Developpment Document

Client :

connection to a server :

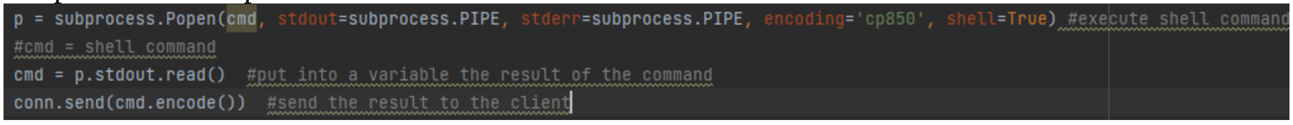
to connect to a server, I use **sockets**, input of ip and port for connecting to a serveur is made from the GUI (**PyQt5**).
When connecting, if it goes well, connection's status changed into « connected », but when it fails, a window will pop up to give the error, and the connection's status remains « disconnected » and the client cannot send message.

Send and receive message :

Thread and **socket**, launching a thread ot receive message at any time, and for sending messages, we just need a loop of inputs and send input
The thread stop when the client receive a specific message from the server, but also the client can no longer send messages, the button to send messages is blocked when the connection's status is « disconnected ».

Getting server's information :

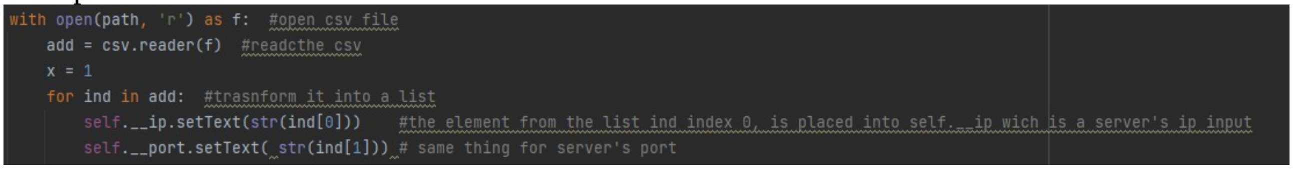
Subprocess and **psutil (server),** using subprocess to execute a shell command from python, and psutil give immediately server's information without executing shell command.
Subprocess : example :



```
p = subprocess.Popen(cmd, stdout=subprocess.PIPE, stderr=subprocess.PIPE, encoding='cp850', shell=True) #execute shell command
#cmd = shell command
cmd = p.stdout.read()  #put into a variable the result of the command
conn.send(cmd.encode())  #send the result to the client
```

Read csv file :

I use **csv** module, to connect to a server by reading the csv file, I need to transform the file into a list, elements are separated by a « , » for example : 127.0.0.1 **,** 10017 or IP **,** port
Once it transfromed into a list, we can take out 1 element by using idex x, list[x], and then put it into a variable that we have defined.
example :



```
with open(path, 'r') as f:  #open csv file
    add = csv.reader(f)  #readcthe csv
    x = 1
    for ind in add:  #trasnform it into a list
        self.__ip.setText(str(ind[0]))    #the element from the list ind index 0, is placed into self.__ip wich is a server's ip input
        self.__port.setText( str(ind[1])) # same thing for server's port
```

Client's exceptions :

if the client try to connect to a server but : the server is down or the informations are incorrects, a pop up window will show up and show the error.
If the client is connected to the server, and suddenly the server stops, a window will show up to tell the client that the server is unavailable and the client will be disconnected from the server automatically.

Server :

receive and send message :

Using the module **Socket**, to send and receive messages. As long as the server is connected to a client, it will start a loop, the loop will not stop until the client send « kill » message. In the loop the server try to receive a message from the client, the server will react depends on the client's message, if its a message that it recognize (by using « if... »), it will answer the client and send what the client asked for (RAM, OS, disconnect, kill etc..), If it's not, it will just stay silent.

recoignizing a message :

psutil and **subprocess** modules. If the server receive a message from the client that it recognize (using « if 'condition' : ... »), the server do the task requested, it will use psutil for cpu and ram (ex : ***psutil.cpu_percent()***), subprocesse for ip, os and name, by executing shell command from python.

Restart, client disconnect and stop server :

the server will restart if the client send « reset », it will close its connection, re-initiate a new socket and get a new IP, and then start to wait a connection from a client.
Disconnecting a client will juste close its connection when it receive « disconnect » from the client and wait a new connection from a client.
Stop the server, the server will close connction, adn stop all process.

Server's exception :

a message will show on terminal the error.

Successful tasks :

- to connect to only one server by using server's IP or name, and of cours the port.
- cannot send messages if the client is not connected to a server.
- Execute commands from the client : **disconnect**, **reset**, **kill**, **OS**, **CPU**, **RAM**, **NAME** and **IP-**.
- Graphic User Interface : connect to a server by chosing a csv file or by typing server's IP and port, if the information are incorrects or the server is down, a pop up window will show and show an error .
- Asynchronous communication between server and client.
- Read csv file.

Unsuccessful tasks :

- The client cannot manage multi-connection to servers.
- **DOS:**, **Linux:** and **Powershell;** commands are not operational.
- Cannot add and save new servers on a new file.