Semaphore

Eina per limitar el nombre de processos que poden accedir a un recurs compartit alhora.

Quan utilitzar-lo:

- Quan es vol limitar accés a un recurs compartit que té un màxim d'usuaris o processos simultanis

Important:

- mp.Semaphore(2) → Numero de processos que permet alhora.
- A la funció es fa un with semaphore: on s'ha de passar per args *

Event

És una eina per comunicar estat entre processos. Actua com una mena de bandera que pot ser activada o desactivada.

Important:

- Dos estats: set (activat) o clear (desactivat)
- Processos poden utilitzar events per sincronitzar-se:
  - set(): Activar event
  - clear(): Desactivar event
  - wait(): Bloquejar procés fins que l'event s'activi
  - mp.Event() i passar event per args
  - Després de ser el start es fa el event.set() i després el join dels processos.

Barrier

Eina de sincronització que espera que un nombre determinat de processos arribin a un punt abans que tots continuin.

Important:

- mp.Barrier(3) → Esperar 3 processos
- En la funció ser un barrier.wait() per esperar que tots processos arribin

Aniol Alorero Satlle

Queue

Per compartir dades entre processos de manera segura, quan necessita comunicació ordenada i sincronitzada

Quan utilitzar-la:
- Quan s'han de passar dades de manera sequencial i segura.
- Per exemple, un procés produeix dades i un altre les consumeix

Important:
- Crear cua: mp.Queue()
- Afegir element: queue.put(i)
- Agafar element: queue.get()
- Mirar si és buida: queue.empty()

Locks

Són mecanismes per assegurar que només un procés accedeix a una secció del codi alhora. Evita condicions de carrera.

Quan utilitzar-los:
- Quan multiples processos poden accedir o modificar dades compartides alhora
- Per evitar inconsistencia de dades durant acces concurrent
- Per sincronitzar processos perque no interfereixen entre ells.

Important:
- Crear lock: mp.Lock()
- Pasar el lock per args
- Fe bloquejar: with lock:

Ser for ràpid de processos.
processes = [mp.Process(target, args=(lock, i)) for i in range(5)]
recorre 2 per ser un start, i un altre per join

Pool

Serveix per distribuir una carrega de treball entre multiples processos de forma automatica

Quan utilitzar-lo:
- Gran nombre de tasques independents que poden ser paral·lelitzades.
- Quan es vol que Py gestioni automàticament la creació i finalització dels processos.
- Simplificar codi i evitar gestió manual de processos

Important:
- No cal ser mp.Process(target, args) ! !
- s'ha de ser with mp.Pool(4) as pool: → Numero de processos paral·lels

```
        result = pool.map(funció, args)
    print(results)
```