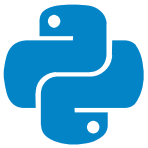


DATA ENGINEERING COURSE

파이썬과 MongoDB
실전 데이터 핸들링

PyMongo 연동부터 웹 크롤링 데이터 저장 실습까지

오늘의 학습 목표 (Agenda)

Part 1. 환경 설정 (Setup)

- MongoDB Atlas (클라우드) 소개
- Python 라이브러리 설치 (pymongo)
- DB 연결 테스트

Part 2. CRUD 마스터

- **C**reate: 데이터 넣기 (Insert)
- **R**ead: 데이터 찾기 (Find)
- **U**pdate: 데이터 수정하기 (Update)
- **D**elete: 데이터 삭제하기 (Delete)

Part 3. 실전 미니 프로젝트

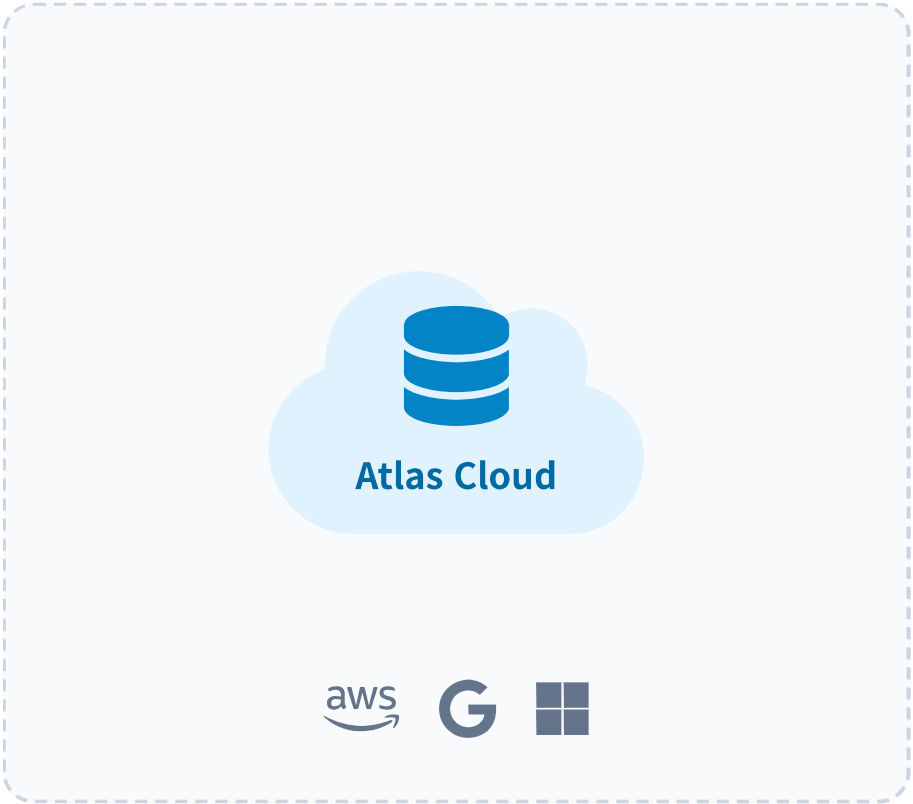
- 주제: 뉴스 제목 수집기
- 구조: Web → Python → MongoDB
- 데이터 수집 및 저장 실습
- 저장된 데이터 확인 (Compass)

PART 01

환경 설정 (Setup)

파이썬에서 데이터베이스를 다루기 위한
준비 운동 단계

1. 어디에 설치하나요? (MongoDB Atlas)



클라우드 DB의 장점

과거에는 DB를 컴퓨터에 직접 설치(Local)했지만, 이제는 **Atlas(클라우드)**를 사용합니다.

- **설치 불필요:** 복잡한 환경 설정 없이 회원가입만 하면 끝.
- **어디서나 접속:** 집, 회사, 카페 어디서든 같은 DB에 접속 가능.
- **무료 티어:** 학습용(M0 Sandbox)은 평생 무료.

2. 파이썬 라이브러리 설치

PyMongo (파이몽고)

파이썬 프로그램이 MongoDB와 대화할 수 있게 해주는 **드라이버(Driver)**입니다.

- **dnspython:** 클라우드(Atlas) 주소를 해석하기 위해 필요한 추가 패키지입니다.

터미널(Terminal) 명령어

```
# 파이썬 가상환경에서 실행하세요
pip install pymongo
pip install "pymongo[srv]"
```

* srv 옵션은 Atlas 연결 시 필수적인 의존성 패키지를 함께 설치해 줍니다.

3. DB 연결하기 (Connection)

```python
from pymongo import MongoClient

# 1. Atlas에서 복사한 연결 문자열 (보안 주의!)
# 실제로는 환경변수나 별도 파일로 관리해야 합니다.
URI = "mongodb+srv://admin:pw@cluster0.abcde.mongodb.net/"

# 2. 클라이언트 생성 (연결 시도)
client = MongoClient(URI)

# 3. 데이터베이스 선택 (없으면 자동으로 생성됨)
db = client['python_lecture']

print("MongoDB 연결 성공!")
```

⚠️ **주의사항:** URI에는 아이디와 비밀번호가 포함되어 있습니다. 깃허브 등에 올릴 때는 반드시 가려야 합니다.

PART 02

CRUD 마스터

생성(Create), 조회(Read), 수정(Update), 삭제(Delete)
데이터 조작의 4대 요소

[Create] 데이터 저장하기

insert_one()

파이썬의 **딕셔너리(Dictionary)**를 그대로 전달하면 됩니다.

- `_id`(고유키)는 지정하지 않으면 MongoDB가 알아서 생성해 줍니다.
- 스키마가 없으므로 필드를 자유롭게 넣을 수 있습니다.

```
# 저장할 데이터 (딕셔너리)
user_data = {
    "name": "김코딩",
    "age": 25,
    "skills": ["Python", "Git"]
}

# 'users' 컬렉션에 저장
db.users.insert_one(user_data)
```

[Read] 데이터 조회하기 (기본)

```
# 1. 한 개만 찾기 (find_one)
# 조건: 이름이 '김코딩'인 사람
user = db.users.find_one({"name": "김코딩"})
print(user)

# 2. 여러 개 찾기 (find)
# 조건: 나이가 25살인 모든 사람
cursor = db.users.find({"age": 25})

# 반복문으로 출력 (커서 순회)
for person in cursor:
    print(person)
```

find_one vs find

- **find_one:** 조건에 맞는 문서 중 가장 먼저 발견된 1개만 딕셔너리로 리턴합니다.
- **find:** 조건에 맞는 모든 문서를 가리키는 **커서(Cursor)**를 리턴합니다.
- **커서(Cursor)**: 데이터 꾸러미를 가리키는 손가락 같은 개념으로, for문을 돌려야 실제 데이터를 꺼낼 수 있습니다.

[Read] 고급 검색 (연산자)

단순 일치 검색 외에 범위 검색 등이 필요할 때 **$연산자**를 사용합니다.

연산자	의미	예시 코드
`$gt`	Greater Than (초과)	`{"age": {"$gt": 20}}` (20세 초과)
`$gte`	Greater Than or Equal (이상)	`{"age": {"$gte": 20}}` (20세 이상)
`$lt`	Less Than (미만)	`{"age": {"$lt": 30}}` (30세 미만)

```
# 예: 나이가 20세 이상인 사람 찾기
results = db.users.find({"age": {"$gte": 20}})
```

[Update] 데이터 수정하기

$set 연산자

문서 전체를 덮어쓰는 것이 아니라, **특정 필드만 수정**할 때 사용합니다.

- 첫 번째 인자: **누구를** (조건)
- 두 번째 인자: **어떻게** (변경 내용)

```
# 이름이 '김코딩'인 사람을 찾아서
# 나이를 26으로 변경 ($set 사용)

db.users.update_one(
    {"name": "김코딩"},
    {"$set": {"age": 26}}
)
```

[Delete] 데이터 삭제하기

```
# 이름이 '김코딩'인 문서 1개 삭제
db.users.delete_one({"name": "김코딩"})

# 나이가 20세 미만인 모든 문서 삭제
db.users.delete_many({"age": {"$lt": 20}})
```

주의사항

- **delete_one:** 조건에 맞는 문서가 여러 개여도 맨 처음 1개만 지웁니다.
- **delete_many:** 조건에 맞는 **모든** 문서를 지웁니다. (신중하게 사용!)
- 빈 조건 {}을 주면 컬렉션의 모든 데이터가 사라집니다.

PART 03

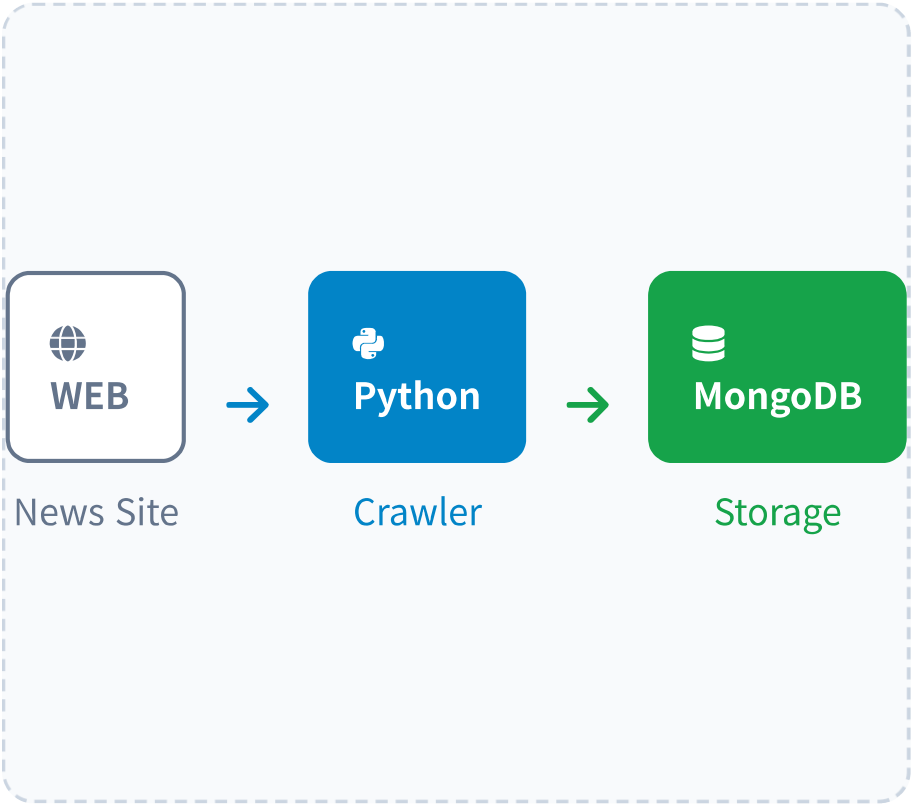
실전 미니 프로젝트

웹상의 데이터를 수집(Crawling)하여
MongoDB에 적재하는 파이프라인 구축하기

프로젝트 구조 (Architecture)

뉴스 제목 수집기

- **목표:** IT 뉴스 사이트의 제목과 링크를 수집한다.
- **도구:**
  - `requests`, `BeautifulSoup` (크롤링)
  - `pymongo` (DB 저장)



[Step 1] 데이터 수집 (Crawling Logic)

가상의 뉴스 리스트를 크롤링하는 코드입니다.

```
import requests
from bs4 import BeautifulSoup

def get_news_data():
    # 실습을 위한 가상의 리스트 (실제로는 requests.get(url) 사용)
    raw_data = [
        {"title": "MongoDB 7.0 출시", "link": "..."},
        {"title": "Python의 미래", "link": "..."},
        {"title": "AI와 데이터베이스", "link": "..."}
    ]
    return raw_data

news_list = get_news_data()
print(f"수집된 뉴스: {len(news_list)}건")
```

[Step 2] MongoDB 적재 (Loading Logic)

수집된 리스트를 한 번에 저장합니다. (Batch Insert)

```
from pymongo import MongoClient

# 1. DB 연결
client = MongoClient("YOUR_ATLAS_URI")
db = client['crawling_project']

# 2. 데이터 저장 (insert_many 사용)
if news_list:
    db.news.insert_many(news_list)
    print("DB 저장 완료!")
else:
    print("저장할 데이터가 없습니다.")

# 3. 확인 출력
saved_count = db.news.count_documents({})
print(f"현재 DB에 저장된 총 뉴스: {saved_count}건")
```

[Step 3] 저장 결과 확인 (Compass)

MongoDB Compass

DB 내부를 엑셀처럼 눈으로 보여주는 GUI 툴입니다.

- Atlas 웹사이트에서도 **'Browse Collections'** 버튼을 통해 확인할 수 있습니다.
- 데이터가 JSON 형태(Document)로 잘 들어갔는지 확인합니다.

자주 발생하는 오류 (Troubleshooting)

1. 인증 오류

URI에 있는 password 부분을 실제 비밀번호로 바꿨는지 확인하세요. <password> 괄호도 지워야 합니다.

2. 네트워크 오류

Atlas 설정의 **Network Access** 메뉴에서 `0.0.0.0/0` (모든 IP 허용)이 추가되어 있는지 확인하세요.

오늘의 핵심 요약

PyMongo

파이썬 딕셔너리를 그대로 DB에 넣을 수 있는 강력한 도구입니다.

CRUD

데이터 생성, 조회, 수정, 삭제의 기본 조작법을 익혔습니다.

활용

크롤링한 비정형 데이터를 저장하기에 NoSQL이 매우 적합함을 확인했습니다.

Q & A

실습 중 안 되는 부분이 있다면 질문해주세요.

과제: 관심 있는 뉴스 사이트를 크롤링해서 DB에 100개 저장해오기