

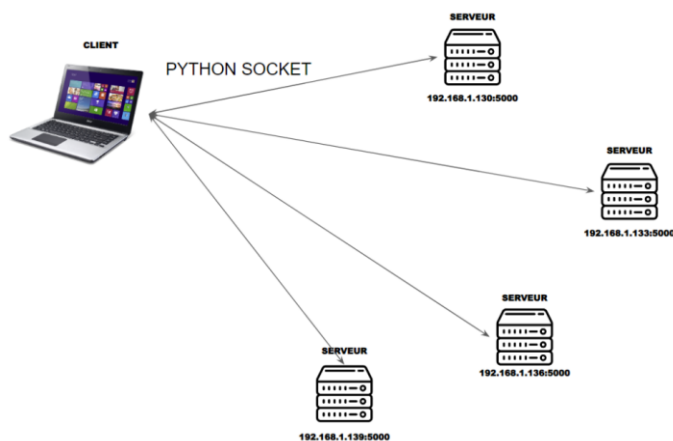
# SAE 3.02

## User manual

To begin with, the purpose of our SAE is to be able to communicate between servers and a client. For this we will create 2 files based on the use of the "socket" and "thread" for communication between the two. All this will be seen directly for the client on a graphical interface that the client will have the right to access.

The SAE project will take place over a period of 4 weeks with dedicated hours of approximately 45 hours. This is a one-off job to do with the possibility that designer teachers can help fix some issues or bugs in the code that each student will create.

### La topologie choisie :



## **The installations to be done in the cmd**

In your cmd of your software where you will put the code files you must install different libraries:

-PyQt5

-psutil

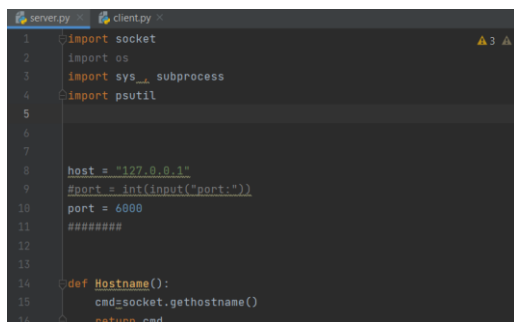
Use the command “pip install PyQt5” and “pip install psutil”

## **Server launch**

For the launch of the server file it is called "server.py" it is very simple I will give you a list of the steps to launch it without problem with visual explanation in support.

- open the "server.py" file and drag it into your software.

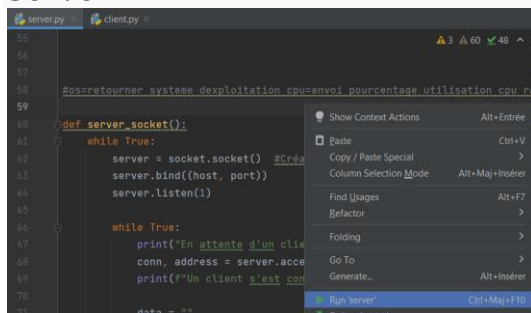
- Go to your server file on your software "this can be like "Pycharm" software like software.



```
1 import socket
2 import os
3 import sys, subprocess
4 import psutil
5
6
7
8 host = "127.0.0.1"
9 #port = int(input("port:"))
10 port = 6000
11 #####
12
13
14 def Hostname():
15     cmd=socket.gethostname()
16     return cmd
```

- Right click on the server.py file and press "Run

Server »



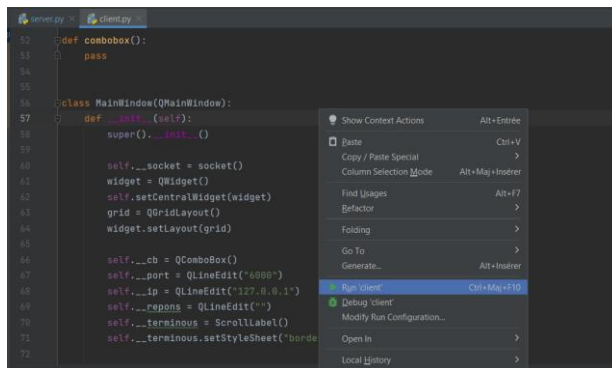
```
55
56
57
58 #os=retourner systeme d'exploitation cpu=envoi pourcentage utilisation cpu
59
60 def server_socket():
61     while True:
62         server = socket.socket()
63         server.bind((host, port))
64         server.listen(1)
65
66     while True:
67         print("En attente d'un client")
68         conn, address = server.accept()
69         print(f"Un client s'est connecté")
70         data = ""
```

-A terminal has just opened below the files, so it is the server terminal. A text has just appeared in it with the registration waiting for a client, which means that our server is waiting for almost the same procedure on our client.

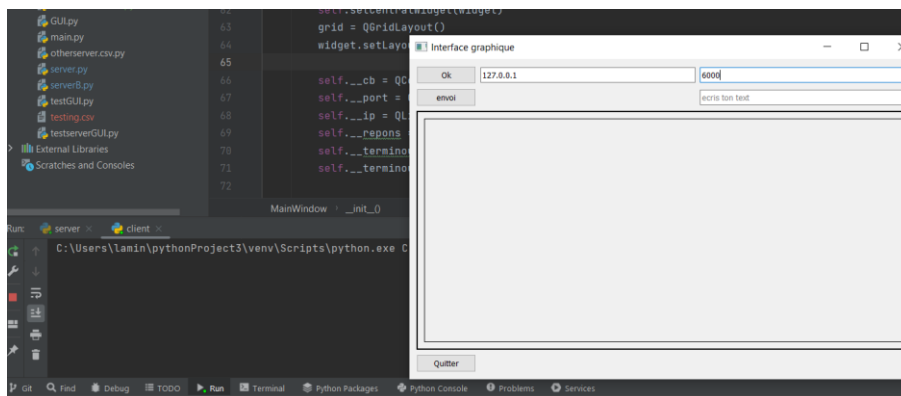
## **Customer part**

For the client I will show you the following so that we have a working client that can communicate with the server:

-After following the same server steps, we need to open the server file as before and right-click on it.

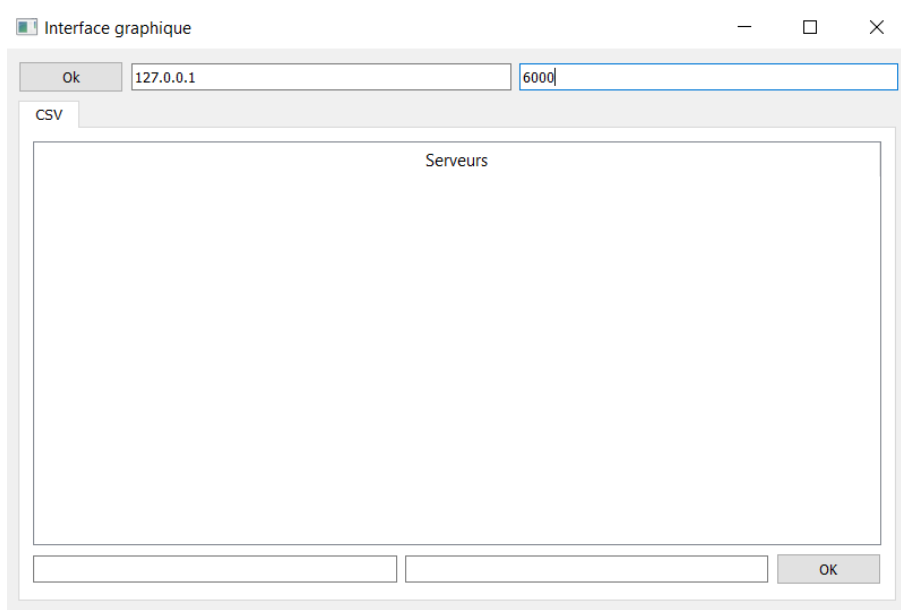


-Then we will click on the “Run ‘client’” button and we will see at the bottom yet another terminal being created which will be our client’s terminal.

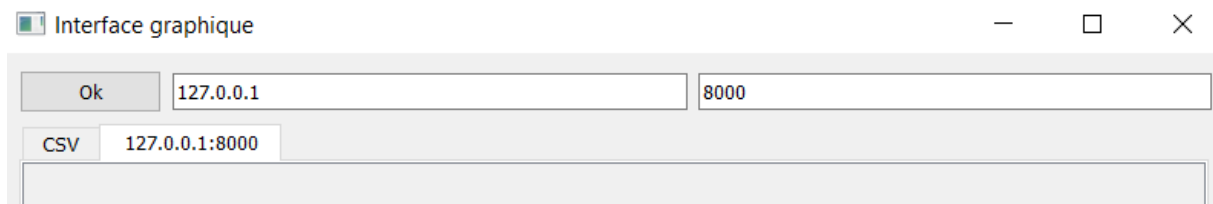


- We also have an interface that will be displayed and that we will see after the "Run client" through this interface, we will connect and communicate

on our server. To allow this communication between the server and the client it will be necessary to give the same IP and the port of the server (which we have at the beginning of the code of the file "server.py" with "the Port and the host" to the client to connect .

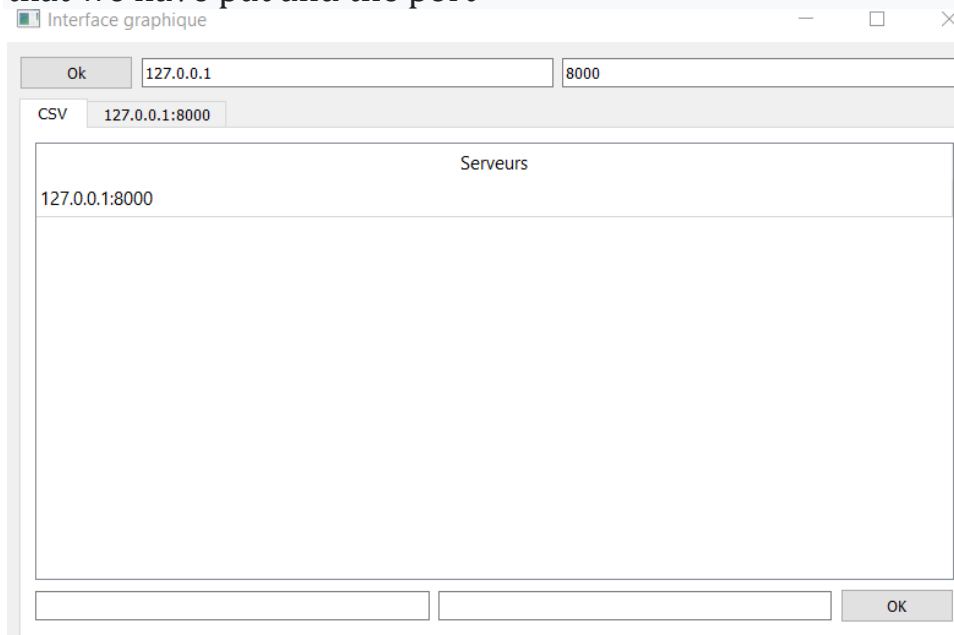


We only have to write our commands on our graphical interface but all the commands do not work if I mark "toto" our interface will tell us that it is "an invalid command".



-After putting the IP and the port We can see that the connection to the server has been created.

-After to set up a csv database you have to go to the csv tab and put the IP that we have put and the port



Here we will show you the different commands you can use in our client's cmd:

- **Possible machine orders :**

- la commande « os » : demande l'OS et le type d'OS par exemple MacOS Darwin ou Linux Ubuntu 21.4
- commande « ram » : mémoire totale, mémoire utilisée et mémoire libre restante
- commande « CPU » : utilisation de la CPU
- Commande « IP » : adresse IP
- Commande « name » : nom de la machine

- **From commands to machine servers:**

- "disconnect" command: disconnection of the interface allowing the monitored machine to be freed to allow the server to be freed for other requests
- "kill" command: kills the server which means that will turn off the server and the client.
- "reset" command: server reset.

**-Command interface will send commands given by the user, for example:**

- - commande « DOS:dir » ou DOS :{la command qui me plairez exemple mkdir,ipconfig)
  - commande « DOS:mkdir toto »
  - Linux:ls -la
  - -commande « Powershell:get-process »
  - commande « python –version »
  - -commande « ping 192.157.65.78 »
  - Linux:ls -la
  - -commande « Powershell:get-process »
  - commande « python –version »
  - -commande « ping 192.157.65.78 »

