Chapter 07:Python과 연동

SQL 활용 프로그래밍

SQL Application Programming

CHAPTER 07 Python과 연동

SECTION 01 PostgreSQL과 응용프로그램의 연결

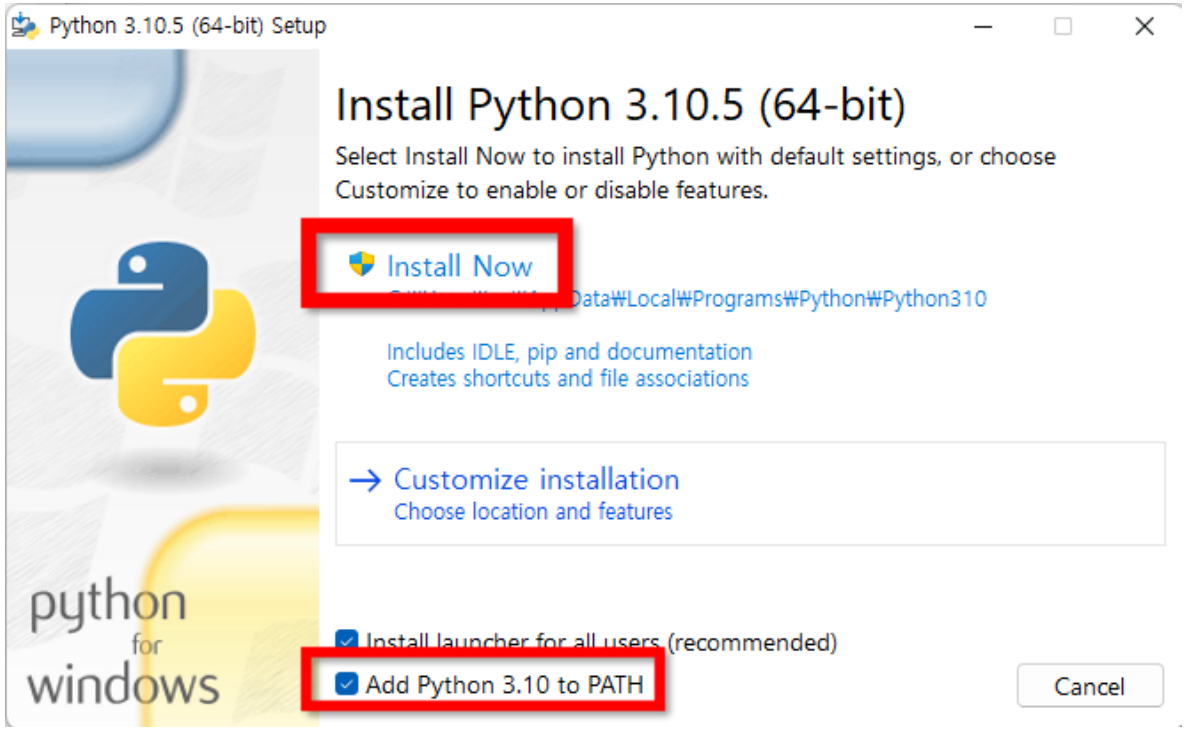
Python을 이용하여 PostgreSQL 사용

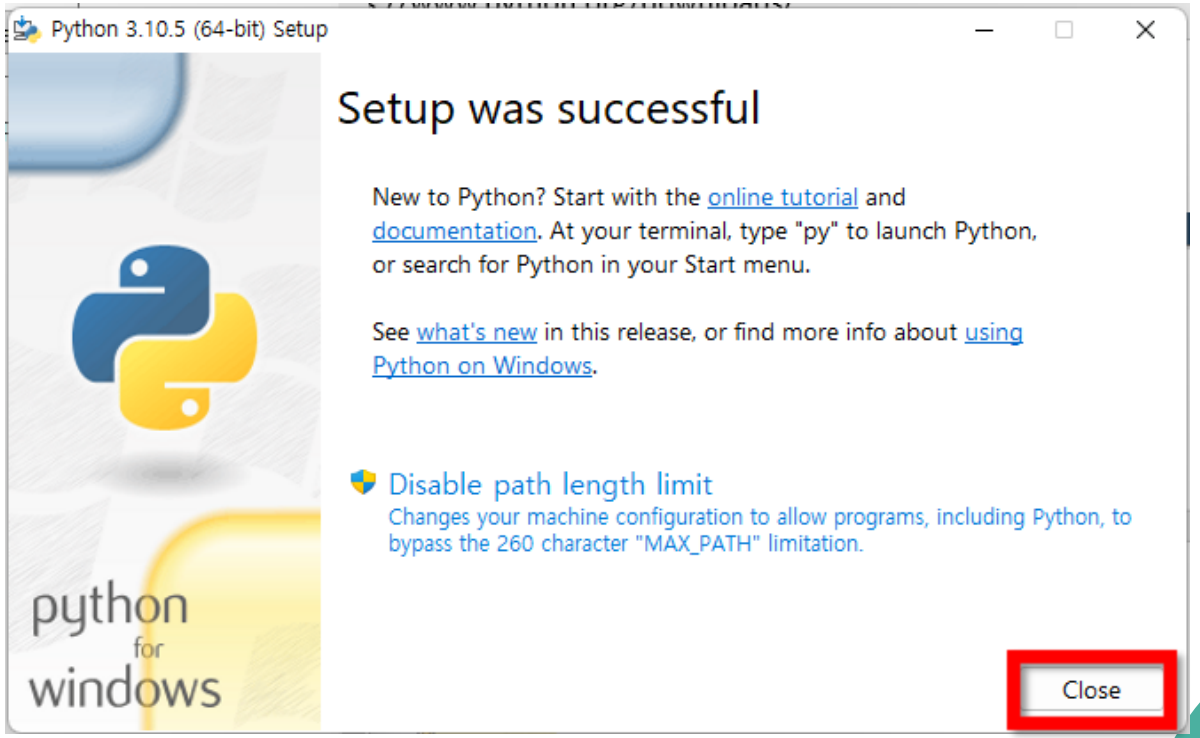
- 개발 툴로 사용할 python 설치
  - https://www.python.org/downloads/ 접속 후 Download Python 3.10.5 버튼

SECTION 01 PostgreSQL과 응용프로그램의 연결

Python을 이용하여 PostgreSQL 사용

- 개발 툴로 사용할 python 설치
  - Add python 3.10 to PATH 체크 >> Install Now >> 설치가 완료되면 Close

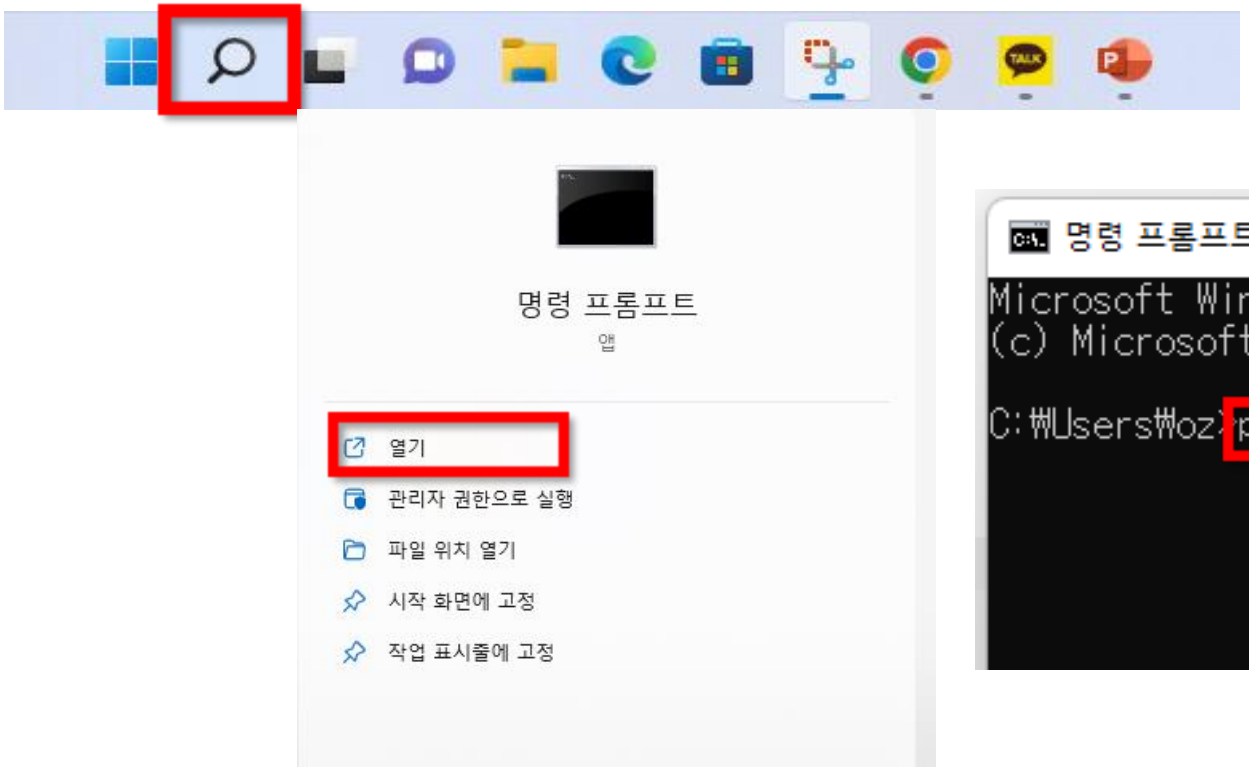


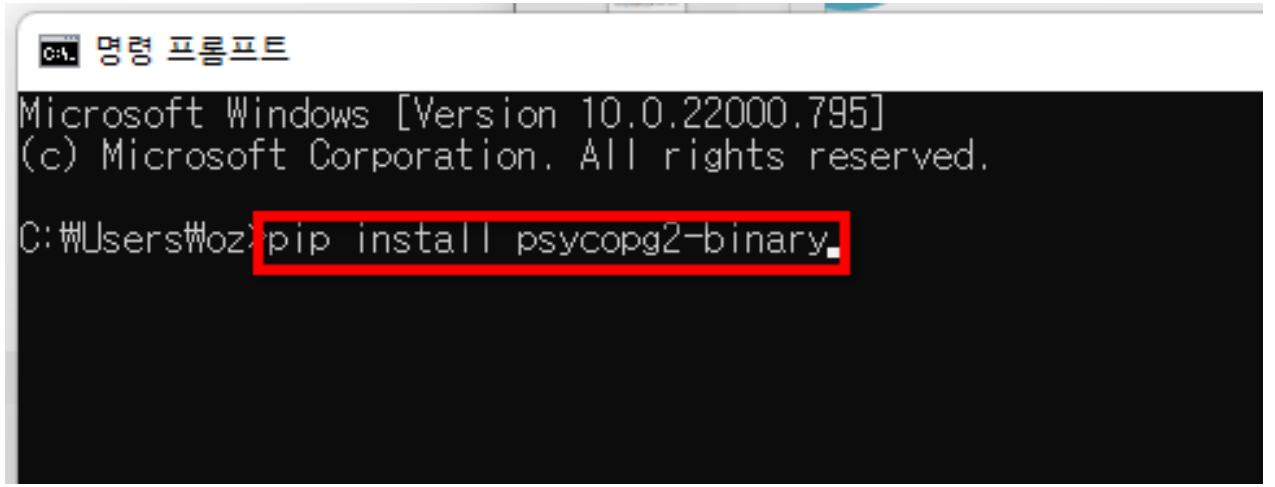


SECTION 01 PostgreSQL과 응용프로그램의 연결

Python을 이용하여 PostgreSQL 사용

- psycopg2 설치
  - Python에서 PostgreSQL을 사용하기 위해서는 psycopg2 라이브러리를 사용
  - 작업표시줄의 검색기능에 'cmd' 입력 후 실행 >> pip install psycopg2-binary 입력 >> 엔터 키





SECTION 01 PostgreSQL과 응용프로그램의 연결

Python을 이용하여 PostgreSQL 사용

- python 에서 PostgreSQL 접속
  1. psycopg2 라이브러리를 사용할 것이라고 정의
  2. Postgresql과 연결하기 위해 psycopg2.connect() 호출
     - Host, db name, user, password, port 지정
  3. db커서 객체생성하기 위해 db 인스턴스로부터 cursor() 호출
  4. SQL문 실행
     - Cursor객체의 메소드를 사용하여 SQL을 DB서버에 전송 & Fetch
       - ✓SQL을 DB서버에 전송: execute()
       - ✓Fetch: fetchall(), fetchone(), fetchmany(n)
     - Connection 객체의 commit(): 삽입, 갱신, 삭제 (INSERT/UPDATE/DELETE)
  5. DB 연결 종료: cur 객체 연결 해제, db인스턴스 연결 해제

SECTION 01 PostgreSQL과 응용프로그램의 연결

Python을 이용하여 PostgreSQL 사용 실습

```
import psycopg2
db = psycopg2.connect(host="localhost",dbname="employees",user="postgres",password="1234",port="5432")
cur = db.cursor()

cur.execute("select * from employees;")
db.commit()

cur.execute("select emp_no, gender from employees;")
result_one = cur.fetchone() #단일 결과 반환
result_many = cur.fetchmany() #여러 결과 반환
result_all = cur.fetchall() #모든 결과 반환

cur.close()
db.close()
```

SECTION 01 PostgreSQL과 응용프로그램의 연결

Python을 이용하여 PostgreSQL 사용하는 기본 클래스

```python
import psycopg2


class Database():
    def __init__(self):
        self.db = psycopg2.connect(
                host = '127.0.0.1',dbname='test',
                user='postgres',password = 'postgres',
                port=5432)

        self.cursor = self.db.cursor()


    def __del__(self):
        self.db.close()
        self.cursor.close()


    def execute(self,query,args={}):
        self.cursor.execute(query, args)
        row = self.cursor.fetchall()
        return row


    def commit(self):
        self.cursor.commit()
```

SECTION 01 PostgreSQL과 응용프로그램의 연결

- Psycopg2.connct는 DB와 통신을 주고 받는 기능 가진 객체를 만듭니다.

```
def __init__(self):
    self.db = psycopg2.connect(
            host = '127.0.0.1',dbname='tabledb',
            user='postgres',password = 'postgres',
            port=5432)

    self.cursor = self.db.cursor()
```

Host: postgresql 서버 ip주소
Dbname: 해당 서버 DB 중 연결할 DB 이름
User: 로그인할 user
Password: user의 비밀번호
Port: 연결할 port 번호

Postgresql 설치 시 사용자가 따로 설정하지 않았을 경우 dbname을 제외하고는 전부 왼쪽 값과 동일한 default값으로 설정됨
(127.0.0.1 localhost를 의미하는 ip이다.)

Context manager로 connect마다 여러 개가 생성될 수 있다.
Cursor를 통하여 사용자는 DB와 통신을 한다.
(cursor를 통하여 query문을 DB 보낼 수 있다.
Pgadmin에 query tool처럼 사용가능)

SECTION 01 PostgreSQL과 응용프로그램의 연결

- Psycopg2.connct는 DB와 통신을 주고 받는 기능 가진 객체를 만든다.

```
def execute(self,query,args={}):
    self.cursor.execute(query, args)
    row = self.cursor.fetchall()
    return row
```

Excute 한수는 사용자가 보낸 query를 DB에 전송하고 이에 대한 결과를 cursor객체 안에 저장한다.

fetchall()함수를 통하여, cursor안에 저장된 결과물을 return 할 수 있다.

fetch를 진행하게 될 경우 cursor안에 값은 사라진다 (중복으로 값을 가지고 올 수 없다)

fetchone(): cursor에 저장된 값을 가져오나 중복된 값이 있을 경우 이를 제거하고 return함

SECTION 01 PostgreSQL과 응용프로그램의 연결

- Commit을 통하여 업데이트를 확정 짓지 않을 경우 업데이트 내용이 DB에 반영되지 않는다.
- DB를 업데이트(변경, 추가, 삭제 등)를 했을 경우, commit을 통하여 내용을 반영시켜야 사용자들에게 반영됨

```
def commit(self):
    self.cursor.commit()
```

DB 트랜잭션을 전부 통합하여 commit을 진행함

트랜잭션(transaction) 이란? 데이터베이스 시스템에서 기본 작업 단위

Commit 이란? 데이터베이스 관리 시스템에서 트랜잭션을 종료 시 업데이트 내용을 확정하는 것

SECTION 01 PostgreSQL과 응용프로그램의 연결

- __del__ 은 class에서 소멸자로, 아래 close 함수를 통하여,connect객체와 contextmanager의 DB통신을 중단하는 명령입니다. Close하지 않을 경우 DB와의 연결이 DB 서버에 남아 자원을 차지할 수 있습니다.

```
def __del__(self):
    self.db.close()
    self.cursor.close()
```

각 객체에서 close() 함수를 통하여 연결 중단

SECTION 01 PostgreSQL과 응용프로그램의 연결

- 기본 Database class를 활용하여 CRUD(Create Read Update Delete) class를 생성

```python
class CRUD(Database):
    def __init__(self):
        super(CRUD,self).__init__()

    def create_table(self,table):
        sql = f'CREATE TABLE IF NOT EXISTS {table} ( id serial, name char(8));'
        try:
            self.cursor.execute(sql)
            self.db.commit()
        except Exception as e :
            print(" insert DB err ",e)


    def insertDB(self,table,colum,data):
        sql = f" INSERT INTO {table}({colum}) VALUES ('{data}') ;"
        try:
            self.cursor.execute(sql)
            self.db.commit()
        except Exception as e :
            print(" insert DB err ",e)
```

```python
def readDB(self,table,colum):
    sql = f" SELECT {colum} from {table}"
    try:
        self.cursor.execute(sql)
        result = self.cursor.fetchall()
    except Exception as e :
        result = (" read DB err",e)

    return result

def updateDB(self,table,colum,value,condition):
    sql = f" UPDATE {table} SET {colum}='{value}' WHERE {condition} "

    try :
        self.cursor.execute(sql)
        self.db.commit()
    except Exception as e :
        print(" update DB err",e)

def deleteDB(self,table,condition):
    sql = f" delete from {table} where {condition} ; "
    try :
        self.cursor.execute(sql)
        self.db.commit()
    except Exception as e:
        print( "delete DB err", e)
```

SECTION 01 PostgreSQL과 응용프로그램의 연결

◦ 앞서 정의한 Database class를 상속받아 부모class의 init을 실행

```
class CRUD(Database):
    def __init__(self):
        super(CRUD,self).__init__()
```

```
def readDB(self,table,colum):
    sql = f" SELECT {colum} from {table}"
    try:
        self.cursor.execute(sql)
        result = self.cursor.fetchall()
    except Exception as e :
        result = (" read DB err",e)

    return result
```

◦ readDB함수는 Select 쿼리를 DB 보낸 결과를 받아옴. 이때 readDB에 경우 기존 DB에 변경사항이 없으므로 commit을 해주지 않는다.

SECTION 01 PostgreSQL과 응용프로그램의 연결

```
def create_table(self,table):
    sql = f'CREATE TABLE IF NOT EXISTS {table} ( id serial, name char(8));'
    try:
        self.cursor.execute(sql)
        self.db.commit()
    except Exception as e :
        print(" insert DB err ",e)
```

◦ create_table 함수를 통하여 예제에서 활용할 table생성하는 create 쿼리를 DB 에 전송. DB에 변동이 있으므로, commit을 해준다.

```
def insertDB(self,table,colum,data):
    sql = f" INSERT INTO {table}({colum}) VALUES ('{data}') ;"
    try:
        self.cursor.execute(sql)
        self.db.commit()
    except Exception as e :
        print(" insert DB err ",e)


def updateDB(self,table,colum,value,condition):
    sql = f" UPDATE {table} SET {colum}='{value}' WHERE {condition} "

    try :
        self.cursor.execute(sql)
        self.db.commit()
    except Exception as e :
        print(" update DB err",e)

def deleteDB(self,table,condition):
    sql = f" delete from {table} where {condition} ; "
    try :
        self.cursor.execute(sql)
        self.db.commit()
    except Exception as e:
        print( "delete DB err", e)
```

◦ 각각의 인자값을 받아 INSERT, UPDATE,DELETE를 쿼리문을 완성하여 DB에 보내는 함수를 정의

◦INSERT,UPDATE,DELETE에 경우 수행시 DB에 변동사항이 발생하므로 COMMIT을 해주어야지 DB에 반영됨

SECTION 01 PostgreSQL과 응용프로그램의 연결

- 앞에서 정의한 class를 기반으로 테이블을 생성한 후 insert, update, delete를 수행해보고 중간 결과값들을 read를 통하여
- 테이블의 값을 출력해본다.

```
if __name__ =='__main__':
    t_name = 'test_t'

    db_con = CRUD()
    db_con.create_table(t_name)
    db_con.insertDB(t_name,'name','홍길동')
    db_con.insertDB(t_name,'name','누군가')

    result = db_con.readDB(t_name,'*')
    print('insert 결과 ',result)
    db_con.updateDB(t_name, 'name','이문형','id = 1')
    result = db_con.readDB(t_name,'*')
    print('update 결과 ',result)
    db_con.deleteDB(t_name, "name = '누군가'")
    result = db_con.readDB(t_name,'*')
    print('delete 결과 ',result)
```

```
insert 결과  [(1, '홍길동      '), (2, '누군가      ')]
update 결과  [(2, '누군가      '), (1, '이문형      ')]
delete 결과  [(1, '이문형      ')]
```