Faculté de physique et ingénierie
Université de Strasbourg

Licence SPI ingénierie L2 S3
TP n°3 Architecture des ordinateurs

Peut-être n'est-il pas possible de faire du multitâche sur MacOS (ça m'étonne mais effectivement j'ai trouvé des questions bizarres sur internet). N'ayant pas de Mac, je n'ai pas la moindre idée de son fonctionnement. Je propose alors une question n°3 de remplacement.
Si quelqu'un a envie de l'essayer sur PC en plus de la Q3 normale, il peut, bien sûr.

3) multitâche

en python, quand vous lancez deux fonctions à la suite, il fait d'abord la première, et quand elle est finie, la seconde. Mais vous pouvez aussi lancer deux fonctions en même temps, et elles tournent en même temps (en fait, en alternance, voir cours). On appelle cela des « threads ». La fonction Afficheur ci-dessous (qui a certes une écriture un peu spéciale) affiche 20 fois la lettre donnée en argument, séparée d'un temps aléatoire (entre 0.2 et 0.8s). Mais on va l'appeler (start) deux fois, EN MEME TEMPS ! Puis on va attendre que les deux soient finies (join) pour que le programme continue (ici, affiche puis s'arrête).
Testez-le. Normalement, les A et les B s'affichent en se mélangeant, dans un ordre variable à chaque essai. Essayez au moins 3 fois (mais il y a toujours 20 A et 20 B). Si ça marche c'est que MacOS est multitâche.

Puis, uniquement pour ceux qui ne peuvent pas faire la Q3 normale, essayez de créer 2 fonctions différentes, appelées « lecteur » et « ecrivain » qui dans un premier temps ne font qu'un print différent. Puis si ça marche, essayez d'y mettre le lecteur et écrivain de la Q3 « normale ».

```
import random
import sys
from threading import Thread
import time

class Afficheur(Thread):
    """Thread chargé simplement d'afficher une lettre dans la console."""
    def __init__(self, lettre):
        Thread.__init__(self)
        self.lettre = lettre

    def run(self):
        """Code à exécuter pendant l'exécution du thread."""
        i = 0
        while i < 20:
            sys.stdout.write(self.lettre)
            sys.stdout.flush()
            attente = 0.2
            attente += random.randint(1, 60) / 100
            time.sleep(attente)
            i += 1

# Création des threads
thread_1 = Afficheur("A")
thread_2 = Afficheur("B")

# Lancement des threads
thread_1.start()
thread_2.start()

# Attend que les threads se terminent
thread_1.join()
thread_2.join()
print("\nfini")
```