파이썬과 데이터베이스 연동

우리는 SQLAlchemy라는 라이브러리를 활용하여 파이썬에서 레먼데이터베이스를 조작할 것임.

가상 환경 세팅

현재 내 PC에 설치되어 있는 파이썬 패키지 목록을 확인하고 싶으면 `pip list` 명령을 터미널에서 실행

보다시피 이미 많은 패키지들이 현재 PC에 설치되어 있음.

작업 공간이 다소 지저분한 상태.

우리는 이 많은 패키지를 다 사용하지 않을 것임.

또한 미리 설치되어 있는 것이 버전 충돌 등의 문제를 일으킬 수 있음.

따라서 우리는 가상 환경을 만들어서 작업을 할 것임.

- 가상환경 생성: `python -m venv .venv`
- 가상환경 활성화
    1. 윈도우: `call .venv/Scripts/activate`
    2. MacOS: `source .venv/bin/activate`

가상환경에 진입한 상황에서 `pip list`를 해보면, 작업 공간이 깨끗함을 확인할 수 있다.

pip를 최신 버전으로 업그레이드하기 위해 `pip install --upgrade pip` 명령을 실행한다.

이제 아래 나열된 패키지들을 설치하면 된다.

필요 패키지 설치

- pip install sqlalchemy

---

python에서 SQL문을 활용해서 데이터 조작하기

STEP 1

crud.py라는 파일 하나 생성.

같은 폴더 안에 lahmansbaseballdb.sqlite가 있어야 함.

```
from sqlalchemy import create_engine, text

SQLALCHEMY_DATABASE_URL = "sqlite:///./lahmansbaseballdb.sqlite"

engine = create_engine(
```

```
    SQLALCHEMY_DATABASE_URL
)

with engine.connect() as conn:
    rows = conn.execute(text("select * from batting where playerID =
'choosh01'"))

print(rows)
print("="*20, "추신수 선수 모든 시즌 타격 기록 조회")

for row in rows:
    print(row)
```

python crud.py로 파이썬 파일을 실행하여 결과 확인.

STEP 2

이번에는 선수 이름을 변수로 받아서 SQL문에 반영해보자.

```
from sqlalchemy import create_engine, text


SQLALCHEMY_DATABASE_URL = "sqlite:///./lahmansbaseballdb.sqlite"

engine = create_engine(
    SQLALCHEMY_DATABASE_URL
)

playerID = 'choosh01'

with engine.connect() as conn:
    rows = conn.execute(
        text("select * from batting where playerID = :playerID"),
        {'playerID': playerID}
    )

print(rows)
print("="*20, "추신수 선수 모든 시즌 타격 기록 조회")

for row in rows:
    print(row)
```

playerID 변수에 다른 선수들의 playerID를 넣어주면 그 선수들의 기록들이 조회될 것이다.

STEP 3

이번에는 함수 처리를 하여 코드를 재사용가능하게 해볼 것이다.

```
from sqlalchemy import create_engine, text


SQLALCHEMY_DATABASE_URL = "sqlite:///./lahmansbaseballdb.sqlite"

engine = create_engine(
    SQLALCHEMY_DATABASE_URL
)


def read_player_batting_data(playerID: str):
    with engine.connect() as conn:
        rows = conn.execute(
            text("select * from batting where playerID = :playerID"),
            {'playerID': playerID}
        )

    row_list = [row for row in rows]
    return row_list


if __name__ == "__main__":
    choo_data = read_player_batting_data('choosh01')
    print("="*20, "추신수 선수 모든 시즌 타격 기록 조회")
    print(choo_data)

    kang_data = read_player_batting_data('kangju01')
    print("="*20, "강정호 선수 모든 시즌 타격 기록 조회")
    print(kang_data)
```

STEP 4

이후 백엔드 API에서 데이터를 활용하려면 함수의 리턴값을 딕셔너리의 형태 또는 딕셔너리의 리스트 형태로 전달해주는 것이 좋음.

특히 튜플이 포함되지 않은 형태로 보내야 함. 튜플은 JSON 양식에 적합하지 않음.

```
from sqlalchemy import create_engine, text


SQLALCHEMY_DATABASE_URL = "sqlite:///./lahmansbaseballdb.sqlite"

engine = create_engine(
    SQLALCHEMY_DATABASE_URL
)


def read_player_batting_data(playerID: str):
    with engine.connect() as conn:
        rows = conn.execute(
```

```
            text("select * from batting where playerID = :playerID"),
            {'playerID': playerID}
        )

    columns = rows.keys()

    data = []
    for row in rows:
        data_dict = {column: row[idx] for idx, column in
enumerate(columns)}
        data.append(data_dict)

    return data


if __name__ == "__main__":
    choo_data = read_player_batting_data('choosh01')
    print("="*20, "추신수 선수 모든 시즌 타격 기록 조회")
    print(choo_data)

    kang_data = read_player_batting_data('kangju01')
    print("="*20, "강정호 선수 모든 시즌 타격 기록 조회")
    print(kang_data)
```

> 실습: 투수 특정 시즌 기록을 조회하는 함수를 crud.py 내에 만들어보고 활용해보자