DataBase

Python DB API

- Python DB API
  - ✓ Python에서 데이터베이스를 엑서스하기 위해서는 Python DB API를 사용
  - ✓ Python DB API는 데이터베이스를 엑세스하는 표준 API 로서 여러 DB 엑세스 모듈에서 이 최소한의 API 인터페이스 표준을 따름
  - ✓ 표준 API는 크게 데이터베이스를 연결하고, SQL 문을 실행하고, 연결을 닫는 등의 기본적인 DB 작업과 관련된 기능들을 정의
  - ✓ 데이터베이스를 엑세스하는 또 다른 방식으로 ORM(Object Relational Mapping)이 있는데 Django ORM, PonyORM, peewee 등이 ORM 방식을 사용한 데이터 엑세스를 제공
  - ✓ Python에서 지원하는 데이터베이스는 매우 다양하기 때문에 Python에서 각 데이터베이스를 사용하기 위해서는 각각의 DB에 상응하는 별도의 DB 모듈을 다운받아야 함(sqlite3 는 Python 2.5 이상에서 기본 내장).
  - ✓ 수많은 DB 모듈들이 있지만 이들이 거의 모두 Python DB API 표준을 따르고 있으므로 동일한 API를 사용해 데이터베이스를 사용할 수 있음
  - ✓ python에서는 거의 모든 데이터베이스를 사용할 수 있는데 MySQL, PostgreSQL, MSSQL, Sqlite, Oracle, Sybase, Informix 등과 같은 대표적인 DB 들을 모두 지원

MySQL 연동

❖ 데이터베이스 접속 확인

✓ MySQL 모듈 설치

```
pip install pyMySQL
```

✓ 데이터베이스 접속 확인

```
import pymysql

#데이터베이스 연결
con = pymysql.connect(host='데이터베이스 위치', port=포트번호,
                      user='계정', passwd='비밀번호',
                          db='데이터베이스 이름', charset ='인코딩방식')

print(con)
con.close()
```

MySQL 연동

- 데이터베이스 접속 확인
  - ✓ 데이터베이스 접속이 되지 않는 경우 아래와 같은 예외 발생

```
Traceback (most recent call last):
  File "C:₩Users₩Administrator₩python₩test₩__main__.py", line 12, in <module>
exception: (<class 'pymysql.err.OperationalError'>, OperationalError(2003, "Can't
    connect to MySQL server on '211.183.2.253' ([WinError 10060] 연결된 구성원으로
    부터 응답이 없어 연결하지 못했거나, 호스트로부터 응답이 없어 연결이 끊어졌습니
    다)"), <traceback object at 0x00C839E0>)
    if con != None:
NameError: name 'con' is not defined
```

MySQL 연동

❖ 샘플 데이터 생성

✓ MySQL에 접속해서 테이블 생성

```
show databases;

use adam;

create table usertbl(
userid char(15) not null primary key,
name varchar(20) not null,
birthyear int not null,
addr char(100),
mobile char(11),
mdate date);
```

✓ 데이터 삽입 및 확인

```
insert into usertbl values('ailee', '에일리',1989,'미국','01000000000', '1989-5-30');

commit;

select * from usertbl;
```

MySQL 연동

- 파이썬에서 DML 수행
  1. 연결 객체의 cursor() 함수를 호출해서 sql 실행 객체를 생성
  2. execute(실행 할 sql문장)
  3. 연결 객체의 commit() 을 호출하면 작업 내용이 반영되고 rollback()을 호출하면 작업 취소

MySQL 연동

❖ 데이터 삽입

```
import sys, pymysql

try:
    # 데이터베이스 연결
    con = pymysql.connect(host='localhost',
                          port=3306, user='adam', passwd='wnddkd', db='adam', charset='utf8')
    cursor = con.cursor()

    # 데이터 삽입 - 파라미터 없이 직접 값 입력
    cursor.execute("insert into usertbl values('ljy', '이진연', 1970, '서울', '01012345678',
     '1970-10-31')")
    # 데이터 삽입 - 파라미터를 이용해서 값 대입
    cursor.execute("insert into usertbl values(%s, %s, %s, %s, %s, %s)",
                   ('sohye', '김소혜', 1999, '서울', '01012121212', '1999-7-19'))
    # 작업 종료
    con.commit()
    print("삽입 성공")
except:
    print('exception:', sys.exc_info())
finally:
    con.close()
```

MySQL 연동

❖ 데이터 수정 과 삭제

```
#데이터 수정 및 삭제
import sys, pymysql
try:
    #데이터베이스 연결
    con = pymysql.connect(host='localhost',
                          port=3306, user='adam', passwd='wnddkd', db='adam', charset='utf8')
    cursor = con.cursor()

    #데이터 수정
    cursor.execute("update usertbl set name='이지연' where name = '이진연'")
    #데이터 삭제
    cursor.execute("delete from usertbl where name = '김소혜'")
    con.commit()
    print("편집 성공")
except:
    print('exception:', sys.exc_info())
finally:
    con.close()
```

MySQL 연동

- 데이터 조회
  1. 연결 객체의 cursor() 함수를 호출해서 sql 실행 객체를 가져옴
  2. execute(실행 할 sql문장)
  3. cursor 객체를 가지고 fetchall 함수를 호출하면 튜플들의 튜플로 결과가 리턴되며 fetchone 함수를 호출하면 첫번째 데이터 1개만 튜플로 리턴

MySQL 연동

❖ 데이터 1개 조회

```
import sys, pymysql

try:
    # 데이터베이스 연결
    con = pymysql.connect(host='localhost',
                          port=3306, user='adam', passwd='wnddkd', db='adam', charset='utf8')
    cursor = con.cursor()

    # 데이터 읽어오는 SQL 실행
    cursor.execute("select * from usertbl")
    # 데이터 1개를 튜플로 가져오기
    data = cursor.fetchone()
    # 튜플을 순회하면서 출력
    for imsi in data:
        print(imsi)
except:
    print('exception:', sys.exc_info())
finally:
    con.close()
```

MySQL 연동

❖ 데이터 여러 개 조회

```
import sys, pymysql

try:
    # 데이터베이스 연결
    con = pymysql.connect(host='localhost',
                          port=3306, user='adam', passwd='wnddkd', db='adam', charset='utf8')
    cursor = con.cursor()

    # 데이터 읽어오는 SQL 실행
    cursor.execute("select * from usertbl")
    # 전체 데이터를 가져와서 튜플의 튜플로 생성
    data = cursor.fetchall()
    for imsi in data:
        print(imsi)
except:
    print('exception:', sys.exc_info())
finally:
    con.close()
```

MySQL 연동

- MySQL 에서 프로시저 생성

```
DELIMITER //
 CREATE PROCEDURE myproc(IN _userid CHAR(15),
  IN _name VARCHAR(20), _birthyear int, _addr CHAR(10), _mobile char(11), _mdate date)
   BEGIN
   insert into usertbl values(_userid, _name, _birthyear, _addr, _mobile, _mdate);
   END //
DELIMITER ;
```

MySQL 연동

❖ 프로시저 실행

```
import sys, pymysql
try:
    con = pymysql.connect(host='localhost',
                          port=3306, user='adam', passwd='wnddkd', db='adam', charset ='utf8')
    cursor = con.cursor()
    cursor.callproc('myproc', args=('momo', '모모', 1996, '교토', '01098765432', '1996-11-
     9'))
    con.commit()
except:
    print('exception:', sys.exc_info())
finally:
    con.close()
```

MySQL 연동

❖ BLOB 사용을 위한 테이블 생성

```
create table blobtable(
    userid char(15) not null primary key,
    filename varchar(1000),
    filecontent longblob);

select * from blobtable;
```

MySQL 연동

- BLOB 저장

```
import sys, pymysql
#자신의 이미지 파일 경로를 사용하세요
# read file
filename = 'data/image1.png'

#파일을 열어서 내용을 읽어서 photo에 저장
f =  open(filename, 'rb')
photo = f.read()
f.close()


#파일이름만 추출
sp = filename.split('/')
```

MySQL 연동

- BLOB 저장

```
query = 'insert into blobtable values(%s, %s, %s)'
args = ('태연', sp[len(sp)-1], photo)
try:
    con = pymysql.connect(host='localhost',
    port=3306, user='adam', passwd='wnddkd', db='adam', charset ='utf8')
    cursor = con.cursor()
    cursor.execute(query, args)
    con.commit()
    print("삽입성공")
except:
    print('exception:', sys.exc_info())
finally:
    cursor.close()
    con.close()
```

MySQL 연동

❖ BLOB 읽기 – 이전에 업로드 한 파일을 삭제하고 실행

```
import sys, pymysql
try:
    con = pymysql.connect(host='localhost',
                          port=3306, user='adam', passwd='wnddkd', db='adam', charset ='utf8')
    cursor = con.cursor()
    cursor.execute("select * from blobtable")
    data = cursor.fetchall()
    for imsi in data:
        print(imsi[0])
        print(imsi[1])
        #파일을 기록
        f = open('data/' + imsi[1], 'wb')
        f.write(imsi[2])
        f.close()
except:
    print('exception:', sys.exc_info())
finally:
    con.close()
```

Oracle 연동

❖ 오라클 연결

✓ 오라클 모듈 설치: pip install cx_Oracle

✓ 연결 만들기

```
# 모듈을 불러옵니다.
import cx_Oracle

# 데이터베이스에 연결합니다.
변수1 = cx_Oracle.makedsn("IP주소 " , " 포트번호 " , "sid " )
변수2 = cx_Oracle.connect(user="계정", password= " 비밀번호", dsn=변수1)
```

Oracle 연동

- ❖ 오라클 연결
  - ✓ Mac 에서의 오라클 접속
    - 아래 사이트에 접속해서 2개의 파일 다운로드

      https://www.oracle.com/database/technologies/instant-client/macos-intel-x86-downloads.html
      - ◆ nstantclient-basic-macos.x64-19.3.0.0.0dbru.zip
      - ◆ instantclient-sqlplus-macos.x64-19.3.0.0.0dbru.zip
      - ◆ instantclient-sdk-macos.x64-19.3.0.0.0dbru.zip
    - opt 디렉토리에 oracle 디렉토리 생성
    - oracle 디렉토리에 3개의 파일의 압축을 해제
      - ◆ Instantclient 디렉토리에 압축 해제
    - 터미널에서 심볼릭 링크 생성: 경로는 자신의 버전에 맞게 수정
      - $ ln -sf /Users/adam/opt/oracle/instantclient_19_3/sdk/include/*.h /usr/local/include/
      - $ ln -sf /usr/local/oracle/instantclient_19_3/sqlplus /usr/local/bin/
      - $ ln -sf /opt/oracle/instantclient_19_3/*.dylib /usr/local/lib/
      - $ ln -sf /opt/oracle/instantclient_19_3/*.dylib.19.1 /usr/local/lib/
      - $ ln -sf /opt/oracle/instantclient_19_3/libclntsh.dylib.19.1 /usr/local/lib/libclntsh.dylib

Oracle 연동

❖ Mac 에서의 오라클 접속

✓ 아래 명령을 수행하고 작성 – 환경 설정 추가

$ sudo nano ~/.zshrc

```
export ORACLE_HOME="/opt/oracle/instantclient_19_3"
export OCI_INC_DIR="$ORACLE_HOME/sdk/include/"
export OCI_LIB_DIR="$ORACLE_HOME"
export LD_LIBRARY_PATH="$ORACLE_HOME"
export DYLD_LIBRARY_PATH="$ORACLE_HOME:$DYLD_LIBRARY_PATH"

// 저장 후, 종료
```

✓ 설정 즉시 적용

$ source ~/.zshrc

Oracle 연동

❖ 오라클 접속

```
# 모듈을 불러옵니다.
import cx_Oracle
import sys

try:
    # 데이터베이스에 연결합니다.
    dsnStr = cx_Oracle.makedsn("localhost", "1521", "xe")
    con = cx_Oracle.connect(user="scott", password="tiger", dsn=dsnStr)
    print(con)
except:
    print('exception:', sys.exc_info())
finally:
    con.close()
```

Oracle 연동

❖ 데이터베이스 접속이 되지 않는 경우 아래와 같은 예외 발생

```
Traceback (most recent call last):
  File "C:₩Users₩Administrator₩python₩test₩__main__.py", line 12, in <module>
exception: (<class 'pymysql.err.OperationalError'>, OperationalError(2003, "Can't connect to MySQL server on '211.183.2.253' ([WinError 10060] 연결된 구성원으로부터 응답이 없어 연결하지 못했거나, 호스트로부터 응답이 없어 연결이 끊어졌습니다)"), <traceback object at 0x00C839E0>)
    if con != None:
NameError: name 'con' is not defined
```

Oracle 연동

- 파이썬에서 DML 수행
  1. 연결 객체의 cursor() 함수를 호출해서 sql 실행 객체를 가져옴
  2. execute(실행 할 sql문장, 파라미터 튜플)
  3. 연결 객체의 commit() 을 호출하면 작업 내용이 반영되고 rollback()을 호출하면 작업 취소
     - 완성된 sql 인 경우 파라미터 생략 가능
     - Sql 문장에 :번호 형태로 파라미터를 설정한 후 파라미터들을 tuple로 만들어서 대입 가능

Oracle 연동

❖ 오라클에 테이블 생성

```
CREATE TABLE DEPT
 (DEPTNO NUMBER(2) CONSTRAINT PK_DEPT PRIMARY KEY,
  DNAME VARCHAR2(14) ,
  LOC VARCHAR2(13));
```

Oracle 연동

❖ 데이터 삽입

```
import cx_Oracle
try:
    # 데이터베이스에 연결합니다.
    dsnStr = cx_Oracle.makedsn("localhost", "1521", "xe")
    con = cx_Oracle.connect(user="scott", password="tiger", dsn=dsnStr)

    cursor = con.cursor()
    #cursor.execute("insert into dept(deptno, dname, loc) values(50, '총무', '목포')")
    cursor.execute('insert into dept values(:1, :2, :3)', (60, '영업', '서울'))

    con.commit()
    print("삽입 성공")
except Exception as e:
    print('exception:', e)
finally:
    con.close()
```

Oracle 연동

❖ 데이터 수정

```
import cx_Oracle
try:
    # 데이터베이스에 연결합니다.
    dsnStr = cx_Oracle.makedsn("localhost", "1521", "xe")
    con = cx_Oracle.connect(user="scott", password="tiger", dsn=dsnStr)

    cursor = con.cursor()
    cursor.execute('update dept set dname = :1 where deptno = :2', ('회계', 60))

    con.commit()
    print("수정 성공")
except Exception as e:
    print('exception:', e)
finally:
    con.close()
```

Oracle 연동

❖ 데이터 삭제

```
import cx_Oracle
try:
    # 데이터베이스에 연결합니다.
    dsnStr = cx_Oracle.makedsn("localhost", "1521", "xe")
    con = cx_Oracle.connect(user="scott", password="tiger", dsn=dsnStr)

    cursor = con.cursor()
    cursor.execute('delete from dept where deptno = :1', (60,))

    con.commit()
    print("삭제 성공")
except Exception as e:
    print('exception:', e)
finally:
    con.close()
```

Oracle 연동

- 데이터 검색
  1. 연결 객체의 cursor() 함수를 호출해서 sql 실행 객체를 생성
  2. execute(실행 할 sql문장)
  3. cursor 객체를 가지고 fetchall 함수를 호출하면 튜플들의 튜플로 결과가 리턴되며 fetchone 함수를 호출하면 첫번째 데이터 1개만 튜플로 리턴

Oracle 연동

- 데이터가 존재하지 않으면 샘플 데이터 삽입

```
INSERT INTO DEPT VALUES(10,'ACCOUNTING','NEW YORK');
INSERT INTO DEPT VALUES (20,'RESEARCH','DALLAS');
INSERT INTO DEPT VALUES(30,'SALES','CHICAGO');
INSERT INTO DEPT VALUES(40,'OPERATIONS','BOSTON');
```

Oracle 연동

❖ 데이터 1개 검색

```
import cx_Oracle
try:
    # 데이터베이스에 연결합니다.
    dsnStr = cx_Oracle.makedsn("localhost", "1521", "xe")
    con = cx_Oracle.connect(user="scott", password="tiger", dsn=dsnStr)

    cursor = con.cursor()

    cursor.execute("select * from dept")
    data = cursor.fetchone()
    print(data)
except Exception as e:
    print('exception:', e)
finally:
    con.close()
```

Oracle 연동

❖ 데이터 여러 개 검색

```
import cx_Oracle
try:
    # 데이터베이스에 연결합니다.
    dsnStr = cx_Oracle.makedsn("localhost", "1521", "xe")
    con = cx_Oracle.connect(user="scott", password="tiger", dsn=dsnStr)

    cursor = con.cursor()

    cursor.execute("select * from dept")
    data = cursor.fetchall()
    for imsi in data:
        print(imsi)
except Exception as e:
    print('exception:', e)
finally:
    con.close()
```

Oracle 연동

- 프로시저 생성 및 실행
  - ✓ 프로시저 생성

```
CREATE OR REPLACE PROCEDURE INSERT_DEPT
( VDEPTNO IN          DEPT.DEPTNO%TYPE,
  VDNAME IN DEPT.DNAME%TYPE,
  VLOC IN DEPT.LOC%TYPE
)
IS
BEGIN
  INSERT INTO DEPT
  VALUES(VDEPTNO, VDNAME, VLOC);
END;
/
```

  - ✓ 프로시저 실행

```
BEGIN
    INSERT_DEPT(98, '영업','신안');
END;
```

  - ✓ 확인

```
SELECT *
FROM DEPT;
```

Oracle 연동

❖ 프로시저 실행

```
import cx_Oracle
try:
    # 데이터베이스에 연결합니다.
    dsnStr = cx_Oracle.makedsn("localhost", "1521", "xe")
    con = cx_Oracle.connect(user="scott", password="tiger", dsn=dsnStr)

    cursor = con.cursor()
    cursor.callproc('INSERT_DEPT', (95, '재무','홍대'))
    con.commit()
    print('프로시저 수행 성공')

except Exception as e:
    print('exception:', e)
finally:
    con.close()
```

Oracle 연동

❖ BLOB 사용을 위한 테이블 생성

```
CREATE TABLE BLOBTABLE(
    USERID CHAR(15) NOT NULL PRIMARY KEY,
    FILENAME VARCHAR2(1000),
    FILECONTENT BLOB);

SELECT * FROM BLOBTABLE;
```

Oracle 연동

- BLOB 저장

```
import cx_Oracle
try:
    # 데이터베이스에 연결합니다.
    dsnStr = cx_Oracle.makedsn("localhost", "1521", "xe")
    con = cx_Oracle.connect(user="scott", password="tiger", dsn=dsnStr)

    cursor = con.cursor()

    #자신의 이미지 파일 경로를 사용하세요
    # read file
    filename = 'data/image1.png'
    f =  open(filename, 'rb')
    photo = f.read()
    f.close()
```

Oracle 연동

❖ BLOB 저장

```
        sp = filename.split('/')
        query = 'insert into BLOBTABLE values(:1, :2, :3)'
        args = ('태연', sp[len(sp)-1], photo)

        cursor.execute(query, args)
        con.commit()
        print('BLOB 삽입 성공')

    except Exception as e:
        print('exception:', e)
    finally:
        con.close()
```

Oracle 연동

- BLOB 읽기

```
import cx_Oracle
import sys
try:
    # 데이터베이스에 연결합니다.
    dsnStr = cx_Oracle.makedsn("localhost", "1521", "xe")
    con = cx_Oracle.connect(user="scott", password="tiger", dsn=dsnStr)
    cursor = con.cursor()
    cursor.execute("select * from blobtable")
    data = cursor.fetchall()
    for imsi in data:
        print(imsi[0])
        print(imsi[1])
        f = open('./data/' + imsi[1], 'wb')
        blob = imsi[2]
        offset = 1
        while True:
            data = blob.read(offset, 65536)
            if data:
                f.write(data)
            if len(data) < 65536:
                break
            offset += len(data)
        f.close()
```

Oracle 연동

- BLOB 읽기

```
except:
    print('exception:', sys.exc_info())
finally:
    con.close()
```

MongoDB 연동

- pymongo 패키지 설치

  pip install pymongo

- 서버 연결

  con = pymongo.MongoClient("서버 IP", 포트번호) # 포트 번호는 27017 이면 생략 가능

- 데이터베이스 생성

  db = con.데이터베이스이름

  없으면 생성됨

- 컬렉션 연결

  collection = db.collection 이름

  없으면 생성됨

MongoDB 연동

- 데이터 삽입

  insert_one, insert_many 함수 이용

  Insert_one 함수의 경우 데이터를 삽입하고 inserted_id 속성을 호출하면 삽입된 ObjectId 확인 가능

- 데이터 개수는 count_documents({}) 함수 이용

MongoDB 연동

❖ 데이터 삽입

```
from pymongo import MongoClient

#데이터베이스 연결
conn = MongoClient('127.0.0.1')

#데이터베이스 설정
db = conn.mymongo

#컬렉션 설정
collect = db.users

#document 생성 : {'key':'value'}
doc1 = {'empno':'10001', 'name':'kim', 'phone':'010-111-111', 'age':35}
doc2 = {'empno':'10002', 'name':'lee', 'age':45}
doc3 = {'empno':'10003', 'name':'park', 'phone':'010-222-222', 'age':25}

doc4 = {'empno':'10004', 'name':'choi', 'age':42}
doc5 = {'empno':'10005', 'name':'yun', 'phone':'010-222-222', 'age':37}
```

MongoDB 연동

❖ 데이터 삽입

```
#데이터를 1개씩 삽입
collect.insert_one(doc1)
collect.insert_one(doc2)
collect.insert_one(doc3)

#데이터를 여러 개 삽입
collect.insert_many([doc4, doc5])


#데이터 여러 개 삽입
collect.insert_many(
    [
        {'num': i} for i in range(10)
    ]
)
```

MongoDB 연동

❖ Mongo DB에 접속해서 명령어 수행

```
> use mymongo
switched to db mymongo
> db.users.find()
{ "_id" : ObjectId("6077f46cbf1adc01e38b9bc6"), "empno" : "10001", "name" : "kim",
"phone" : "010-111-111", "age" : 35 }
{ "_id" : ObjectId("6077f46cbf1adc01e38b9bc7"), "empno" : "10002", "name" : "lee", "age" :
45 }
{ "_id" : ObjectId("6077f46cbf1adc01e38b9bc8"), "empno" : "10003", "name" : "park",
"phone" : "010-222-222", "age" : 25 }
{ "_id" : ObjectId("6077f46cbf1adc01e38b9bc9"), "empno" : "10004", "name" : "choi",
"age" : 42 }
{ "_id" : ObjectId("6077f46cbf1adc01e38b9bca"), "empno" : "10005", "name" : "yun",
"phone" : "010-222-222", "age" : 37 }
{ "_id" : ObjectId("6077f46cbf1adc01e38b9bcb"), "num" : 0 }
{ "_id" : ObjectId("6077f46cbf1adc01e38b9bcc"), "num" : 1 }
{ "_id" : ObjectId("6077f46cbf1adc01e38b9bcd"), "num" : 2 }
{ "_id" : ObjectId("6077f46cbf1adc01e38b9bce"), "num" : 3 }
{ "_id" : ObjectId("6077f46cbf1adc01e38b9bcf"), "num" : 4 }
{ "_id" : ObjectId("6077f46cbf1adc01e38b9bd0"), "num" : 5 }
{ "_id" : ObjectId("6077f46cbf1adc01e38b9bd1"), "num" : 6 }
{ "_id" : ObjectId("6077f46cbf1adc01e38b9bd2"), "num" : 7 }
{ "_id" : ObjectId("6077f46cbf1adc01e38b9bd3"), "num" : 8 }
{ "_id" : ObjectId("6077f46cbf1adc01e38b9bd4"), "num" : 9 }
```

MongoDB 연동

- 데이터 검색
  - ✓ find_one() 과 find() 를 이용
  - ✓ find_one 은 하나의 데이터만 리턴
  - ✓ find()는 모든 데이터를 리턴

MongoDB 연동

❖ 데이터 검색

```
from pymongo import MongoClient

#데이터베이스 연결
conn = MongoClient('127.0.0.1')

#데이터베이스 설정
db = conn.mymongo

#컬렉션 설정
collect = db.users

# 전체 문서 조회
result = collect.find()

#print(result)
# 조회 문서 출력
print('전체 데이터 조회')
for r in result :
    print(r)
```

MongoDB 연동

- 데이터 검색

```
# 조건 검색
print('조건 검색')
result2 = collect.find({ 'age':{'$gte':30} }) #크거나 같다
for r in result2 :
    print(r)

print('조건 검색 및 정렬')
result2 = collect.find({ 'age':{'$gte':30} }).sort("age")
for r in result2 :
    print(r)
```

MongoDB 연동

- 데이터 수정
  - ✓ update_one() 과 update_many()
  - ✓ update_one() : 가장 먼저 검색되는 한 Document만 수정
  - ✓ update_many() : 조건에 맞는 모든 Document 수정

- 데이터 삭제
  - ✓ delete_one() 과 delete_many()
  - ✓ delete_one() : 가장 먼저 검색되는 한 Document만 삭제
  - ✓ delete_many() : 조건에 맞는 모든 Document 삭제

MongoDB 연동

❖ 데이터 수정 및 삭제

```
from pymongo import MongoClient

# 데이터베이스 연결
conn = MongoClient('127.0.0.1')

# 데이터베이스 설정
db = conn.mymongo

# 컬렉션 설정
collect = db.users

# 전체 문서 조회
result = collect.find()

# 조회 문서 출력
for r in result :
    print(r)
```

MongoDB 연동

❖ 데이터 수정 및 삭제

```
# 데이터 1개 수정
collect.update_one(
    { "empno" : "10001" },
    { "$set" :
        { "name" : "kkk" }
    }
)

# 전체 문서 조회
result = collect.find()

# 조회 문서 출력
for r in result :
    print(r)
```

MongoDB 연동

- 데이터 수정 및 삭제

```
# 데이터 여러 개 수정
collect.update_many(
    { 'age':{'$gte':30} },
    { "$set" :
        { "name" : "park" }
    }
)

# 전체 문서 조회
result = collect.find()

# 조회 문서 출력
for r in result :
    print(r)
```

MongoDB 연동

- 데이터 수정 및 삭제

```
# 데이터 삭제
collect.delete_many( {"age": { "$gte": 30 } } )

# 전체 문서 조회
result = collect.find()

# 조회 문서 출력
for r in result :
    print(r)
```

Python Redis

❖ 패키지 설치 및 데이터베이스 접속

✓ 패키지 – redis

✓ 연결

```
#데이터베이스 접속
import redis

# 호스트와 포트 설정, 데이터베이스 설정하여 connection 취득
with redis.StrictRedis(host='IP', port=포트번호 …) as 연결 객체
```

Python Redis

❖ 접속 및 Set을 이용한 저장

```
import redis

# 호스트와 포트 설정, 데이터베이스 설정하여 connection 취득
with redis.StrictRedis(host='localhost', port=6379) as conn:
    # name 키로 adam 값을 입력
    conn.set('name', 'adam')
    # name 키로 데이터를 조회
    data = conn.get('name')
    # 콘솔 출력
    print(data)
```

Python Redis

❖ 접속 및 Set을 이용한 저장

```
# 호스트와 포트 설정, 데이터베이스 설정하여 pool를 설정
redis_pool = redis.ConnectionPool(host='localhost', port=6379, max_connections=4)
# pool로 connection을 취득
with redis.StrictRedis(connection_pool=redis_pool) as conn:
    # album 키로 list를 입력
    conn.set('album', '제네시스')
    # album 키로 데이터를 조회
    data = conn.get('album')
    # 콘솔 출력
    print(data)
    print(data.decode('utf-8'))
```

```
b'adam'
b'₩xec₩xa0₩x9c₩xeb₩x84₩xa4₩xec₩x8b₩x9c₩xec₩x8a₩xa4'
제네시스
```

Python Redis

❖ 만료 시간 설정

```
# redis 라이브러리 선언
import redis
import time
with redis.StrictRedis(host='localhost', port=6379) as conn:
    # name 키로 adam 값을 입력, 10초의 만료시간
    conn.set('name', 'adam', 10)
    # name 키의 만료시간을 출력
    print(conn.ttl('name'))
    # 만료시간 설정 100초
    conn.expire('name', 30)
    # test 키의 만료시간을 출력
    print(conn.ttl('name'))
    # test를 키로 데이터를 출력
    data = conn.get('name')
    # 콘솔 출력
    print(data)
    time.sleep(60)
    # test를 키로 데이터를 출력
    data = conn.get('name')
    # 콘솔 출력
    print(data)
    # 모든 키 지우기
    # conn.flushall()
```

Python Redis

- 만료 시간 설정

```
10
30
b'adam'
None
```

Python Redis

❖ 여러 종류의 데이터 저장

```
#다른 자료형의 데이터 저장
import redis
with redis.StrictRedis(host='localhost', port=6379) as conn:
    # 리스트에서 오른쪽 입력
    conn.rpush("list", "1");
    # 리스트에서 왼쪽 입력
    conn.lpush("list", "2");
    # 리스트 데이터 출력 - 2, 1
    for i in conn.lrange('list', 0, 10):
        # 콘솔 출력
        print(i)
    # 콘솔 출력
    print('lpop list')
    # 리스트 왼쪽 pop - 2
    print(conn.lpop('list'))
    # 콘솔 출력
    print('list')
    # 리스트 데이터 출력 - 1
    for i in conn.lrange('list', 0, 10):
        # 콘솔 출력
        print(i)
    # 개행
    print()
```

Python Redis

- 여러 종류의 데이터 저장

```
# Hash 형식의 key-value 값 입력
conn.hset("map", "a", "1")
conn.hset("map", "b", "2")
conn.hset("map", "c", "3")
# Hash 형식의 모든 데이터 출력
print(conn.hgetall('map'))
# Hash 형식의 a key의 데이터 출력
print(conn.hget('map', 'a'))
# 개행
print()
```

Python Redis

- 여러 종류의 데이터 저장

```
# Set 형식의 값 입력
conn.sadd('set', 'C')
conn.sadd('set', 'B')
conn.sadd('set', 'A')
# 중복은 입력 안된다.
conn.sadd('set', 'A')
conn.sadd('set', 'A')
# Set 형식 출력
print(conn.smembers('set'))
# Set 형식의 값 입력 - 변수 명 set1
conn.sadd('set1', 'A')
conn.sadd('set1', 'D')
# set과 set1의 교집합
print(conn.sinter('set', 'set1'))
# set과 set1의 합집합
print(conn.sunion('set', 'set1'))
# 개행
print()
```

Python Redis

❖ 여러 종류의 데이터 저장

```
# SortedSet 형식의 값 입력
conn.zadd('sortedset', {'B': 1})
conn.zadd('sortedset', {'A': 3})
conn.zadd('sortedset', {'C': 0})
# 10개의 데이터 출력
print(conn.zrange('sortedset', 0, 9))
# 10개의 데이터 출력 - 내림차순
print(conn.zrange('sortedset', 0, 9, desc=True))
# 2개의 데이터 출력 - 내림차순
print(conn.zrange('sortedset', 0, 1, desc=True))
# 모든 키 지우기
# conn.flushall()
```

Python Redis

- 여러 종류의 데이터 저장

```
b'2'
b'1'
lpop list
b'2'
list
b'1'

{b'a': b'1', b'b': b'2', b'c': b'3'}
b'1'

{b'B', b'A', b'C'}
{b'A'}
{b'B', b'A', b'C', b'D'}

[b'C', b'B', b'A']
[b'A', b'B', b'C']
[b'A', b'B']
```