**AMALIY MASHG‘ULOT UCHUN O‘QUV MATERIALLARI**

**3-Mavzu:** Pythonda tarmoq dasturlashga kirish.

**4-mashg‘ulot.** TCP klient-server dasturini testlash.

O‘quv savollari:

1. TCP client-server dasturi yordamida ma’lumot almashish.

**1. TCP client-server dasturi yordamida ma’lumot almashish.**

**Serverni ulanish**

Server kiruvchi ulanishlarni kutib turadi (listining), ularni qandaydir tarzda qayta ishlaydi va javob yuboradi. Server o‘z ishini boshlashi uchun, avvalo, u ulanishlarni tinglaydigan manzilni aniqlab olishi kerak. Buning uchun **bind()** metodi qo‘llaniladi.

socket.bind(address)

Bu usul server ishga tushadigan manzilni qabul qiladi. Odatiy bo‘lib, manzil ikki elementdan iborat to‘plamdir:

(host, port)

Birinchi element string sifatida xost hisoblanadi. Bu, masalan, "127.0.0.1" shaklidagi IP-manzil yoki mahalliy xost nomi bo‘lishi mumkin. Ikkinchi parametr raqamli port raqamidir. Port 0 dan 65535 gacha bo‘lgan 2 baytlik qiymatni ifodalaydi. Bir xil manzilda (bir xil mashinada) bir nechta turli tarmoq ilovalari ishga tushirilishi mumkinligi sababli, port ushbu ilovalarni farqlash imkonini beradi. Masalan:

import socket

server = socket.socket()

hostname = socket.gethostname()

port = 12345

server.bind((hostname, port))

Bu yerda server 12345-portda ishlaydi. E’tibor bering, barcha portlar ham bepul bo‘lmasligi mumkin. Ammo, qoida tariqasida, band bo‘lgan portlar unchalik ko‘p emas.

Manzil sifatida joriy xost nomidan foydalanamiz. Uni olish uchun funktsiyadan foydalaniladi **socket.gethostname()** (odatda bu joriy kompyuterning nomi).

**Ulanishlarni tinglash**

Serverni bog‘lagandan so‘ng ulanishlarni tinglash uchun uni ishga tushirish kerak. Buning uchun **listen()** metodi qo‘llaniladi.

socket.listen([backlog])

Bu backlog parametrini qabul qiladi. backlog – bu socket uchun ruxsat etilgan navbatdagi kiruvchi ulanishlarning maksimal soni. Ya’ni, mijozlar ulanganda, ular navbatda turishadi va server joriy mijozni qayta ishlaguncha kutishadi. Agar navbatda server tomonidan qayta ishlanishini kutayotgan mijozlar soni allaqachon belgilangan bo‘lsa, barcha yangi mijozlar rad etiladi. Masalan:

import socket

server = socket.socket()

hostname = socket.gethostname()

port = 12345

server.bind((hostname, port))

server.listen(5)

**Mijozni qabul qilish va qayta ishlash**

**accept()** metodi kiruvchi ulanishlarni qabul qilish uchun ishlatiladi . Bu usul 2 ta tuple tipli qiymatni qaytaradi

(conn, address)

Birinchi element conn - server mijoz bilan aloqa qiladigan boshqa soket ob’ektini ifodalaydi. Ikkinchi element address - ulangan mijozning manzili hisoblanadi. Shuni ta’kidlash kerakki, mijoz bilan o‘zaro aloqalar tugagandan so‘ng, conn socketini close() usuli bilan yopish kerak bo‘ladi.

Tuplening birinchi elementi - conn yordamida mijozga xabar yuborish yoki aksincha ma’lumotlarni qabul qilish mumkin. Soketdan ma’lumotlarni qabul qilish uchun **socket.recv()** usuli qo‘llaniladi.

bytes = socket.recv(bufsize)

Bu zarur parametr sifatida bir vaqtning o‘zida boshqa socketdan olinishi mumkin bo‘lgan baytlardagi maksimal bufer hajmini oladi. Qaytish qiymati boshqa socketdan olingan baytlar to‘plamidir.

Ma’lumotlarni yuborish uchun **socket.send()** metodi qo‘llaniladi, u parametr sifatida yuboriladigan ma’lumotlar to‘plamini oladi.

socket.send(bytes)

Endi bir misolni ko‘rib chiqaylik. Server.py faylida serverni quyidagi kod bilan aniqlaylik:

import socket

server = socket.socket()

hostname = socket.gethostname()

port = 12345

server.bind((hostname, port))

server.listen(5)

print("Server starts")

con, addr = server.accept()

print("connection: ", conn)

print("client address: ", addr)

message = "Hello Client!"

con.send(message.encode())

con.close()

print("Server ends")

server.close()

Bu yerda server mijozni qabul qiladi, uning ulanishi haqidagi ma’lumotlarni chiqaradi va mijozga “Hello Client!” qatorini qaytarib yuboradi.

Va client.py faylida quyidagi mijoz kodini aniqlanadi:

import socket

client = socket.socket()

hostname = socket.gethostname()

port = 12345

client.connect((hostname, port))

data = client.recv(1024)

print("Server sent: ", data.decode())

client.close()

Bizning serverimiz va mijozimiz bitta kompyuterda ishlaganligi sababli, **socket.gethostname()** server kodidagi kabi ulanish uchun server manzilini aniqlash uchun funksiya va port 12345 ishlatiladi. Serverga ulangandan so‘ng undan ma’lumotlarni olinadi va uni konsolda ko‘rsatadi.

Avval server kodini ishga tushiraylik. Va keyin mijoz kodini ishga tushiriladi. Natijada, server ulanishni qabul qiladi, u haqidagi ma’lumotlarni konsolda ko‘rsatadi va mijozga xabar yuboradi:

c:\python>python server.py

Server starts

connection: <socket.socket fd=432, family=2, type=1, proto=0, laddr=(‘192.168.0.102’, 12345), raddr=(‘192.168.0.102’, 61824)>

client address: (‘192.168.0.102’, 61824)

Server ends

Xususan, bu yerda server va mijoz 192.168.0.102 da ishlayotganini va mijoz 61824 portidan foydalanayotganini ko‘ramiz.

Va mijoz serverdan xabar oladi va uni konsolga chop etadi:

c:\python>python client.py

Server sent: Hello Client!

**Bir nechta mijozlar bilan ishlash**

Bunday holda, server bitta mijozga xizmat qiladi va ishlashni to‘xtatadi. Agar biz server ko‘p mijozlar bilan ishlashni istasak, unda cheksiz tsikldan foydalanishga to‘g‘ri keladi:

import socket

from datetime import datetime

server = socket.socket()

hostname = socket.gethostname()

port = 12345

server.bind((hostname, port))

server.listen(5)

print("Server running")

while True:

con, addr = server.accept()

print("client address: ", addr)

message = datetime.now().strftime("%H:%M:%S")

con.send(message.encode())

con.close()

Bunday holda, masalan, o‘rnatilgan modul datetime va datetime.now().strftime() funksiyasidan foydalanib, joriy vaqtni satr sifatida olish, keyinchalik uni mijozga yuborish mumkin. Natijada, mijoz so‘rov bilan joriy vaqthaqidagi ma’lumotlarni ham oladi.

**Ikki tomonlama aloqa**

Yuqoridagi misollarda aloqa bir tomonlama edi - server ma’lumotlarni yuboradi va mijoz uni qabul qiladi. Keling, mijoz ham, server ham ma’lumotlarni jo‘natgan va qabul qilganda eng oddiy ikki tomonlama aloqani ko‘rib chiqaylik. Serverga mijozdan bir nechta satr olishiga ruxsat beraylik, uni o‘zgartirib so‘ngra mijozga qaytarib yuboraylik:

import socket

server = socket.socket()

hostname = socket.gethostname()

port = 12345

server.bind((hostname, port))

server.listen(5)

print("Server running")

while True:

con, \_ = server.accept()

data = con.recv(1024)

message = data.decode()

print(f"Client sent: {message}")

message = message[::-1]

con.send(message.encode())

con.close()

Mijoz konsoldan satr kiritish va uni serverga yuborish uchun kodni aniqlasin:

import socket

client = socket.socket()

hostname = socket.gethostname()

port = 12345

client.connect((hostname, port))

message = input("Input a text: ")

client.send(message.encode())

data = client.recv(1024)

print("Server sent: ", data.decode())

client.close()

Yuqoridagi ishning natijasi:

**Client**:

c:\python>python client.py

Input a text: hello

Server sent: olleh

c:\python>

**Server**:

c:\python>python server.py

Server running

Client sent: hello

**Nazorat savollari:**

1. Server bilan aloqani qanday amalga oshiriladi?
2. socket.bind() metodi nima amalni bajaradi?
3. socket.gethostname() metodi qanday vazifani bajaradi?
4. listen() metodining ishlash tamoyili qanday?
5. socket.recv() metodidan nima maqsadda foydalaniladi?