Python and DB Applications

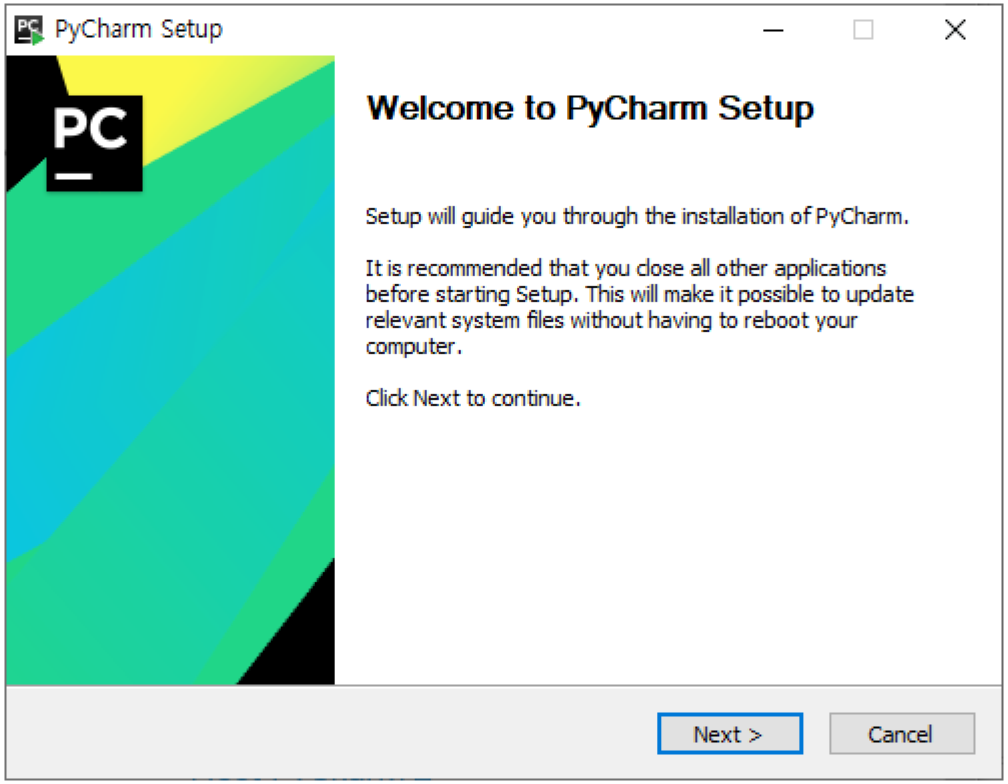
1. PyMySQL: Python DB API

Hyeokman Kim
School of Computer Science
Kookmin Univ.

Python and DB Applications

INSTALLATION

PyCharm

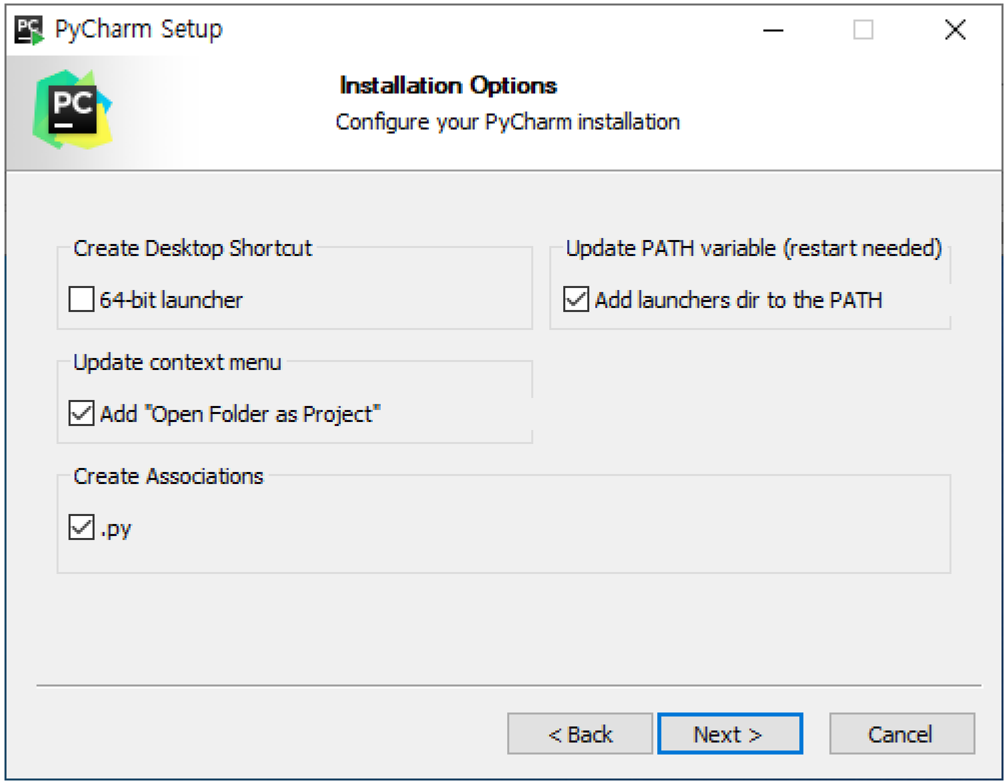


Welcome to PyCharm Setup

Setup will guide you through the installation of PyCharm.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.


PyCharm Setup

Installation Options

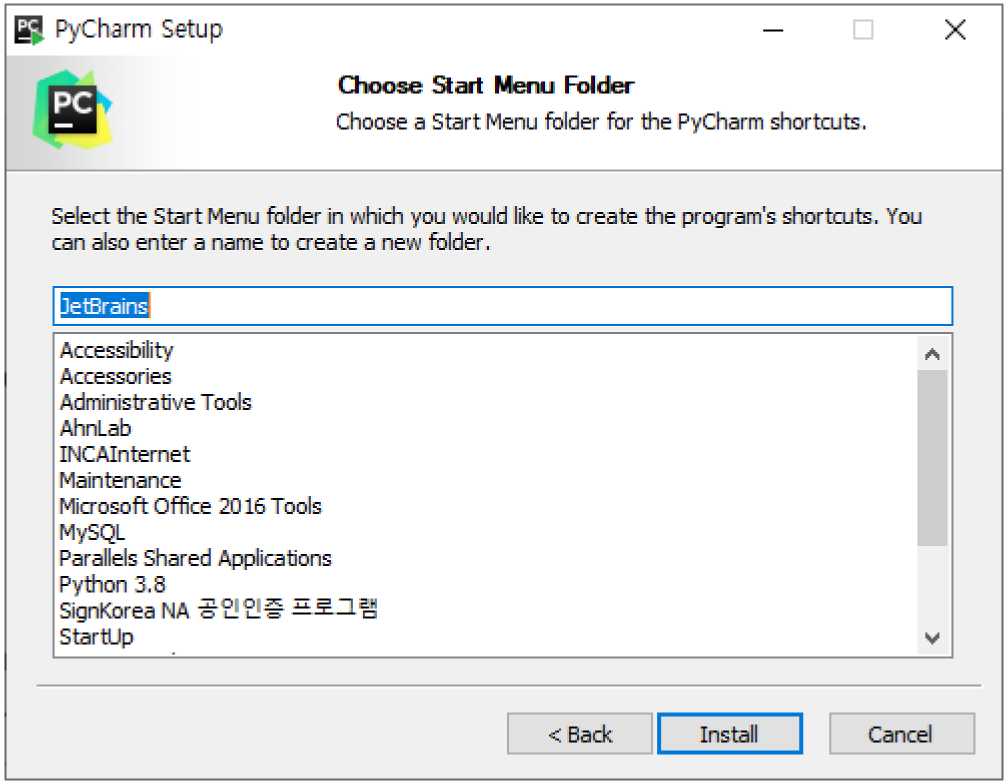
Configure your PyCharm installation

Create Desktop Shortcut
- [ ] 64-bit launcher

Update PATH variable (restart needed)
- [✓] Add launchers dir to the PATH

Update context menu
- [✓] Add "Open Folder as Project"

Create Associations
- [✓] .py


PyCharm Setup

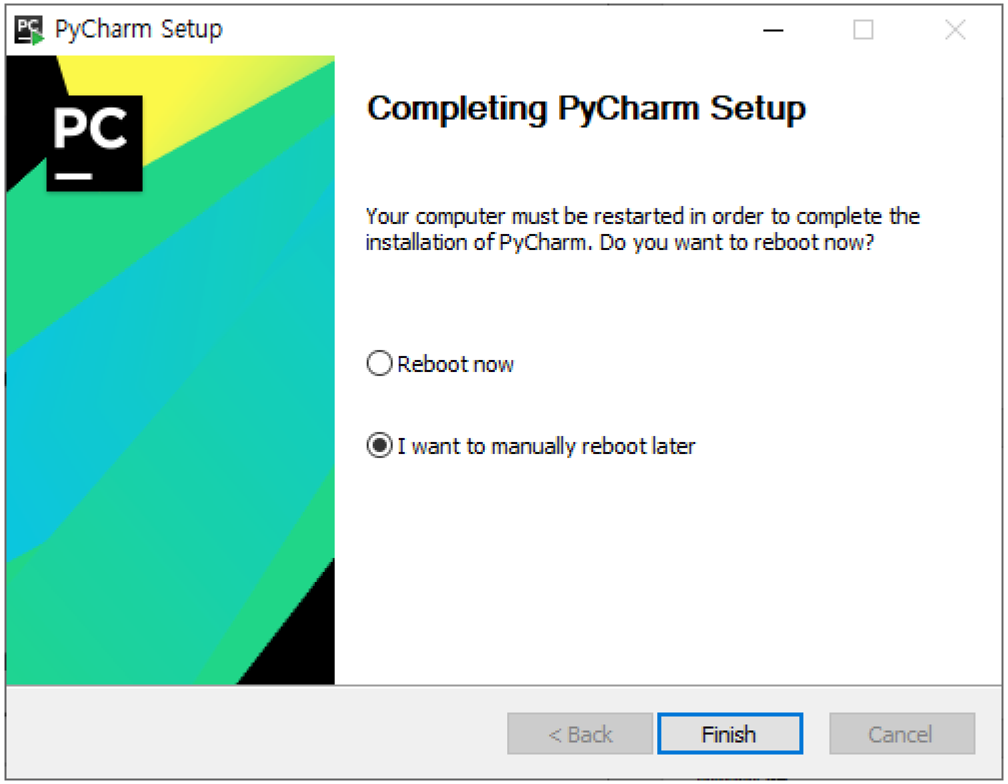
Choose Start Menu Folder

Choose a Start Menu folder for the PyCharm shortcuts.

Select the Start Menu folder in which you would like to create the program's shortcuts. You can also enter a name to create a new folder.

JetBrains

Accessibility
Accessories
Administrative Tools
AhnLab
INCAInternet
Maintenance
Microsoft Office 2016 Tools
MySQL
Parallels Shared Applications
Python 3.8
SignKorea NA 공인인증 프로그램
StartUp



PyCharm Setup

Completing PyCharm Setup

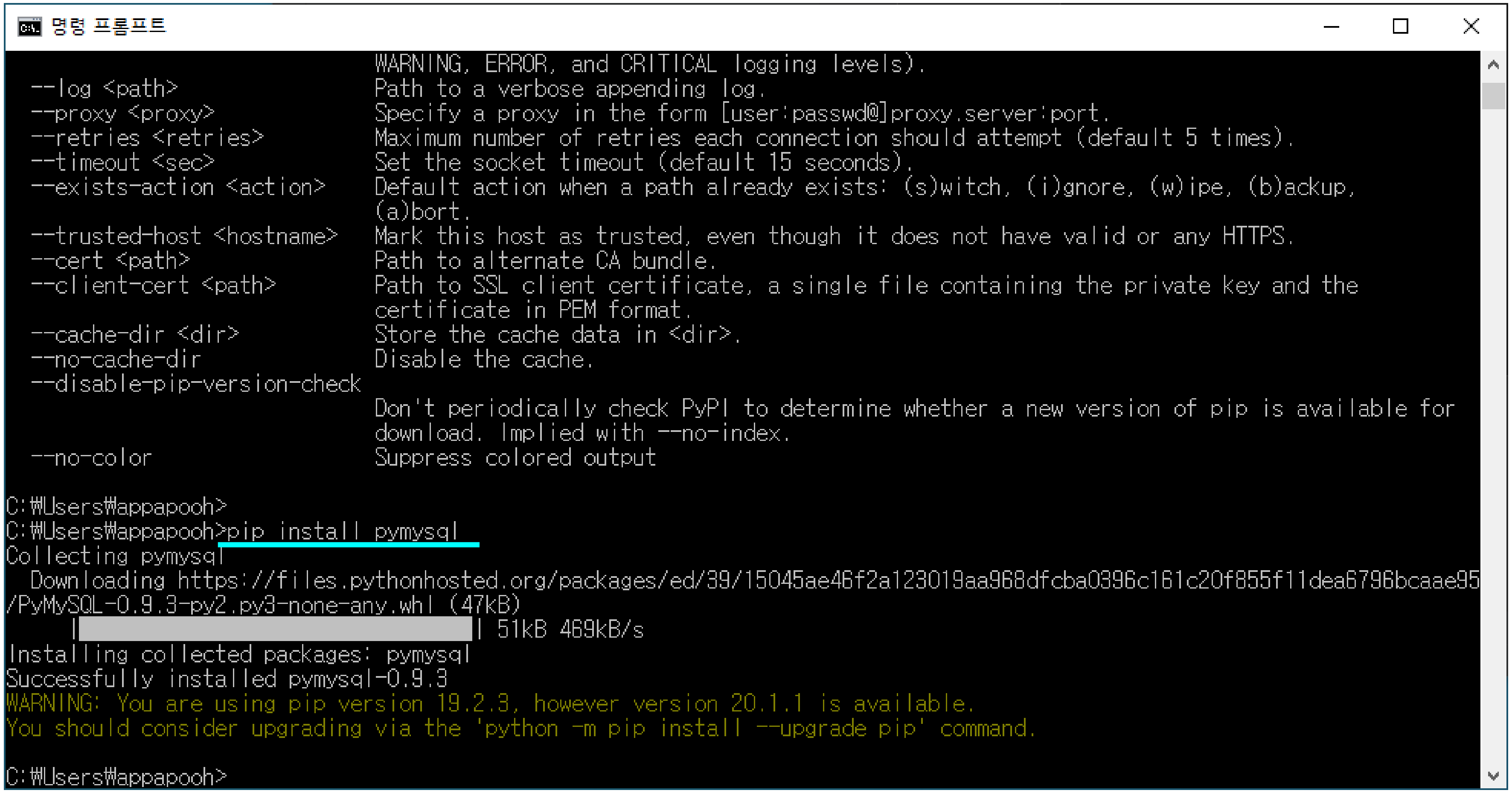
Your computer must be restarted in order to complete the installation of PyCharm. Do you want to reboot now?

Reboot now [✓ I want to manually reboot later]

PyMySql – Windows에 설치

- 홈페이지
  - https://github.com/PyMySQL/PyMySQL
- 설치 절차: 커맨드 창을 실행하여 다음 명령어를 실행
  - (python이 설치된 폴더 하위의 Scripts 폴더로 이동)
    - path 명령으로 python 설치 폴더가 path 변수에 포함되어 있는지 확인함.
  - pip install pymysql 명령어 실행
    - 다운로드가 자동으로 진행됨.
  - 설치 완료 확인
- PyCharm에 PyMySql 모듈 연결

ㅇ 설치

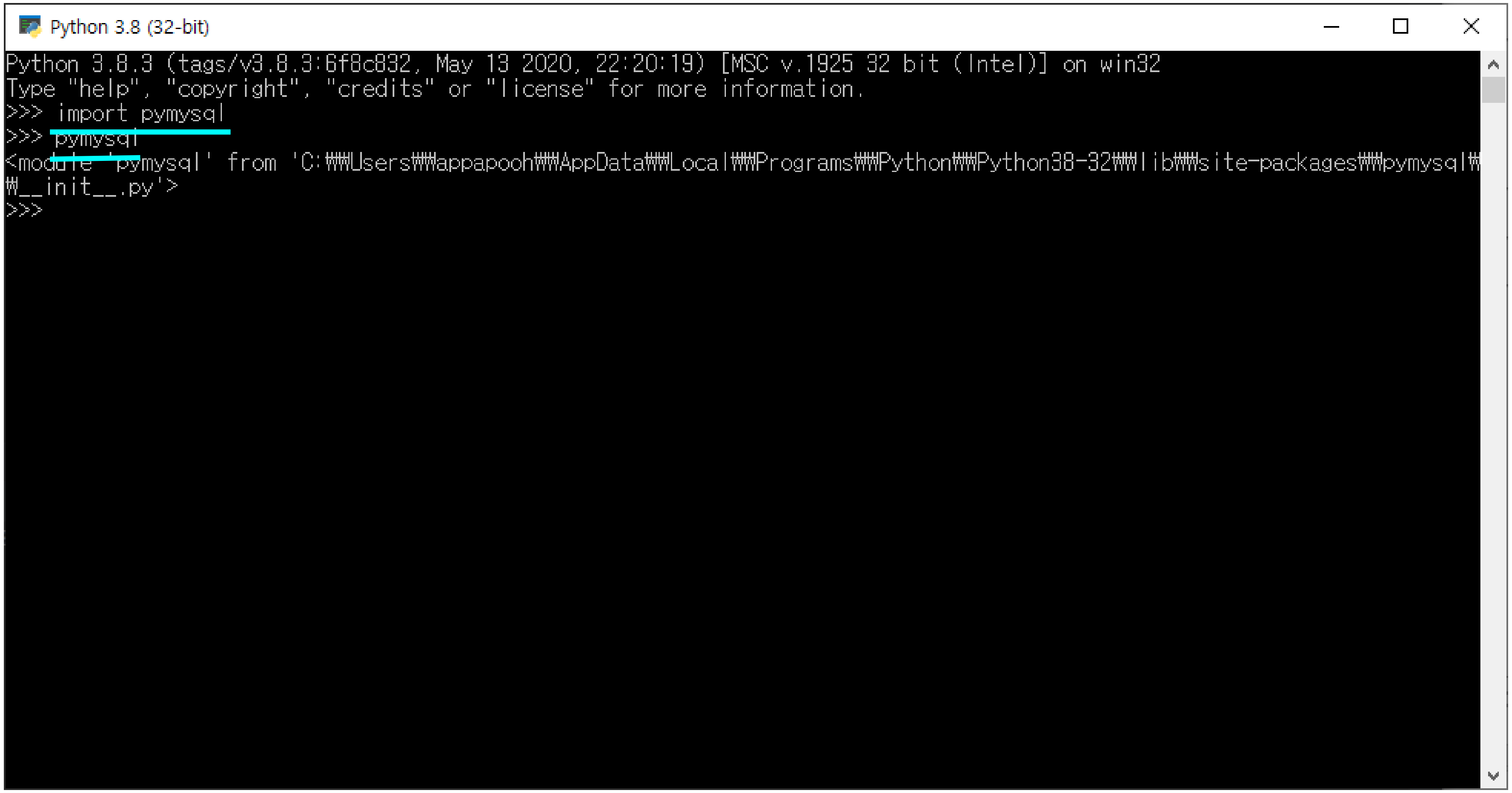


```
                              WARNING, ERROR, and CRITICAL logging levels).
  --log <path>                Path to a verbose appending log.
  --proxy <proxy>             Specify a proxy in the form [user:passwd@]proxy.server:port.
  --retries <retries>         Maximum number of retries each connection should attempt (default 5 times).
  --timeout <sec>             Set the socket timeout (default 15 seconds).
  --exists-action <action>    Default action when a path already exists: (s)witch, (i)gnore, (w)ipe, (b)ackup,
                              (a)bort.
  --trusted-host <hostname>   Mark this host as trusted, even though it does not have valid or any HTTPS.
  --cert <path>               Path to alternate CA bundle.
  --client-cert <path>        Path to SSL client certificate, a single file containing the private key and the
                              certificate in PEM format.
  --cache-dir <dir>           Store the cache data in <dir>.
  --no-cache-dir              Disable the cache.
  --disable-pip-version-check
                              Don't periodically check PyPI to determine whether a new version of pip is available for
                              download. Implied with --no-index.
  --no-color                  Suppress colored output

C:₩Users₩appapooh>
C:₩Users₩appapooh>pip install pymysql
Collecting pymysql
  Downloading https://files.pythonhosted.org/packages/ed/39/15045ae46f2a123019aa968dfcba0396c161c20f855f11dea6796bcaae95
/PyMySQL-0.9.3-py2.py3-none-any.whl (47kB)
     |████████████████████████████████| 51kB 469kB/s
Installing collected packages: pymysql
Successfully installed pymysql-0.9.3
WARNING: You are using pip version 19.2.3, however version 20.1.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

C:₩Users₩appapooh>
```

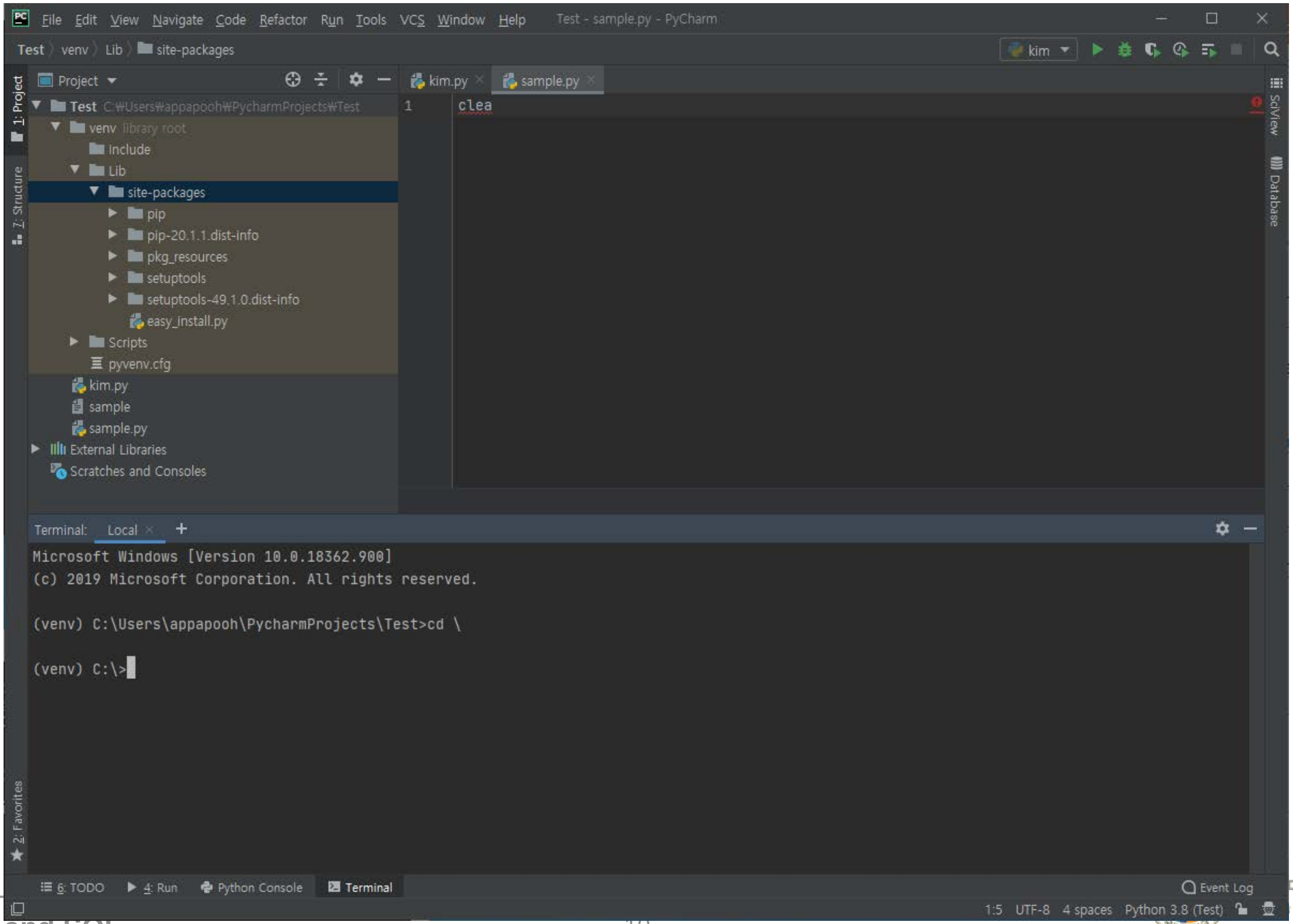
ㅇ 설치 확인


Python 3.8 (32-bit)

```
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:20:19) [MSC v.1925 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import pymysql
>>> pymysql
<module 'pymysql' from 'C:₩₩Users₩₩appapooh₩₩AppData₩₩Local₩₩Programs₩₩Python₩₩Python38-32₩₩lib₩₩site-packages₩₩pymysql₩
₩__init__.py'>
>>>
```

PyMySql – PyCharm의 Virtual Environment에 설치

o 설치 전

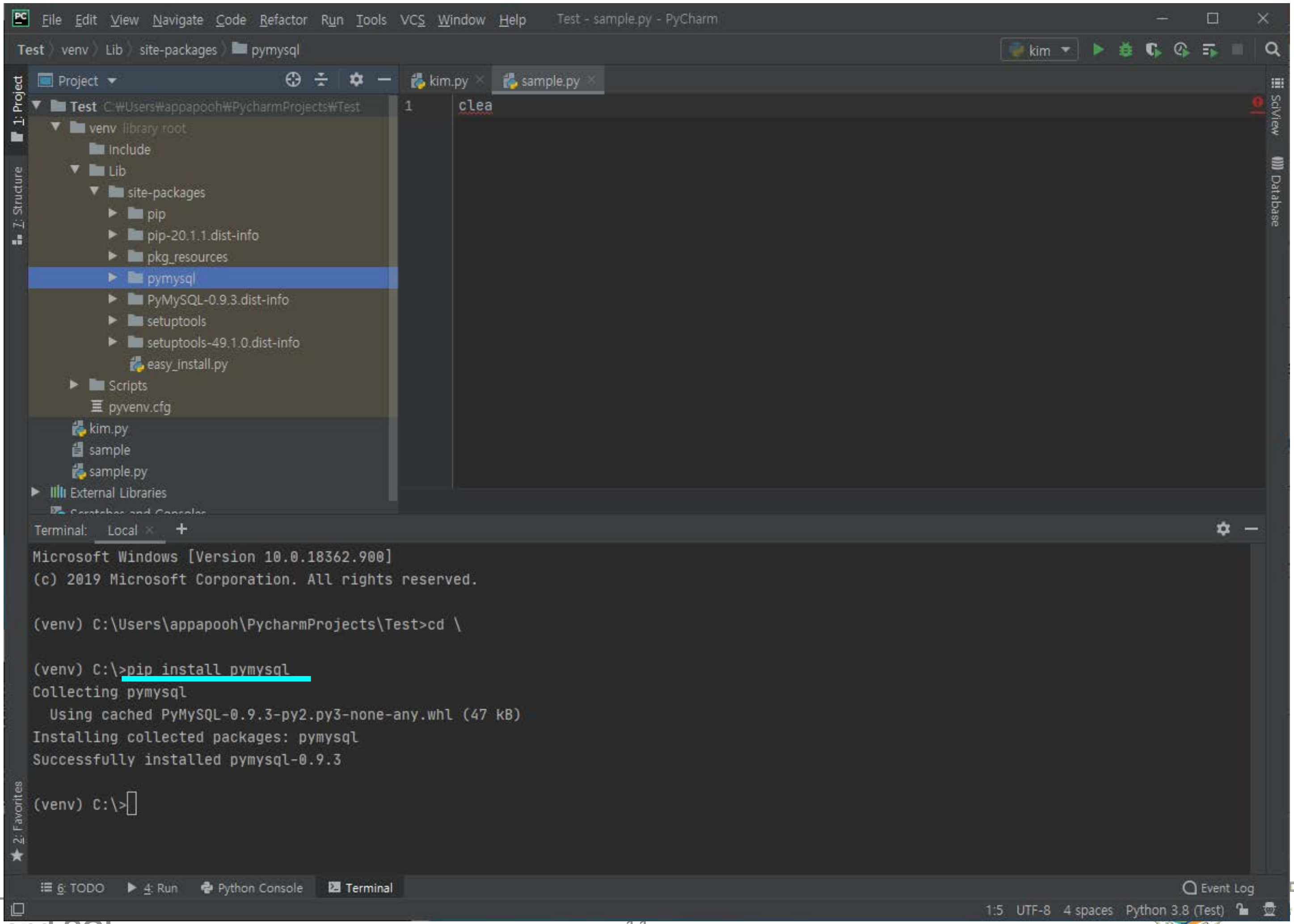


```
Microsoft Windows [Version 10.0.18362.900]
(c) 2019 Microsoft Corporation. All rights reserved.

(venv) C:\Users\appapooh\PycharmProjects\Test>cd \

(venv) C:\>
```

ㅇ 설치



```
clea
```

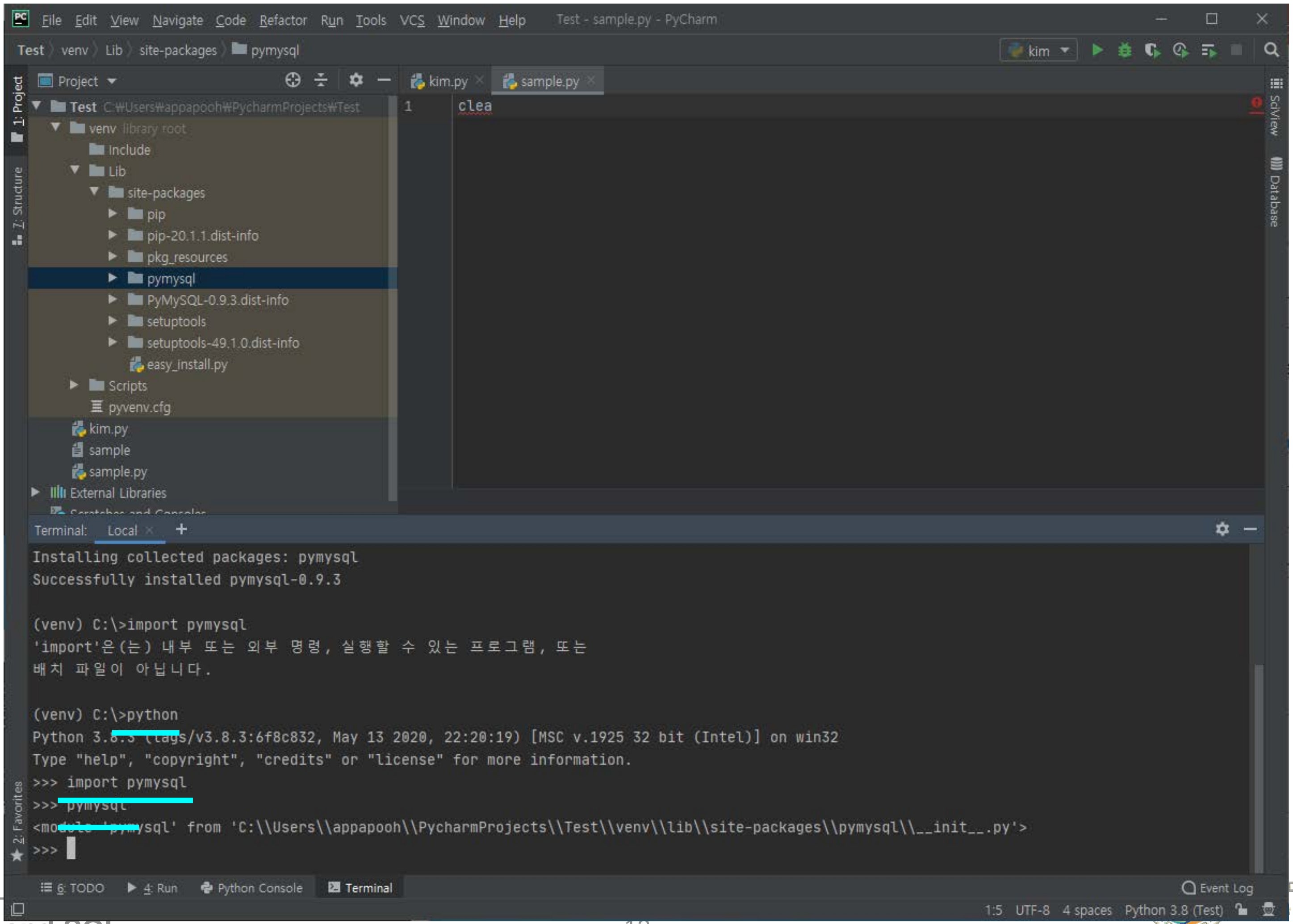
```
Microsoft Windows [Version 10.0.18362.900]
(c) 2019 Microsoft Corporation. All rights reserved.

(venv) C:\Users\appapooh\PycharmProjects\Test>cd \

(venv) C:\>pip install pymysql
Collecting pymysql
  Using cached PyMySQL-0.9.3-py2.py3-none-any.whl (47 kB)
Installing collected packages: pymysql
Successfully installed pymysql-0.9.3

(venv) C:\>
```

ㅇ 설치 확인



```
clea
```

```
Installing collected packages: pymysql
Successfully installed pymysql-0.9.3

(venv) C:\>import pymysql
'import'은(는) 내부 또는 외부 명령, 실행할 수 있는 프로그램, 또는
배치 파일이 아닙니다.

(venv) C:\>python
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:20:19) [MSC v.1925 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import pymysql
>>> pymysql
<module 'pymysql' from 'C:\\Users\\appapooh\\PycharmProjects\\Test\\venv\\lib\\site-packages\\pymysql\\__init__.py'>
>>> 
```

Python and DB Applications

PYMYSQL : PYTHON DB API

PyMySQL & API Reference

- PyMySQL은 PSF (Python Software Foundation) 권고안 PEP 249를 구현한 코드임.
  - PEP 249: Python Database API Specification V2.0
  - https://www.python.org/dev/peps/pep-0249/
  - ☞ PEP (Python Enhancement Proposal)

- PyMySQL Documentation
  - https://pymysql.readthedocs.io/en/latest/
  - 구성
    - ◆ User Guide
    - ◆ API Reference

- API Reference
  - Connection object : DBMS와 연결에 필요한 메소드를 정의
  - Cursor objects : SQL 문을 DBMS에 보내고, 결과를 리턴 받기 위한 메소드를 정의

Connection Object

```
class pymysql.connections.Connection(host=None, user=None,
    password='', database=None, port=0, unix_socket=None,
    charset='', sql_mode=None, read_default_file=None, conv=None,
    use_unicode=None, client_flag=0,
    cursorclass=<class 'pymysql.cursors.Cursor'>, init_command=None,
    connect_timeout=10, ssl=None, read_default_group=None,
    compress=None, named_pipe=None, autocommit=False, db=None,
    passwd=None, local_infile=False, max_allowed_packet=16777216,
    defer_connect=False, auth_plugin_map=None, read_timeout=None,
    write_timeout=None, bind_address=None, binary_prefix=False,
    program_name=None, server_public_key=None)

    begin():                 Begin transaction.
    close():                 Send the quit message and close the socket.
    commit():                Commit changes to stable storage.
    rollback():              Roll back the current transaction.
    cursor(cursor=None): Create a new cursor to execute
                             queries with
    open():                  Return True if the connection is open.
    ping(reconnect=True): Check if the server is alive.
    select_db(db):           Set current db.
    show_warning():          Send the “SHOW WARNINGS” SQL command.
```

Cursor Objects

- Cursor 객체
  - 일반 커서

```
class pymysql.cursors.Cursor(connection)

   max_stmt_length=1024000

   execute(query, args=None):    Execute a query.
   executemany(query, args):     Run several data against one query.
   fetchone():                   Fetch the next row.
   fetchmany(size=None):         Fetch several rows.
   fetchall():                   Fetch all the rows.
   callproc(procname, args=()):  Execute stored procedure with args.
   mogrify(query, args=None):    Returns the exact string that is
                                 sent to the database by calling the
                                 execute() method.
   setinputsizes(*args):         Does nothing, required by DB API.
   setoutputsizes(*args):        Does nothing, required by DB API.
   close():                      Closing a cursor just exhausts all
                                 remaining data.
```

- SSCursor 객체
  - Unbuffered cursor : 많은 데이터를 리턴할 때 사용.
  - 제약점
    - ◆ MySQL은 전체 투플 수를 제공하지 않으므로, 리턴되는 투플들을 모두 세어야만 전체 투플 수를 알 수 있음.
    - ◆ 결과에서 역방향으로 스크롤할 수 없음.

```
class pymysql.cursors.SSCursor(connection)

   fetchone():                Fetch the next row.
   fetchmany(size=None): Fetch many.
   fetchall():                Fetch all the rows.
   fetchall_unbuffered(): Fetch all, implemented as a generator,
                           which isn’t to standard, however, it doesn’t
                           make sense to return everything in a list,
                           as that would use ridiculous memory for
                           large result sets.
   read_next():               Read next row.
   close():                   Closing a cursor just exhausts all
                           remaining data.
```

- DictCursor 객체

```
class pymysql.cursors.DictCursor(connection)
```

- SSDictCursor 객체

```
class pymysql.cursors.SSDictCursor(connection)
```

Python and DB Applications

PYMYSQL 패키지의 구조

Pymysql 패키지의 __init__.py

```
class DBAPISet(frozenset):
    def __ne__(self, other):
    def __eq__(self, other):
    def __hash__(self):

def Binary(x):

def Connect(*args, **kwargs):

def get_client_info():

def thread_safe():

def install_as_MySQLdb():
```

Pymysql 패키지의 connections.py 모듈

```
class Connection(object):
    def __init__(self, host=None, user=None, password="",
    database=None, port=0, unix_socket=None, charset='',
    sql_mode=None, read_default_file=None, conv=None,
    use_unicode=None, client_flag=0, cursorclass=Cursor,
    init_command=None, connect_timeout=10, ssl=None,
    read_default_group=None, compress=None, named_pipe=None,
    autocommit=False, db=None, passwd=None, local_infile=False,
    max_allowed_packet=16*1024*1024, defer_connect=False,
    auth_plugin_map=None, read_timeout=None,
    write_timeout=None, bind_address=None, binary_prefix=False,
    program_name=None, server_public_key=None):

    def _create_ssl_ctx(self, sslp):
    def close(self):
    def open(self):
    def _force_close(self):
    def autocommit(self, value):
    def get_autocommit(self):
    def _read_ok_packet(self):
    def _send_autocommit_mode(self):
    def begin(self):
```

```
    def commit(self):
    def rollback(self):
    def show_warnings(self):
    def select_db(self, db):
    def escape(self, obj, mapping=None):
    def literal(self, obj):
    def escape_string(self, s):
    def _quote_bytes(self, s):
    def cursor(self, cursor=None):
    def __enter__(self):
    def __exit__(self, exc,value, traceback):
    def query(self, sql, unbuffered=False):
    def next_result(self, unbuffered=False):
    def affected_rows(self):
    def kill(self, thread_id):
    def ping(self, reconnect=True):
    def set_charset(self, charset):
    def connect(self, sock=None):
    def write_packet(self, payload):
    def _read_packet(self, packet_type=MysqlPacket):
    def _read_bytes(self, num_bytes):
```

```
    def _write_bytes(self, data):
    def _read_query_result(self, unbuffered=False):
    def insert_id(self):
    def _execute_command(self, command, sql):
    def _request_authentication(self):
    def _process_auth(self, plugin_name, auth_packet):
    def _get_auth_plugin_handler(self, plugin_name):
    def get thread_id(self):
    def character_set_name(self):
    def get_host_info(self):
    def get_proto_info(self):
    def get_server_information(self):
    def get_server_info(self):
```

```
class MySQLResult(object):
    def __init__(self, connection):

    def __del__(self):
    def read(self):
    def init_unbuffered_query(self):
    def _read_ok_packet(self, first_packet):
    def _read_load_local_packet(self, first_packet):
    def _check_packet_is_eof(self, packet):
    def _read_result_packet(self, first_packet):
    def _read_rowdata_packet_unbuffered(self):
    def _finish_unbuffered_query(self):
    def _read_rowdata_packet(self):
    def _read_row_from_packet(self, packet):
    def _get_descriptions(self):

class LoadLocalFile(object):
    def __init__(self, filename, connection):

    def send_data(self):
```

Pymysql 패키지의 cursors.py 모듈

```
class Cursor(object):
    def __init__(self, connection):

    def close(self):
    def __ente__(self):
    def __exit__(self, *exc_info):
    def _get_db(self):
    def _check_executed(self):
    def _conv_row(self, row):
    def setinputsizes(self, *args):
    def setoutputsizes(self, *args):
    def _nextset(self, unbuffered=False):
    def nextset(self):
    def _ensure_bytes(self, x, encoding=None):
    def _escape_args(self, args, conn):
    def mogrify(self, query, args=None):
    def execute(self, query, args=None):
    def executemany(self, query, args):
    def _do_execute_many(self, prefix, values, postfix, args,
                                  max_stmt_length, encoding):
    def callproc(self, procname, args=()):
```

```
def fetchone(self):
def fetchmany(self, size=None):
def fetchall(self):
def scroll(self, value, mode=‘relative’):
def _query(self, q):
def _clear_result(self):
def _do_get_result(self):
def _show_warning(self):
def __iter__(self):
```

```
class DictCursorMixin(object):
    def _do_get_result(self):
    def _conv_row(self, row):

class DictCursor(DictCursorMixin, Cursor):

class SSCursor(Cursor):
    def _conv_row(self, row):
    def close(self):
    def _query(self, q):
    def nextset(self):
    def read_next(self):
    def fetchone(self):
    def fetchall(self):
    def fetchall_unbuffered(self):
    def __iter__(self):
    def fetchmany(self, size=None):
    def scroll(self, value, mode='relative'):

Class SSDictCursor(DictCursorMixin, SSCursor):
```

Python and DB Applications

PYMYSQL 모듈 사용법

MySQL 모듈의 사용 절차

- MySQL 사용 절차
  - PyMySql 모듈을 import 한다.
  - pymysql 모듈의 connect() 함수를 사용하여, MySQL에 연결한다.
  - connection 객체로부터 cursor() 메서드를 호출하여, cursor 객체를 가져온다.
  - cursor 객체의 execute() 메서드를 사용하여, SQL 문장을 DB 서버에 보낸다.
    - ◆ 검색문의 경우, cursor 객체의 fetchall(), fetchone(), fetchmany() 등의 메서드를 사용하여 데이터를 DB 서버로부터 가져온다.
    - ◆ 갱신문의 경우, INSERT/DELETE/UPDATE 후 connection 객체의 commit() 메소드를 사용하여 데이터 갱신을 확정한다.
  - connection 객체의 close() 메서드를 사용하여 MySQL 연결을 해제한다.

pymysql 모듈

- pymysql.connect() 함수
  - 호스트명, 로그인, 암호, 접속할 DB, 문자셀 등을 파라미터로 지정함.
  - 한글 깨지는 문제를 방지하려면 charset=‘utf8’으로 지정함.

connection 객체

- connection.cursor() 메소드
  - DB 커서는 fetch 동작을 관리하는데 사용함.
  - Cursor의 종류
    - ◆ Array based cursor : 질의의 결과를 tuple 타입의 리스트로 리턴함.
      - 컬럼 인덱스로 컬럼을 지정함.
      - 디폴트 값
    - ◆ Dictionary based cursor : 질의의 결과를 dictionary 타입의 리스트로 리턴함.
      - 컬럼명으로 컬럼을 지정함.
- connection.commit() 메소드
- connection.close() 메소드

cursor 객체

- Execute 관련 메소드
  - SQL 문을 클라이언트에서 DB 서버로 전송함.
  - cursor.execute()
  - cursor.executemany()

- Fetch 관련 메소드
  - SQL문 실행 결과를 DB 서버에서 클라이언트로 가져옴.
  - cursor.fetchone() : 한번 호출에 하나의 투플만 가져옴.
  - cursor.fetchmany(n) : 한번 호출에 n개의 투플을 가져옴.
  - cursor.fetchall() : 모든 투플을 한번에 가져옴.

Dynamic SQL

- Dynamic SQL
  - SQL 문에 동적으로 컬럼 데이터를 넣어야 하는 경우

- Parameter placeholder, %s
  - 숫자 혹은 문자열에 관계 없이, 변수명으로 %s를 사용함.
  - SQL 문의 해당 컬럼 데이터에 %s를 사용하고, cursor 객체의 execute() 메소드의 첫번째 파라미터에 SQL 문, 두번째 파라미터에 컬럼 데이터를 넣어 줌.

  ☞ Note

  - ◆ %s를 대체할 컬럼 데이터 안에 단일 인용부호가 있는 경우 SQL Syntax 에러를 유발함.

☞ Note

- SQL injection 공격에 노출되지 않도록 주의해야함.

예제: 검색문 실행 (DBAPI_1.1.py)

```
import pymysql

conn = pymysql.connect(host='localhost', user='guest',
            password='bemyguest', db='kleague', charset='utf8')

cursor = conn.cursor()     # tuple based cursor

sql = "SELECT * FROM player“
cursor.execute(sql)

tuples = cursor.fetchall()      # 튜플 타입의 리스트
print(tuples)
print(len(tuples))

print(len(tuples[0]))
# ('2000001', '김태호', 'K10', None, None, None, 'DF', None,
#   None, datetime.date(1971, 1, 29), '1', None, None)
```

```
for rowIDX in range(len(tuples)):
    for columnIDX in range(len(tuples[0])):
        print(tuples[rowIDX][columnIDX], end=' ')
    print('')

conn.close()
```

예제: 갱신문 하나씩 실행 (DBAPI_1.2.py)

```
import pymysql

conn = pymysql.connect(host='localhost', user='guest',
            password='bemyguest', db='kleague', charset='utf8')

cursor = conn.cursor()     # tuple based cursor

sql = "INSERT INTO player(player_id, player_name, team_id, position)
            VALUES (%s, %s, %s, %s)"
cursor.execute(sql, ('2020001', '손흥민', 'K01', 'FW'))
cursor.execute(sql, ('2020002', '호날두', 'K02', 'FW'))
conn.commit()

sql = "SELECT * FROM player"
cursor.execute(sql)
tuples = cursor.fetchall()
print(tuples)
print(len(tuples))
```

```python
sql = "DELETE FROM player WHERE player_id = %s"
cursor.execute(sql, '2020001')
cursor.execute(sql, '2020002')
conn.commit()

sql = "SELECT * FROM player"
cursor.execute(sql)
tuples = cursor.fetchall()
print(tuples)
print(len(tuples))

conn.close()
```

예제: 갱신문 n번 실행 (DBAPI_1.3.py)

```
import pymysql

conn = pymysql.connect(host='localhost', user='guest',
            password='bemyguest', db='kleague', charset='utf8')

cursor = conn.cursor()     # tuple based cursor

newPlayers = (
    ('2020001', '손홍민', 'K01', 'FW'),
    ('2020002', '호날두', 'K02', 'FW'),
)
sql = "INSERT INTO player(player_id, player_name, team_id, position)
                VALUES (%s, %s, %s, %s)"
cursor.executemany(sql, newPlayers)
conn.commit()

sql = "SELECT * FROM player"
cursor.execute(sql)
tuples = cursor.fetchall()
print(tuples)
print(len(tuples))

conn.close()
```

예제: Dictionary Based Cursor (DBAPI_2.py)

```
import pymysql

conn = pymysql.connect(host='localhost', user='guest',
password='bemyguest', db='kleague', charset='utf8')

# dictionary based cursor
cursor = conn.cursor(pymysql.cursors.DictCursor)

sql = "SELECT * FROM player"
cursor.execute(sql)

tuples = cursor.fetchall()      # 딕셔너리 타입의 리스트
print(tuples)
print(len(tuples))

print(tuples[0])
# {'PLAYER_ID': '2000001', 'PLAYER_NAME': '김태호',
#  'TEAM_ID': 'K10', 'E_PLAYER_NAME': None, 'NICKNAME': None,
#  'JOIN_YYYY': None, 'POSITION': 'DF', 'BACK_NO': None,
#  'NATION': None, 'BIRTH_DATE': datetime.date(1971, 1, 29),
#  'SOLAR': '1', 'HEIGHT': None, 'WEIGHT': None}
# value만 출력할 때
columnNames = list(tuples[0].keys())
for tuple in tuples:
    for columnName in columnNames:
        print(tuple[columnName], end=‘  ’)
    print(‘ ’)

# key와 value를 같이 출력할 때
for tuple in tuples:
    kvlist = list(tuple.items())
    for (k, v) in kvlist:
        print(k, v, end=', ')
    print('')

conn.close()
```

예제: Connection Leak의 방지 (DBAPI_3.py)

```
import pymysql

conn = pymysql.connect(host='localhost', user='guest',
password='bemyguest', db='kleague', charset='utf8')

try:
    with conn.cursor(pymysql.cursors.DictCursor) as cursor:
        sql = "SELECT * FROM player"    # 이 문장을 코멘트 처리할 경우
        cursor.execute(sql)
        tuples = cursor.fetchall()
        print(tuples)
except Exception as e:      # 예측 불가능한 모든 에러
    print(e)
    print(type(e))
finally:
    conn.close()
```

- SQL connection을 열고 중간에 에러가 발생하는 경우가 쌓여, 나중에 새로운 connection을 오픈할 수 없는 현상

예제: Dynamic SQL (DBAPI_4.py)

```
import pymysql

conn = pymysql.connect(host='localhost', user='guest',
password='bemyguest', db='kleague', charset='utf8')

try:
    with conn.cursor() as cursor:
        sql = "SELECT * FROM %s WHERE position = %s"
        params = ('player', 'GK')
        cursor.execute(sql, params)
        tuples = cursor.fetchall()
        print(tuples)
except Exception as e
    print(e)
    print(type(e))
finally:
    conn.close()
```

예제: 기능을 클래스화 (DBAPI_5.py)

```
import pymysql

class DB_Utils:

    def queryExecutor(self, db, sql, params):
        conn = pymysql.connect(host='localhost', user='guest',
                password='bemyguest', db=db, charset='utf8')

        try:
            with conn.cursor(pymysql.cursors.DictCursor) as cursor:
                cursor.execute(sql, params)
                tuples = cursor.fetchall()
                return tuples
        except Exception as e:
            print(e)
            print(type(e))
        finally:
            conn.close()
    def updateExecutor(self, db, sql, params):
        conn = pymysql.connect(host='localhost', user='guest',
                password='bemyguest', db=db, charset='utf8')

        try:
            with conn.cursor() as cursor:
                cursor.execute(sql, params)
            conn.commit()
        except Exception as e:
            print(e)
            print(type(e))
        finally:
            conn.close()
```

```
class DB_Queries:
    # 모든 검색문은 여기에 각각 하나의 메소드로 정의

    def selectPlayer(self, position):
        sql = "SELECT * FROM player WHERE position = %s"
        params = (position)

        util = DB_Utils()
        tuples = util.queryExecutor(db="kleague", sql=sql,
                                    params=params)
        return tuples

class DB_Updates:
    # 모든 갱신문은 여기에 각각 하나의 메소드로 정의

    def insertPlayer(self, player_id, player_name, team_id, position):
        sql = "INSERT INTO player (player_id, player_name, team_id,
               position) VALUES (%s, %s, %s, %s)"
        params = (player_id, player_name, team_id, position)

        util = DB_Utils()
        util.updateExecutor(db="kleague", sql=sql, params=params)
```

```
########################################

# DBAPI_5.py가 실행될 때 __main__, import될 때는 모듈명 즉 DBAPI_5
if __name__ == "__main__":
    query = DB_Queries()
    players = query.selectPlayer("GK")
    print(players)
    print(len(players))

    update = DB_Updates()
    update.insertPlayer("2020001", "홍길동", "K01", "GK")

    players = query.selectPlayer("GK")
    print(players)
    print(len(players))
```