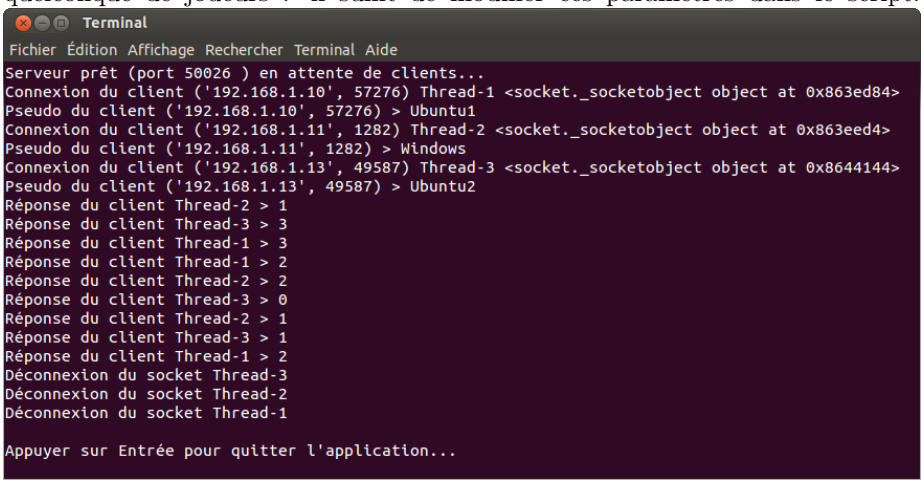
Les applications et jeux en réseau se basent sur le modèle clients/serveur. Au niveau de la programmation, la tâche est ardue car il faut avoir des connaissances en réseau (module `socket`) et en multithreading (module `threading`) : - Un `socket` permet de connecter deux machines à travers un réseau. Ainsi, pour un jeu en réseau avec 10 joueurs (soit 10 clients et 1 serveur), il faut créer 10 sockets (chaque client est connecté au serveur). Dans le cas du réseau Internet, les sockets se servent du protocole IP (couche réseau) et du protocole TCP pour la couche transport (il existe aussi le protocole UDP qui est plus simple, plus rapide mais non fiable). - Le `multithreading` est une technique qui permet d'exécuter plusieurs tâches (`thread`) en même temps et de manière indépendante dans un même processus. Pour un jeu en réseau avec 10 joueurs, le serveur utilisera 10 threads pour communiquer individuellement avec chaque client. Chaque thread gère donc le socket du client. On retrouve le multithreading du côté de l'application cliente : pour traiter l'émission et la réception en parallèle, on utilise un thread pour parler au serveur, et un autre pour écouter le serveur.

### Exemple : un QCM en réseau

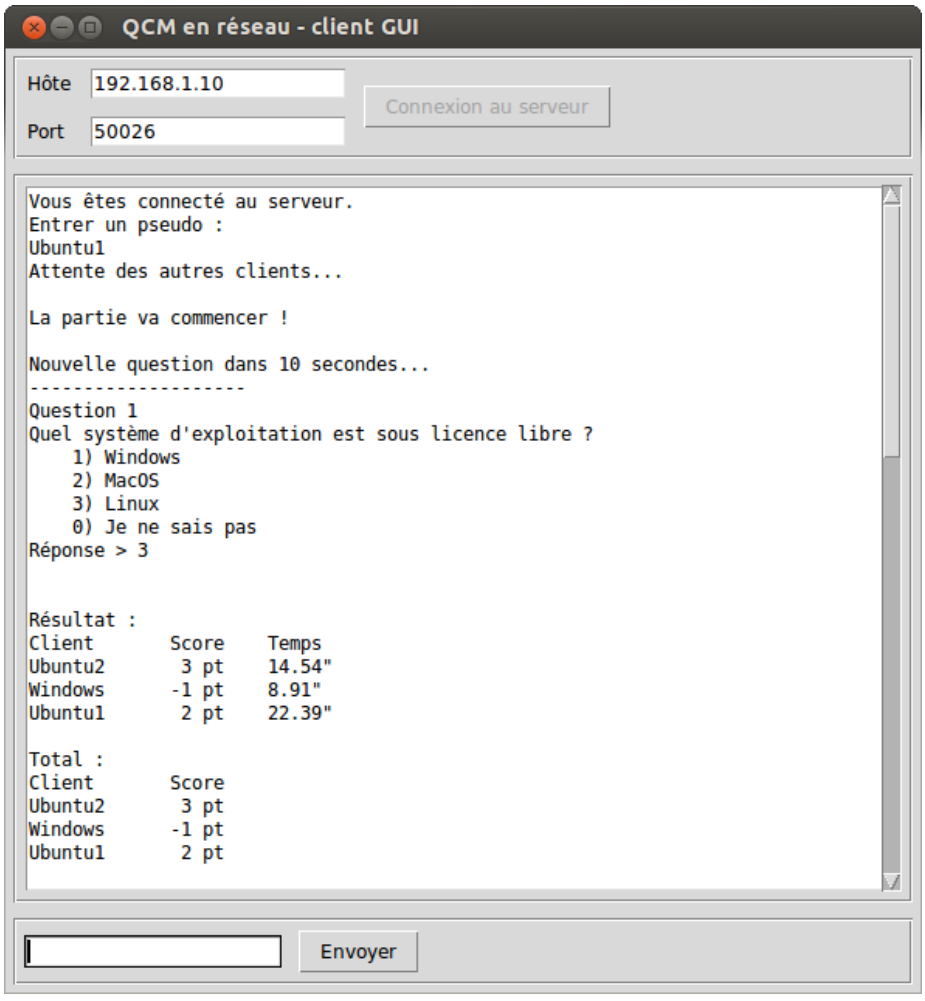
Il s'agit d'un jeu multijoueurs. On peut y jouer en réseau local (LAN) ou sur Internet. Chaque joueur lance son application cliente pour se connecter au serveur. La première étape consiste à créer un pseudonyme, puis quand le nombre de joueurs est suffisant (3 dans l'exemple ci-dessous), la partie commence. Le serveur envoie alors simultanément la première question à l'ensemble des joueurs : le but est d'y répondre correctement et le plus rapidement possible. Quand tout le monde a répondu ou si la limite de temps est dépassée, on passe à la question suivante... Un classement est établi à partir d'un système de points.

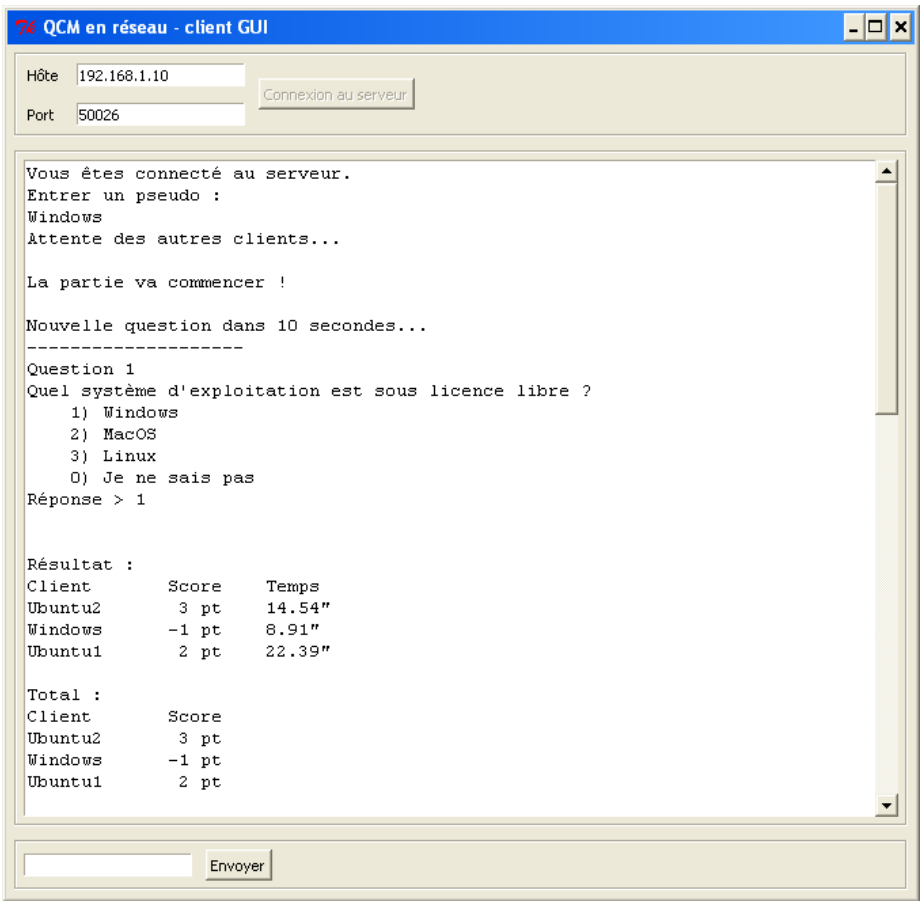
**Le serveur** Avant toute chose, il faut lancer le serveur. Le port par défaut est 50026 et le nombre de joueurs est 3. Mais on peut choisir pour le numéro de port une valeur arbitraire comprise entre 49152 et 65535, ainsi qu'un nombre quelconque de joueurs : il suffit de modifier ces paramètres dans le script.

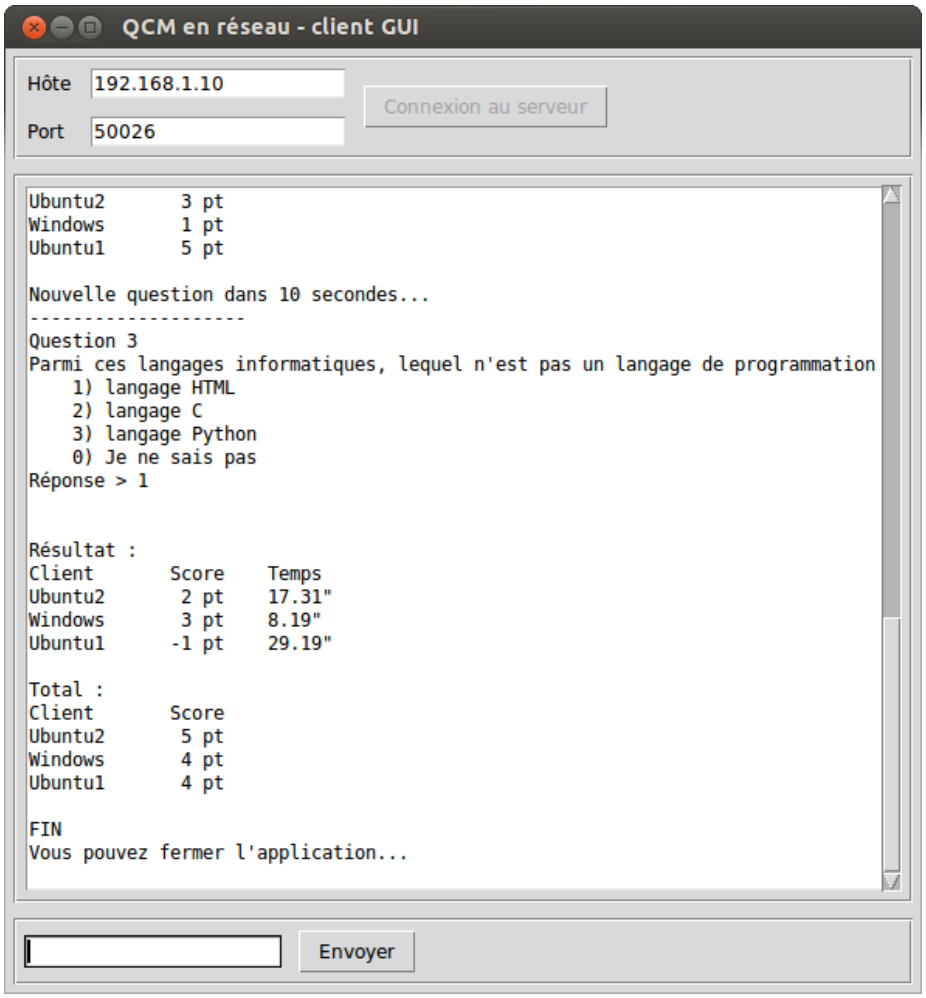


**Les clients** Ici, on joue à trois en réseau local (chez vous, derrière votre box, par exemple). Pour se connecter au serveur, il faut connaître son adresse IP (192.168.1.10) et le port qu'il utilise (50026). Python étant multiplateforme, on peut mélanger les systèmes d'exploitation : Windows, Linux, Mac... Dans cet exemple :

- le serveur et un client tournent sur le même ordinateur sous Linux/Ubuntu (192.168.1.10)
- un deuxième client sur un ordinateur sous Windows (192.168.1.11)
- un troisième client sur un troisième ordinateur sous Linux/Ubuntu (192.168.1.13)




QCM en réseau - client GUI
Hôte
192.168.1.10
Port
50026
Connexion au serveur
Vous êtes connecté au serveur.
Entrer un pseudo :
Windows
Attente des autres clients...
La partie va commencer !
Nouvelle question dans 10 secondes...
Question 1
Quel système d'exploitation est sous licence libre ?
1) Windows
2) MacOS
3) Linux
0) Je ne sais pas
Réponse > 1
Résultat :
Client Score Temps
Ubuntu2 3 pt 14.54"
Windows -1 pt 8.91"
Ubuntu1 2 pt 22.39"
Total :
Client Score
Ubuntu2 3 pt
Windows -1 pt
Ubuntu1 2 pt
Envoyer




Finalement, c'est Ubuntu qui gagne devant Windows ;)

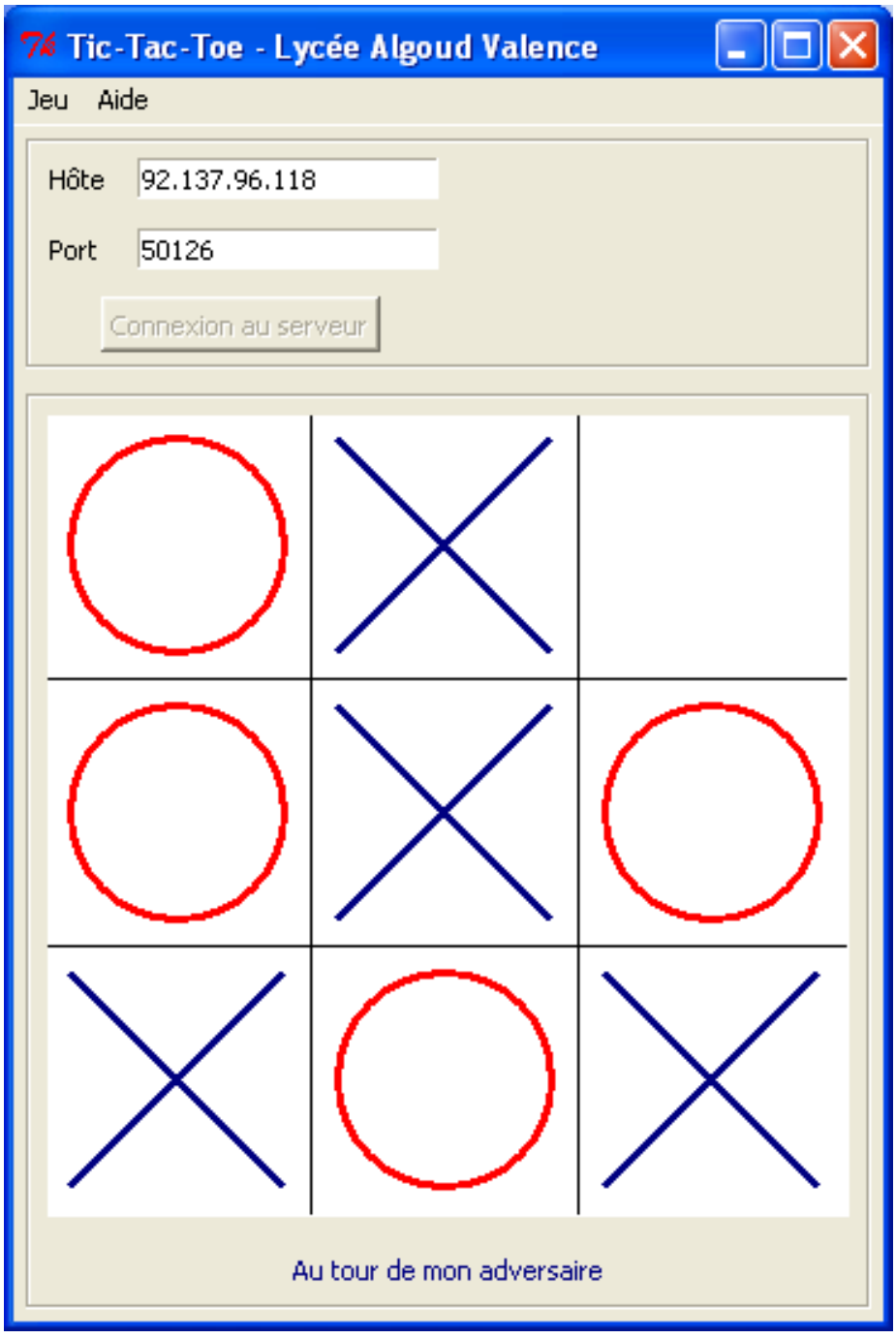
**Remarques** Cette application fonctionne mais est loin d'être parfaite : les pertes de connexion ne sont pas gérées, etc... Si vous avez testé cette application, cela m'intéresse d'avoir un retour de votre part.

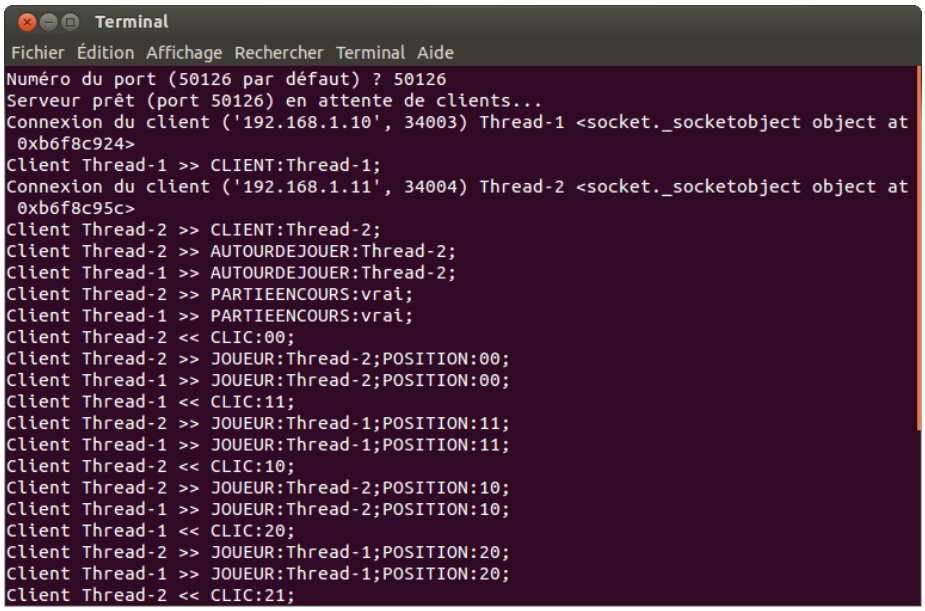
### Télécharger les scripts

- serveur
- client (mode graphique avec Tkinter)
- client (mode console)

**Projet**

**Jeu Tic-Tac-Toe en réseau (jeu du morpion)** Ecrire l'application serveur et l'application cliente d'un jeu de morpion en réseau local (LAN) ou sur Internet (plus d'informations ici) :


Tic-Tac-Toe - Lycée Algoud Valence
Jeu
Aide
Hôte
92.137.96.118
Port
50126
Connexion au serveur
Au tour de mon adversaire




### Bibliographie

Le cours de Gérard Swinnen, avec deux exemples concrets à essayer (un chat et un jeu de bombardes) :

- inforef.be/swi/python.htm
- python.developpez.com/cours/TutoSwinnen/?page=page_20#L18

Le livre de Brandon Rhodes et John Goerzen :

- Foundations of Python Network Programming (Apress)

La documentation officielle de Python :

- socket : low-level networking interface
- threading : higher-level threading interface