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

データベース演習

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

概要

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 (Database Name)

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のに使われる

■ 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
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5	3.4	1.5	0.2	setosa
9	4.4	2.9	1.4	0.2	setosa

■ テーブル定義 (Table Definition)

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 (SQL Data Types)

代表的なものを下にまとめる。The followings are important SQL data types.

- **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，*A*1, *A*2, ..., *A*n を属性名とするとき， リレーションスキーマを「*R*(*A*1, *A*2, ..., *A*n)」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．

```
In [27]: cur.execute(u"select * from iris where id = ?", [4])
Out[27]: <sqlite3.Cursor at 0x66afa60>

In [28]: for t in cur:
    ...:     print t
    ...:
(4, 4.6, 3.1, 1.5, 0.2, u'setosa')
```

■ 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

■ CVS ファイル CSV File

カンマで区切られたデータファイル

準備

1. **Python のインストール**

   Windows での Python のインストールは次のページで説明している．

   https://www.kkaneko.jp/tools/win/python.html

2. **pandas のインストール**

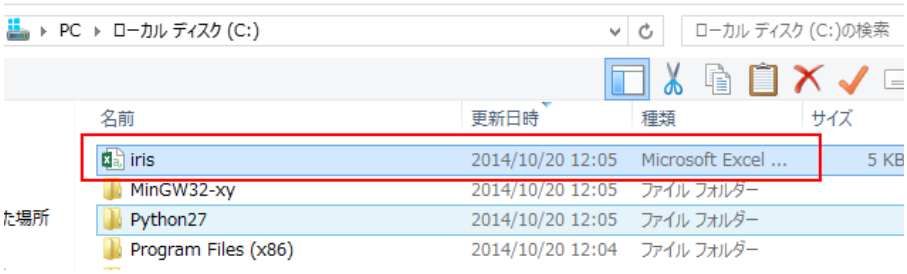
   Windows のコマンドプロンプトを**管理者として開き**，次のコマンドを実行．

   pip install -U pandas

CSV ファイル iris.csv のダウンロード

① Web ブラウザで https://www.kkaneko.jp/sample/iris.csv を開き、ダウンロードする

② ダウンロードされたファイル **iris.csv** を **C:¥iris.csv にコピー**



③ iris.csv は次のようなファイルである.

	A	B	C	D	E	F	G
1	id	sepal_leng	sepal_wid	petal_leng	petal_widt	species	
2	0	5.1	3.5	1.4	0.2	setosa	
3	1	4.9	3	1.4	0.2	setosa	
4	2	4.7	3.2	1.3	0.2	setosa	
5	3	4.6	3.1	1.5	0.2	setosa	
6	4	5	3.6	1.4	0.2	setosa	
7	5	5.4	3.9	1.7	0.4	setosa	
8	6	4.6	3.4	1.4	0.3	setosa	
9	7	5	3.4	1.5	0.2	setosa	
10	8	4.4	2.9	1.4	0.2	setosa	
11	9	4.9	3.1	1.5	0.1	setosa	
12	10	5.4	3.7	1.5	0.2	setosa	
13	11	4.8	3.4	1.6	0.2	setosa	
14	12	4.8	3	1.4	0.1	setosa	
15	13	4.3	3	1.1	0.1	setosa	
16	14	5.8	4	1.2	0.2	setosa	
17	15	5.7	4.4	1.5	0.4	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 **os.getcwd()** により**カレントディレクトリを確認.**

import os
os.getcwd()

```
C:¥Users¥user>python
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 23:03:10) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import os
>>> os.getcwd()
'C:¥¥Users¥¥user'
>>>
```

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.

※ 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.

```
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.

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

c.commit()
```

```
<sqlite3.Cursor object at 0x000001F00C7131F0>
<sqlite3.Cursor object at 0x000001F00C713570>
<sqlite3.Cursor object at 0x000001F00C7132D0>
<sqlite3.Cursor object at 0x000001F00C7131F0>
<sqlite3.Cursor object at 0x000001F00C713570>
<sqlite3.Cursor object at 0x000001F00C7132D0>
<sqlite3.Cursor object at 0x000001F00C7131F0>
>>> c.commit()
>>>
```

(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')
```

- 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

⑤ テーブルをすべて読みだす．カーソルを使う．_

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

c.close()
```

```
>>> for t in cur:
...     print (t)
...
(0, 5.1, 3.5, 1.4, 0.2, 'setosa')
(1, 4.9, 3.0, 1.4, 0.2, 'setosa')
(2, 4.7, 3.2, 1.3, 0.2, 'setosa')
(3, 4.6, 3.1, 1.5, 0.2, 'setosa')
(4, 5.0, 3.6, 1.4, 0.2, 'setosa')
(5, 5.4, 3.9, 1.7, 0.4, 'setosa')
(6, 4.6, 3.4, 1.4, 0.3, 'setosa')
(7, 5.0, 3.4, 1.5, 0.2, 'setosa')
```

- 「**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．

⑥ exit() で終了

```
>>> exit()

C:¥Users¥user>
```

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．

⑧ この hoge.sqlite は不要なので**削除**する．さきほどダウンロードした iris.csv も不要なので**削除**する．