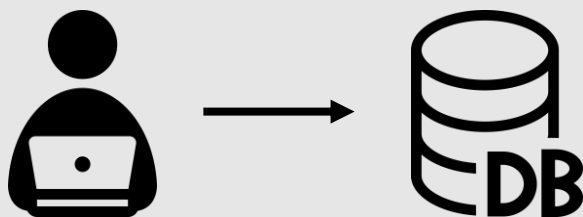


# CSVファイル出力マニュアル – python

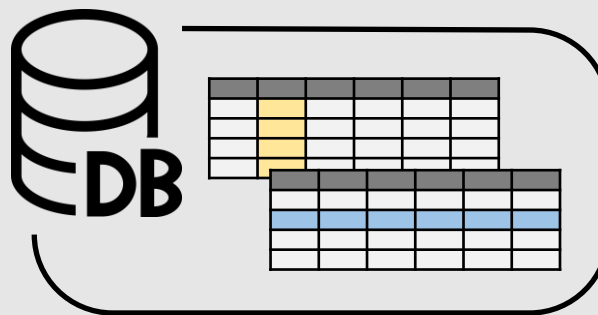
配布資料「**csv\_from\_database.ipynb**」に沿って、データベースへの接続から**csv**ファイル出力までの流れを説明する補足資料となります。



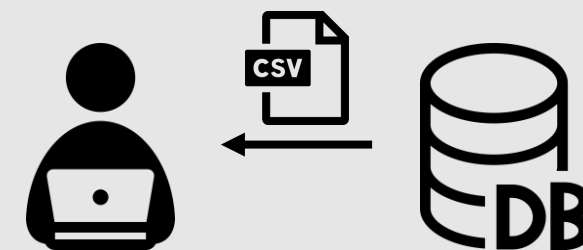
## 1. データベースへの接続



## 2. データベースの観察

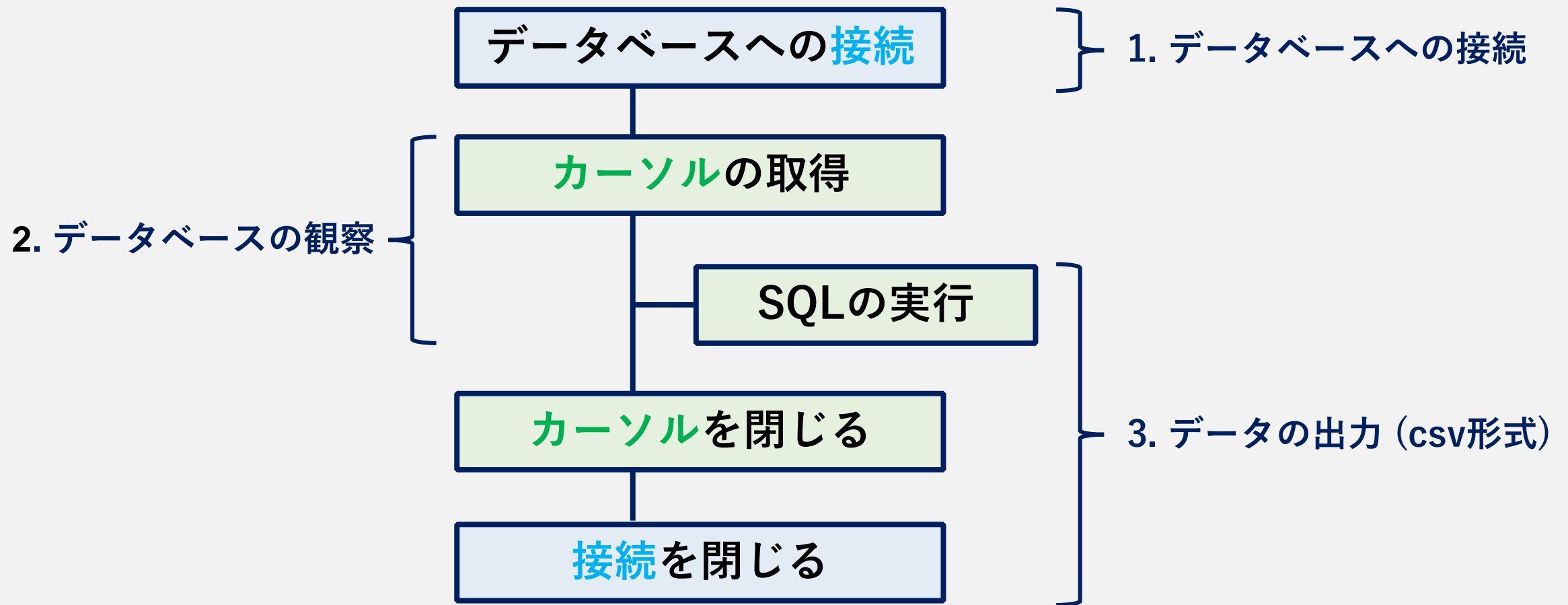


## 3. データの出力(csv形式)



# 0. データベース操作の概観

データベース操作は次の流れで行います。  
カーソルを取得し、イテレータ (p4) として扱うことが特徴です。



# 1. データベースへの接続

まずは、データベースに接続します。

ここでは、データベース「**trial.sqlite3**」が用意されているとします。

## 必要なモジュールのインポート

```
[1] import sqlite3  
import csv
```

## GoogleColabでのドライブのマウント

```
[2] from google.colab import drive  
drive.mount('/content/drive')
```

## データベースファイルのパスを指定

```
[3] dbpath = '/content/drive/---/trial.sqlite3'
```

## データベースへの接続

```
[4] conn = sqlite3.connect(dbpath)
```

データの読み込み方が不明な場合は、配布済みの「[ドライブマウント説明](#)」と「[フォルダ構造](#)」に関するpdfも参照ください。

データベースに接続されます。  
データベースが存在しない場合には、新規作成された上で接続されます。

## 2. データベースの観察

データベースに接続すると、テーブルに対して操作が可能となります。  
ここでは、出力に必要なデータベースの情報について確認します。

### カーソルの取得

```
[5] cur = conn.cursor()
```

SQLite3では、“**sqlite\_master**”  
にメタ情報が格納されています。

### テーブル名の確認

```
[6] cur.execute('select name from sqlite_master where type = "table"')  
iter_cur = cur.fetchall()  
for row in iter_cur:  
    print(row)
```

### イテレータ(反復子)とは

配列やそれに類似するデータ構造をもち、  
for文など反復的处理をする際に、**1件ずつ  
処理可能**なオブジェクト。

“iter\_cur” がテーブルのデータを1行ずつ返す  
**イテレータ**となっている。

```
( '<テーブル名1>', )  
( '<テーブル名2>', )
```

※テーブル数に合わせて、  
1件ずつ全件出力されます。

## 2. データベースの観察

### カラム名の確認 (CREATE TABLE文の確認)

```
[7] cur.execute('select sql from sqlite_master where type = "table"')
for row in cur.fetchall():
    print(row)
```

```
(CREATE TABLE "<テーブル名1>" (
    "<カラム名1>" データ型,
    "<カラム名2>" データ型
),)
(CREATE TABLE "<テーブル名2>" (
    "<カラム名1>" データ型,
    "<カラム名2>" データ型
),)
```

※テーブル数に合わせて、  
1件ずつ全件出力されます。

### データ型について

SQLiteには5種類のデータ型があります。

INTEGER ... 符号付き整数 (1,2,3,4,6,8 Byte)

REAL ... 浮動小数点 (8 Byte)

TEXT ... 文字列

BLOB ... Binary Large Objectの略。  
入力データのまま格納。

NULL ... NULL値

## 2. データベースの観察（補足：カーソルの扱い方）

ここでカーソルオブジェクト (`cur`) の関数について補足します.

**execute:** sqlite3上でSQLコマンドを実行する関数

```
[ - ] cur.execute("SQL コマンド")
```

**fetchall:** テーブルデータを1行ずつ所得するイテレータ (`iter_cur`) を返す関数

```
[ - ] iter_cur = cur.fetchall()
```

**fetchone:** テーブルデータを1行 (`row`) のみ返す関数

```
[ - ] row = cur.fetchone()
```

### 3. データの出力 (csv形式)

テーブル名とカラム名から、データの選択が可能になりました。  
まずは、区切り文字等を見るためにデータを出力します。

#### データの取得 (1件ずつ)

```
[8] cur.execute('select * from <テーブル名>')  
     print(cur.fetchone()) # 1レコード目  
     print(cur.fetchone()) # 2レコード目
```

#### データの取得 (全件)

```
[9] cur.execute('select * from <テーブル名>')  
     for row in cur.fetchall():  
         print(row)
```

※レコード数に合わせて、  
1件ずつ全件出力されます。

全件取得する場合、**GoogleColab**では**5000**行までしか表示されないことに  
注意してください。

### 3. データの出力 (csv形式)

続いて, **csv**形式でデータを出力します.

#### テーブルから特定の列を取得

```
[10] cur.execute('select <カラム名> from <テーブル名>')
with open('out.csv', 'w', newline = '') as csv_file:
    csv_writer = csv.writer(csv_file)
    csv_writer.writerow([i[0] for i in cur.description])
    csv_writer.writerows(cur)
```



#### openモジュール – 組み込み関数

open(**file**, **mode**, buffering, encoding, errors, **newline**, closed, opener)

file : ファイル名の指定

mode : 開くファイルのモード指定

    ‘r’ 読み込み/入力用(デフォルト), ‘w’ 書き込み/出力用

newline : 改行の変換先の指定

    ‘\n’ “変換なし



### 3. データの出力 (csv形式)

全ての列を取得する場合は、次のようになります。

テーブルから全ての列を取得

```
[11] cur.execute('select * from <テーブル名>')
with open('out.csv', 'w', newline = '') as csv_file:
    csv_writer = csv.writer(csv_file)
    csv_writer.writerow([i[0] for i in cur.description])
    csv_writer.writerows(cur)
```



#### csvモジュール

`csv.writer(csv_file, dialect)`

引数に`open()`で開いたcsvファイルを指定し、`csv_writer`を取得する。

`csv_writer.writerow(リスト)` ※ `cur.description`はカラムを提供

引数にリストを指定し、1行だけカンマ区切りでcsvファイルに書き込む。

`csv_writer.writerows(リスト)`

引数にリストを指定し、複数行をカンマ区切りでcsvファイルに書き込む。

### 3. データの出力 (csv形式)

以上で、処理を確定します。

処理を確定

```
[12] conn.commit()
```

左サイドバーに**csv**ファイルが出力されていることを確認してください。確認が出来たら、最後にカーソルと接続を閉じて終了します。

カーソルを閉じる

```
[13] cur.close()
```

接続を閉じる

```
[14] conn.close()
```

