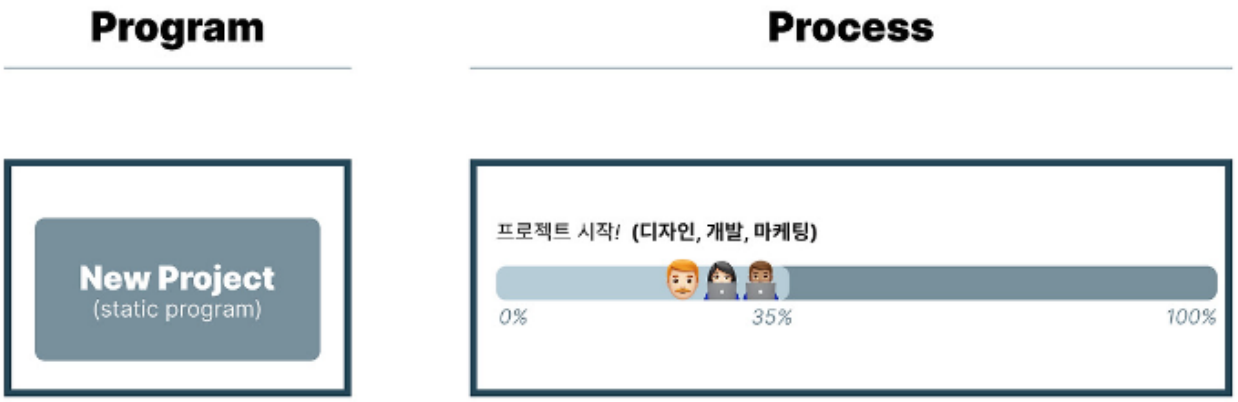
single multi-core CPU & num_thread = 1

Program

컴퓨터에서 실행할 수 있는 파일(.py, .exe).

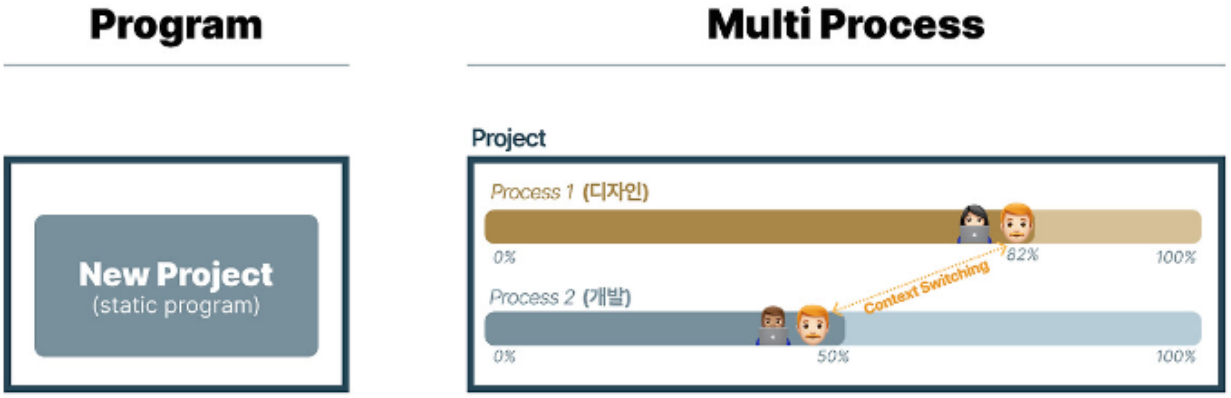
Process

실행되는데 필요한 모든 자원과 함께 메모리에 load된 실행 Program.
Program이 작동되는 과정.
ex) train process, test process



Multi-process

Program과 Process 가 $1 : n \ (n \geq 2)$의 관계를 가질 때.
프로그램이 여러 개 작동된다는 의미이다.



Core

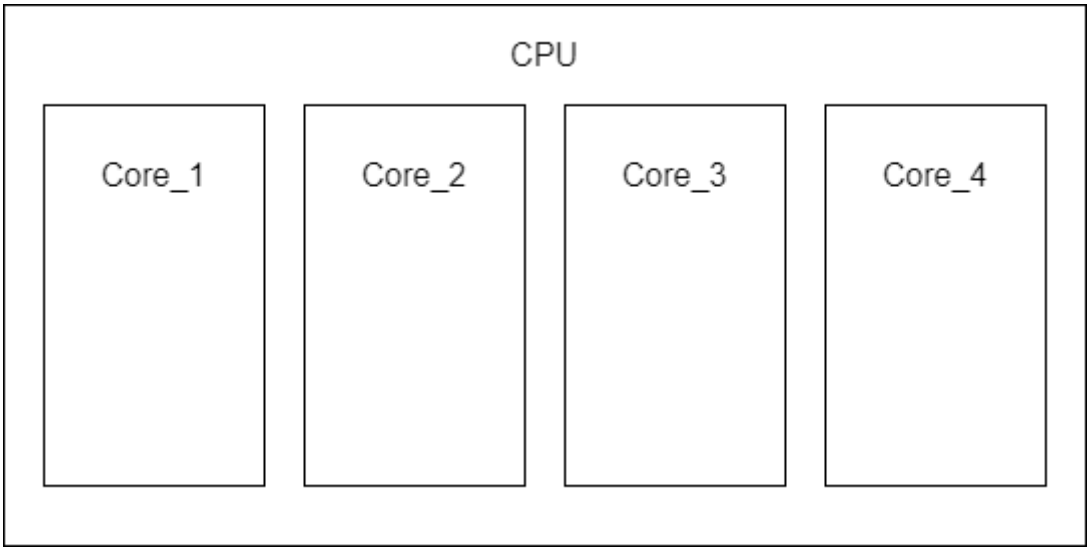
- CPU가 가지고 있는 **핵심 부품**으로 기본 연산과 계산 작업 역할을 한다.
- 각 Core는 **독립적으로** 작업을 수행한다.

ex) CPU : 회사 사장 / Core : 직원

Multi-core

ex) 2코어 (듀얼 코어), 4코어 (쿼드 코어), 8코어 (옥타 코어)
CPU 는 1개 이상의 Core 를 가진다.

4코어 (쿼드 코어)

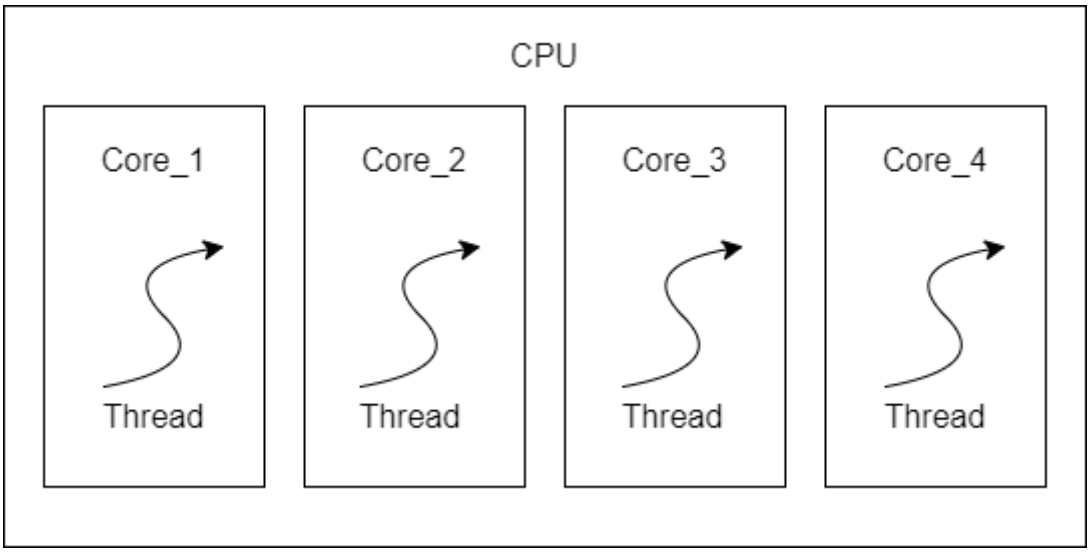


Thread

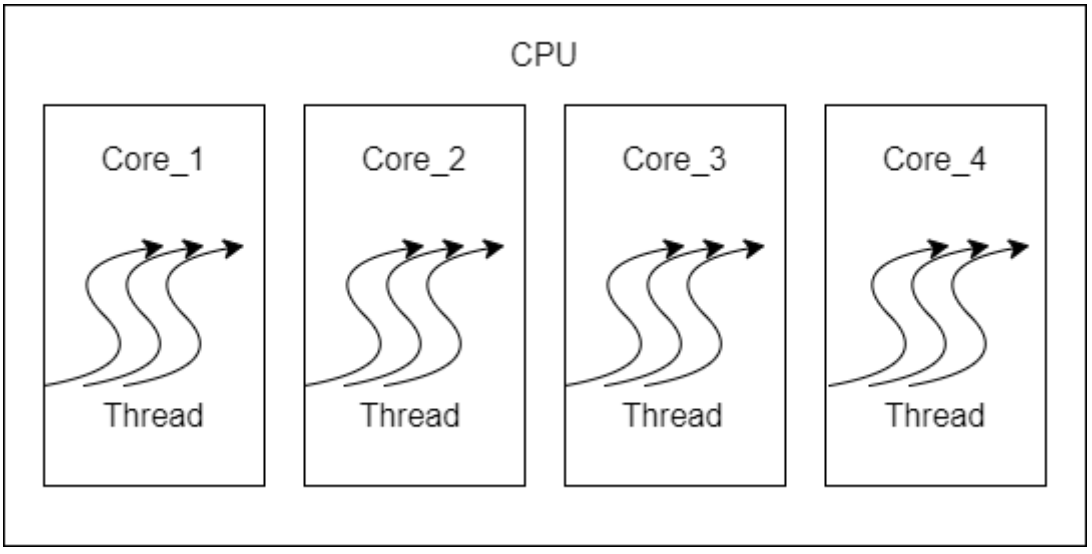
- 하나의 Process 에 대해 추상화된 실행 단위
- Core가 할 수 있는 최소 단위의 일 혹은 도구

Core는 1개 이상의 Thread 를 가진다.

Num_Threads = 1

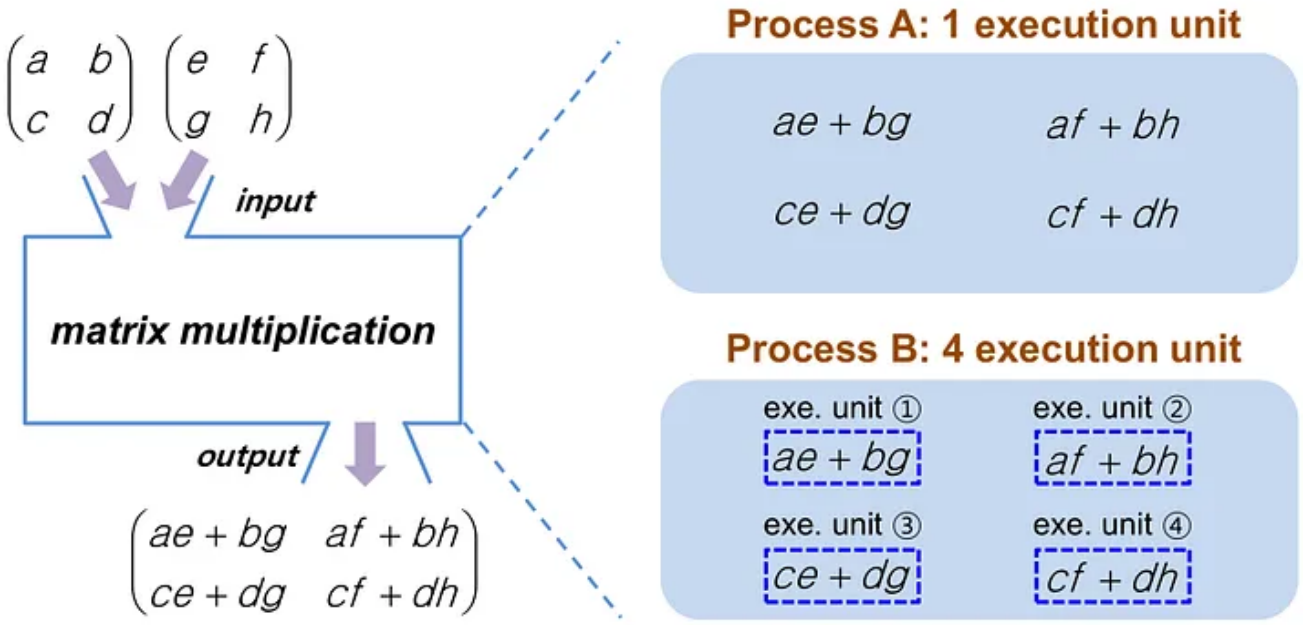


Num_Threads = 3



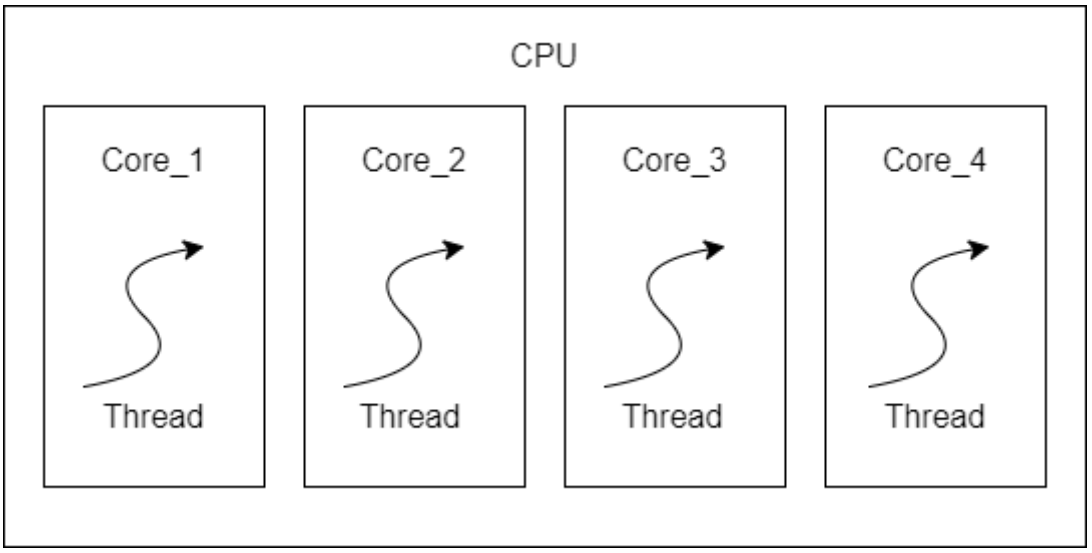
ex) 1대의 모니터당 1개의 업무 처리를 할 수 있다고 가정하자.
따라서, 여러 대의 모니터를 보유함으로서 다중적 업무 처리가 가능해진다.

- CPU(Processor) : 논리적 사고를 담당하는 회사 사장
- Core : CPU의 논리적 사고를 바탕으로 연산 업무를 처리하는 직원
- Thread : Core가 할 수 있는 최소한의 작업 단위, 업무를 처리하는 모니터



Process A 는 *한 개*의 thread 상에서 연산을 수행한다.
Process B 는 *네 개*의 thread 상에서 연산을 수행한다.
multi-thread 는 여러 개의 thread 가 역할을 나누어서 process 를 처리할 수 있다.

→ **single multi-core CPU & num_thread = 1** :



A3C

A3C에서 multiprocessing 을 활용하여 여러 개의 workers가 동시에 작업을 수행할 수 있도록 한다.
이번 코드에서는 4개의 train process와 1개의 test process를 생성한다.

1. **Train process**

- *worker*를 학습시키는 역할을 한다.
- 각 프로세스는 독립된 환경에서 학습을 수행하고, 중앙의 공유 모델을 주기적으로 갱신한다.
- 4개의 Train Process를 생성한다 = Worker의 개수가 4개다

3. **Test process**

- 주기적으로 *현재 모델의 성능을 평가*한다.
- worker을 학습시키지 않고, 모델의 성능을 평가하기 위해 환경을 실행한다.

Conclusion

4 train processes & *1 test process* & *a single multi-core CPU* & *num_thread = 1*

Discussion

1. *Train Process의 개수 = Worker의 개수* (o)
2. *Process의 개수 = Core의 개수* (x)

- *Process의 개수 = Core의 개수

→ 하나의 Core에 하나의 Process가 할당된다.

- *Process의 개수* $\leq$ *Core의 개수*

→ 하나의 Core에 하나의 Process가 할당된다. 운영 체제가 각 Process를 스케줄링하고 실행하는 방식에 따라 달라진다.

- *Process의 개수* $\geq$ *Core의 개수*

→ 하나의 Core에 동시에 여러 개의 Process가 직접적으로 할당되지 못한다. 스케줄링을 통해 각 Core에 할당될 Process를 순차적으로 선택하게 된다.