Python Async I/O server Live Coding

(use async/await keyword, no use asyncio lib)

Socker server

sample code

async/await 적용

sample code

async/await

- PEP-380 : 중첩된 코루틴을 다룰 수 있다.
  - `yield from` 추가 : 제너레이터에서 다른 제너레이터를 부르는 문법.
  - 최상위 caller 는 얼마다 코루틴(or 제너레이터)이 중첩되어있는지 상관없이 **가장 안쪽** 코루틴이 `yield` 한 값을 받는 것이 항상 보장된다.

ref) Making asyncio on our own

async/await

- PEP-492: `async` 와 `await` keyword 추가
  - `yield from` 은 제너레이터 관점의 문법 (`yield` keyword 존재에 따라 def 의 의미가 바뀜. func vs generator)
  - `yield from` 을 그래도 `await` 로 바꾸어도 통함. 차이점 ->
    - `async def` 안에서만 `await` 사용하도록 강제: 실수 가능성 감소
    - `await` 포함되지 않은 `async` 함수 선언 가능: 인터페이스 일관성 향상
  - `async for` , `async with` 와 같은 syntax sugaring 추가
  - 향후 async generator 구현을 위해 yield 키워드의 구분 필요성

ref) Making asyncio on our own

Reference

PyCon APAC 2018 - Jonas Obrist's "Artisanal Async Adventures"

Making asyncio on our own!

Differences between Futures in Python3 and Promises in ES6

파이썬 소켓 연결 사용법

select 와 epoll