université
PARIS-SACLAY

Programmation en Python
Master 2 Réseaux Télécoms
TP 6

Ce TP étant consacré à la communication client-serveur, il sera nécessaire de lancer plusieurs scripts de Python simultanément. Notez qu'actuellement VS Code ne permet pas de le faire de façon naturelle ; il est à vous d'ouvrir manuellement les terminaux supplémentaires et de copier la commande de lancement du script dans ces terminaux.

1 Interrogation d'un service Web REST

On se propose de créer une fonction nommée `temperature` qui affiche la température courante dans une ville dont le nom est passé en son argument. À cette fin on utilisera l'API du serveur *Open Weather Map* :
`https://openweathermap.org/current`
avec la clé suivante : `29f14a7a8876ecb1219ecccd26ce4c1e`. Voici un exemple de la réponse donnée par le serveur :

```
<current>
<city id="2989204" name="Orsay">
<coord lon="2.1873" lat="48.6957"/>
<country>FR</country>
<timezone>3600</timezone>
<sun rise="2021-11-06T06:45:45" set="2021-11-06T16:24:02"/>
</city>
<temperature value="278.93" min="277.13" max="281.25" unit="kelvin"/
<feels_like value="278.93" unit="kelvin"/>
<humidity value="89" unit="%"/>
<pressure value="1027" unit="hPa"/>
<wind>
<speed value="0.89" unit="m/s" name="Calm"/>
```

```
<gusts value="2.68"/>
<direction value="219" code="SW" name="Southwest"/>
</wind>
<clouds value="0" name="clear sky"/>
<visibility value="10000"/>
<precipitation mode="no"/>
<weather number="800" value="clear sky" icon="01n"/>
<lastupdate value="2021-11-06T20:49:19"/>
</current>
```

On utilisera la classe `request` du module `urllib` ainsi que la classe `ElementTree` du module `xml.etree`. N'oubliez pas de soustraire 273,16 de la valeur de la température donnée par le serveur (elle est donnée en degrés Kelvin !)

2 Serveur horloge mono-tâche

Dans cet exercice on créera un serveur mono-tâche qui enverra au client connecté une chaîne de caractères contenant l'heure courante. On utilisera le module `socket` pour réaliser ce serveur. Le code du client vous est fourni dans le fichier `horloge_client.py`. Le serveur doit fermer la connexion avec le client immédiatement après lui avoir répondu, et se remettre à l'écoute de nouvelles connexions. Le serveur sera attaché au port numéro 51793 et enverra l'heure dans le format `HH:mm:ss`.

N.B. : N'oubliez pas que le type des données envoyées et reçues par les sockets de réseau est `bytes`, et non `str`. Il est donc nécessaire de procéder au codage et décodage lors de transformation de données entre ces deux types.

3 Serveur écho mono-tâche

Dans cet exercice on doit réaliser un serveur dit *écho*, qui répond au client en lui renvoyant son propre message, précédé par un préfixe `echo:`. Le code du client vous est fourni dans le fichier `echo_client.py`. Notez que si l'utilisateur du client entre le mot `quit`, le client s'arrête. Si le serveur détecte la perte de connexion avec le client, il doit immédiatement se remettre à l'écoute sur son propre port (49631).

Essayez de lancer deux clients en parallèle. Est-ce que le second client arrive à ce connecter au serveur tant que le premier est toujours connecté ? Pourquoi ?

4 Serveur écho multi-tâche

On va maintenant modifier le code du serveur *écho* de sorte qu'il puisse desservir plusieurs clients de façon concurrentielle. A chaque nouvelle connexion, le serveur maître (la fonction `echo_multi`) créera un *thread* dédié qui lance la fonction `speak_to_client`. Cette dernière s'occupera de toute communication avec le client nouvellement connecté.

Vérifiez que maintenant le serveur peut communiquer avec plusieurs clients simultanément.