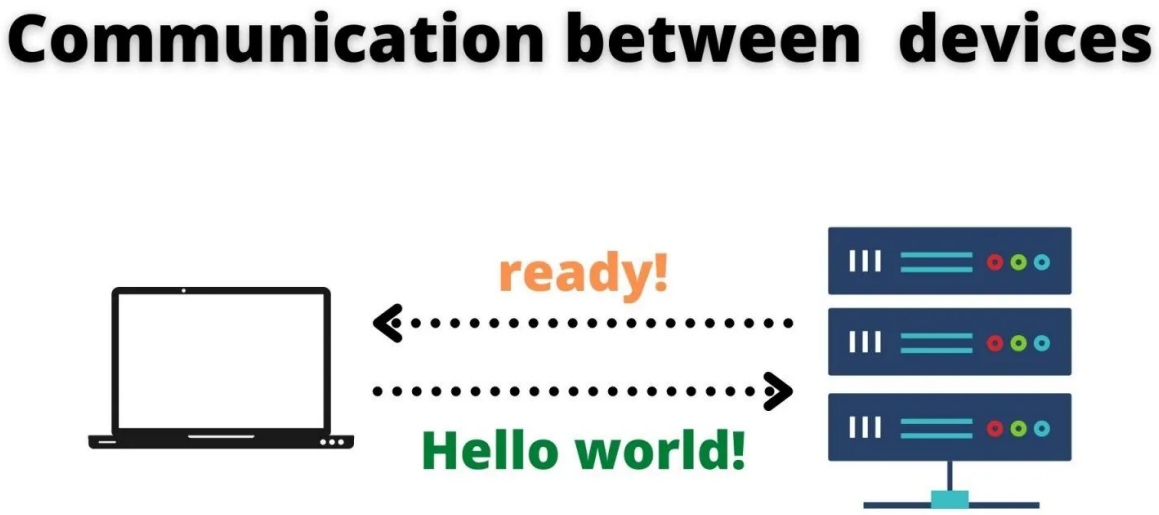
CoolplayDev

TCP Client: How to create a simple TCP client with Python

A TCP client is the counterpart to the important TCP server. Learn how to create a basic TCP client on your own with Python!

OLIVER ZINK / MARCH 14, 2022 / NETWORKING, PROGRAMMING

TCP Clients are the counterpart of TCP Servers and they are necessary to make connections between computers with the TCP protocol possible. TCP plays a very big role in networking and you need to understand how it works if you want to deal with networking with computers. Read this article now to find out how to create your own TCP client in Python!



Introduction

Fortunately, creating a basic TCP client, which can communicate with a server, isn't very difficult in Python. However, I **briefly** want to explain some important key terms.

- TCP stands for **Transmission Control Protocol** and is part of the TCP/IP suite. The difference between TCP and UDP (**User Datagram Protocol**) is that TCP makes sure that there is a connection between devices and all packages are received by the receiver. UDP, however, just sends the packages no matter whether they are even received by anyone.
- A socket is an interface for the exchange of data on the same device or between different devices. It is provided and managed by the operating system.

Creating a TCP Client in Python

The first thing to do is, like in almost every other Python code, to import some modules. Luckily we only need one module that automatically comes with Python. It is the **socket** module and we use it for creating a network socket for TCP communication.

```
import socket
```

Next, we define two variables for saving the IP address of the server and the port it listens on.

```
target_host = "192.168.178.73" # Change this to the IP address of your server
target_port = 27700 # Change this to the port of your server
```

After we've defined those 2 variables, we need to create a new socket and try to connect it to the defined IP address on the defined port.

```
# create a socket
client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.settimeout(1)
# connect to the server
client.connect((target_host, target_port))
```

client.settimeout() determines the time (in seconds) the client will wait for a response from the server before the connection times out and the client disconnects.

Now, we wait to receive a specific keyword and then send hello world to the server.

```
# receive
response = client.recv(4096)
if response.decode() == "ready":
    print("Successful")
else:
    print("Not successful")
# send
client.send("hello world".encode())
```

The number **4096 in client.recv(4096)** specifies the maximum amount of bytes the client receives from the server. That means if the server sends a string that is bigger than the specified amount of bytes, the client will not receive the complete string.

That's it, you now have a very basic TCP client that you can include into any project and update however you want. To update this client, you'll need some basic lines of code for sending and receiving data:

- **response = client.recv(4096)**
    - With this line of code the client will wait for the server to send a message to it. After receiving the message you can use it as a usual string with **response.decode()**

- **client.send("any string".encode())**
  - This line of code encodes any string and sends it to the server

Here is the complete code:

```
import socket

target_host = "192.168.178.73" # Change this to the IP address of your server
target_port = 27700 # Change this to the port of your server

# create a socket
client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.settimeout(1)
# connect to the server
client.connect((target_host, target_port))
# receive
response = client.recv(4096)
if response.decode() == "ready":
    print("Successful")
else:
    print("Not successful")
# send
client.send("hello world".encode())
```

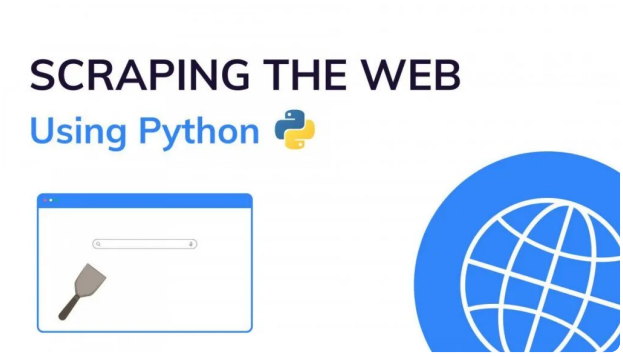
Summary

That's it! You now got your own TCP client that you can upgrade and use in any project you want. If you now want to test your client with a TCP server it can connect to, you can read this article on how to create a TCP server.

Thanks for reading!

Share this article

Interesting Reads

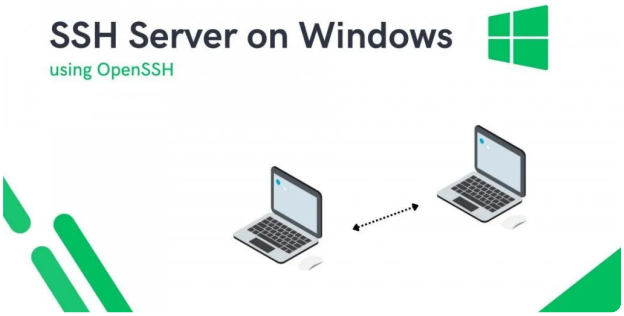


A Beginner's Guide to Web Scraping with Python [2023]

August 25, 2023

Python to Executable: Easily Convert a Python File to an Executable

April 23, 2023



SSH Server on Windows: How to set up OpenSSH on Windows

April 9, 2023

Navigation

- Home
- Blog
- Contact

Categories

- Programming
- Computer
- Networking

Tools

- Bin Dec Hex Converter
- LZW Encoder

Important Links

- Privacy Policy
- Terms & Conditions
- Imprint