프로세스

- 프로세스 개념
  - 프로세스(Process) : 실행 중인 프로그램의 인스턴스. CPU, 메모리 등 시스템 자원을 소비하며 실행됨
  - PID(Process ID) : 프로세스를 식별하기 위한 고유번호
  - PPID(Parent Process ID) : 부모 프로세스의 PID
  - 상태(State) : 프로세스의 현재 상태로 다음과 같은 상태가 있음

상태 코드	설명
R	Running: 실행 중
S	Sleeping: 대기 상태(예: I/O 대기)
D	Disk sleep: 입출력 대기 상태(디스크 I/O 작업 대기)
Z	Zombie 좀비 상태. 종료됐지만 부모 프로세스가 처리 안됨
T	Stopped 중지 상태(Ctrl+z)
I	Idle: 커널 수준에서 유휴 상태
X	Dead: 프로세스가 존재하지 않음

프로세스 명령어

```
# 프로세스 조회
ps
ps aux
ps -ef
        # 주요 옵션
        ps a    # a : 다른 사용자 프로세스
        ps u    # u : 사용자와 cpu/메모리 표시
        ps x    # x : 터미널과 연결되지 않은 프로세스

# 프로세스 종료
kill -SIGNAL PID
        # 주요 시그널(SIGNAL)
        kill -SIGTERM    # 정상 종료
        kill -SIGKILL    # 강제 종료
        kill -SIGSTOP    # 실행 중지
        kill -SIGCONT    # 중지된 프로세스 재실행

# 백그라운드 실행
        # 터미널 제어권은 돌려주면서 프로세스는 계속 실행되고 있음
        # 백그라운드로 실행하는 방법은 &를 명령어 끝에 추가
$ python3 app.py &

# 백그라운드 작업 확인
$ jobs

# 백그라운드 작업 포어그라운드로 가져오기
$ fg %작업번호

# 프로세스 중지
프로세스 실행 중 ctrl + z

# 중지된 작업을 백그라운드에서 계속 실행
$ bg %작업번호
```

파이썬 플라스크 백그라운드로 실행

- 기존 명령어

```
(venv) $ python3 app.py
```

- 백그라운드 실행 명령어

```
(venv) $ nohup python3 app.py > mylog.log 2>&1 &
```

- **nohup** : 터미널이 종료되어도 프로세스는 계속 실행되도록 하는 명령어
- **python3 app.py > mylog.log** : 프로젝트 실행 후 나오는 출력을 리다이렉션하여 mylog.log 파일에 저장
- **2>** : 표준 에러를 리다이렉션 하라는 의미 (2주차 강의자료 참고)
- **&1** : 표준 출력이 향하는 곳(여기서는 mylog.log). 즉 프로젝트 실행시 나오는 에러도 mylog.log 파일에 저장
- **&** : 백그라운드에서 실행하기 위한 키워드