Python + SQL

เก็บข้อมูลราคาเรือดำน้ำแต่ละประเทศ

SQLite

Uncle Engineer

ลุงวิศวกร สอนคำนวณ

โครงสร้างฐานข้อมูล (database)

1 ฐานข้อมูลอาจมีหลายตาราง

โครงสร้างตาราง (table)

table name = ‘submarine’

ID	brand	price
1	China	10000
2	Russia	30000
3	USA	50000

1 ตารางอาจมีหลายฟิลด์

แต่ละฟิลด์ของตารางชื่อ submarine ประกอบด้วย ID, brand, price

มาเริ่มต้นสร้างฐานข้อมูลใน python กัน

1- อิมพอร์ทไลบรารี่สำหรับ SQL

```
import sqlite3
```

ใช้ไลบรารี่ sqlite3 ซึ่งแถมมากับ python แล้ว ไม่ต้องติดตั้งแต่อย่างใด อิมพอร์ทมาใช้งานได้เลยหนุ่มๆ

2- สร้าง table เก็บเรือดำน้ำก่อน

ขั้นแรกถามลุงตู่ก่อนว่าอยากเก็บค่าอะไรบ้าง ลุงตู่เข้าฝัน บอกว่าให้ table ชื่อว่า submarine เก็บ

- เรือดำน้ำลำนี้ใครผลิต ให้ตั้งชื่อ field ว่า brand
- ราคาเรือดำน้ำลำนี้ราคากี่ล้าน ตั้งชื่อ filed ว่า price

*** ฟิลด์ ID ควรจะมีไว้ทุก table

2.1 ก่อนสร้างตารางได้ ต้องสร้าง connection ก่อน

```
import sqlite3

conn = sqlite3.connect('submarine.db')
c = conn.cursor()
```

conn = sqlite3.connect('submarine.db')

สร้างตัวแปร conn เพื่อแทนที่เครื่องมือสำหรับติดต่อกับฐานข้อมูลที่ชื่อว่า " submarine.db " (.db คือ นามสกุล)
ส่วนเจ้า c = conn.cursor() คือ เครื่องที่ใช้ในการสร้างตาราง, เพิ่มข้อมูล, แก้ไขข้อมูล , ลบข้อมูล, เรียกดูข้อมูลจากฐานข้อมูล และอื่นๆ

conn = sqlite3.connect(‘submarine.db’) เปรียบเสมือนเจ้าของบริษัทรับเหมาที่ต้องไปติดต่อลูกค้า ที่ชื่อว่า submarine.db ถ้ายังไม่ติดต่อตกลงทำสัญญากัน เจ้า c = conn.cursor() เปรียบเสมือช่างก่อสร้าง ก็ยังไม่สามารถสร้างอะไรขึ้นมาได้ ดังนั้นต้องมีการติดต่อกันมาก่อน

2.2 คราวนี้ก็สร้างตารางได้แล้ว

```
c.execute("""CREATE TABLE IF NOT EXISTS submarine (
                ID INTEGER PRIMARY KEY AUTOINCREMENT,
                brand text,
                price int)""")
```

- c.execute แปลว่า ลงมือทำตามคำสั่งต่อไปนี้
- CREATE TABLE IF NOT EXISTS submarine

create table คือ คำสั่งสร้างตาราง
if not exists คือ ถ้าตารางนี้ยังไม่มีก็ให้สร้างมันขึ้นมา
ถ้าเคยสร้างไว้แล้วก็ไม่ต้องสร้างเพิ่ม
submarine คือ ชื่อตารางที่จะใช้เก็บเรือดำน้ำ

2.2 คราวนี้ก็สร้างตารางได้แล้ว (ต่อ)

```
c.execute("""CREATE TABLE IF NOT EXISTS submarine (
                ID INTEGER PRIMARY KEY AUTOINCREMENT,
                brand text,
                price int)""")
```

- ID INTEGER แปลว่า สร้าง feild ชื่อ ID สำหรับเก็บไอดีของข้อมูลแต่ละชุด
- PRIMARY KEY AUTOINCREMENT แปลว่า ให้ฟิลด์ตัวนี้เป็นคีย์หลักและให้เพิ่มอัตโนมัติเมื่อมีการสร้างข้อมูลใหม่ขึ้นมา ส่วนคำว่า brand text คือบอกว่า ให้สร้างฟิลด์ชื่อว่า brand เป็นชนิด text และคำว่า price คือ ชื่อฟิลด์ใช้เก็บราคาเรือดำน้ำ int คือชนิดจำนวนเต็ม (integer)

3- คราวนี้ทดลองป้อนค่าเข้าไปใน table

```
def insert_submarine(brand,price):
    ID = None
    with conn:
        c.execute("""INSERT INTO submarine VALUES (?,?,?)""",
                  (ID,brand,price))
    conn.commit()
    print('Data was inserted')
```

- สร้างฟังชั่นสำหรับเพิ่มค่าเข้าไปในตาราง
- ID ให้ค่าเป็น None เพราะเราตั้งค่าอัตโนมัติไว้แล้ว
- with conn: ใช้สำหรับเรียกให้ติดต่อฐานข้อมูลนั้นๆมา

3- คราวนี้ทดลองป้อนค่าเข้าไปใน table (ต่อ)

```
def insert_submarine(brand,price):
    ID = None
    with conn:
        c.execute("""INSERT INTO submarine VALUES (?,?,?)""",
                  (ID,brand,price))
    conn.commit()
    print('Data was inserted')
```

- การเพิ่มข้อมูลเข้าไปในตารางให้ใช้คำสั่ง
  INSERT INTO submarine (submarine คือชื่อตาราง)
- VALUES (?,?,?) ใส่เครื่องหมาย ? แทนฟิลด์ทั้งหมด (มีกี่ฟิลด์ใส่หมด)
  (ID, brand, price) คือ ตัวแปรที่จะเอามาใส่ในตาราง
  ID = None, brand และ price = ค่าที่จะใส่เข้ามาตอนเรียกใช้ฟังชั่น
- conn.commit() เป็นคำสั่งสำหรับการบันทึกข้อมูลต้องใช้ทุกครั้ง

4- ทดลองบันทึกราคาเรือดำน้ำจากจีน

```
insert_submarine('China',10000)
insert_submarine('Russia',30000)
```

เรียกฟังชั่น insert_submarine แล้วใส่ brand = ‘China’

price = 10000 ....เมื่อใส่ค่าแล้วให้ลองกดรัน 1 ครั้ง

คราวนี้ต้องเขียนฟังชั่นสำหรับการอ่านข้อมูลในตาราง

```
Data was inserted
>>> |
```

5- เมื่อใส่ค่าเข้าไปแล้วเรามาดูข้อมูลกัน

```
def view_submarine():
    with conn:
        c.execute("SELECT * FROM submarine")
        submarine = c.fetchall()
        print(submarine)
```

สร้างฟังชั่น view_submarine

ใช้คำสั่ง SELECT * FROM submarine

submarine = c.fetchall() คือคำสั่งบอกว่าให้ดึงค่าที่เลือกมาได้ใส่เข้าไปในตัวแปรชื่อว่า submarine

6- ลองเรียกฟังชั่นการดูข้อมูลเรือดำน้ำ

```
view_submarine()
```

ผลลัพท์ที่ได้ จะได้เป็น list ออกไปใช้งานต่อ

```
[(1, 'China', 10000), (2, 'Russia', 30000)]
>>>
```

7- ถ้าอยากได้ค่าไปใช้งานต่อก็ใช้ return

```
def view_submarine():
    with conn:
        c.execute("SELECT * FROM submarine")
        submarine = c.fetchall()
        print(submarine)

    return submarine
```

เราจะได้ List submarine ไปใช้งานต่อโดยการสร้างตัวแปร เช่น

```
allsubmarine = view_submarine()
print("First Submarine Brand: ",allsubmarine[0][1])
```

จะได้ result ออกมาเป็น

```
[(1, 'China', 10000), (2, 'Russia', 30000)]
First Submarine Brand:  China
>>> |
```

เข้าไปดูโค้ดฉบับเต็ม แถมฟังชั่นการ delete และ update ด้วย

https://github.com/UncleEngineer/BasicSQL

เริ่มหัวร้อนแล้ว...
ติดตามชมตอนต่อไป

つづく

หนังกลางแปลงจบแล้ว ขายยาวิเศษแป๊ป 555

Uncle Engineer
ลุงวิศวกร สอนคำนวณ

สนใจเรียน Python ฉบับใช้งานได้จริง สมัครด่วน

มีทั้งคอร์สเสาร์อาทิตย์, คอร์สภาคค่ำ,
คอร์สเรียนสด online conference

http://uncle-engineer.com/python/
สายด่วน : 080-3943058

สถานที่อบรม: ณ BTS สนามเป้า(ทางออก 4)
อาคารวิเศษศิริชั้น 3 ห้อง 301