Python Study 2

Date	
Property	
Weeks	4주차

프로세스가 가지고 있는 메모리 영역

- 뭐... 뭐요?
- code, data, heap, stack

code

- 하드웨어가 알아들을 수 있는 코드

data

- 초기화가 안 된 애들을 저장하는 bss라는 곳도 있는데, 파이썬은 초기화가 항상 있어야 하니까 상관 없음

heap

- 런타임에 결정되는 데이터
- 실행하면서 append, key추가 등등
- 프로그램에 뭘 입력하냐에 따라 어떻게 될지 모르는 영역
- stack overflow의 반대가
- heap overflow

stack

- call stack
- 파이썬 같은 경우에는 default로 1000을 가지고 있음
- 1000 이상이 필요하다면 언어 환경변수를 건드려야 함

- 재귀알고리즘을 실제구현에서 무턱대고 쓰면 안 되는 이유이기도 함
  - 손코딩 시키는 코테에서는 물어볼 수도?
- 그러면 1000번 이상 호출이 나면 무슨일이 벌어질까요?
  - stack overflow

Threading

- 자 그러면 heap에 저장이 되는 데이터들 중에 보면 비교적 간단

Concurrency programming in Python

- 그러면 여러 콜스택이 실행되고, 하나의 데이터영역에서 쓰기작업을 한다면 뭐든 빨리 할 수 있지 않을까?
- 파이썬에서는 그렇기도 하고 아니기도 합니다.
  - CPU-bound, IO-bound

  → 크롤링은 IO-bound . 이건 파이썬에서도 병렬화 가능.

  → CPU-bound 는 파이썬에서 막힘 (Thread 관리가 어려움)
  - 저한테 도커 배우신 분은 기억하실텐데, CPU와 메모리가 있고 그게 아닌 장비들이 있다 (e.g. 스토리지, 소켓, 입출력장치)
- GIL(Global Interpreter lock)
  - 결론적으로 CPython에서는 한번에 한 쓰레드만 실행된다
- SIMD(Single Instruction Multiple Data) heap

→ for문과 다르게 한 방에 베이킹 반죽 탕 (pandas list), Pandas가 왜 빠른가?

mutable vs immutable

- 자원소비의 효율성을 생각하면
- 근데 사용의 편의성과 안정성을 생각하면 추상화된 데이터들이 immutable 한 게 더 좋을 수도 있어요
- 사고 방지

Garbage Collector

- 쓰고, 다음 것을 만들어 냈으면 나머지는 *쓰레기가 아닐 것인가...*

Functional Programming

- lambda, function as parameter, function as 1st class citizen