

Python chatbots con asyncio

Marco Bianchi @akita8



Concurrency in Python today

- **Multiprocess**
- **Threads**
- **Async**



Multiprocess

Pro

- Rende possibile sfruttare architetture multi-core
- Must per applicazioni cpu intensive

Contro

- Necessità di molta infrastruttura per rendere possibile la comunicazione tra i processi



Threads

Pro

- **It just works**
- **Non c'è necessità di una speciale sintassi**
- **Non ha problemi con chiamate bloccanti**

Contro

- **Race conditions**
- **Peso della creazione di nuove threads e del context switch**



Async

Pro

- **Veloce**
- **Vive in una sola thread quindi niente switch o race condition**

Contro

- **Richiede molto più boilerplate come setup iniziale**
- **Lo sviluppatore deve decidere dove e quando il contesto di esecuzione ridà il controllo alla funzione chiamante**



Async != asyncio

- La nuova sintassi introdotta in 3.5 non è framework dependant
- **AsyncIO** è un'implementazione del ciclo eventi, l'unica inclusa nella stdlib
- Ci sono molteplici implementazioni, Twisted, Tornado, Curio....
- La maggior parte dei framework async hanno già introdotto la possibilità di usare la nuova sintassi.



Async/await

- L'utilizzo principale della keyword `async` è per definire funzioni o metodi `async`.
- Le funzione e i metodi `async` se chiamati non eseguono il codice ma ritornano un oggetto chiamato `coroutine`.
- Solo nelle funzioni e nei metodi definiti con `async` si può usare la keyword `await`.
- Si possono “awaitare” solo oggetti `coroutine`.



Esecuzione asincrona in asyncio

- **Await** denota le parti del codice dove ci sono chiamate che potrebbero possibilmente bloccare o dove la coroutine corrente cede il contesto di esecuzione ad un'altra coroutine
- Con solo **await** non si può scrivere un'applicazione veramente asincrona
- **AsyncIO** espone in particolare due funzioni per eseguire codice in maniera asincrona: `asyncio.get_event_loop.create_task` e `asyncio.gather`



Asyncio API e Docs (a.k.a i punti dolenti)

- **La svolta mainstream riguardo tutte le cose async della community di sviluppatori python è recente.**
- **L'api stà ancora maturando e le docs, anche se eccellenti, sono un po' datate e non sono particolarmente user friendly.**



Usability

- Posso usare tutte le librerie su PyPI?
- Si ma... non devono fare chiamate bloccanti.
- Se la possibilità esiste `asyncio` espone una funzione per non bloccare l'event loop, `asyncio.get_event_loop().run_in_executor` che può eseguire la chiamata in una thread o in un process

