**Zip function in python:**

What is zip: It connects two/multiple components. In the context of python it is used to connect/join 2 lists(tuples)

Ex: below 2 tuples are zipped and observe the output.

Graphical user interface, text, application

Description automatically generated

Set: unordered

Graphical user interface, text, application

Description automatically generated

Dict: will give the dictionary format, observe the output

Graphical user interface, text, application

Description automatically generated

* You can use a loop as well and iterate between the values as shown below. Run the code and observe the output

Graphical user interface, text, application

Description automatically generated

name = ('uday', 'rishi', 'bunny')

companies = ('accenture', 'dell', 'amazon')

zipped = zip(name,companies)

for a,b in zipped:

    print(a,b)

Conclusion: zip function can be used for various purposes, however here we are using it for joining 2 lists.

**Socket programming using python:**

Before we start working on sockets, we need to know a bit about internet where all the devices(mobiles/pcs/servers/laptops etc..) are interacting with each other or talking to each other. Consider the example of google search request and response. Discuss about google search for a moment.

IP Address

Public IP

Private IP

A person wearing glasses

Description automatically generated with medium confidence

Base of the internet or base of the network which is sockets. We will do socket programming with python.

When we say we are doing socket programming. We will use 2 things here 1) server(server.py) 2) client(client.py).

Let’s discuss some basic understanding of port numbers and type of connection we are going to build. Discuss OSI Model

What is IP Address and IPv4, IPv6? Why is it important? When we type Google.com on our browser what happens. Discuss about how DNS works and it’s port numbers. HTTP service and it’s port number

Port numbers and protocols:

Types of connection: TCP and UDP

<https://docs.python.org/3/library/socket.html>

* As discussed above we will create 2 files **server.py** and **client.py** to see how the socket works.

To work with socket in python, we need to import a module called ***socket***

Create a server socket s = socket.socket

import socket

s = socket.socket() #we will create a socket with socket module and socket function and this function takes 2 parameters IPV4/IPV6 and the other is TCP/UDP - we will leave it blank it takes ipv4 and tcp by default

print("socket created") #once we had created the socket we will print it.

s.bind(("localhost", 9999)) #we need to make this as server socket to accept the connection. it accept it we need a ip address and port number. We are binding a socket with a port number.

#since we are using our own machine as server we will use localhost and which port are we using- any port that is available or not assigned for some other task. bind takes one argument NOT two hence there are two brackets.

#discuss port number ranges 0 - 65535

s.listen(3) #now let's define how many connections we want to accept/queue them - here we will allow 3 clients to connect

print("waiting for connections")

#how many connections you want to accept, think of when you are working with client 1, client 2 comes in with new connection and same with client 3. We need to do this in continuously - hence we will use a loop.

while True:

    c, addr = s.accept() #accept a conenction from a client, and it will return/give 2 things 1) client socket 2)address - we need to accept both

    print("Connected with ", addr) #connected with we can print the addr of the client

    c.send("Connection established successfully") #after establishing the connection we will send something to the client.

    c.close()

* Now this will produce and output “connections created” “waiting for connections” and it started serving and now we need a client to connect. How do we do it? Check the below code
* import socket
* c = socket.socket()
* #server will bind to the particular socket and client will connect to it.
* c.connect(('localhost', 9999))#mention ip addr of the server and port number
* # now if you run the code till this part it won't return anything as we are not printing, however if the server it running we will notice an error "a byte like object is required, not str"

Graphical user interface, text, application

Description automatically generated

So now let’s make a change to server.py for ‘connection established successfully’ line as shown below.

Bytes for converting the string to bytes and it’s format is utf-8. After updating run the server.py and client.py and the connection should get established. Observe the output ipaddress and port number of the client.

Text

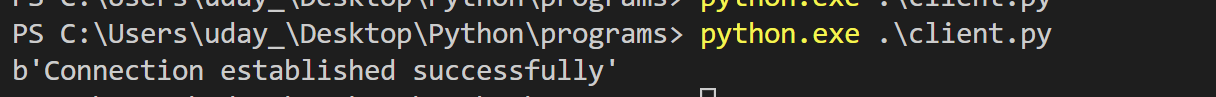
Description automatically generated

* Now server is sending something and client should receive it and to receive it we need to write another line of code in client.py using recv() function and buffer size 1024

Graphical user interface, text, application

Description automatically generated

Now run the client.py again and we should observe that we had received “connection established successfully”.



Now on the server side we can see that we have 2 connections and if we run client.py again we can see that server has 3 connections with different ports. In the output we can see that we are printing **b** as well. B – bytes which is a format, now we don’t want to print it and we need to convert it to str format and we can achieve it by using decode as shown below in **client.py**

A picture containing graphical user interface

Description automatically generated

Text, chat or text message

Description automatically generated

* Now how to make this more interesting: what if when client sends a request which means client is sending the IP address, but all the clients on this machine have the same IP Address and what if we can send the name as well after establishing the connection, we will ask the user to input/enter their name. We will send this name to the server using **send()** function in byte format and utf-8 format.
* On the server side once you accept the connection before sending data you have to accept the value from the client. Go to server.py receive the name and print the name along with address as highlighted below.

Graphical user interface, text, application, email

Description automatically generated

* Rerun the server.py and we get usual output saying waiting for connections, now go to client.py run it and it should ask for the name – enter the name you want and hit enter. Output on the client side is usual and observe the server side which will print the name you had entered before.
* Let’s discuss what is utf-8 : <https://www.w3schools.com/charsets/ref_html_utf8.asp>

The same can be done on two different machines and instead of localhost you can use public/private IP’s considering the networking settings and you can create a chat bot or something similar and play around.

Client.py

import socket

c = socket.socket()

c.connect(('localhost', 4444))

name = input('enter your name: ')

c.send(bytes(name, 'utf-8'))

print(c.recv(1024).decode())

server.py

import socket

s = socket.socket()

print("Socket created")

s.bind(("localhost",4444))

s.listen(3)

print("waiting for connections")

while True:

    c, addr = s.accept()

    name = c.recv(1024).decode()

    print("connected with ", addr, name)

    c.send(bytes("welcome to the chat room", 'utf-8'))

    c.close()