

Réalisation du projet de threading avec asynchronisation

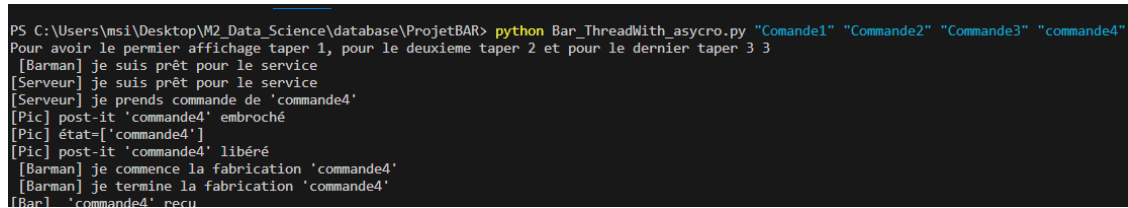
Youness Chaanani

Octobre 2023

1 Utilisation du programme

Vous devrez exécuter le programme dans le terminal en utilisant la commande suivante : `python Nomdu-fichier.py "commande1" "commande2" "commande3" ...`. Ensuite, le programme vous demandera de saisir un nombre entier compris entre 1 et 3, qui déterminera le niveau de verbosité de l’affichage. Si vous souhaitez afficher toutes les informations, y compris les détails des pics et des barres, saisissez 3. Si vous préférez un affichage sans les détails des pics et des barres, saisissez 2. Pour un affichage normal, saisissez 1.

À chaque exécution, le programme créera un fichier dans le même répertoire où il se trouve, et ce fichier sera nommé en fonction de la date. Ce fichier enregistrera tous les détails de l’exécution, y compris le temps d’exécution de chaque opération.



```
PS C:\Users\msi\Desktop\M2_Data_Science\database\ProjetBAR> python Bar_ThreadWith_asyncro.py "Comande1" "Comande2" "Comande3" "commande4"
Pour avoir le premier affichage taper 1, pour le deuxieme taper 2 et pour le dernier taper 3 3
[Barman] je suis prêt pour le service
[Serveur] je suis prêt pour le service
[Serveur] je prends commande de 'commande4'
[Pic] post-it 'commande4' embroché
[Pic] état-['commande4']
[Pic] post-it 'commande4' libéré
[Barman] je commence la fabrication 'commande4'
[Barman] je termine la fabrication 'commande4'
[Bar] 'commande4' reçu
```

2 Les résultats

Dans la première version que j’ai développée en utilisant uniquement des threads, le programme traitait les commandes séquentiellement, c’est-à-dire qu’il prenait la première commande, la préparait, la servait, puis passait à la deuxième commande, en répétant ce processus jusqu’à la fin de la liste des commandes.

Ensuite, j’ai intégré la partie d’asynchronisation en ajustant les temps d’attente de manière à ce que les différentes étapes soient exécutées en parallèle par rapport à chaque méthode, permettant ainsi un traitement plus efficace et simultané des différentes commandes.

Ci-dessus, vous pouvez voir une capture d’écran affichant le résultat que j’ai obtenu lors de l’exécution.

```

[Bar] état=[]
PS C:\Users\msi\Desktop\M2_Data_Science\database\ProjetBAR> python Bar_ThreadWith_asyncro.py "Comande1" "Comande2" "Comande3"
Pour avoir le premier affichage taper 1, pour le deuxieme taper 2 et pour le dernier taper 3 3
[Barman] je suis prêt pour le service
[Serveur] je suis prêt pour le service
[Serveur] je prends commande de 'Comande3'
[Pic] post-it 'Comande3' embroché
[Pic] état=['Comande3']
[Pic] post-it 'Comande3' libéré
[Barman] je commence la fabrication 'Comande3'
[Barman] je termine la fabrication 'Comande3'
[Bar] 'Comande3' reçu
[Serveur] je sers 'Comande3'
[Bar] état=[]
[Bar] 'Comande3' évacué
[Barman] j'encaisse le client qui a demandé la commande 'Comande3'
[Serveur] je prends commande de 'Comande2'
[Pic] post-it 'Comande2' embroché
[Pic] état=['Comande2']
[Serveur] je prends commande de 'Comande1'
[Pic] post-it 'Comande1' embroché
[Pic] état=['Comande2', 'Comande1']
[Pic] post-it 'Comande1' libéré
[Barman] je commence la fabrication 'Comande1'
[Barman] je termine la fabrication 'Comande1'
[Bar] 'Comande1' reçu
[Bar] état=['Comande1']
[Serveur] il n'y a plus de commande à prendre
plus de commande à prendre
[Serveur] je sers 'Comande1'
[Bar] 'Comande1' évacué
[Barman] j'encaisse le client qui a demandé la commande 'Comande1'
[Pic] post-it 'Comande2' libéré
[Barman] je commence la fabrication 'Comande2'
[Barman] je termine la fabrication 'Comande2'
[Bar] 'Comande2' reçu
[Bar] état=['Comande2']
[Serveur] je sers 'Comande2'
[Bar] 'Comande2' évacué
[Barman] j'encaisse le client qui a demandé la commande 'Comande2'
[Pic] état=[]
Pic est vide
[Bar] état=[]

```