RAPPORT PROJET COCKTAIL (ASYNCHRONISME/THREADING)

1) Comment lancer le programme :

```
PS C:\Users\rpaul\Downloads> python3 BarThreading.py "1 cafés" "2 mojito" "3 rhums" --verb2
```

```
PS C:\Users\rpaul\Downloads> python3 BarThreading.py --verb3 "1 cafés" "2 mojito" "3 rhums"
```

```
PS C:\Users\rpaul\Downloads> python3 BarThreading.py "1 cafés" "2 mojito" "3 rhums"
```

Ci-dessus différentes manières de lancer le programme. Pour pouvoir le lancer, il faut aller dans sa console, se rendre à l'endroit où est situé le programme ''**BarThreadingv2.py**'' et utiliser une des lignes de commande ci-dessus. La ligne de commande est composée de plusieurs éléments, ''python3 BarThreadingv2.py'' qui permet d'indiquer quel programme nous souhaitons lancer, et ensuite il y a les commandes avec les guillemets, et le niveau de verbosité.

Il existe 3 **niveaux de verbosité** : --verb1, --verb2 et --verb3, du moins au plus complet, suite aux consignes de l'exercice, et il est possible d'indiquer le niveau de verbosité avant ou après les commandes. Si un niveau qui n'existe pas est indiqué, le programme retournera l'erreur associée, et si aucun niveau n'est indiqué, alors le niveau de verbosité par défaut est le premier niveau.

Pour ce qui concerne **les commandes**, tout ce qui est écrit et qui n'est pas une ligne de commande (donc les éléments commençant par un -), sera considéré comme un élément que le serveur va devoir commander. C'est pour cela que si on souhaite commander plusieurs fois le même élément, il faudra bien ajouter les guillemets pour les commandes (exemple : si les commandes sont composées de [''4 pintes'' 3 chocolats chauds], quatre éléments seront ajoutés à la liste de commande, ce qui pose problème).

2) Fonctionnement

Le programme est construit de la manière suivante :

- Une classe **Serveur**, composée de deux fonctions, l'une permettant de prendre les commandes, la seconde permettant de servir.
- Une classe **Barman**, elle aussi composée de deux fonctions, la première qui permet la préparation des commandes, et la deuxième qui sert à l'encaissement.
- Une classe **Accessoire**, de laquelle dépendent les classes **Pic**, **Bar**, **Encaiss**. Cette classe permet de créer une liste, de la remplir et de la vider. Elle est donc utilisée dans les classes **Serveur** et **Barman**, en effet ses fonctions sont utiles à chacune des étapes du programme, pour passer d'une étape du serveur à une étape du barman par exemple.
- Des fonctions subsidiaires qui permettent de calculer le temps écoulé depuis le début du programme **format_time** et **temps**.
- Deux fois 4 variables globales, les **gl_work** qui permettent de savoir où en sont les tâches principales du serveur et du barman.

A toutes ces fonctions, sont ajoutés de l'asynchronisme et du multithreading, à tous niveaux dans les fonctions. Les await sont placés aux endroits où le serveur et le barman doivent attendre pour réaliser leurs tâches. Pour lancer les threads, on utilise les commandes :

```
serveur.start()
barman.start()
```

```
serveur.join()
barman.join()
```

Pour le bon fonctionnement de mon programme, j'ai dû ajouter des variables globales **gl_work**, qui permettent de mettre à jour les états du serveur et du barman. La fonction **prendre_commande** fonctionne tant qu'il reste des éléments dans la liste des commandes, et lorsqu'elle se termine la variable **gl_work['cmd']** est mise à 0. La fonction **preparer** fonctionne elle tant que le serveur prend des commandes ou qu'il reste des éléments dans la liste des commandes prises, lorsqu'il n'y a plus de préparations, sa variable globale devient donc nulle. Ensuite, **servir** fonctionne tant qu'il reste des éléments dans la liste du bar ou qu'il y a des préparations en cours, d'où l'intérêt de mettre les variables globales sur 0 au préalable. Enfin, **encaisser** fonctionne quand il reste des éléments dans la liste de ce qui doit être encaissé, ou lorsque n'importe qu'elle autre fonction tourne encore.

3) Echecs / Réussites :

Pour ce qui concerne les échecs et les réussites, le plus gros problème est sûrement le fonctionnement de ma ligne de commande, car je n'ai pas trouvé de solution simple pour prendre en compte les guillemets et seulement les guillemets, c'est la limite du module **argparse** que j'ai utilisé.

J'aurai aussi aimé pouvoir prendre en compte le fait que si je commande '3 mojito', le temps soit donc triplé, je n'ai pas eu le temps d'ajouter cette fonction à mon programme.

Il aurait été aussi possible de personnaliser un peu plus les classes Pic, Bar et Encaiss, pour les différencier, mais j'ai utilisé au maximum le principe de classe accessoire.

Pour ce qui concerne, les réussites, mon barman n'encaisse pas pendant ses préparations, ce qui est une volonté de ma part, tout comme le serveur ne sert que lorsqu'il a fini de prendre les commandes.

Le programme fonctionne bien en multithread avec l'asynchronisme, bien qu'il ne soit pas forcément intéressant d'utiliser les deux dans un seul et même programme, étant donné que les deux approches ont un but similaire. L'asynchronisme n'a qu'un seul thread et n'a pas de verrous, tandis que le multithreading permet d'exécuter deux threads en simultané.