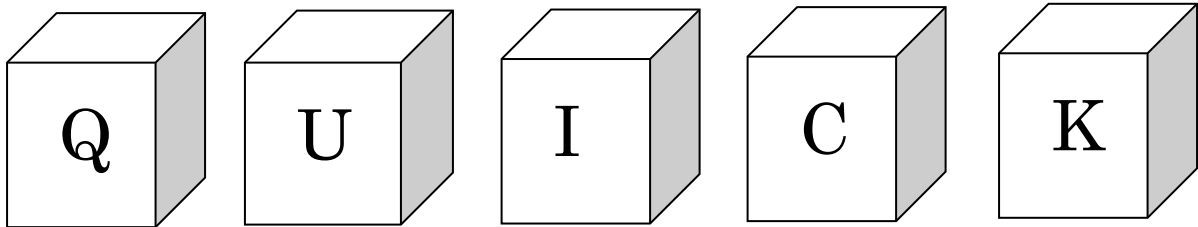


Quick Check

▶ Grammer & Usage



>> PySimpleGUI で GUI ツール作成

～取り込んだ csv ファイルのデータを sqlite データベースに登録してみよう

業務改善や社内の取り組み（事例）として、VBA や Python 等を使って今後、ツール導入を考えていきたいとお考えの方もいらっしゃるかもしれません。（小さい規模での）導入作業を繰り返し行うことで、周囲に合意を得られる形で情報が共有され、実際に導入実績につながるということもあるのかもしれない。

Python で GUI ツールを作成する際に、取り込んだ Excel (csv) ファイルを操作するという方法として、前回「PySimpleGUI」ライブラリを使った例を確認しました。ライブラリを用いてデータを読み書きしたり、DB 操作をしたりといった具合に、使用用途に合わせて作成していけると良いですね。

今回、読み込んだ csv のデータを SQLite データベースに登録・更新する操作について確認します。前回同様、「PySimpleGUI」ライブラリを使って GUI ツールの使い方について簡単にサンプルで確認後、実際に csv ファイルを読み込んで DB にデータ登録・更新を行う方法についてみていきたいと思います。


[添付資料のダウンロードはこちらから](#)

添付資料にあるテーブル定義書のデータ「FileToRead.csv」を登録してみたいと思います。

▶ 家計簿テーブル (HouseholdAccount)

日付	費目	メモ	入金額	出金額
2022/5/22	食費	1週間分の食材購入	0	6,680
2022/5/25	給料	4月の給料	200,000	0
2022/5/27	教養娯楽費	書籍を購入	0	2,260
2022/5/27	通信費	4月の携帯代	0	4,000
2022/5/27	楽天カード利用料	主にAmazonで購入	0	55,000
2022/5/27	サブスク費(その他)	定額サービス利用料	0	7,000

FileToRead.csv



	A	B	C	D	E	F
1	AccountDate	Expenseltem	Memo	DepositAmount	WithdrawalAmount	
2	2022/5/22	食費	1週間分の食材購入	0	6,680	
3	2022/5/25	給料	4月の給料	200,000	0	
4	2022/5/27	教養娯楽費	書籍を購入	0	2,260	
5	2022/5/27	通信費	4月の携帯代	0	4,000	
6	2022/5/27	楽天カード利用料	主にAmazonで購入	0	55,000	
7	2022/5/27	サブスク費 (その他)	定額サービス利用料	0	7,000	
8						
9						
10						

それではまず、PySimpleGUI ライブラリを使って GUI ツールの画面レイアウトを作成していきましょう。

○ PySimpleGUI の基礎～基本的な使い方

問題：サンプルコードを実行して画面レイアウトを作成

以下のサンプルのように（PySimpleGUI の基礎解説の参考サイトや他の Web サイトを参考にしながら）csv ファイルを読み込む画面レイアウトを作成してみましょう。



【参考サイト】：PySimpleGUI の使い方についての基礎解説
http://www.k-techlabo.org/blog2/?page_id=1481

・インストール

pip という Python のパッケージ管理ツールを使って、「PySimpleGUI」をインストールします。

※ Python インストール時に標準で入っているパッケージ以外をインストールする場合、
「**pip install パッケージ名**」のコマンドで、ローカル環境の PowerShell にてインストールをします。

pip install PySimpleGUI

```
PS C:\Users\... pip install PySimpleGUI
Collecting PySimpleGUI
  Downloading PySimpleGUI-4.57.0-py3-none-any.whl (491 kB)
    | 491 kB 1.6 MB/s
Installing collected packages: PySimpleGUI
Successfully installed PySimpleGUI-4.57.0
```

▶ サンプルコード（画面レイアウト作成）： layout_PySimpleGUI.py

```
import PySimpleGUI as sg

layout = [
    [sg.Text("ファイル"), sg.InputText(), sg.FileBrowse('ファイル選択', key="inputFilePath")],
    [sg.Button('DB 登録')],
    [sg.Text(" ↓ ↓ DB に登録した内容を表示: ")],
    [sg.Multiline(default_text="", size=(60,10), border_width=2, key='tb1')],
    [sg.Button('クリア')]
]

window = sg.Window("ファイル読込", layout)

# イベントループ
while True:
    event, values = window.read()
    if event == sg.WIN_CLOSED:
        break
    elif event == 'DB 登録':
        # 処理の内容を記載
        sg.popup('処理内容の実装前')

    elif event == 'クリア':
        # 処理の内容を記載
        window['tb1'].Update("")

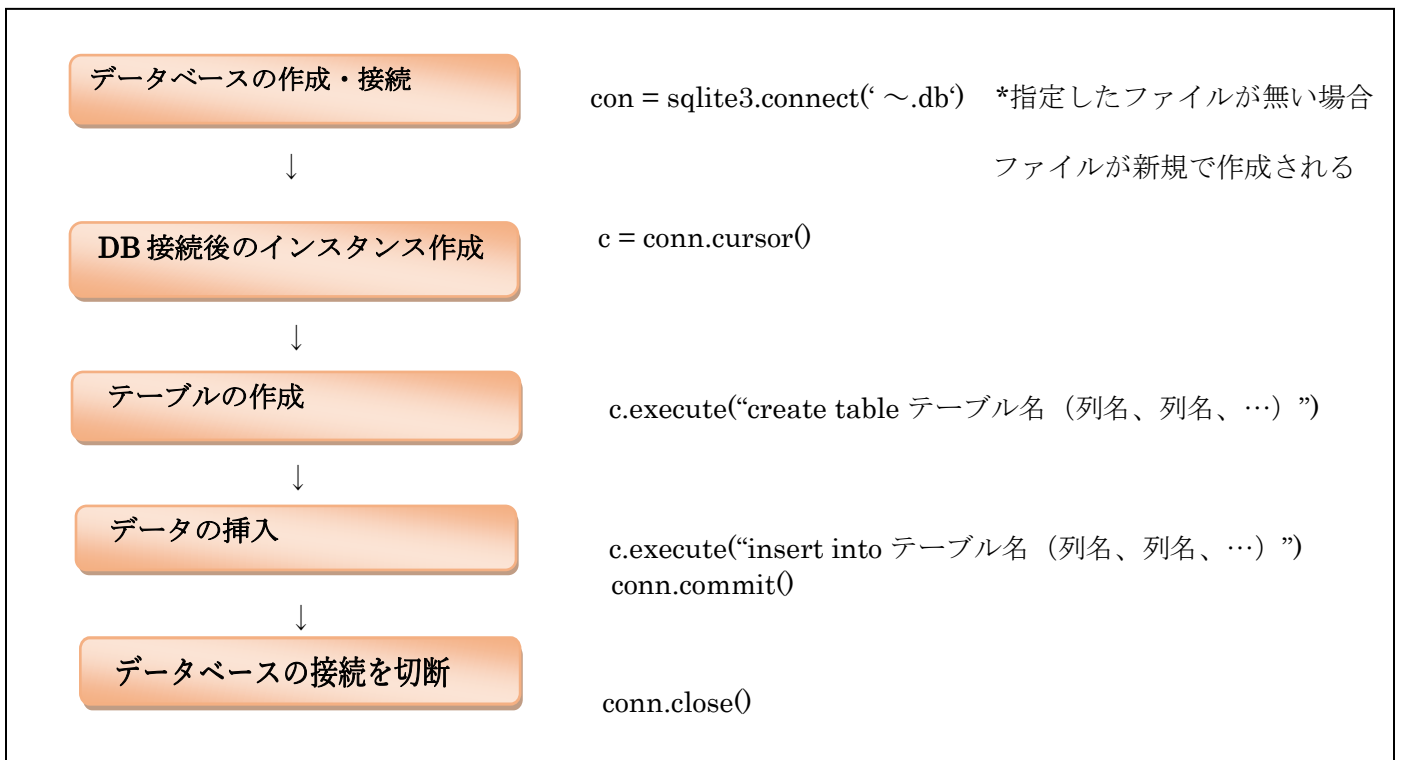
window.close()
```

○Python で DB 操作 ～ SQLite でデータを登録・更新する

まずは、データベース操作方法の手順を理解していきましょう！

標準ライブラリに入っている sqlite3 モジュールを import します。

```
import sqlite3
```



問題

Python で DB 操作の処理を記述し、以下の家計簿テーブル（HouseholdAccount）にデータを挿入して結果を確認してみましょう。

テーブル名(和)			家計簿テーブル			RDBMS		
テーブル名(英)			HouseholdAccount			備考		
No	PK	UK	カラム名	項目名	概要	データ型	長さ	NOT NULL
1	-	-	AccountDate	日付	費目の入出があった日	TEXT (DATETIME)	10	○
2	-	-	ExpenseItem	費目	記載する項目名	TEXT (VARCHAR)	50	○
3	-	-	Memo	メモ	メモ	TEXT (VARCHAR)	50	-
4	-	-	DepositAmount	入金額	入金額	INTEGER	7	-
5	-	-	WithdrawalAmount	出金額	出金額	INTEGER	7	-

挿入するデータ

日付	費目	メモ	入金額	出金額
2021/5/28	食費	外食費	0	2,500
2021/5/28	交通費	電子マネーチャージ	0	1,000

▶ コード (insertSQL_sample.py)

```
import sqlite3

dbname = 'Household.db'
conn = sqlite3.connect(dbname)
cur = conn.cursor()

# テーブルを作成
cur.execute('CREATE TABLE HouseholdAccount(
    AccountDate DATETIME NOT NULL, ExpenseItem VARCHAR(50) NOT NULL,
    Memo VARCHAR(50), DepositAmount INTEGER(7), WithdrawalAmount INTEGER(7)
)')

# " HouseholdAccount "にデータを入れる
cur.execute('INSERT INTO HouseholdAccount(
AccountDate, ExpenseItem, Memo, DepositAmount, WithdrawalAmount) VALUES("2022/5/28",
食費", "外食費", 0, 2500), ("2022/5/28", "交通費", "電子マネーチャージ", 0, 1000); ')

conn.commit()

cur.close()
conn.close()
```

参考記事 : 「Python SQLite3 でデータベースを扱う」

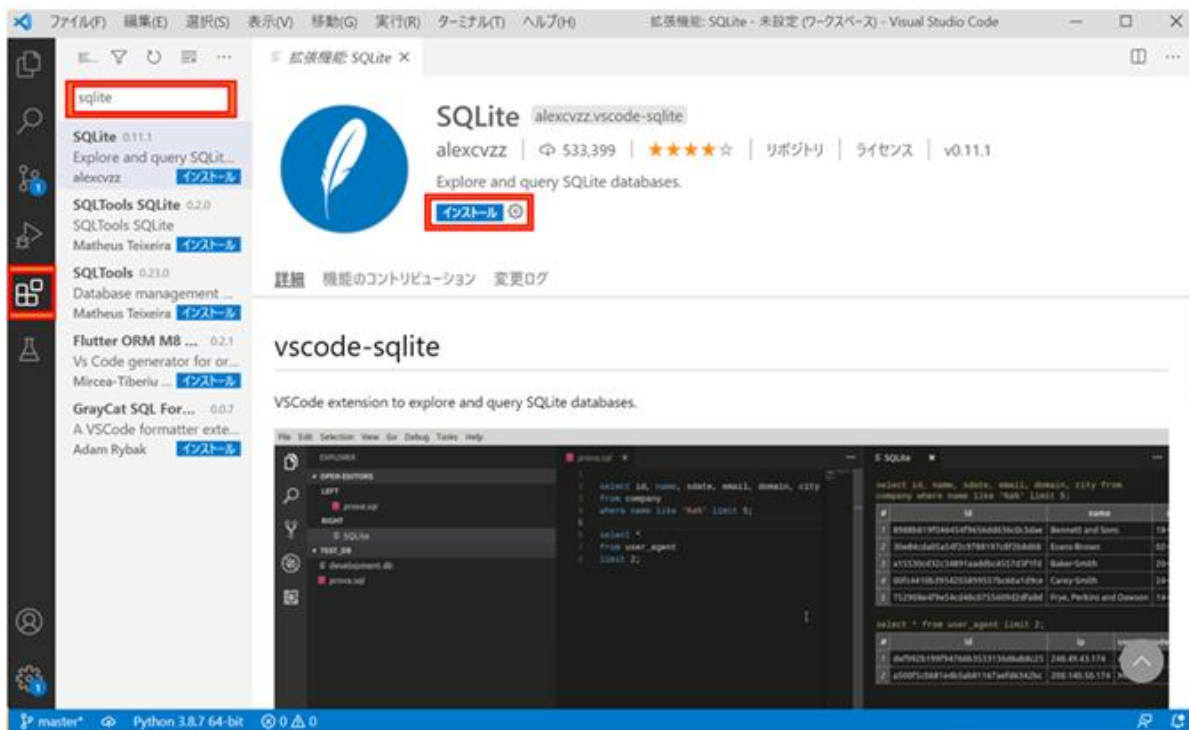
<https://blog.novonovo.jp/python/python-sqlite3%E3%81%A7%E3%83%87%E3%83%BC%E3%82%BF%E3%83%99%E3%83%BC%E3%82%B9%E3%82%92%E6%89%B1%E3%81%86/>

補足：必要に応じて、テーブル作成時に「テーブルの存在チェック」処理を追加してください。

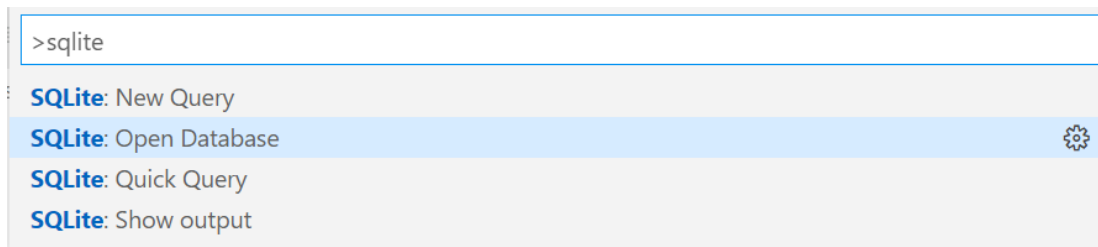
```
#テーブルの存在チェック
def table_isExist(conn, cur):
    cur.execute("""
        SELECT COUNT(*) FROM sqlite_master
        WHERE TYPE='table' AND name='テーブル名'
        """)
    if cur.fetchone()[0] == 0:
        return False
    return True

if table_isExist(conn, cur) == False:
    # テーブル作成処理を記載
```

拡張機能から「sqlite」と検索して、「SQLite」インストール



コマンドパレットを開き、「SQLite: Open Database」を選択し、作成したデータベースファイルを選択



先ほど作成したデータベースファイルを選択

表示された「SQLITE EXPLORE」から作成したテーブルを右クリックし、「Show Table」を選択

▶ 実行結果

SQL ▾				
< 1 / 1 > 1 - 2 of 2				
AccountDate	ExpenseItem	Memo	DepositAmount	WithdrawalAmount
2022/5/28	食費	外食費	0	2500
2022/5/28	交通費	電子マネーチャージ	0	1000

問題

先程の家計簿テーブル（HouseholdAccount）に pandas のデータフレームを利用して読み込んだ CSV ファイルのデータを DB に追加して結果を確認してみましょう。

* このとき、実行ファイルと同じディレクトリに「FileToRead.csv」を追加すること

○ Pandas とは？

Python でデータ処理をするために作られた高機能なライブラリ。代表的な使い方として Series や DataFrame を使ったデータの処理方法があります。

Pandas を利用するには、以下のようにインポートを行います。As キーワードを使用して pd で呼び出せるようにします。

In

```
import pandas as pd
```


▷使用例

```
# 1 次元データの利用
ser = pd.Series( [10, 20, 30, 40] )
ser

# 2 次元データの利用
df = pd.DataFrame( [10, "a", True],
                   [20, "b", False],
                   [30, "c", False],
                   [40, "d", True] )
df
```

csv や Excel のデータを読み込んだり、列や行を削除したり、フィルターをかけて抽出をしたりといった Excel やデータベース言語の SQL でできることが Pandas の機能にはあります。

▶ コード (insertSQL_dataframe.py)

```
import sqlite3
import pandas as pd
# pandas で csv ファイルの内容を読み込む
# csv ファイルの、1 列目に AccountDate, 2 列目に ExpenseItem, 3 列目に Memo, 4 列目に
DepositAmount, 5 列目に WithdrawalAmount が入っているとする
df = pd.read_csv('FileToRead.csv', encoding='SHIFT_JIS')
dbname = 'Household.db'
conn = sqlite3.connect(dbname)
cur = conn.cursor()

# 名前が Household の DB に、読み込んだ csv ファイルのデータを SQL に書き込む
df.to_sql('HouseholdAccount', conn, if_exists='replace', index = None)

conn.commit()
# 作成したデータベースを 1 行ずつ見る
select_sql = 'SELECT * FROM HouseholdAccount'
for row in cur.execute(select_sql):
    print(row)

cur.close()
conn.close()
```

▶ 実行結果

```
xe c:/VSCoFolder/PySimpleGUIProject/insertSQL_dataframe.py
('2022/5/22', '食費', '1週間分の食材購入', '0', '6,680')
('2022/5/25', '給料', '4月の給料', '200,000', '0')
('2022/5/27', '教養娯楽費', '書籍を購入', '0', '2,260')
('2022/5/27', '通信費', '4月の携帯代', '0', '4,000')
('2022/5/27', '楽天カード利用料', '主にAmazonで購入', '0', '55,000')
('2022/5/27', 'サブスク費（その他）', '定額サービス利用料', '0', '7,000')
(None, None, None, None, None)
```

問題【課題】：DB 操作処理の実装～作成した画面レイアウトの「DB 登録」ボタン押下時の処理を記述

「DB 登録」ボタンを押下した際の処理を実装してみましょう。



=====

業務で Python を使ってツールを作ったり、自動化処理を記述したりする機会がある方もいらっしゃると思います。自身の PC で Visual Studio Code を使って簡単に今回のサンプルを試すことができますので、この機会に Python に触れる時間を作っていただけると嬉しい限りです。

以上となります。