BAC Pro CIEL	Python – Module 13 Communiquer en réseau avec socket	Année 2025/2026

A. A quoi sert socket ?

Le module socket permet de :

- ⇨ Créer des communications réseau
- ⇨ Faire du client/serveur
- ⇨ Envoyer et recevoir des données

C'est la base de :

- ⇨ TCP/UDP
- ⇨ Serveurs Web
- ⇨ Scripts réseau
- ⇨ Cybersécurité

B. Principe client / serveur

La communication s'organise tel que :

- ⇨ Le serveur attend
- ⇨ Le client se connecte

C. Importation du module

```
import socket
```

D. Création d'un socket

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

AF_INET : IPv4

SOCK_STREAM : TCP

E. Base d'un serveur TCP

```
1  import socket
2
3  serveur = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
4
5  serveur.bind(("0.0.0.0", 5000))  # IP + PORT
6  serveur.listen()
7
8  print("Serveur en attente...")
9
10 client, adresse = serveur.accept()
11
12 print("Client connecté :", adresse)
13
14 data = client.recv(1024)
15 print("Reçu :", data.decode())
16
17 client.send("Message reçu".encode())
18
19 client.close()
20 serveur.close()
```

Ligne	
1	Importation du module socket
3	Création d'un socket serveur TCP en IPv4
5	On indique au serveur sur quelle IP écouter et sur quel port (0.0.0.0 écoute toutes les interfaces réseau) et 5000 est le port utilisé
6	Mise en écoute, le serveur passe en mode attente de connexion
10	Le serveur reste bloqué ici jusqu'à ce qu'un client se connecte. *client* étant le socket dédié à ce client et *adresse* le tuple (IP,PORT) du client.
14	Réception des données depuis le client, le serveur reçoit des données (1024 octets maximum)
15	.decode() est obligatoire pour lire du texte, il permet la conversion byte ➔ texte.
17	Envoi de données au client. .encode() permet la conversion texte ➔ byte
19	Coupe la connexion avec le client
20	Fermeture du socket et donc arrêt complet du serveur d'écoute TCP.

F. Base d'un client TCP

```
1  import socket
2
3  client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
4
5  client.connect(("127.0.0.1", 5000))
6
7  client.send("Bonjour serveur".encode())
8
9  data = client.recv(1024)
10
11 print("Réponse :", data.decode())
12
13 client.close()
```

Ligne	
1	Importation du module socket
3	Création d'un socket client TCP en IPv4
5	Connexion vers le serveur dont l'adresse IP est 127.0.0.1 et le port 5000
7	Envoi d'un message vers le serveur. .encode() permet la conversion texte ➔ byte
9	Réception des messages depuis le serveur dans la limite de 1024 octets
11	Affichage de la donnée reçue. .decode() permet la conversion byte ➔ texte
13	Coupe la connexion avec le serveur.

G. Fonctions importantes

socket()	Crée le socket
bind()	Associe IP + port
listen()	Met en attente
accept()	Accepte connexion
connect()	Connexion client
send()	Envoie
recv()	Reçoit
close()	Ferme

H. Traitement du texte

Les sockets travaillent en bytes ! Il faut toujours encoder ou décoder les données.

Envoyer une donnée :

```
client.send("Bonjour".encode())
```

Recevoir une donnée

```
data = client.recv(1024).decode()
```

I. Serveur multi-clients (boucle)

```
while True:
    client, adresse = serveur.accept()
    print("Connexion :", adresse)

    data = client.recv(1024)
    print(data.decode())

    client.close()
```

J. Erreurs fréquentes

Oublier .encode()

⇨ Erreur de type

Mauvaise IP / port

⇨ Connexion refusée

recv(1024) mal compris

⇨ Ce n'est pas garanti de recevoir tout (limité à 1024 octets)

K. A retenir

socket permet de créer une communication réseau

AF_INET correspond à IPv4

SOCK_STREAM correspond à TCP

bind() est utilisé côté serveur

connect() est utilisé côté client

send() envoie des données

recv() reçoit des données

encode() convertit texte → bytes

decode() convetit bytes → texte

close() ferme la connexion

L. Synthèse