numbers between 1 and 35

        lst.append(rvalue) #append to list

    return lst

Funksioni GAME ka për qëllim që të krijoj një listë me 5 numra arbitrar, me kufi prej 1 deri në 35. Kjo është realizuar me përdorimin e modulit random, ku prej saj janë përdorur funksionet built-in **seed()** dhe **randint().** Që ky funksion të krijoj njëmend numra të çfarëdoshëm duhet të perdoret funksioni seed() I cili gjeneron një numër(parametër) që inicializon një sekuencë të numrave dhe së bashku me përdorimin e randint() do të krijohet një varg me numra arbitrar.

**Funksioni TIME()**

from datetime import datetime

def Time(): #ketu hyhet ne perdorim librari e gatshme datetime

    time = datetime.now()

    date\_string = time.strftime("%d/%m/%Y %H:%M:%S") #string format for time

    return date\_string

Vetë emri I funksionit tregon se çfarë bën, pra kthen orën e saktë por edhe datën, në të cilën kërkohet ky funksion nga klienti. Kjo është bërë e mundur nga moduli datetime, ku ky modul na jep klasa te gatshme për manipulimin e datave dhe kohës. Për këtë funksion ndihmë ka qenë faqja docs.python. Për format të stringut është përdorur metoda strftime() që kthen datën dhe kohën e formatuar në bazë të kushteve brenda kllapave të funksionit.

**Funksioni GCF TE SERVERI DUHET ME BO IF ELSE…**

def GCF(x, y):

    if x > y:

        little = y

    else:

        little = x

    for i in range(1, little+1): #for loop prej 1 deri te me i vogli +1

        if ((x % i == 0) and (y % i == 0)): #operacioni modulo

            gcf = i

    return gcf

Funksioni GCF do të thotë funksioni **greatest common factor** I cili ka për detyrë të kthejë faktorin më të madh të përbashkët të dy numrave që janë në hyrje të funksionit, kjo është bërë e mundur përmes këtij algoritmi relativisht të thjeshtë ku me një for-loop prej 1 deri te numri më vogel në hyrje të funksionit, shiqojmë me ndihmën e operacionit modulo dhe me anë të njohurive elementare të matematikës të gjejmë faktorin më të madh të përbashkët të ata dy numrave në hyrje.

**Dy funksionet extra që I kam bërë janë:**

* **PasswordStrength**
* **IpClass**

**Funksioni PasswordStrenth**

import re

def passwordStrength(s):

    # Enter password text

    length = re.compile(r'(\w{8,})')  # Check if password has atleast 8 characters

    lower = re.compile(r'[a-z]+') # Check if at least one lowercase letter

    upper = re.compile(r'[A-Z]+')# Check if atleast one upper case letter

    digit = re.compile(r'[0-9]+') # Check if at least one digit.

    special = re.compile(r'[!@#$%^&\*\_.]+')

    if length.findall(s) == []:  # Checks if the password length is not 8 characters long, findall() kthen string array

        result = 'Your Password must contain at least 8 characters'

    elif lower.findall(s)==[]: # Checks if the password does not contain a  lowercase character

        result = 'Your Password must contain at least one lowercase character'

    elif upper.findall(s)==[]: # Checks if the password does not contain an  uppercase character

        result = 'Your Password must contain at least one uppercase character'

    elif digit.findall(s)==[]: # Checks if the password does not contain a  digit character

        result = 'Your Password must contain at least one digit character'

    elif special.findall(s)==[]: # Checks if the password does not contain a  special char

        result = 'Your Password must contain at least one special character'

    else:  # if the above 4 conditions are successfully passed, pringt out a message saying the password is strong.

        result = 'Your password is strong, congratulations!'

    return result

Si funksion extra që është dashur t’a bëjmë, unë kam përzgjedhur funksionin të ashtuquajtur **passwordStrength,** nga vetë emri tregohet se teston nje string hyrës se a është password I sigurt apo jo, dhe nëpërmjet built-in funksioneve te modulit re(*regular expression*) siç është re.compile() e cila kompajllon një “mostër” të regular expression në regular expression objekt, dhe për të punuar me atë objekt ne manipulojmë me funksionet **.findall()** apo **.match()**. Në këtë rast me .findall() kemi kërkuar se a ka mjaftueshëm karaktere, numra, etj. Pastaj me if, elif, else statements kemi manipuluar me të dhënat që I kemi.

**Funksioni ipClass:**

from ipaddress import IPv4Address, IPv4Network

def ipClass(s):

    try:

        ip = IPv4Address(s) #represent and manipulate single ipv4 addresses

        # ipv4network represents and manipulates 32-bit IPv4 network and addresses.

        classA = IPv4Network(("1.0.0.1", "255.0.0.0")) # or IPv4Network("10.0.0.0/8")

        classB = IPv4Network(("128.1.0.1", "255.255.0.0")) # or IPv4Network("128.1.0.1/16")

        classC = IPv4Network(("192.0.1.1", "255.255.255.0")) # or IPv4Network("192.0.1.1/24")

        if ip in classA:

            return "Given ip is in Class A"

        elif ip in classB:

            return "Given ip is in Class B"

        elif ip in classC:

            return "Given ip is in Class C"

    except:

        return "error"

Funksioni I mësipërm quhet ipClass pra dhe në input ka një string I cili duhet të jetë një IP adresë e versionit 4 dhe ky funksion ka për detyrë që të gjejmë se IP Adresa në hyrje cilës klasë I takon, e kam zgjedhur këtë metodë për aryse që edhe në klasë e kemi trajtuar çështjen e klasave të IP-ve, ku në total janë 5 ndarje, por që 2 të fundit janë out of bounds me modulin ipaddress pra nuk kemi si te manipulojmë me ato dhe se **klasa e katërt është për multicast dhe e pestë për arsye eksperimentale**, nuk janë edhe shumë të përdorshme prandaj kam vendosur që vetëm **tri klasat e para** t’I implementoj në këtë funksion pasi që edhe janë më të shpeshta në jetën e përditshme, kjo mënyrë mendoj se është një mënyrë mjaft efektive dhe optimale.

Kemi importuar modulin **ipaddress** në të cilën gjenden built-in funksionet sikur ajo IPv4Network dhe IPv4Address me të cilat kemi manipuluar adresat në hyrje, apo edhe kërkuar në një varg adresash, siç është rasti me IPv4Network ku kemi jepur ip adresën fillestare të klasës