Python et les réseaux

Les réseaux informatiques permettent la communication et l'échange de données entre plusieurs machines. Avec Python, il est possible de manipuler facilement les protocoles réseau, d'automatiser des tâches de communication, ou de créer des applications client-serveur. Grâce à des bibliothèques comme `socket`, `requests` ou `asyncio`, Python offre une approche simple et puissante pour explorer et développer des solutions réseau, que ce soit pour l'apprentissage ou pour des projets professionnels.

Le socket

Un **socket** est un point de communication permettant à deux machines d'échanger des données sur un réseau. Il agit comme une interface entre une application et la couche réseau, facilitant l'envoi et la réception de messages. En Python, la bibliothèque `socket` permet de créer, configurer et utiliser des sockets pour établir des connexions réseau, que ce soit en mode client ou serveur.

Exemple d'utilisation des sockets

1. Socket TCP (Transmission Control Protocol)

Le protocole TCP permet une communication fiable et orientée connexion entre deux machines.

Serveur TCP :

```
import socket

server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind(('localhost', 12345))
server.listen(1)
conn, addr = server.accept()
data = conn.recv(1024)
conn.sendall(b'Recu: ' + data)
conn.close()
server.close()
```

Client TCP :

```
import socket

client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect(('localhost', 12345))
client.sendall(b'Bonjour serveur')
response = client.recv(1024)
print(response.decode())
client.close()
```

2. Socket UDP (User Datagram Protocol)

Le protocole UDP permet une communication rapide, sans connexion et sans garantie de livraison.

Serveur UDP :

```
import socket

server = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
server.bind(('localhost', 12345))
data, addr = server.recvfrom(1024)
server.sendto(b'Recu: ' + data, addr)
server.close()
```

Client UDP :

```
import socket

client = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
client.sendto(b'Bonjour serveur', ('localhost', 12345))
response, addr = client.recvfrom(1024)
print(response.decode())
client.close()
```

3. Socket HTTP (avec la bibliothèque `socket`)

Bien que des bibliothèques comme `requests` soient plus simples, il est possible de faire une requête HTTP basique avec un socket.

```
import socket

host = 'example.com'
port = 80
request = b"GET / HTTP/1.1\r\nHost: example.com\r\n\r\n"

client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect((host, port))
client.sendall(request)
response = client.recv(4096)
print(response.decode())
client.close()
```