

## TCP/UDP SOKET PROG. KOD AÇIKLAMALARI

### TCP İLE SOKET PROGRAMLAMA – SERVER ;

Öncelikle socket oluşturmak için gerekli kütüphanemizi import ediyoruz.

```
import socket
```

Host\_num ve port\_num'a TCP için socket programlarken kullanacağımız gerekli host ve port numaralarının atamasını yapıyoruz. 127.0.0.1 bizim localhostmuzu gösteren IP adresidir.

```
host_num = '127.0.0.1'  
port_num = 5000
```

Socketimizi oluştururken s değişkeninde tutup IPv4 internet protokol adresi ve TCP için geçerli socket olan SOCK\_STREAM ile ilişkilendiriyoruz.

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

Bind( ) fonksiyonu yardımıyla socketimizin host ve port numarası ile ilişkilendirilmesini sağlıyoruz.

```
s.bind((host_num, port_num))
```

TCP portu dinlemek için listen() fonk. kullanıyor ve kaç kez dinleyeceğimizi belirtiyoruz.

```
s.listen(1)
```

Servera gelen bağlantıyı kabul etmek için accept() fonk. kullanıyoruz. Ve kabul sonrası bağlantı adresini terminal ya da ekrana yazdırıyoruz.

```
conn,addr = s.accept()  
print('Bağlantı adresi', addr)
```

Recv() ile data değişkeni içerisine kabul ettiğimiz TCP verisinin, mesajının alımını yapıyoruz. Ve print fonk. ile ekrana yazdırıyoruz. Decode() ekleyerek gelen mesajın formunun bozulmamasını sağlıyoruz.

```
data = conn.recv(1024)  
print("Alınan data:", data.decode())
```

Eco amacıyla kodlama yaptığımız için yansıma adına bize gelen mesajın send ile karşı tarafa da gönderimini sağlıyoruz. Ve close() ile socketimizi kapatıyoruz.

```
conn.send(data)  
conn.close()
```

## TCP İLE SOKET PROGRAMLAMA – CLIENT ;

Client için de socket oluşturmak için gerekli kütüphanemizi import ediyoruz.

```
import socket
```

Host ve port numaralarımızın atamasını yine aynı şekilde client kısmı içinde uyguluyoruz. Bu kısımda iletimde kullanacağımız mesajı, mesaj değişkenine atıyoruz

```
host_num = '127.0.0.1'  
port_num = 5000  
mesaj= "Merhaba :)!"
```

Socketimizi oluştururken s değişkeninde tutup IPv4 internet protokol adresi ve TCP için geçerli socket olan SOCK\_STREAM ile ilişkilendiriyoruz.

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

Connect( ) yardımı ile socketimizin host ve port numarasıyla bağlantı kurmasını sağlıyoruz.

```
s.connect((host_num, port_num))
```

Mesaj gönderimi için send() fonksiyonunu kullanarak mesaj formunun bozulmaması için mesajı encode() yardımıyla utf-8 formatında tutuyoruz.

```
s.send(mesaj.encode())
```

Data içine TCP için kullanılan recv() fonksiyonu ile buffer size'ımızın yani saniyedeki max. veri boyutumuzun belirlemesini yapıyoruz. 1024'ten farklı olarak 1024'ün katları şeklinde 2048,4096... gibi değerler de alabilir. Ve s.close() ile socketimizi kapatıyoruz.

```
data = s.recv(1024)  
s.close()
```

Client kısmı için de eco amacıyla datayı ekran ya da terminale yazdırıyoruz ve gelirken de format bozulmasın diye decode()'dan faydalıyoruz.

```
print("Alınan data:", data.decode())
```

## UDP İLE SOKET PROGRAMLAMA – SERVER ;

Öncelikle soket oluşturmak için gerekli kütüphanemizi import ediyoruz.

```
import socket
```

Gerekli host numaramızı ve port numaramızı host\_num ve port\_num değişkenlerine atıyoruz. Host\_num da yer alan IP adresi ("127.0.0.1") aslında bizim lokal hostumuzu göstermektedir.

```
host_num = "127.0.0.1"  
port_num = 5005
```

AF\_INET ile IPv4'e bağlı internet adres belirtecimizi ve UD protokolüne bağlı bir soket oluşturacağımız için SOCK\_DGRAM olduğunu belirtiyoruz.

```
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
```

Soketimizi ağ ve port numarasıyla ilişkilendiriyoruz.

```
sock.bind((host_num, port_num))
```

Eğer clientle olan durumumuz True halindeyse bu süreçte datamızın iletimini ve alımını devam ettirmek için bir while döngüsü açıyoruz ve mesajın teslim alındığını, aynı zamanda yeni haline dair bilgiyi yazdırmak için print ve formunu bozmadan yazdırmak için decode() kullanıyoruz. Sock.recvfrom(1024) max 1024 bite kadar alım yapabileceğimizi belirtmek içindir. Aslında bizim buffer sizeimizi temsil eder. 1024 ve katları olarak ilerlenebilir. Client verilerini algılayan recvfrom( ) da yine UDP için soket programlarken kullanılır. TCP'de recv( ) kullanılır.

```
while True:  
    data, addr = sock.recvfrom(1024)  
    print ("Mesaj teslim alındı.")  
    print ("Mesajın yeni hali:", data.decode())
```

## UDP İLE SOKET PROGRAMLAMA - CLIENT ;

Soket oluşturmak ve programda döngü bitişinde kısa süreli uyku durumu kullanacağımız için gerekli kütüphaneleri import ediyoruz;

```
import socket  
import time
```

Gerekli host numaramızı ve port numaramızı host\_num ve port\_num değişkenlerine atıyoruz. Host\_num da yer alan IP adresi ("127.0.0.1") aslında bizim lokal hostumuzu göstermektedir.

```
host_num = "127.0.0.1"  
port_num = 5005
```

Yine bir while ile konumumuz True olduğu sürece devam edebileceğimiz bir döngü açıyor ve gerekli kuralların girişini yapıyoruz.

Öncelikle kullanıcıdan kelime ya da cümle olarak bir girdi belirtmesini istiyoruz. Daha sonrasında texti utf-8 formatına uygunluk sağlaması için message içine text.encode() ile atıyoruz. Encode() parametre almadan kısaca utf-8 formatına uygun kullanım sağlamaya yarar. İstersek ascii gibi farklı formatlarda kullanabilir ya

da utf-8 için farklı şekilde format belirleme yapabiliriz ama encode kısa ve net olduğundan ben encode() ile belirttim.

```
while True:
    text = input('Bir kelime ya da cumle giriniz:')
    message = text.encode()
```

Text girişi ve çevrim işlemini yaptıktan sonra client ekranımıza bilinmesi açısından host,port numaraları ve formu bozulmaması adına decode() kullanarak mesajımızın orijinal halini yazdırıyoruz.

```
print ("UDP IP adresi", host_num)
print ("UDP port numarası:", port_num)
print ("Mesajınız:", message.decode())
```

Serverda olduğu gibi client için de AF\_INET ile IPv4'e bağlı internet adres belirtecimizi ve UD protokolüne bağlı bir soket oluşturacağımız için SOCK\_DGRAM olduğunu belirtiyoruz.

```
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
```

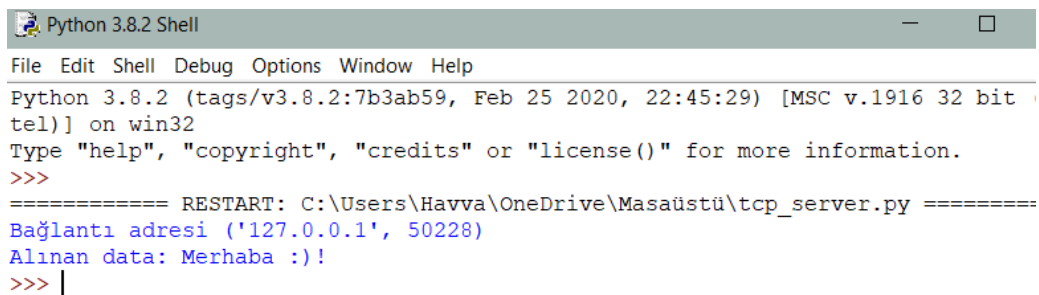
Ve mesajımızı upper ile tamamen büyük harflerden oluşan bir mesaj haline getirip sendto ile ilişkili IP'miz ve portumuz aracılığı ile server ekranımızı iletiyoruz.

```
sock.sendto(message.upper(), (host_num, port_num))
```

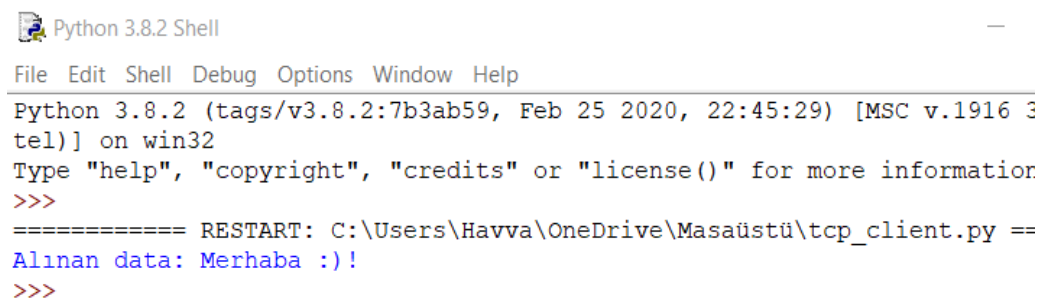
Döngünün tekrar işleminin üzerinden geçmesi gereken süreyi de time.sleep() ile saniye cinsinden belirliyoruz ki sürekli olarak peş peşe sormasın 1 dk kadar sonra tekrar aktif halde çalışsın gibi bir ayarlama yapabilmek için.

```
time.sleep(60)
```

## TCP SOKET PROGRAMLAMA EKRAN ÇIKTILARI; (eco)



```
Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Havva\OneDrive\Masaüstü\tcp_server.py =====
Bağlantı adresi ('127.0.0.1', 50228)
Alınan data: Merhaba :)!
>>> |
```



```
Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Havva\OneDrive\Masaüstü\tcp_client.py =====
Alınan data: Merhaba :)!
>>>
```

## UDP SOKET PROGRAMLAMA EKRAN ÇIKTILARI; (Girdiyi değiştirerek yansıtma – upper())

```
*Python 3.8.2 Shell*
File Edit Shell Debug Options Window Help
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.191
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more informat
>>>
= RESTART: C:\Users\Havva\OneDrive\Masaüstü\BİLG. AĞLARI PROJE ÖDEVİ
PY
Mesaj teslim alındı.
Mesajın yeni hali: HAVVA
```

```
*Python 3.8.2 Shell*
File Edit Shell Debug Options Window Help
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.19
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more informa
>>>
= RESTART: C:\Users\Havva\OneDrive\Masaüstü\BİLG. AĞLARI PROJE ÖDEV
PY
Bir kelime ya da cumle giriniz:havva
UDP IP adresi 127.0.0.1
UDP port numarası: 5005
Mesajınız: havva
|
```

1700002090 – HAVVANUR SELAMET