데이터베이스 연동

안 화 수

pip

❖ 파이썬에서 MySQL에 접속할 경우에 pymysql 모듈을 설치

C:₩>pip install pymysql

❖ 설치된 모듈 목록

C:₩> pip list

❖ 모듈 삭제

C:₩> pip uninstall pymysql

MySQL 연동

MySQL 연동

❖MySQL 모듈 설치

C:₩>pip install pymysql

❖데이터베이스 접속 확인

```
import pymysql

#데이터베이스 연결
con = pymysql.connect(host='211.183.2.100', port=3306,
                      user='root', passwd='1234',
                      db='mysql', charset ='utf8')

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

MySQL 연동

❖MySQL 연동 테스트

mysqltest.py

```
import pymysql

# 데이터 베이스 연결
con = pymysql.connect(host='localhost', user='root',
                      passwd='1234',  port=3306,
                      db='mysql', charset='utf8')
cursor = con.cursor()
cursor.execute('select * from user')

list = cursor.fetchone()                    # 1개의 데이터를 구해옴
print(type(list))                           # 'tuple'
print(list)
```

MySQL 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 '211.183.2.100' ([WinError 10060] 연
결된 구성원으로부터 응답이 없어 연결하지 못했거나, 호스트로부터 응답
이 없어 연결이 끊어졌습니다)"), <traceback object at 0x00C839E0>)
    if con != None:
NameError: name 'con' is not defined
```

MySQL 연동

❖MySQL에 접속해서 테이블 생성

```
create table contact(
    num int primary key auto_increment,
    name varchar(100) not null,
    phone varchar(20));
```

❖데이터 삽입 및 확인

```
insert into contact(name, phone) values( ' ahn', '01098094320')

commit;

select * from contact;
```

MySQL 연동

❖파이썬에서 삽입 및 삭제 또는 갱신

1. 연결 객체의 cursor() 메소드를 호출해서 sql 실행 객체를 가져옵니다.

2. execute(실행 할 sql문장)

3. 연결 객체의 commit() 을 호출하면 작업 내용이 반영되고 rollback()을 호출하면 작업 취소

MySQL 연동

❖ 데이터 삽입 : contact 테이블에 데이터 입력

insert.py

```
import pymysql
try:
    con = pymysql.connect(host='localhost', user='jspid',
                          passwd='jsppass', port=3306,
                          db='jsptest', charset='utf8')
    cursor = con.cursor()
    cursor.execute("insert into contact(name, phone)
     values('kim','01011112222')")
    con.commit()
    print('입력 성공')
except Exception as err:
    print(err)
finally:
    con.close()
```

MySQL 연동

❖파이썬에서 데이터 검색

1. 연결 객체의 cursor() 메소드를 호출해서 sql 실행 객체를 가져옵니다.

2. execute(실행 할 sql문장)

3. cursor 객체를 가지고 fetchall 메소드를 호출하면 튜플들의 튜플로 결과가 리턴되며 fetchone 메소드를 호출하면 첫번째 데이터 1개만 튜플로 리턴됩니다.

MySQL 연동

❖ 데이터 검색 : contact 테이블에서 데이터 1개 검색

selectone.py

```
import pymysql
try:
    con = pymysql.connect(host='localhost',  user='jspid',
                          passwd='jsppass', port=3306,
                          db='jsptest',  charset='utf8')
    cursor = con.cursor()
    cursor.execute('select * from contact')
    row = cursor.fetchone()                    # 데이터 1개만 구해옴
    print(row)
except Exception as err:
    print(err)
finally:
    con.close()
```

MySQL 연동

❖ 데이터 검색 : contact 테이블에서 모든 데이터 검색

selectall.py

```
import pymysql
try:
    con = pymysql.connect(host='localhost',  user='jspid',
                          passwd='jsppass',  port=3306,
                          db='jsptest', charset='utf8')
    cursor = con.cursor()
    cursor.execute('select * from contact')
    rows = cursor.fetchall()                    # 모든 데이터 검색
for r in rows:
        print(r)
except Exception as err:
    print(err)
finally:
    con.close()
```

MySQL 연동

❖ execute 안의 sql 문장을 아래처럼 수정하고 데이터베이스 확인

```
cursor.execute("update contact set phone='01098094320'
                where name = ' 안화수'")

cursor.execute("delete from contact where name = ' 안화수' ")
```

MySQL 연동

❖ 데이터 수정 : contact 테이블 데이터 수정

update.py

```
import pymysql
try:
    con = pymysql.connect(host='localhost',  user='jspid',
                          passwd='jsppass',  port=3306,
                          db='jsptest', charset='utf8')
    cursor = con.cursor()
    cursor.execute("update contact set phone='1234' where num=1")
    con.commit()
    print('수정 성공')
except Exception as err:
    print(err)
finally:
    con.close()
```

MySQL 연동

❖ 데이터 삭제 : contact 테이블 데이터 삭제

delete.py

```
import pymysql
try:
    con = pymysql.connect(host='localhost',  user='jspid',
                          passwd='jsppass',  port=3306,
                          db='jsptest', charset='utf8')
    cursor = con.cursor()
    cursor.execute("delete from contact where num=1")
    con.commit()
    print('삭제 성공')
except Exception as err:
    print(err)
finally:
    con.close()
```

Sqlite 연동

Sqlite 연동

- sqlite3 모듈은 파이썬 표준 라이브러리(파이썬이 설치될 때 기본적으로 설치되는 모듈)로 SQLite에 대한 인터페이스를 제공합니다. 표준 라이브러리인 sqlite3 모듈을 사용하면 따로 모듈을 설치할 필요 없이 데이터베이스를 쉽게 이용할 수 있습니다.

- **sqlite3 모듈 설치 확인**

  C:₩Program Files₩Anaconda3₩Lib₩sqlite3 설치

Sqlite 연동

- **makedb.py**

```
import sqlite3                          # SQLite3 탑재


# 테이블 생성용 함수
def create_table():
    conn = sqlite3.connect('naverDB')  # 데이터베이스 커넥션 생성

    cur = conn.cursor()                 # 커서 확보

    # usertable 테이블 생성
    cur.execute('''create table user(
                        id char(20),
                        username char(20),
                        email char(20),
                        birth char(20) )'''
                )

    conn.commit()                       # 데이터베이스 반영

    conn.close()                        # 커넥션 닫기


if __name__ == "__main__":              # 외부에서 호출 시
    create_table()                      # 테이블 생성 함수 호출
```

Sqlite 연동

- insert.py

```
# user 테이블에 데이터 입력

import sqlite3

con = sqlite3.connect('naverDB')

cursor = con.cursor()

while True:
    data1 = input('사용자 ID ?')
    if data1 =='':
        break
    data2 = input('사용자 이름 ?')
    data3 = input('사용자 이메일 ?')
    data4 = input('사용자 출생연도 ?')

    sql="insert into user values('"+data1+"','"+data2+"','"+data3+"','"+data4+"')"

    cursor.execute(sql)

con.commit()
con.close()
```

Sqlite 연동

- select.py

```
# user 테이블 데이터 검색

import sqlite3

con = sqlite3.connect('naverDB')

cursor = con.cursor()

cursor.execute('select * from user')

print('사용자ID ₩t 사용자이름 ₩t 이메일 ₩t 출생연도')
print('-----------------------------------------------')

rows = cursor.fetchall()              # 모든 데이터 검색 , list
print(rows)

for r in rows:
    print(r[0],r[1],r[2],r[3])
```

Sqlite 연동

- update.py

```
# user 테이블 데이터 수정

import sqlite3

con = sqlite3.connect('naverDB')

cursor = con.cursor()

sql = "update user set email='toto@daum.net' where id='toto' "
cursor.execute(sql)
con.commit()
print('수정 성공')

con.close()
```

Sqlite 연동

- delete.py

```
# user 테이블 데이터 삭제

import sqlite3

con = sqlite3.connect('naverDB')

cursor = con.cursor()

cursor.execute("delete from user where id='toto' ")
con.commit()
print('삭제 성공')

con.close()
```

Oracle 연동

Oracle 연동

- Oracle 모듈 설치 (방법1)
  > pip install cx_Oracle

- 모듈 설치 확인
  > pip list

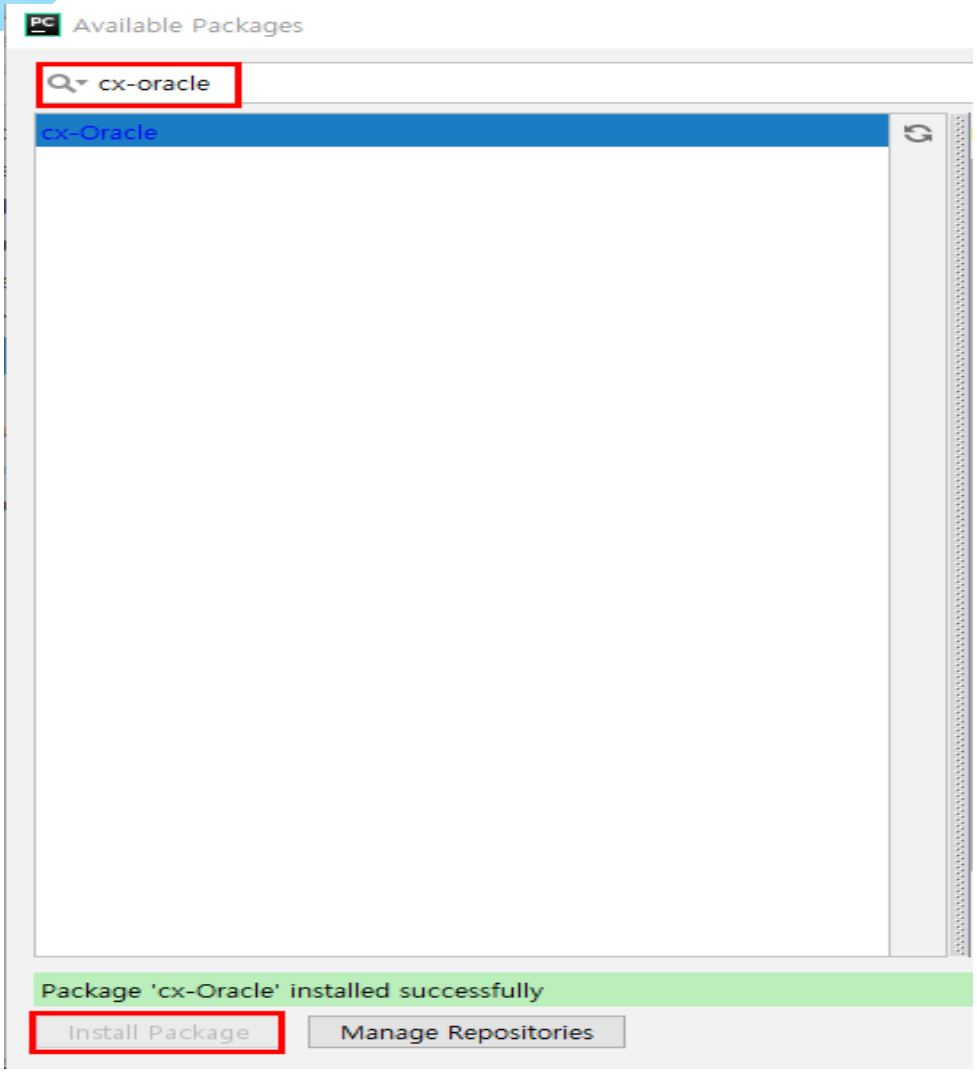
- Oracle 모듈 다운로드 및 설치 (방법2)

  cx-Oracle 5.3
  https://pypi.org/project/cx-Oracle/5.3/#files

  cx-Oracle 7.1
  https://pypi.org/project/cx-Oracle/#files

Oracle 연동

- Oracle 모듈 설치 (방법3)



Oracle 연동

- python – oracle 연동 테스트
- oracletest.py

```
import cx_Oracle

# con = cx_Oracle.connect('아이디/암호@아이피:1521/인스턴스명')
# con = cx_Oracle.connect('scott/tiger@localhost:1521/xe')

dsn = cx_Oracle.makedsn("localhost", 1521, "xe")
con = cx_Oracle.connect("scott", "tiger", dsn)
cursor = con.cursor()

cursor.execute("select * from dept")
row = cursor.fetchone()                    # 데이터 1개 검색     tuple
# row = cursor.fetchall()                  # 모든 데이터 검색    list
print(row)

cursor.close()
con.close()
```

Oracle 연동

- 데이터 입력
- insert.py

```
# 부서테이블(DEPT) 테이블에 데이터 입력

import cx_Oracle

try:
    con = cx_Oracle.connect('scott/tiger@localhost:1521/xe')

    cursor = con.cursor()

    cursor.execute("insert into dept values(50,'개발부','서울')")
    con.commit()

    print('데이터 입력 성공')
except Exception as err:
    print(err)
finally:
    con.close()
```

Oracle 연동

- 데이터 입력
- insert1.py

```
# 부서테이블(DEPT) 테이블에 데이터 입력
# 사용자가 키보드로 입력한 정보를 DEPT 테이블에 입력

import cx_Oracle

no = int(input('부서번호를 입력하세요?'))
name = input('부서명을 입력하세요?')
local = input('지역명을 입력하세요?')

try:
    con = cx_Oracle.connect('scott/tiger@localhost:1521/xe')

    cursor = con.cursor()

    sql = "insert into dept(deptno,dname,loc) values(:no, :name, :local)"
    cursor.execute(sql, no=no, name=name, local=local)
    con.commit()
    print('데이터 입력성공')

except Exception as err:
    print(err)
finally:
    con.close()
```

Oracle 연동

- 데이터 1개 검색
- selectone.py

```
import cx_Oracle

try:
    con = cx_Oracle.connect('scott/tiger@localhost:1521/xe')

    cursor = con.cursor()

    cursor.execute('select * from dept')

    row = cursor.fetchone()              # 1개의 데이터 구해옴
    print(type(row))                     # 'tuple'
    print(row)

except Exception as err:
    print(err)

finally:
    con.close()
```

Oracle 연동

- 모든 데이터 검색
- selectall.py

```
# 부서 테이블(DEPT)테이블 데이터 검색

import cx_Oracle

try:
    con = cx_Oracle.connect('scott/tiger@localhost:1521/xe')

    cursor=con.cursor()

    cursor.execute('select * from dept order by deptno')
    rows = cursor.fetchall()                    # 모든 데이터를 구해옴
    print(type(rows))                           # 'list'
    print(rows)

    for r in rows:
        print(r[0], r[1], r[2])

except Exception as err:
    print(err)
finally:
    con.close()
```

Oracle 연동

- 데이터 수정
- update.py

```
# 부서테이블(DEPT) 테이블 수정

import cx_Oracle

try:
    con = cx_Oracle.connect('scott/tiger@localhost:1521/xe')

    cursor = con.cursor()

    cursor.execute("update dept set loc='SEOUL' where deptno=40")
    con.commit()
    print('수정 성공')

except Exception as err:
    print(err)
finally:
    con.close()
```

Oracle 연동

- 데이터 삭제
- delete.py

```
# 부서테이블(DEPT) 테이블 데이터 삭제

import cx_Oracle

try:
    con = cx_Oracle.connect('scott/tiger@localhost:1521/xe')

    cursor = con.cursor()

    cursor.execute("delete from dept where deptno = 50")
    con.commit()
    print('삭제 성공')

except Exception as err:
    print(err)
finally:
    con.close()
```

MongoDB

MongoDB 환경 구축

1. MongoDB 다운로드

   http://www.mongodb.org/downloads

   MongoDB를 다운로드 받는다.

2. MongoDB 설치 및 PATH 설정

   다운로드 받은 MongoDB 프로그램 설치하면 아래 위치에 설치된다.
   C:₩Program Files₩MongoDB₩Server₩3.0₩bin

   환경변수 등록
   [제어판]-[시스템 및 보안]-[시스템]-[고급 시스템 설정]으로 이동
   [시스템 속성]에서 [환경변수] 버튼 클릭
   [시스템 변수]-[Path] 변수 더블클릭후
   기존 Path의 마지막 부분에 추가
   ;C:₩Program Files₩MongoDB₩Server₩3.0₩bin

MongoDB 환경 구축

3. 데이터베이스가 생성될 물리적인 경로 생성

```
c:₩> mkdir  c:₩mongodb₩test
```

4. MongoDB 인스턴스 활성화

```
c:₩> mongod  --dbpath  c:₩mongodb₩test
```

cf. 이 메세지들은 MongoDB를 위한 기본적인 메모리 영역을 활성화하고 관련 데이터 파일의 무결성을 확인하는 절차입니다.
이 단계에서 어떤 장애 메세지가 발생하거나 프롬프트가 떨어지지 않으면 정상적인 시작이 실패한 것을 의미합니다.

MongoDB 환경 구축

5. 다음은 활성화된 MongoDB에 클라이언트 프로그램인 mongo.exe를 통해서 접속

```
c:₩> mongo                  <-- MongoDB에 접속하는 Client Shell Program

>                           <-- 에러없이 Prompt가 나타나면 정상 접속된 상태

> help                      <-- Mongo Shell 상태에서 실행할 수 있는 명령어 help 기능

> show dbs                  <-- 데이터베이스 목록출력
  local                     <-- 현재 기본 local 데이터베이스가 생성되어 있음.

> use test                  <-- test 데이터베이스로 이동할때 사용하는 명령어
                                만약 test 데이터베이스가 존재하지 않는 경우에는 첫번째
                                컬렉션(=테이블)을 생성할 때 자동으로 test테이블이 생성됨

> db.stats()                <-- 현재 데이터베이스에 대한 정보를 보여줌.

> show collections          <-- Collection(=테이블) 목록 출력

> db.collection명.drop()    <-- Collection 삭제

> db.logout()               <-- 접속만 해제된 상태(정상처리:1, 실패:0)

> exit                      <-- 종료
```

MongoDB 환경 구축

- RDBMS와 MongoDB 용어 비교

RDBMS	MongoDB
Database	Database
table	collection
row	document
column	field
primary key	object_id field

RDBMS와 MongoDB의 SQL문 비교

❖ **RDBMS와 MongoDB의 SQL문 비교**

1. DDL문(create, alter, drop)

1) create table member(id varchar2(30), age number, type varchar2(10),
primary key(id));

db.member.insert({id:"toto", age:25, type:"ace"})
또는 db.createCollection("member")

2) drop table member;

db.member.drop()

2. insert문

insert into member(id, age, type) values('toto', 30, 'gold');

db.member.insert({id : "toto", age : 30, type : "y"})

RDBMS와 MongoDB의 SQL문 비교

3. select문

1) select * from member;

db.member.find()

2) select * from member where type = 'y';

db.member.find({type:"y"})

3) select * from member where type != 'y';

db.member.find({type: {$ne:"y"} })

4) select * from member where age = 25 and type = 'y';

db.member.find({age:25, type:"y"})

RDBMS와 MongoDB의 SQL문 비교

4. update문

```
update member set age = age + 5  where type = 'y';
```

```
db.member.update( {type : 'y'}, {$inc: {age:5}},  multi = true )
```

5. delete문

```
delete from member where type = 'y';
```

```
db.member.remove({type : "y"})
```

MongoDB

- MongoDB 모듈 설치 (방법 1)

  > pip install pymongo

  > conda install pymongo

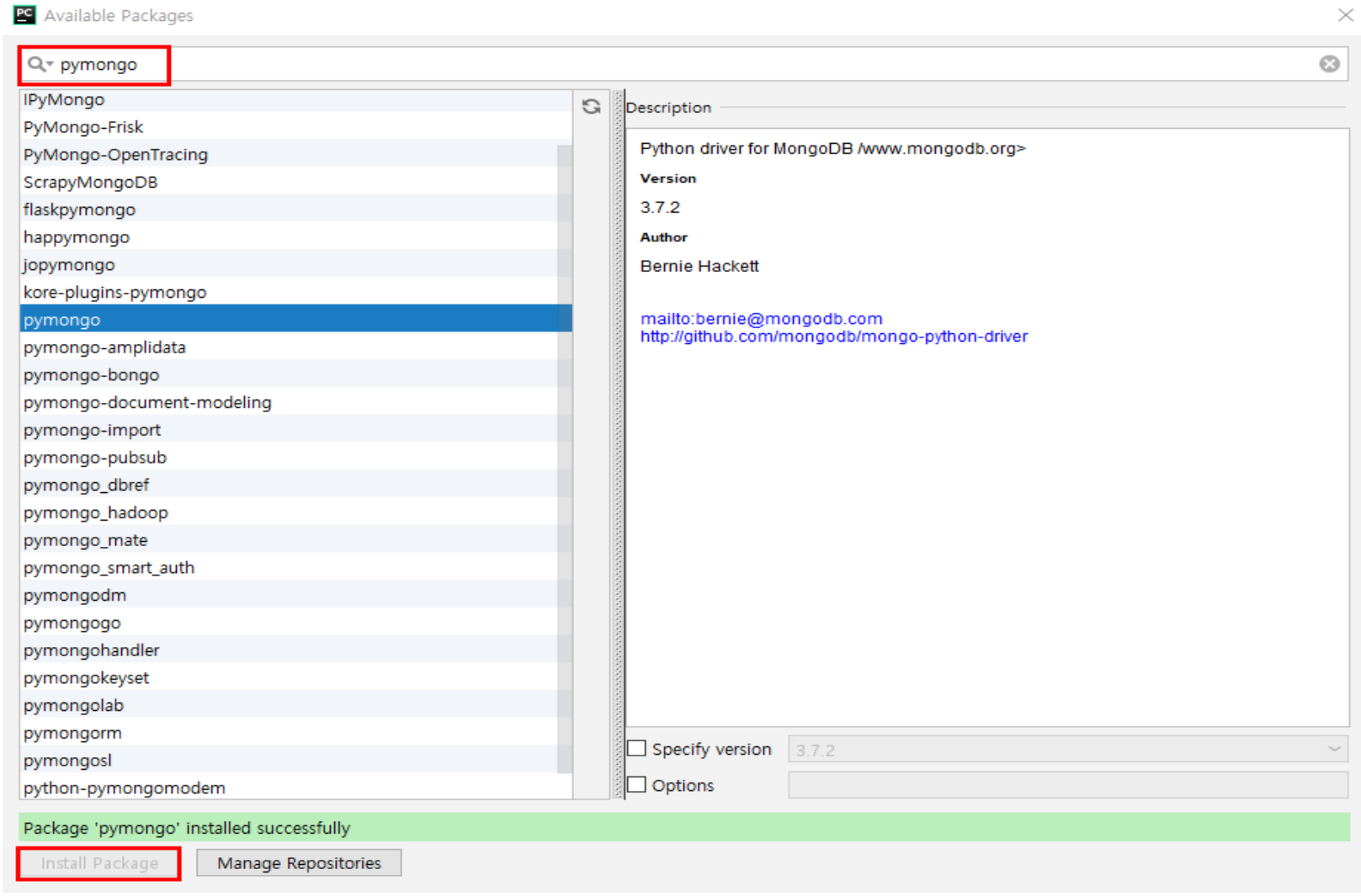
  pip나 conda 명령으로 pymongo 모듈을 설치 한다.

- 모듈 설치 확인

  > pip list

MongoDB

- MongoDB 모듈 설치 (방법 2)



MongoDB

❖ Python – MongoDB 연동 예제

mongodb_connect.py

```
import pymongo

conn = pymongo.MongoClient("localhost", 27017)

# db 생성 : testdb
db = conn.testdb

# collection 생성 : collect
collect = db.collect

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

MongoDB

```
# collection에 문서 추가
db.collect.insert(doc1)
db.collect.insert(doc2)
db.collect.insert(doc3)

# 전체 문서 조회
print('전체 문서 조회')
result1 = db.collect.find()
for r in result1:
    print(r)

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

MongoDB

```
# 문서 수정
# empno 가 10001인 경우에 name을 kim 으로 수정
db.collect.update( {'empno':'10001'}, {'$set':{'name':'kim'}}, multi=True)

# 전체 문서 조회(수정)
print('문서 수정 결과')
result3 = db.collect.find()
for r in result3:
    print(r)
```

MongoDB

```
# 문서 삭제
db.collect.remove({'empno' : '10002'})

# 전체 문서 조회(삭제)
print('문서 삭제 결과')
result4 = db.collect.find()
for r in result4:
    print(r)

# 컬렉션 제거
db.collect.drop()
```

MongoDB

- Python – MongoDB 연동 결과