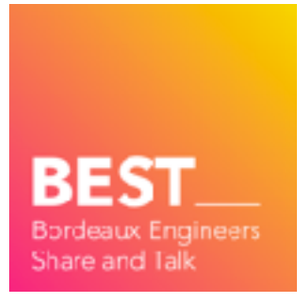
BEST
Bordeaux Engineers
Share and Talk

Awesome async / await avec Python 3.5

Bordeaux Engineers Share & Talk #3
27 juin 2018



Thibaut Séguy

CTO @ Actiplay

Data marketing / Adtech

https://github.com/thibauts

https://twitter.com/thibautseguy

Python et concurrence, un historique tourmenté

- Threading / polling (vanilla)
- Callbacks, deferred, futures, promises (twisted, tornado)
- Green threads (gevent, greenlet, coroutines)
- yield from (Python 3.3)
- asyncio (Python 3.4)
- async / await (Python 3.5)

Convergence des mécanismes de concurrence

- Javascript : callbacks, promises … async / await !

- C#, Javascript, Scala, Dart, Python l’ont adopté

- Une syntaxe uniforme, qui masque les complexités d’implémentation

Commençons par la documentation d’asyncio :

Bounded semaphores

Stream writers

Futures

Unix signals

Coroutines

Pipes

Locks

Protocols

Delayed calls

Tasks

Executors

Queues

Stream readers

Coroutine chaining

Event loop

Events

Cancellation

Transports

Mais si, c’est beaucoup plus simple ! Reviens !

Live coding time !

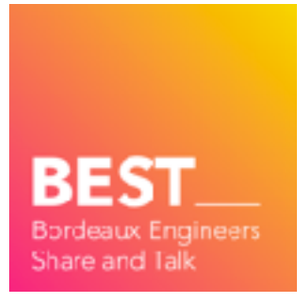
Woohoo !

Le nécessaire pour commencer

- async def, async with, async for
- await
- asyncio.get_event_loop()
- loop.run_forever()

Utilisation en production

- Warning: modules spécifiques
- Ecosystème en évolution constante (https://github.com/aio-libs)
- OK si le périmètre est bien défini (notamment, choix des datastores)



Merci !

https://github.com/thibauts/talk-asyncio-best