1. La notion de Sockets

Comme nous l'avons vu précédemment la communication entre machines est complexe et beaucoup d'applications peuvent échanger des informations en même temps sur une seule machine. Pour faire le tri entre toutes ces informations, chaque connexion possède des « fils » différents appelés « ports » qui évitent que toutes ces informations se télescopent.

Les machines sont identifiées par leur adresse IP et les applications qui communiquent, par le port utilisé.
Exemple :

- le port 21 utilisé par le protocole FTP (transfert de fichiers)
- le port 80 utilisé par le protocole HTTP (serveur web),
- le port 443 utilisé par le protocole HTTPS (serveur web sécurisé).

Il en existe 65535 (codé sur 16 bits = 2^{16}) et les 1024 premiers sont réservés à des usages courants.

Dans le document de cours, vous avez vu qu'il existe 2 protocoles de communication entre machines :

- l'UDP qui envoie des informations au travers d'un réseau sans se préoccuper de savoir si elles ont été bien réceptionnées. Il n'y a pas de connexion à établir.
- Le TCP qui est plus sûr car il vérifie que les informations ont bien été réceptionnées, nécessite d'établir une connexion. C'est ce dernier protocole qui va nous intéresser pour la suite.
-

Pour établir une connexion entre un serveur et un client, il est nécessaire de définir deux éléments :

- l'adresse IP du serveur (ou le nom d'hôte)
- le port utilisé, que le serveur va scruter en continu à l'écoute de toute information provenant du ou des clients.

Ces 2 éléments sont définis dans un élément logiciel que l'on appelle un **SOCKET**.

Nous allons définir 2 machines :

- Un serveur :
    1. Attend une connexion
    2. Accepte la connexion
    3. Echange des informations
    4. Ferme la connexion

- Un client :
    1. Se connecte au serveur
    2. Echange des informations
    3. Ferme la connexion

2. Les sockets

Un socket est un élément informatique qui permet d'associer une application installée sur la machine avec un port de communication.

- Création du serveur

La première chose à faire est donc de **définir le socket** que l'on va utiliser.

```
Import socket

connexion = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
```

SOCK_STREAM est utilisé pour le protocole TCP

Ensuite on **connecte notre socket :**

```
connexion.bind(('X.X.X.X', 20180))
```

X.X.X.X est l'adresse IP de notre serveur
20180 est notre port d'écoute.

Ensuite on **demande à notre socket d'écouter** le port désigné :

```
connexion.listen(5)
```

On utilise la méthode listen souvent avec le paramètre 5. Si 5 clients différents demandent la connexion et que le serveur les refuse, la connexion se ferme.

Si un client demande la connexion on va **l'accepter** :

```
Connexion_avec_client,info_connexion = connexion.accept()
```

Ensuite on peut faire communiquer le serveur avec son client connecté (envoi de message, envoi de fichier) puis il faut **fermer la connexion** :

```
Connexion_avec_client.close()
```

- Création du client

On suit presque les mêmes étapes pour créer le client :

```
Import socket

Connexion_avec_serveur = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

On connecte le client :

```
Connexion_avec_serveur.connect = ('IP_serveur', port))
```

Rques : si le serveur et le client sont situés sur la même machine, on peut noter 'localhost' comme IP_serveur. Le port de connexion client est différents de celui du serveur.

A ce moment-là, les deux machines sont connectées.

- Scripts de base

Pour ce qui suit, vous allez travailler en binôme.

A l'aide de votre IDE python, chargez sur une machine, le script « serveur.py » et sur l'autre, le script « client.py ».

Modifier le script du client en renseignant l'adresse de la machine sur laquelle tourne le scipt serveur.

Examiner le fonctionnement de chaque script.