PROJET

Etape 1

Etude des bibliothèques Python permettant de créer des flots d’exécution pouvant s’exécuter en parallèle

Sur la forme

- Le fichier doit contenir les noms des membres du groupe
- Le document rendu doit être un notebook exécutable
  - Pas de rar, pas de multiples fichiers
  - Pas de pdf
- Le document doit alterner
  - Présentation d'un concept (texte)
  - si necessaire-si possible illustration de ce contexte (code)
- Rendez votre texte intéressant à lire, vous pouvez aussi comparer d'autres langages (Java par exemple)
  - Il faut toujours conclure : qu'est-ce qui vous semble le plus adapté ? Pourquoi ?

Sur la forme

- On va a l'essentiel

  - Il vaut mieux un texte court qui met en évidence ce qui est essentiel qu'un texte long qui n'apprend rien ou noie le poisson

  - Inutile de raconter ce qui est connu mais on se concentre sur les différences entre ce qui est 'enseigné' et ce qui est présent dans la bibliothèque Python.

- Exemples:
  - La classe Multiprocessing Python permet de mettre en oeuvre différents processus qui ont chacun leur propre espace virtuel et d'attendre leur terminaison. L'exemple montre comment on crée 3 processus et comment on attend la terminaison du processus 1 et 3 avant que le main continue son exécution. L'exemple met aussi en évidence que les espaces virtuels sont disjoints.

  - La classe Thread Python permet de mettre en oeuvre différentes thread qui ont le même espace virtuel. L'exemple suivant met en oeuvre le même scenario que précédemment mais met en évidence que les thread partage le même espace virtuel

Sur la forme

- Pas de banalité ou d’arguments non étayés
  - Les threads prennent peu de place dans la mémoire, alors que les processus sont plus coûteux
  - Les threads permettent de faire des tâches légères et les processus sont mieux adaptés à des tâches plus lourdes.
  - En conclusion ces deux bibliothèques ont des avantages et des inconvénients
  - Il faudra choisir si l'on utilise des processus ou des threads pour que le programme soit le optimisé
- Pas de choses fausses
  - Un thread est un processus qui s’exécute dans un programme
- Les exemples doivent être juste – un exemple
  - Ici, la variable `a` est partagée. On s'attendait à obtenir comme résultat 99+99=198
  - J’exécute le code proposé.
    - J’obtiens 9900
    - Je modifie le code pour exécuter d’abord Thread1, puis Thread2. J’obtiens le même résultat

Attention à la representation des données

- Pour partager un objet, je le passe en paramètre
- Pour partager un entier, je dois le passer par référence (ou adresse)
  - N'existe par dans tous les langages

- Attention à l'usage de mot clé global en Python
  - Toute variable déclarée à l'extérieur d'une procédure est connue dans une procédure mais ne peut pas être modifiée
  - L'usage du mot clé global, permet de modifier à l'intérieur d'une procédure une variable déclarée à l'extérieur

Sur le notebook

- Inutile de mettre if __name__ == '__main__’:
  - Surtout s’il y en a un dans plusieurs cellules

- Le résultat de la dernière ligne est imprimé à l’écran, même résultat que
  - display(dernière ligne)

- A priori, vous ne devez pas utiliser « sleep » qui perturbe l’ordonnancement des flots d’exécution

Les points qui doivent être mis en evidence dans les différentes bibliothèques

- Est-ce que le flot d'exécution du père et du fils sont associés au même processus Unix
  - Parler de l'entrelacement, et si possible, mettez le en évidence (sans sleep)
  - Comment se passe l'affectation des flots d'exécution aux cœurs
  - Si multi-cœurs, est-il possible d'y avoir une exécution en parallèle ou est-celle systématiquement entrelacée comme s'il y avait un seul cœur
- Est-ce que par défaut le père et le fils, ou les fils peuvent partager la même variable ?
  - La variable existe dans le code du père
  - On crée le fils
    - Si on ne passe pas la variable en paramètre du fils :
      - est-ce que le fils la connait ?
      - si le fils la modifie, est-ce que le père connait la modification
      - Si un fils la modifie, est-ce que les autres fils connaissent les modifications
    - Idem en utilisant le mot clé global
    - Idem en passant la variable du père en paramètre au fils lors de sa création
    - A tester pour un type simple (entier) et une structure plus complexe (list) et un objet
- Est-ce que le fils démarre dès sa création ou faut il le démarrer explicitement
- Comment le père attend la terminaison d'un fils, de tous les fils