Calcul asynchrone en Python

Calcul asynchrone

- Principe : lancer plusieurs calculs en parallèle sans attendre la fin de l'un pour débuter le suivant
- Map-reduce est un cas particulier
- Il existe des cas plus complexes
  Exemple :

(3 + 7) * (2 + 8) * ((2 * 3) - 1)

Calcul asynchrone

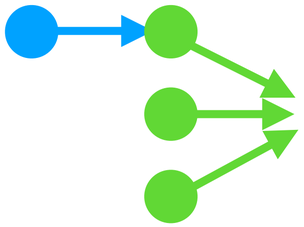
- Principe : lancer plusieurs calculs en parallèle sans attendre la fin de l'un pour débuter le suivant
- Map-reduce est un cas particulier
- Il existe des cas plus complexes
  Exemple :

(3 + 7) * (2 + 8) * ((2 * 3) - 1)

Calcul asynchrone

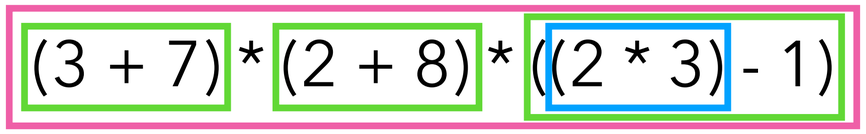
- Principe : lancer plusieurs calculs en parallèle sans attendre la fin de l'un pour débuter le suivant
- Map-reduce est un cas particulier
- Il existe des cas plus complexes
  Exemple :

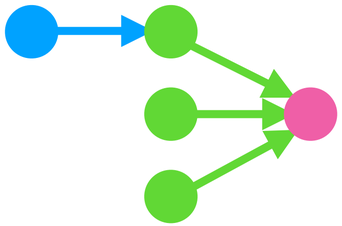
(3 + 7) * (2 + 8) * ((2 * 3) - 1)



Calcul asynchrone

- Principe : lancer plusieurs calculs en parallèle sans attendre la fin de l'un pour débuter le suivant
- Map-reduce est un cas particulier
- Il existe des cas plus complexes
  Exemple :





Calcul asynchrone en Python

Utilisation de la librairie `concurrent.futures`

- Définition d'un pool (de *threads* ou de *process*)

```
from concurrent.futures import ThreadPoolExecutor

with ThreadPoolExecutor(max_workers=1) as e:
    # Code utilisant le pool `e` ici
```

- 1 pool = 1 ensemble
  - Taille fixée par `max_workers`
- Encapsulé dans un `with`
  - Garantit que le pool est détruit à la fin du `with`

Calcul asynchrone en Python

Utilisation de la librairie `concurrent.futures`

- On peut soumettre des travaux à un pool

```
with ThreadPoolExecutor(max_workers=1) as e:
    fut = e.submit(f, x, y, z)
    print("Coucou")
```

- Ici, le travail demandé est d'appliquer la fonction `f` en lui passant les arguments `x`, `y` et `z` (équivalent à `f(x, y, z)`)
  - La valeur de retour d'un `submit` est appelé un `future`
- Calcul effectué dès qu'un élément (*thread* ici) du pool est disponible
- La ligne qui suit peut être exécutée sans que le travail soit fini

Calcul asynchrone en Python

Utilisation de la librairie `concurrent.futures`

- Pour récupérer le résultat d'un travail :

```
with ThreadPoolExecutor(max_workers=1) as e:
    fut = e.submit(f, x, y, z)
    print("Coucou")
    ret = fut.result()
    print("Re-Coucou")
```

- La ligne `ret = fut.result()` est bloquante
- `ret` récupère la valeur de retour de l'appel `f(x, y, z)`

Calcul asynchrone en Python

Utilisation de la librairie `concurrent.futures`

- Résumé
  - `submit` lance un appel de fonction en tâche de fond
  - `submit` + boucle `for` = `map`
  - `submit` retourne un `future`, pour lire son résultat on appelle `.result()`
  - Intérêt : plus général que `map`

Calcul asynchrone en Python

Utilisation de la librairie `concurrent.futures`

- Exercice : implémentez l'exemple de début du cours à l'aide de `concurrent.futures`

(3 + 7) * (2 + 8) * ((2 * 3) - 1)