DATA 저장하기

목차

1. pickle
2. json
3. sqllite3
4. maria DB

1. pickle

pickle은 객체의 형태를 그대로 유지하면서 파일에 저장하고 불러올 수 있게 하는 모듈이다.

```
import pickle

# dict타입의 데이타
data = {1:'python',2:'you need'}

print(type(data))  # dict 타입

# 파일로 저장
with open("./python_basic/test.pickle",'wb') as f:
    pickle.dump(data,f)

# 파이썬 내에서 사용 바이트 형태
datab=pickle.dumps(data)
print(type(datab))
```

1. pickle

pickle은 객체의 형태를 그대로 유지하면서 파일에 저장하고 불러올 수 있게 하는 모듈이다.

```
import pickle

#파일을 읽음.
with open("./python_basic/test.pickle",'rb') as f:
    data=pickle.load(f)
    print(data)

# 바이트 타입을 파이썬 형태의 데이타 타입으로 읽음.
data1 = pickle.loads(datab)
print(data1)
print(type(data1))
```

2. json

JSON (JavaScript Object Notation)은 경량의 DATA-교환 형식이다.

```
import json

# 테스트용 Python Dictionary
customer = {
    'id': 152352,
    'name': '강진수',
    'history': [
        {'date': '2015-03-11', 'item': 'iPhone'},
        {'date': '2016-02-23', 'item': 'Monitor'},
    ]
}
```

2. json

JSON (JavaScript Object Notation)은 경량의 DATA-교환 형식이다.

```
import json

# JSON 인코딩
jsonString = json.dumps(customer)

# 문자열 출력
print(jsonString)
print(type(jsonString))  # class str

# json.dumps 파이썬 내에서 바로 사용
jsonString = json.dumps(customer, indent=4)
print(jsonString)

# json.dump 파일로 바로 저장
with open('./python_basic/data.json','wt') as f:
    json.dump(customer,f,indent=4)
```

```
# json.loads json 문자를 읽어서 파이썬 객체로 변경
customer1 = json.loads(jsonString)
print(customer1)

# json.load json 파일을 읽어서 파이썬 객체로 변경
with open('./python_basic/data.json','rt') as f:
    customer2 = json.load(f)
    print(type(customer2))
    print(customer2)
```

3. sqllite3

파이썬 내장 데이터베이스 관리 시스템이지만, 서버가 아니라 응용 프로그램에 넣어 사용하는 비교적 가벼운 데이타베이스 (SQLiteExpert)

01.sqlite-test.py

```
import sqlite3

#SQLlite3모듈 자체 버전
print(sqlite3.version)

#SQLlite 버전
print(sqlite3.sqlite_version)
```

01.sqlite-test.py

```
conn = sqlite3.connect('sqlite/example.db')
c = conn.cursor()

# Create table
c.execute('''CREATE TABLE if not exists stocks
             (date text, trans text, symbol text, qty real, price real)''')

# Insert a row of data
c.execute("INSERT INTO stocks VALUES ('2006-01-05','BUY','RHAT',100,35.14)")

# Save (commit) the changes
conn.commit()

conn.close()
```

3. sqllite3

02.sqlite-test.py

```
import sqlite3

conn = sqlite3.connect('sqlite/example.db')
c = conn.cursor()

# Never do this -- insecure!
symbol = 'RHAT'
c.execute("SELECT * FROM stocks WHERE symbol = '%s'" % symbol)

# Do this instead
t = ('RHAT',)
sql='SELECT * FROM stocks WHERE symbol=?'
c.execute(sql, t)     #c.execute(sql, (t,t1))
print(c.fetchone())
```

02.sqlite-test.py

```
# Larger example that inserts many records at a time
purchases = [('2006-03-28', 'BUY', 'IBM', 1000, 45.00),
             ('2006-04-05', 'BUY', 'MSFT', 1000, 72.00),
             ('2006-04-06', 'SELL', 'IBM', 500, 53.00),
            ]
c.executemany('INSERT INTO stocks VALUES (?,?,?,?,?)', purchases)
conn.commit()

c.execute('select * from stocks ORDER BY price')
rows=c.fetchall()
for row in rows:
    print(row)

for row in c.execute('SELECT * FROM stocks ORDER BY price'):
    print(row)

c.close()
```

3. sqllite3

03.sqlite-CSVtoDB.py

```
import csv, sqlite3

input_file = 'sqlite/input.csv'

conn=sqlite3.connect('sqlite/suppliers.db')
cursor=conn.cursor()
sql='''
    create table if not exists suppliers(
        supplier_name varchar(20),
        invoice_number varchar(20),
        part_number varchar(20),
        cost float,
        purchase_date date
    )
    '''
cursor.execute(sql)
```

03.sqlite-CSVtoDB.py

```
sql='delete from suppliers'
cursor.execute(sql)

# csv 파일에서 데이터를 읽어서 테이블에 insert
# file_reader=csv.reader(open(input_file,'r',encoding='utf-8'),delimiter=',')
file_reader=csv.reader(open(input_file,'r'),delimiter=',',quotechar='"')

# 첫 라인을 읽음(제목행)
header=next(file_reader,None)
print('header',header)

# header 이후의 2번째 행부터 끝까지 읽어 들이며 insert
for row in file_reader:
    data=[]
    # idx에는 0~4가 입력됨
    for idx in range(len(header)):
        data.append(row[idx])
    cursor.execute('insert into suppliers values(?,?,?,?,?)',data)

conn.commit()
```

03.sqlite-CSVtoDB.py

```
output=cursor.execute('select * from suppliers')
rows=output.fetchall()
print('행의 갯수:',len(rows))
for row in rows:
    print('필드의 갯수:',len(row))
    output=[]
    for column_index in range(len(row)):
        output.append(str(row[column_index]))
    print(output)
```

3. sqllite3

04.sqlite-def.py

```
import sqlite3

#테이블 생성 함수
def create_table():
    conn=sqlite3.connect('my_books.db')
    cursor=conn.cursor()
    #my_books테이블 생성(제목,출판일자,출판사,페이지수,추천여부)
    cursor.execute('''create table if not exists books(
        title text,
        published_date text,
        publisher text,
        pages integer,
        recommend integer
        )''')
    conn.commit()
    conn.close()

#테이블 생성 함수 호출
create_table()
```

04.sqlite-def.py

```
#데이터 입력 함수
def insert_books():
    conn=sqlite3.connect("my_books.db") #db연결
    cursor=conn.cursor()
    #데이타 입력방법1
    cursor.execute("insert into books values('Java','2019-05-20','길벗',500,10)")
    #데이타 입력방법2
    sql='insert into books values(?,?,?,?,?)'
    cursor.execute(sql,('Python','201001','한빛',584,20))
    #데이타 입력방법3
    items=[
        ('빅데이터','2014-07-02','삼성',296,11),
        ('안드로이드','2010-02-02','삼성',526,20),
        ('spring','2013-12-02','삼성',248,15)
    ]
    cursor.executemany(sql,items)
    conn.commit()
    conn.close()

#데이터 입력 함수 호출
insert_books()
```

04.sqlite-def.py

```
#전체 데이터 출력 함수
def all_books():
    conn=sqlite3.connect("my_books.db")
    cursor=conn.cursor()
    cursor.execute("select * from books")
    print('[1] 전체 데이터 출력하기')
    books=cursor.fetchall()
    print(type(books))
    print(len(books)) #레코드 개수 출력

    for book in books:
        for i in book:
            print(i,end=" ")
        print()

    conn.close()


#전체 데이타 출력 함수 호출
all_books()
```

04.sqlite-def.py

```
#레코드 개수 정하여 출력
def some_books(number):
    conn=sqlite3.connect("my_books.db")
    cursor=conn.cursor()
    cursor.execute("select * from books")
    books=cursor.fetchmany(number)

    for book in books:
        for i in book:
            print(i,end=" ")
        print()

    conn.close()

#
some_books(3)
```

04.sqlite-def.py

```
#1개의 데이타 출력하는 함수
def one_book():
    conn=sqlite3.connect("my_books.db")
    cursor=conn.cursor()
    cursor.execute("select * from books")
    book=cursor.fetchone()
    print(type(book))
    print(book)
    conn.close()

one_book()
```

04.sqlite-def.py

```
#조건 지정 및 정렬하여 검색
def big_books():
    conn=sqlite3.connect("my_books.db")
    cursor=conn.cursor()
    #페이지수가 300이상이고 페이지수가 많은 순서대로 title,page 출력
    cursor.execute('''select title,pages from books
                      where pages>300 order by pages desc''')
    books=cursor.fetchall()

    for book in books:
        for i in book:
            print(i,end=" ")
        print()

    conn.close()

big_books()
```

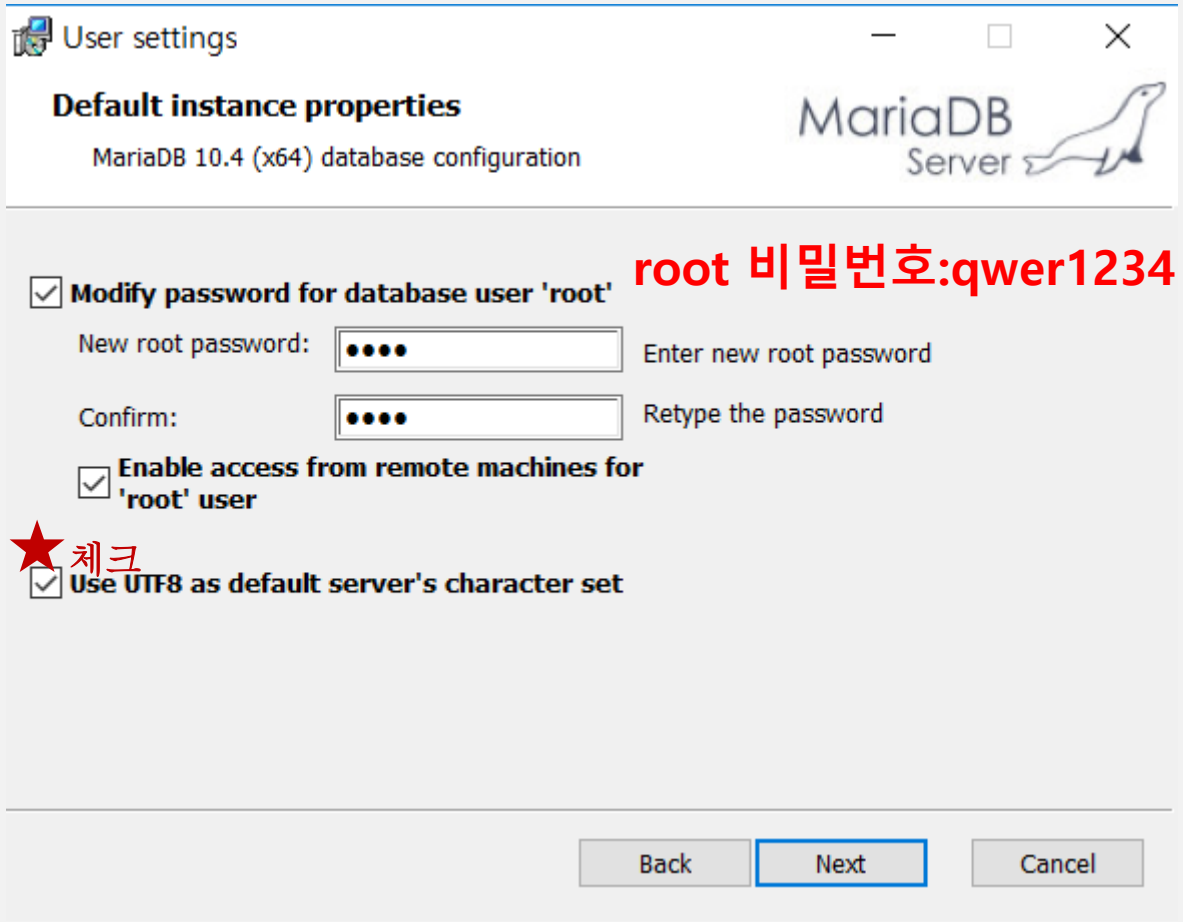
04.sqlite-def.py

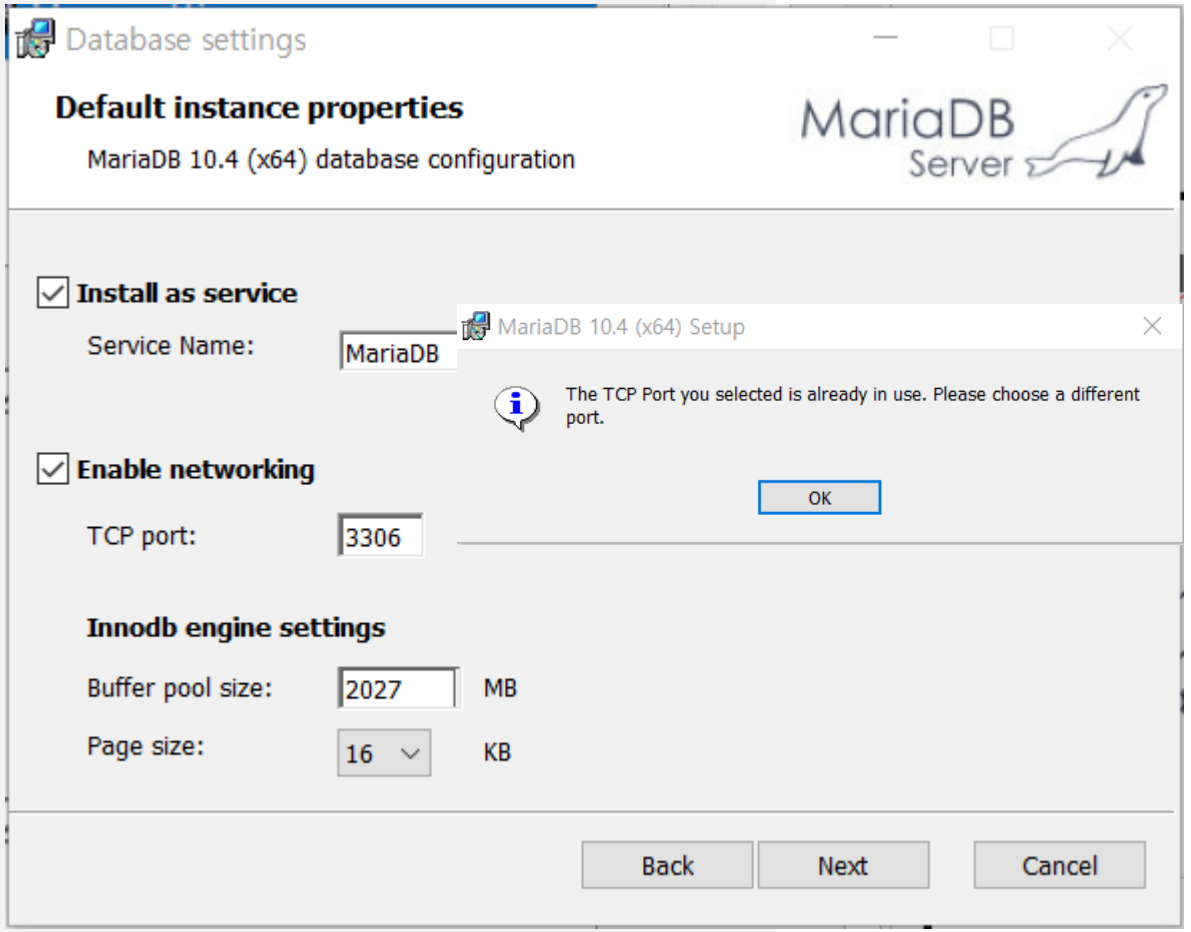
```
# 데이터 수정 함수
def update_books():
    conn=sqlite3.connect("my_books.db")
    cursor=conn.cursor()
    #title이 java인 recommend를 200으로 수정
    sql='''update books
            set recommend=? where title=?'''
    sql='''update books
            set recommend=:1 where title=:2'''
    cursor.execute(sql,(200,'Java'))
    conn.commit()
    conn.close()

update_books()
all_books()
```

04.sqlite-def.py

```
#데이터 삭제 함수
def delete_books():
    conn=sqlite3.connect("my_books.db")
    cursor=conn.cursor()
    #publisher가 한빛인 데이터 삭제
    sql="delete from books where publisher='한빛'"
    cursor.execute(sql)
    # sql="delete from books where publisher=:1"
    # cursor.execute(sql,('한빛',))
    conn.commit()
    conn.close()

delete_books()
all_books()
```

4. mariadb


User settings
Default instance properties
MariaDB 10.4 (x64) database configuration
MariaDB Server
root 비밀번호:qwer1234
Modify password for database user 'root'
New root password:
Enter new root password
Confirm:
Retype the password
Enable access from remote machines for 'root' user
체크
Use UTF8 as default server's character set
Back
Next
Cancel


Database settings
Default instance properties
MariaDB 10.4 (x64) database configuration
MariaDB Server
Install as service
Service Name:
MariaDB
MariaDB 10.4 (x64) Setup
The TCP Port you selected is already in use. Please choose a different port.
OK
Enable networking
TCP port:
3306
Innodb engine settings
Buffer pool size:
2027
MB
Page size:
16
KB
Back
Next
Cancel

4. mariadb

01.mariadb-pymysql.py

```
import pymysql

conn = pymysql.connect(host='localhost',
                       user='root',
                       password='qwer1234',
                       db='test',
                       charset='utf8mb4',
                       cursorclass=pymysql.cursors.DictCursor)

c = conn.cursor()

c.execute('''CREATE TABLE if not exists stocks
          (date text, trans text, symbol text, qty real, price real)''')

c.execute("INSERT INTO stocks VALUES ('2006-01-05','BUY','RHAT',100,35.14)")

conn.commit()
conn.close()
```

4. mariadb

02.mariadb-pymysql.py

```
import pymysql

conn = pymysql.connect(host='localhost',
                       user='root',
                       password='qwer1234',
                       db='test',
                       charset='utf8mb4',
                       cursorclass=pymysql.cursors.DictCursor)

c = conn.cursor()

# Do this instead
t = ('ibm')
sql='SELECT * FROM stocks WHERE symbol=%s'
c.execute(sql, t)

print(c.fetchall())
print()
```

02.mariadb-pymysql.py

```
# Larger example that inserts many records at a time
purchases = [('2006-03-28', 'BUY', 'IBM', 1000, 45.00),
             ('2006-04-05', 'BUY', 'MSFT', 1000, 72.00),
             ('2006-04-06', 'SELL', 'IBM', 500, 53.00)
            ]
c.executemany('INSERT INTO stocks VALUES (%s,%s,%s,%s,%s)', purchases)
conn.commit()

c.execute('select * from stocks ORDER BY price')
rows=c.fetchall()
for row in rows:
    print(row)


c.close()
```

4. mariadb

mariadb_pymysql.py

```
import pymysql

def conn_db():
    conn = pymysql.connect(host='localhost',
                           user='root',
                           password='qwer1234',
                           db='test',
                           charset='utf8mb4',
                           cursorclass=pymysql.cursors.DictCursor)
    return conn
```

mariadb_pymysql.py

```
#테이블 생성 함수
def create_table():
    conn=conn_db()
    cursor=conn.cursor()
    #my_books테이블 생성(제목,출판일자,출판사,페이지수,추천여부)
    cursor.execute('''create table if not exists books(
        title text,
        published_date text,
        publisher text,
        pages integer,
        recommend integer
        )''')
    conn.commit()
    conn.close()
```

mariadb_pymysql.py

```
#데이터 입력 함수
def insert_books():
    conn=conn_db()
    cursor=conn.cursor()
    #데이타 입력방법1
    cursor.execute("insert into books values('Java','2019-05-20','길벗',500,10)")
    #데이타 입력방법2
    sql='insert into books values(%s,%s,%s,%s,%s)'
    cursor.execute(sql,('Python','201001','한빛',584,20))
    #데이타 입력방법3
    items=[
        ('빅데이터','2014-07-02','삼성',296,11),
        ('안드로이드','2010-02-02','삼성',526,20),
        ('spring','2013-12-02','삼성',248,15)
    ]
    cursor.executemany(sql,items)
    conn.commit()
    conn.close()
```

mariadb_pymysql.py

```
#전체 데이터 출력 함수
def all_books():
    conn=conn_db()
    cursor=conn.cursor()
    cursor.execute("select * from books")

    print('[1] 전체 데이터 출력하기')

    books=cursor.fetchall()
    print(type(books))
    print(len(books)) #레코드 개수 출력
    print(books)

    for book in books:
        for i in book:
            print(book[i],end=" ")
        print()

    conn.close()
```

mariadb_pymysql.py

```
#레코드 개수 정하여 출력
def some_books(number):
    conn=conn_db()
    cursor=conn.cursor()
    cursor.execute("select * from books")
    books=cursor.fetchmany(number)

    for book in books:
        for i in book:
            print(book[i],end=" ")
        print()

    conn.close()
```

mariadb_pymysql.py

```
# 데이터 수정 함수
def update_books():
    conn=conn_db()
    cursor=conn.cursor()
    #title이 java인 recommend를 200으로 수정
    sql='''update books
           set recommend=%s where title=%s'''
    sql='''update books
           set recommend=%s where title=%s'''
    cursor.execute(sql,(200,'Java'))
    conn.commit()
    conn.close()
```

mariadb_pymysql.py

```
#데이터 삭제 함수
def delete_books():
    conn=conn_db()
    cursor=conn.cursor()
    #publisher가 한빛인 데이터 삭제
    sql="delete from books where publisher='한빛'"
    cursor.execute(sql)
    # sql="delete from books where publisher=%s"
    # cursor.execute(sql,('한빛',))
    conn.commit()
    conn.close()
```