1. テーブル定義，テーブル生成，問い合わせ（SQLite3，Python を使用）

URL: https://www.kkaneko.jp/cc/dbenshu/index.html

Python 言語からのリレーショナルデータベースの使用について説明する．リレーショナルデータベースは，複数のテーブルから構成される．テーブルは，数値，テキスト，日時などの情報を，テーブル形式で格納している．リレーショナルデータベースの各種操作を行う言語の世界標準が SQL である．ここでは，Python と SQL を組み合わせる．

■　データベース名

データベース名は，データベース接続時に，使用したいデータベースを指定するために使われる．

■ テーブル

リレーショナルデータベースでは，データベースをテーブルの集まり（collection of tables）として記述する．各テーブルには，識別に使うテーブル名があり，テーブルの各列には，識別に使う列名がある．

【テーブルの例】

id	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

■　テーブル定義

テーブル名，属性名の並び，各属性のデータ型，各属性の制約（たとえば主キー）などを記述すること．

■ SQL のデータ型

代表的なものを下にまとめる．

NULL: 空値 (a NULL value)

INTEGER: 符号付きの整数 (signed integer)

REAL: 浮動小数点値 (floating point value)

TEXT: 文字列 (text string)

■ 主キー

テーブルのタップルを唯一に識別するのに使える属性あるいは属性の組のうち極小なものをキーという．その中で，データベース管理者が，データベースの管理上最も適当と判断し，かつ，空値 (NULL) をとることがありえないものを，リレーションスキーマの設計時に選んだものが主キーである．

■ リレーション・スキーマ（スキーマともいう）(Relation schema)

R をテーブル名，A1, A2, …, An を属性名とするとき，リレーションスキーマを「R(A1, A2, …, An)」のように書く．

■ ユニコード文字列

複数行にわたるユニコード文字列を使いたいところでは，「u""" ・・・ """」のように書く．

■ Python の繰り返し処理

次のプログラムでは，変数 t はデータの集まりであるとする．row は t の各要素について処理を繰り返すための変数になる．

```
for row in t:
    print (row)
```

■ Python の SQL プレースホルダー

次のプログラムでは，SQL 文の中に，プレースホルダーを示す記号「?」が使われている．「?」の部分が「4」で置き換わり，"select * from iris where id = 4" が評価される．これは Python の機能である．

■ カーソル (Cursor)

カーソルはテーブルの各行を指し示すために使われる．

■ CSV ファイル CSV File

カンマで区切られたデータファイル．

Iris データセットのファイルは次のようなファイルである．

kaggle のデータセット： https://www.kaggle.com/datasets/uciml/iris から入手可能．

```
Id,SepalLengthCm,SepalWidthCm,PetalLengthCm,PetalWidthCm,Species
1,5.1,3.5,1.4,0.2,Iris-setosa
2,4.9,3.0,1.4,0.2,Iris-setosa
3,4.7,3.2,1.3,0.2,Iris-setosa
4,4.6,3.1,1.5,0.2,Iris-setosa
5,5.0,3.6,1.4,0.2,Iris-setosa
6,5.4,3.9,1.7,0.4,Iris-setosa
7,4.6,3.4,1.4,0.3,Iris-setosa
8,5.0,3.4,1.5,0.2,Iris-setosa
```

テーブル定義，テーブル生成，問い合わせ

① Python を起動する

python

② カレントディレクトリの確認

【プログラムの説明】

実行環境の確認のため，os モジュールを使用してカレントディレクトリを取得する．このパスはデータベ

ースファイルと CSV ファイルの保存場所となり，後続の処理で重要な役割を果たす．

```
import os
os.getcwd()
```

③ テーブル定義を行う，次の Python プログラムを実行
データ型の指定は，各列で，integer, real などのデータ型を指定している．
id は主キーであるので「primary key」を指定している．

【プログラムの説明】
SQLite3 データベースの初期設定とテーブル作成を行う．create table 文で各列の型を指定し，id を主キーに設定．ヒアドキュメント形式で SQL を記述することで，複数行の SQL を見やすく表現している．

```
import pandas as pd
import sqlite3
c = sqlite3.connect('hoge.sqlite')
sql = u"""
create table iris (
   id integer          primary key,
   sepal_length      real,
   sepal_width       real,
   petal_length      real,
   petal_width       real,
   species             text );
"""
c.execute(sql)
```

④ テーブル生成（空のテーブルにレコードを挿入）を行う，次の Python プログラムを実行
プログラム中の「?」は SQL プレースホルダーである．

【プログラムの説明】
pandas（pd.read_csv）で CSV ファイルを読み込み，SQLite テーブルにデータを格納する．SQL プレースホルダー（?）と iterrows()を組み合わせることで，安全かつ効率的なデータ挿入を実現している．

(1) オブジェクト x に CSV ファイルを読み込む

```
x = pd.read_csv('c:/iris.csv', header=0)
```

Windows でのファイル名「C:¥iris.csv」は，Python のプログラム中では「'C:/iris.csv'」のように書く．
読み込みたい CSV ファイルの先頭行に，「id, sepal_length, sepal_width, petal_length, petal_width, species」のようなヘッダーがない場合には「header=None」を付ける．
（今回は，ヘッダーがあるので，「header=None」を付けない）

(2) オブジェクト x に格納されたデータを iris テーブルに挿入する

```
for index, r in x.iterrows():
    sql = u"insert into iris values (?, ?, ?, ?, ?, ?)"
    c.execute(sql, (r[0], r[1], r[2], r[3], r[4], r[5]))
```

「for r in x:」は x の各行について繰り返すという Python プログラム.

「insert into iris values」は，テーブルに 1 行挿入するという SQL プログラム.

「?」は，SQL プレースホルダー.

「r[0], r[1], r[2], r[3], r[4], r[5]」は，もとの CSV データファイルの 0 列目，1 列目，2 列目，3 列目，4 列目，5 列目を使うという意味.

```
c.commit()
```

⑤ テーブルをすべて読みだす．カーソルを使う.

【プログラムの説明】

カーソルを使用してテーブルの内容を確認する．SQL 問い合わせ（select * from iris）の結果をカーソルで取得し，for ループで表示．データベース操作終了後は close()で確実に接続を終了する.

```
cur = c.cursor()
cur.execute(u"select * from iris")
for t in cur:
    print (t)
```

「select * from iris」は，SQL 問い合わせ.

「cur」は，カーソルである．問い合わせ結果を得るのに使う.

```
c.close()
```

⑥ exit() で終了

【プログラムの説明】

プログラムの終了処理．exit()で Python インタプリタを終了する.

```
exit()
```

⑦ さきほど調べたカレントディレクトリに，データベースファイル hoge.sqlite ができているので確認する.