2025

프로그래밍원리와실습 16주차
쓰레드

JEONBUK NATIONAL UNIVERSITY

쓰레드

- 파이썬 프로그램은 기본적으로 하나의 쓰레드(Single Thread)에서 실행된다. 즉, 하나의 메인 쓰레드가 파이썬 코드를 순차적으로 실행한다. 코드를 병렬로 실행하기 위해서는 별도의 쓰레드(Subthread)를 생성해야 하는데, 파이썬에서 쓰레드를 생성하기 위해서는 threading 모듈 (High 레벨) 혹은 thread 모듈 (Low 레벨)을 사용할 수 있다. 일반적으로 쓰레드 처리를 위해서는 thread 모듈 위에서 구현된 threading 모듈을 사용하고 있으며, thread 모듈은 (deprecate 되어) 거의 사용하고 있지 않다.

- 파이썬(오리지날 파이썬 구현인 CPython)은 전역 인터프리터 락킹(Global Interpreter Lock) 때문에 특정 시점에 하나의 파이썬 코드만을 실행하게 되는데, 이 때문에 파이썬은 실제 다중 CPU 환경에서 동시에 여러 파이썬 코드를 병렬로 실행할 수 없으며 인터리빙(Interleaving) 방식으로 코드를 분할하여 실행한다. 다중 CPU 에서 병렬 실행을 위해서는 다중 프로세스를 이용하는 multiprocessing 모듈을 사용한다.

출처: http://pythonstudy.xyz/python/article/24-%EC%93%B0%EB%A0%88%EB%93%9C-Thread

쓰레드

- 파이썬에서 쓰레드를 실행하기 위해서는, threading 모듈의 threading.Thread() 함수를 호출하여 Thread 객체를 얻은 후 Thread 객체의 start() 메서드를 호출하면 된다. 서브쓰레드는 함수 혹은 메서드를 실행하는데, 일반적인 구현방식으로 (1) 쓰레드가 실행할 함수 혹은 메서드를 작성하거나 또는 (2) threading.Thread 로부터 파생된 파생클래스를 작성하여 사용하는 방식 등이 있다.

- 먼저 첫번째 함수 및 메서드 실행 방식은 쓰레드가 실행할 함수 (혹은 메서드)를 작성하고 그 함수명을 hreading.Thread() 함수의 target 아큐먼트에 지정하면 된다. 예를 들어, 아래 예제에서 sum 이라는 함수를 쓰레드가 실행하도록 threading.Thread() 함수의 파라미터로 target=sum 을 지정하였다. 여기서 한가지 주의할 점은 target=sum() 처럼 지정하면, 이는 sum() 함수를 실행하여 리턴한 결과를 target에 지정하는 것이므로 잘못된 결과를 초래할 수 있다. 만약 쓰레드가 실행하는 함수(혹은 메서드)에 입력 파라미터를 전달해야 한다면, args (혹은 키워드 아규먼트인 경우 kwargs) 에 필요한 파라미터를 지정하면 된다. args는 튜플로 파라미터를 전달하고, kwargs는 dict로 전달한다. 아래 예제에서 sum() 함수는 두 개의 파라미터를 받아들이기 때문에 "args=(1, 100000)" 와 같이 입력파라미터를 지정하였다.

출처: http://pythonstudy.xyz/python/article/24-%EC%93%B0%EB%A0%88%EB%93%9C-Thread

쓰레드

```
import threading

def sum(low, high):
    total = 0
    for i in range(low, high):
        total += i
    print("Subthread", total)

t = threading.Thread(target=sum, args=(1, 1000000))
t.start()

print("Main Thread")
```

출처: http://pythonstudy.xyz/python/article/24-%EC%93%B0%EB%A0%88%EB%93%9C-Thread

파이썬 멀티 쓰레드(thread)와 멀티 프로세스(process)

GIL(Global Interpreter Lock)

언어에서 자원을 보호하기 위해 락(Lock) 정책을 사용하고 그 방법 또한 다양하다. 파이썬에서는 하나의 프로세스 안에 모든 자원의 락(Lock)을 글로벌(Global)하게 관리함으로써 한번에 하나의 쓰레드만 자원을 컨트롤하여 동작하도록 한다.

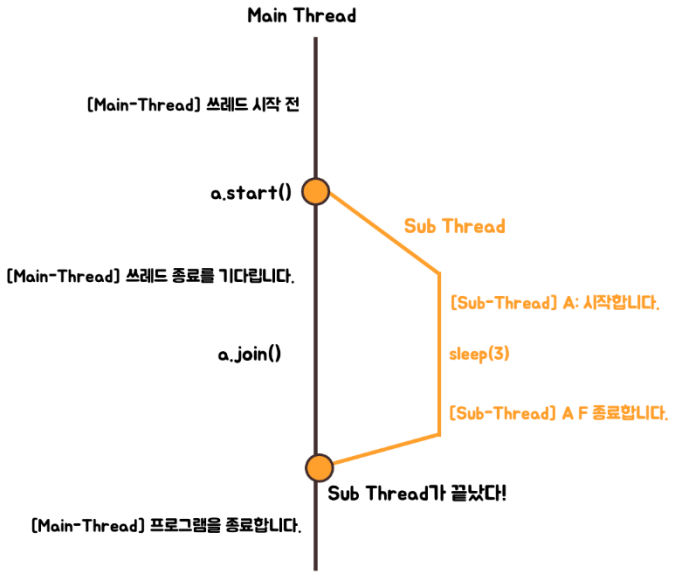
GIL 덕분에 자원 관리(예를 들어 가비지 컬렉팅)를 더 쉽게 구현할 수 있었지만, 지금처럼 멀티 코어가 당연한 시대에서는 조금 아쉬운 것이 사실이다. 그렇다고 파이썬의 쓰레드가 쓸모 없는 것은 아니다. GIL이 적용되는 것은 CPU 동작에서이고 쓰레드가 CPU 동작을 마치고 I/O 작업을 실행하는 동안에는 다른 쓰레드가 CPU 동작을 동시에 실행할 수 있다. 따라서 CPU 동작이 많지 않고 I/O동작이 더 많은 프로그램에서는 멀티 쓰레드만으로 성능적으로 큰 효과를 얻을 수 있다.

계산을 병렬로 처리하는데 도움을 주는 것이 바로 **multiprocessing** 모듈이다. multiprocessing 모듈은 쓰레드 대신 프로세스를 만들어 병렬로 동작한다.

출처: https://monkey3199.github.io/develop/python/2018/12/04/python-pararrel.html

멀티쓰레드 흐름도

```
import logging
import threading
import time

# 스레드에서 실행할 함수
def thread_func(name):
    logging.info("[Sub-Thread] %s: 시작합니다.", name)
    time.sleep(3) # 3초간 sleep합니다.
    logging.info("[Sub-Thread] %s: 종료합니다.", name)

# 메인 영역
if __name__ == "__main__": # main thread 흐름을 타는 시작점
    format = "%(asctime)s: %(message)s" # Logging format 설정
    logging.basicConfig(format=format, level=logging.INFO, datefmt="%
    logging.info("[Main-Thread] 쓰레드 시작 전")

    x = threading.Thread(target=thread_func, args=('A',))

    logging.info("[Main-Thread] 쓰레드 시작 전")

    a.start() # 서브 스레드 시작

    logging.info("[Main-Thread] 쓰레드 종료를 기다립니다.")

    a.join() # join() 추가!!!

    logging.info("[Main-Thread] 프로그램을 종료합니다.")
```



출처: https://covenant.tistory.com/231

관련 학습자료

- https://www.pythontutorial.net/advanced-python/python-threading/
- https://realpython.com/intro-to-python-threading/
- https://blog.floydhub.com/multiprocessing-vs-threading-in-python-what-every-data-scientist-needs-to-know/