

# Embedded Pythonで データベースプログラミング SQLアクセス編



# この資料の主な目的

---

この資料では、Embedded Pythonを使用して、IRIS にテーブルを作成し、SQL文でアクセスする方法、Python でストアドプロシージャを記述する方法を習得し、データベース内でPythonが実行できることをご理解いただきます。

具体的には、以下の内容を学習します。

- `irispython`でPythonシェルにログインする
- `iris`パッケージを使ってSQLを実行する
- `language=python` のストアドプロシージャを作って実行する
- `irispython` を使用して Python スクリプトファイルを動かす

# irispthonでPythonシェルにログインする

- IRISのインストールで用意されるirispthonコマンドを使用してPythonシェルにログインします。
- irispthonコマンドでログインしたシェルからは、IRISのUSERネームスペースにアクセスできます。
- irispthonコマンドは、[<インストールディレクトリ>/bin](#) 以下にあります。

例)

```
c:¥>cd intersystems¥iris¥bin
```

```
c:¥InterSystems¥IRIS¥bin>irispthon
```

```
Python 3.9.5 (default, Mar 11 2022, 10:30:25) [MSC v.1927 64 bit (AMD64)] on win32
```

```
Type "help", "copyright", "credits" or "license" for more information.
```

```
>>>
```

```
>>> import datetime
```

```
>>> dt=datetime.datetime(2022,4,25,12,30,15,1500)
```

```
>>> print(dt)
```

```
2022-04-25 12:30:15.001500
```

# 用語のふりかえり

## IRISのネームスペースとデータベースについて

ネームスペースは、仮想の作業環境で使いたいデータベースを指定する論理定義

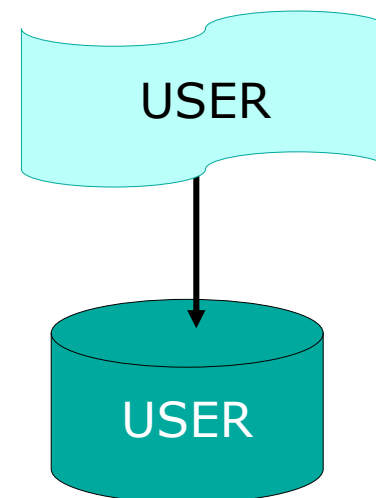
データベースは、データやロジック、クラスやテーブルのスキーマを格納する場所

ネームスペースを特定する



データベースが特定できる

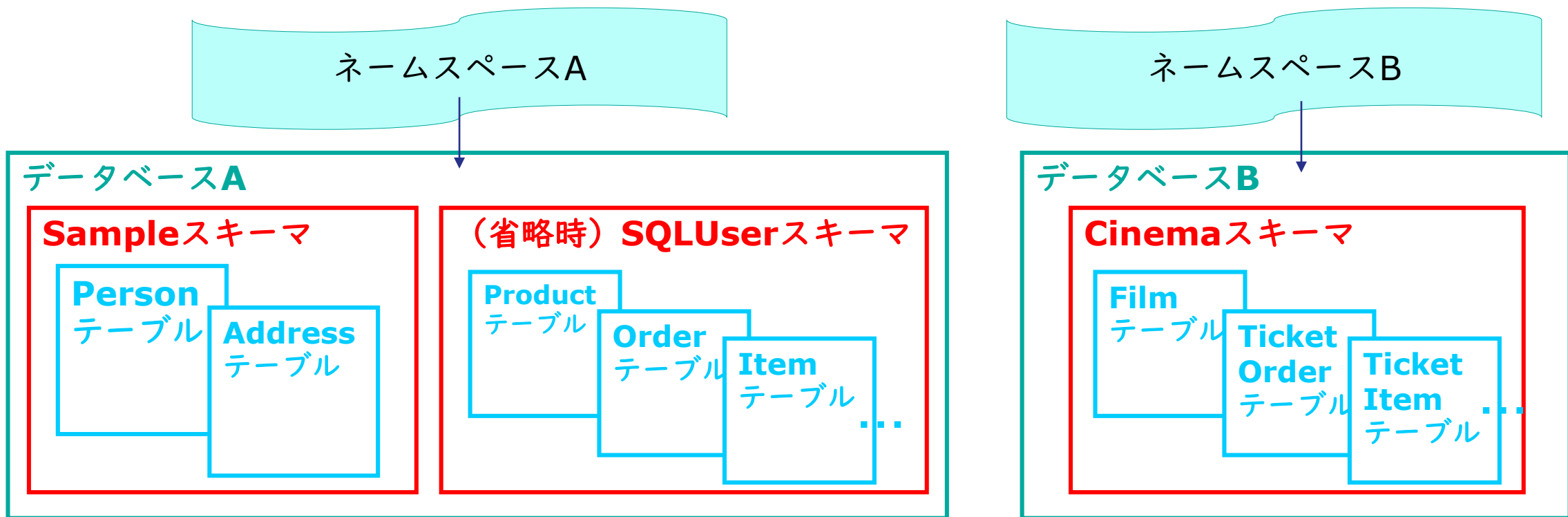
使いたい情報をすべて利用できる



# IRISにテーブルを作成する場合のルール

テーブルは、スキーマ名.テーブル名 で表現します。

スキーマ名は省略できますが、省略した場合は、システムデフォルトで設定されている **SQLUser** が使用されます (**SQLUser.テーブル名**)。



# テーブルを作ってみよう！その1

irispythonコマンドで起動したシェルから、以下の手順でCREATE TABLE文を実行してみましょう。

SQLの実行には、**iris.sql.exec()**メソッドを使用します。

1. irisモジュールをインポートします。

```
import iris
```

2. テーブルを作成します。

```
iris.sql.exec("CREATE TABLE Simple.Person (Name VARCHAR(50),Email VARCHAR(50))")
```

3. INSERT文を実行します。

```
rs=iris.sql.exec("INSERT INTO Simple.Person (Name,Email) VALUES('山田太郎','yamada@mail.com')")  
rs=iris.sql.exec("INSERT INTO Simple.Person (Name,Email) VALUES('杉村花子','sugi@mail.com')")
```

4. SELECT文を実行します。例では実行結果をdataframeに格納しています。

```
df=iris.sql.exec("SELECT * FROM Simple.Person").dataframe()
```

# 管理ポータルを使ってテーブルを参照してみよう！

## 管理ポータル > システムエクスプローラ > SQL

1. ネームスペースを確認し、USER以外に接続している場合は、Switchのリンクよりネームスペース選択画面を開き、変更します。

サーバ iijima-letsn2    **ネームスペース USER** [Switch](#)

### ネームスペース選択

ネームスペース選択

利用可能なネームスペース

%SYS  
TRAINING  
**USER**

キャンセル    **OK**

4. [フィールド]を選択し  
カラムの一覧を確認します。

2. スキーマで  
Simpleを選択

3. [テーブル]を展開し、  
Simple.Personを選択

**カタログの詳細**    クエリの実行    参照    SQLステートメント

テーブル: Simple.Person    ○ テーブル情報    **○ フィールド**    ○ マップ/インデックス    ○ トリガ

フィールド名	データタイプ	カラム#	必須	ユニーク	照合	隠し	最大長	MaxVal	MinVal	スカラー
ID	%Library.BigInt	1	Yes			Yes				No
Name	%Library.String	2								No
Email	%Library.String	3								No

5. [クエリの実行]でSQL  
文を実行できます。



# 管理ポータルを使ってテーブルを参照してみよう！

## 管理ポータル > システムエクスプローラ > **SQL**

システム > SQL

フィルタ Simple.\* ⊗適用先 すべて

システム ☐ スキーマ Simple

テーブル

> Simple.Person

ビュー

プロシージャ

キャッシュ

テーブル名をドラッグし、クエリ実行欄でドロップすると、全カラムをSELECTする文が準備できます。

ウィザード > アクション > テーブルを開く ツール > ドキュメント

カタログの詳細 クエリの実行 参照 SQLステートメント

実行 プラン表示 履歴を表示 クエリビルダ 表示モード 最大 1000

```
SELECT
Name, Email
FROM Simple.Person
```

行数: 2 パフォーマンス: 0.005 秒 324 グローバル参照 698 実行されたコマンド 0 ディスク読み込み

Name	Email
山田太郎	yamada@mail.com
杉村花子	sugi@mail.com

2 行が影響を受けました



# ここまでの流れで確認できたこと

---

- PythonからIRISに「接続するための記述」がないこと
- irispythonを使用してPythonシェルを起動すると、irisモジュールがインポートできること
- テーブル作成、データ作成、更新、検索など、普通にSQLでアクセスできること
- iris.sql.exec()メソッドでSQLを実行できること
- Python以外にも、IRISの管理ポータルでもSQLが実行できること

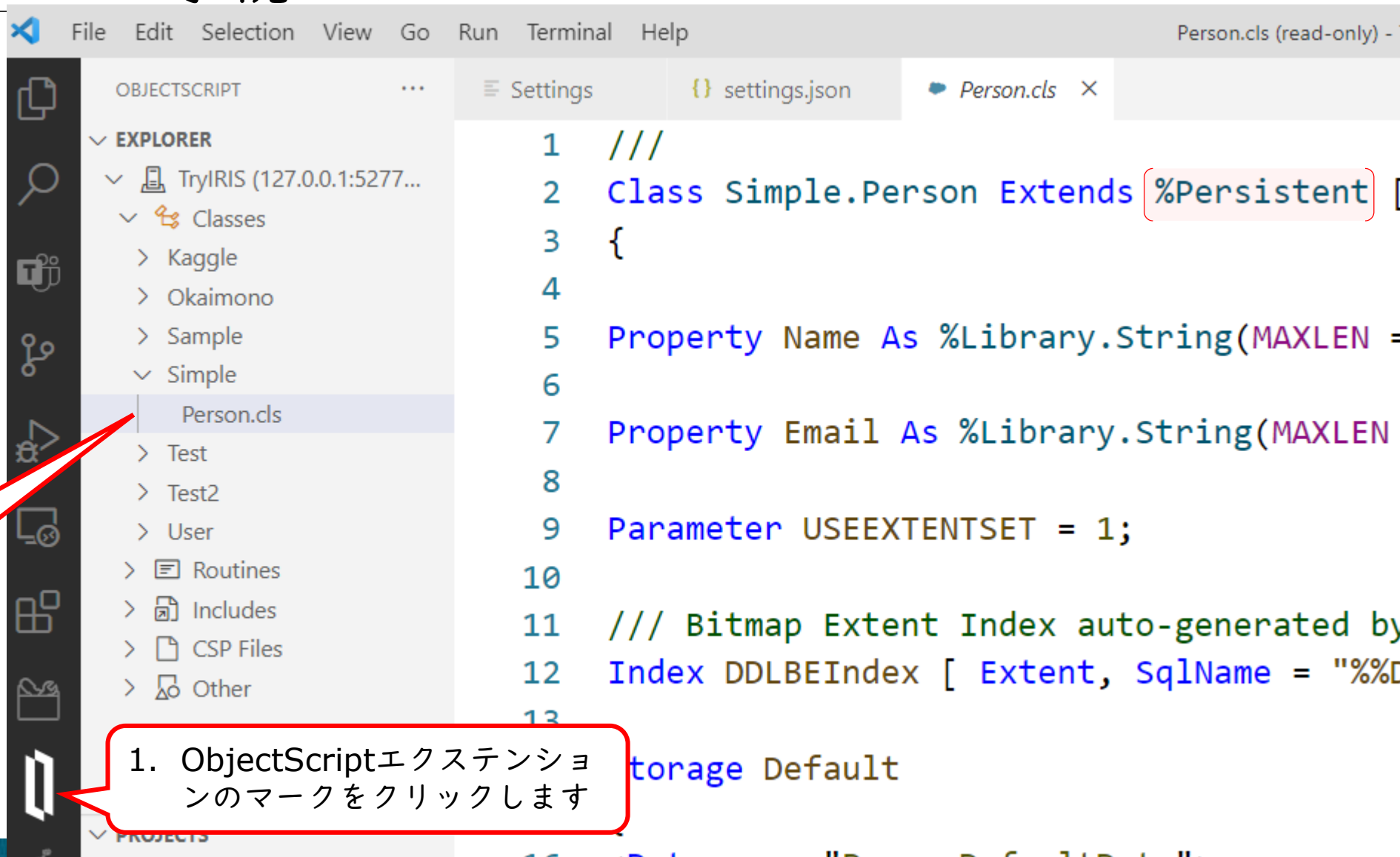
# ちょっと休憩

## 作成したテーブル=永続クラス

別添のVSCode-演習補足.pdf  
をご覧くださいながらテー  
ブルの作成により、永続クラス  
定義も用意されていることを  
確認してください。

2. Classes >  
Simple >  
Person.cls  
を選択します。

1. ObjectScriptエクステンシ  
ョンのマークをクリックします



The screenshot shows the VS Code interface. The Explorer sidebar on the left displays the project structure under 'TryIRIS (127.0.0.1:5277...)'. The 'Classes' folder is expanded, showing 'Kaggle', 'Okaimono', 'Sample', and 'Simple'. The 'Simple' folder is further expanded, showing 'Person.cls' selected. The editor window on the right shows the code for 'Person.cls' (read-only). The code is as follows:

```
1  ///  
2  Class Simple.Person Extends (%Persistent) [  
3  {  
4  
5  Property Name As %Library.String(MAXLEN =  
6  
7  Property Email As %Library.String(MAXLEN  
8  
9  Parameter USEEXTENTSET = 1;  
10  
11  /// Bitmap Extent Index auto-generated by  
12  Index DDLBEIndex [ Extent, SqlName = "%%  
13  
storage Default
```

# ここからは・・・

---

別のテーブル定義を作成しながら、以下の流れを確認します。

- 計算フィールドを用意し、ロジックをPythonで記述する。
- ストアドプロシージャを用意し、ロジックをPythonで記述する。

# Sample.Personテーブルを作成する

DOB（誕生日）から年齢は算出できるので、Ageカラムを計算フィールドとして定義します。

計算フィールドとは？

- アクセス時に計算（処理）結果を返すフィールドの定義
- IRISには、値を保持しないタイプと保持するタイプを選択できます。
  - 演習では値を保持しない**CALCULATED**キーワードを使用します。
- 値を保持するカラムの場合、**COMPUTEDONCHANGE**キーワードを利用して、以下の用途で処理を追加できます。
  - 他のカラム値が変更されたときに処理させる
  - INSERTやUPDATEの時だけ処理させる

## Sample.Person

Name VARCHAR(50)

DOB DATE

**Age INTEGER**

Ageカラムにアクセスしたとき、DOBカラムに登録された誕生日から年齢を算出する処理を記述します。

# Ageカラムの定義（計算フィールド）

指定するキーワードは以下の通りです。

- **CALCULATED**

年に1度、年齢が変わるので、アクセスされたときに処理が動くように、値を保持しない  
常時計算タイプのCALCULATEDキーワードを使用します。

- **COMPUTECODE PYTHON{}**

実際の処理内容が記述できます。COMPUTECODEキーワードの後にPYTHONを指定することで、PYTHONで処理内容を記述できます。

IRISのDATE型は、内部形式では数値で表現され、起源地（0）は1840年12月31日であるため、Pythonの先発グレゴリオ暦の数値表現を合わせるため、以下計算式を指定します。

- （本日の日付 - 1840年12月31日 - 誕生日） // 365

コードの中でテーブルの別カラム値を取得するには、**cols.getfield("カラム名")**を使用します。

```
((datetime.date.today().toordinal() - datetime.date(1840,12,31).toordinal() - cols.getfield("DOB"))) // 365
```

# CREATE TABLE文

---

```
CREATE TABLE Sample.Person (  
  Name VARCHAR(50),  
  DOB DATE,  
  Age INTEGER CALCULATED COMPUTECODE PYTHON {  
    if cols.getfield("DOB") == "":  
      return ""  
    import datetime  
    return ((datetime.date.today().toordinal() -  
datetime.date(1840,12,31).toordinal() - cols.getfield("DOB")) // 365) }  
)
```

# 引数のあるSQLを実行する方法

- 引数が入る場所の置き換え文字として **?** を使用し、  
**iris.sql.prepare()** メソッドを使用して、SQL実行の準備をします。

```
rs=iris.sql.prepare("INSERT INTO Sample.Person (Name,DOB) VALUES(?,?)")
```

- SQLの実行には、**execute()** メソッドを使用します。

```
dob=iris.system.SQL.TODATE("1999-12-31","YYYY-MM-DD")  
rs=iris.sql.execute("山田太郎",dob)
```

- IRISの内部日付を操作する場合は、以下メソッドを利用します。

```
#表示形式から内部形式
```

```
dob=iris.system.SQL.TODATE("1999-12-31","YYYY-MM-DD")
```

```
#内部形式から表示形式
```

```
irisdob=iris.system.SQL.TOCHAR(58073,"YYYY-MM-DD")
```



# 演習

---

1. CREATE TABLE文を実行し、テーブルを作成してください。  
実行は、irispythonコマンドで起動したPythonシェル上で実行してください。
2. INSERT文を前頁で紹介した**iris.sql.prepare()**メソッドを利用して1~2件登録して下さい。登録後、SELECT文を利用してAgeの値が算出できるかご確認ください。
3. Simple.Personと同様に、永続クラスSample.Person が作成されていることと、Ageカラムを算出する処理がどのように定義されているかご確認ください。

コピー元に演習サポート.txt をご利用ください

# 確認：永続クラス (Sample.Person)

```
2 Class Sample.Person Extends [%Persistent] [ ClassType = persistent, DdlAllowed, Final, Owner = {UnknownUser}, ProcedureBlock,
3 {
4
5 Property Name As %Library.String(MAXLEN = 50) [ SqlColumnNumber = 2 ];
6
7 Property DOB As %Library.Date [ SqlColumnNumber = 3 ];
8
9 Property Age As %Library.Integer(MAXVAL = 2147483647, MINVAL = -2147483648) [ [Calculated], SqlColumnNumber = 4, [SqlComputed]
10
11 Debug this method
12 ClassMethod AgeComputation(cols As %Library.PropertyHelper) As %Library.Integer [ Language = python ]
13 {
14     if cols.getfield("DOB") == "":
15         return ""
16     import datetime
17     return ((datetime.date.today().toordinal() - datetime.date(1840,12,31).toordinal() - cols.getfield("DOB")) // 365)
18 }
19
20 Parameter USEEXTENTSET = 1;
21
22 /// Bitmap Extent Index auto-generated by DDL CREATE TABLE statement. Do not edit the SqlName of this index.
23 Index DDLBEIndex [ Extent, SqlName = "%DDLBEIndex", Type = bitmap ];
24
25 Storage Default
```

ストアドや計算フィールド用コードは、  
クラスメソッドとして定義されます。

# ストアドプロシージャを追加します

ストアドプロシージャの処理はPythonで記述します。

SELECT文の結果をCSVファイルに出力する処理を記述します。

- dataframe()の実行のために、pandasモジュールのインポートが必要になります。「No module named 'pandas'」のエラーが発生した場合は、pandasモジュールをインストールしてください。

※Windowsの場合はirisipを使用してインストールします

```
c:¥InterSystems¥IRIS¥bin>irisip install --target c:¥intersystems¥iris¥mgr¥python pandas
```

入力引数にファイル名（フルパス）を指定できるように作成します。

```
CREATE PROCEDURE ToCSV(filename VARCHAR(50))
```

```
  (For Sample.Utills)
```

```
  LANGUAGE PYTHON
```

```
  {
```

```
    import iris
```

```
    df=iris.sql.exec("select Name,DOB,Age from Sample.Person").dataframe()
```

```
    df.to_csv(filename,encoding="utf-8")
```

```
  }
```

LANGUAGE PYTHON  
を指定します。

Forの後ろにストアドプロシージャの保存先クラス名を指定します。（任意指定ですが、指定することで複数のストアドプロシージャを1つのクラスにまとめて定義できます。クラスは、インスタンスが生成できるクラスとして定義されます。）

# ストアードプロシージャの追加と実行

irispythonコマンドでPythonシェルにログインし、以下実行します。（管理ポータルでも実行できます。）

プロシージャの追加

```
>>> import iris
>>> sql="""
... CREATE PROCEDURE ToCSV(filename VARCHAR(50))
... For Sample.Utils
... LANGUAGE PYTHON
... {
...     import iris
...     df=iris.sql.exec("select Name,DOB,Age from Sample.Person").dataframe()
...     df.to_csv(filename,encoding="utf-8")
... }
... """
>>> rs=iris.sql.exec(sql)
```

プロシージャの実行

```
>>> rs=iris.sql.exec("call Sample.ToCSV('c:¥¥temp¥¥test.csv')")
```

# 確認：ストアードプロシージャはクラスメソッドに

クラスメソッドとして定義され、ストアードプロシージャとして実行できるようにSqlProc属性が設定されます。

```
1 Class Sample.Utills Extends %Library.RegisteredObject [ DdlAllowed, Owner = {_SYSTEM}, Not ProcedureBlock ]
2 {
3
4     Debug this method
5     ClassMethod ToCSV(filename As %Library.String(MAXLEN=50)) [ Language = python, SqlName = ToCSV, SqlProc ]
6     {
7         import iris
8         df=iris.sql.exec("select Name,DOB,Age from Sample.Person").dataframe()
9         df.to_csv(filename,encoding="utf-8")
10    }
11
12 }
```

クラスメソッドとして実行することもできます。  
do ##class(Sample.Utills).ToCSV("c:¥temp¥test123.csv")

# Pythonスクリプトファイルを作成します

サンプルCSVファイルの中身をSample.PersonテーブルにINSERTする流れを作成します。

1. irisモジュール、pandasモジュールをインポートします。

```
import iris
import pandas as pd
```

2. pandasモジュールのread\_csv()を使用してCSVファイルからdataframeを作成します。

```
df=pd.read_csv(inputfile,encoding="utf-8")
```

3. INSERT文を**iris.sql.prepare()**を使用して準備します。

```
sql="INSERT INTO Sample.Person (Name,DOB) VALUES(?,?)"
stmt=iris.sql.prepare(sql)
```

4. dataframeをiterrows()を使用して1行ずつ取得しながらINSERTを実行 (**execute()**) します。

DOBはDATE型が設定されています。  
IRISの内部日付の数値に変換する必要があるため、IRIS内ユーティリティを実行しています。

```
for idx,row in df.iterrows():
    record=list(row)
    [record[1]=iris.system.SQL.TODATE(record[1],"YYYY-MM-DD")]
    stmt.execute(*record)
```

# irispythonコマンドを使用してスクリプトファイルを実行します

サンプルでは、fromCSV()関数を用意して第1引数に入力ファイルを指定できるように記載しています。

```
PS C:¥WorkSpace¥TryPython> c:¥intersystems¥iris¥bin¥irispython
Python 3.9.5 (default, Mar 11 2022, 10:30:25) [MSC v.1927 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import sys
>>> sys.path+=['C:¥WorkSpace¥TryPython']
>>> import script1
>>> script1.fromCSV("c:¥¥temp¥¥inputtest1.csv")
```

```
import iris
import pandas as pd

def fromCSV(inputfile):
    df=pd.read_csv(inputfile,encoding="utf-8")
    sql="INSERT INTO Sample.Person (Name,DOB) VALUES(?,?)"
    stmt=iris.sql.prepare(sql)
    for idx,row in df.iterrows():
        record=list(row)
        record[1]=iris.system.SQL.TODATE(record[1],"YYYY-MM-DD")
        stmt.execute(*record)
```



# ここまでの流れで確認できたこと

---

- Embedded Pythonを利用して、Pythonでストアドプロシージャや計算フィールドのロジックを記述できること
- 作成したストアドプロシージャや計算フィールドの処理は、クラスメソッドとしてIRIS内に定義されること
- irispythonコマンドを利用してirisモジュールの利用を含むPythonスクリプトファイルをOSから実行できること

# テーブル定義 = 永続クラス定義

## どちらも使えます (マルチモデルの利点)

```
Method Print() [ Language = python ]  
{  
    print(f"名前は : {self.Name}、年齢は{self.Age}")  
}
```

Objectでアクセス

### Sample.Person

Name As %String  
DOB As %Date  
Age As %Integer

AgeComputation()  
Print()

Name

DOB

Age

VARCHAR

DATE

INTEGER

SQLでアクセス

CREATE TABLEの実行で  
作成されたIRIS内永続ク  
ラス定義にインスタンス  
メソッドを追加し、実行  
させることもできます。

計算フィールドAge用ロジック (python)

オブジェクトアクセス

リレーショナルアクセス

テーブル/クラス定義 (=生成コードができます)

データ・インデックス用グローバル

コンパイルによってユーザ  
コードを含む「生成コード」  
が作成されます。

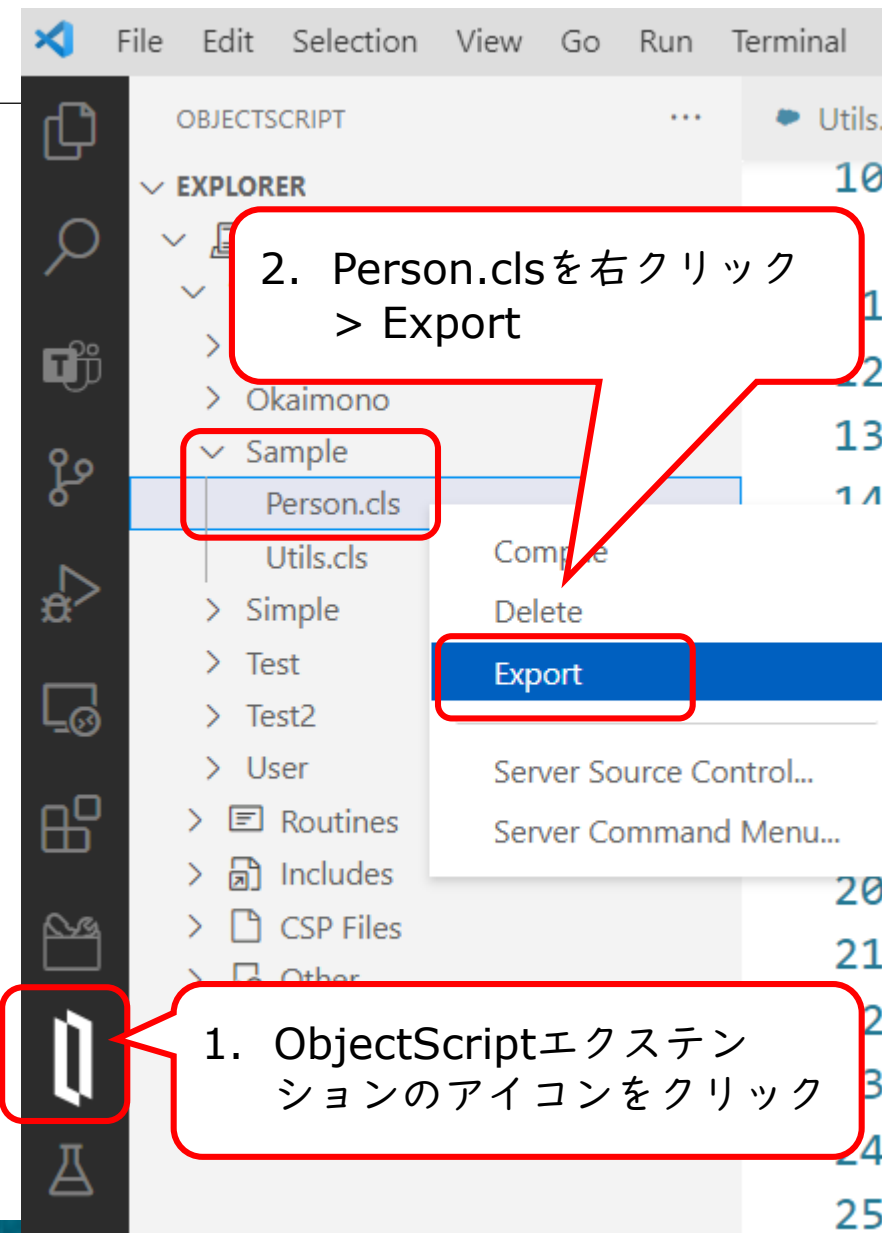
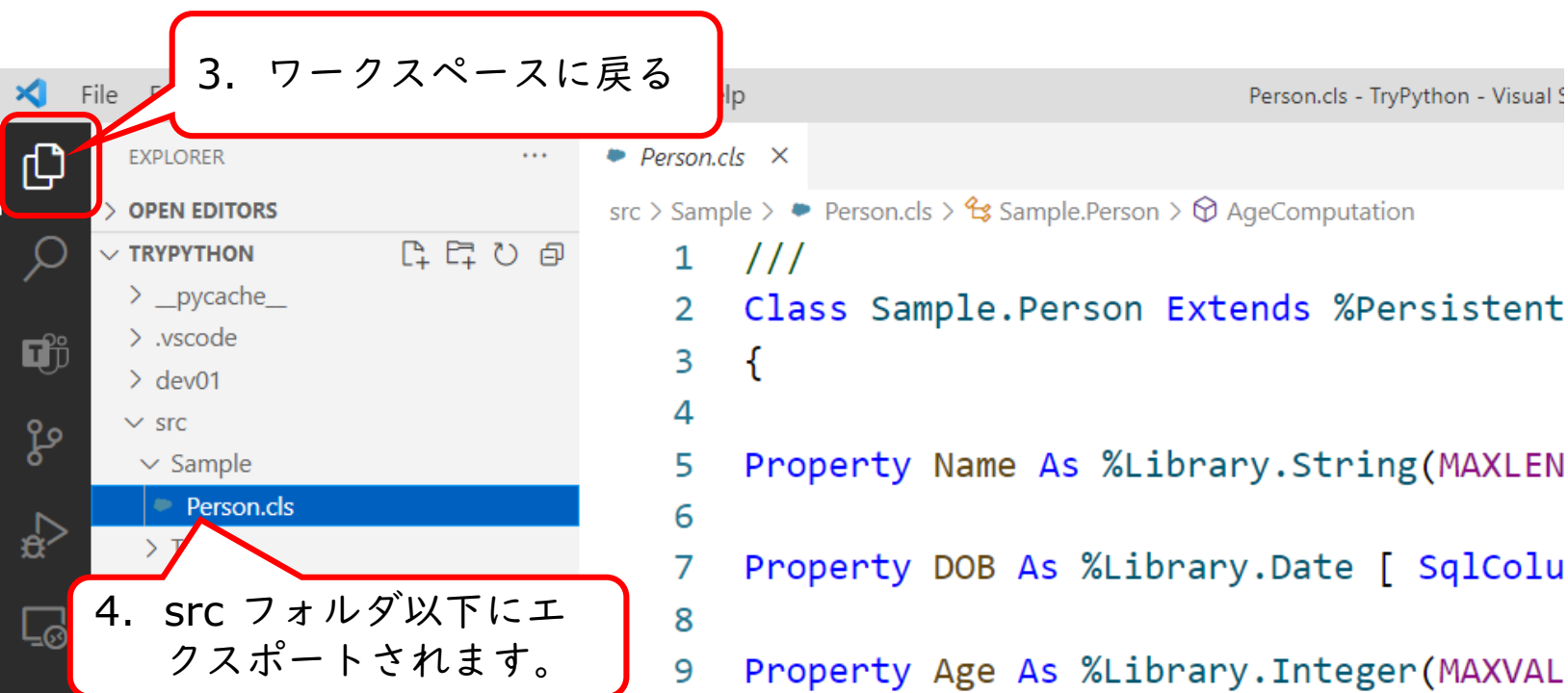
```
1: ^EW3K.wPC9.1 = 3  
2: ^EW3K.wPC9.1(1) = $1b("山田太郎",58073)  
3: ^EW3K.wPC9.1(2) = $1b("佐々木小太郎",57435)  
4: ^EW3K.wPC9.1(3) = $1b("鈴木花子",57810)
```

```
1: ^EW3K.wPC9.3(" 佐々木小太郎",2) = ""  
2: ^EW3K.wPC9.3(" 山田太郎",1) = ""  
3: ^EW3K.wPC9.3(" 鈴木花子",3) = ""
```

# おまけ：インスタンスメソッドを追加して実行する (ソースコードのエクスポート)

CREATE TABLEにより定義された永続クラスに、インスタンスメソッドを追加し、実行することもできます。

VSCodeでクラス定義を修正するために、サーバ側のSample.Personクラスをエクスポートします。



# おまけ：インスタンスメソッドを追加して実行する

Print()メソッドを以下のように追加し、コンパイルします（保存時にコンパイルも実施されます）。  
保存済オブジェクトをオープン／新規インスタンスを生成し、Print()メソッドを実行します。

- オブジェクト操作の体験について詳しくは別コースでご紹介しています。

src > Sample > Person.cls > Sample.Person > Print

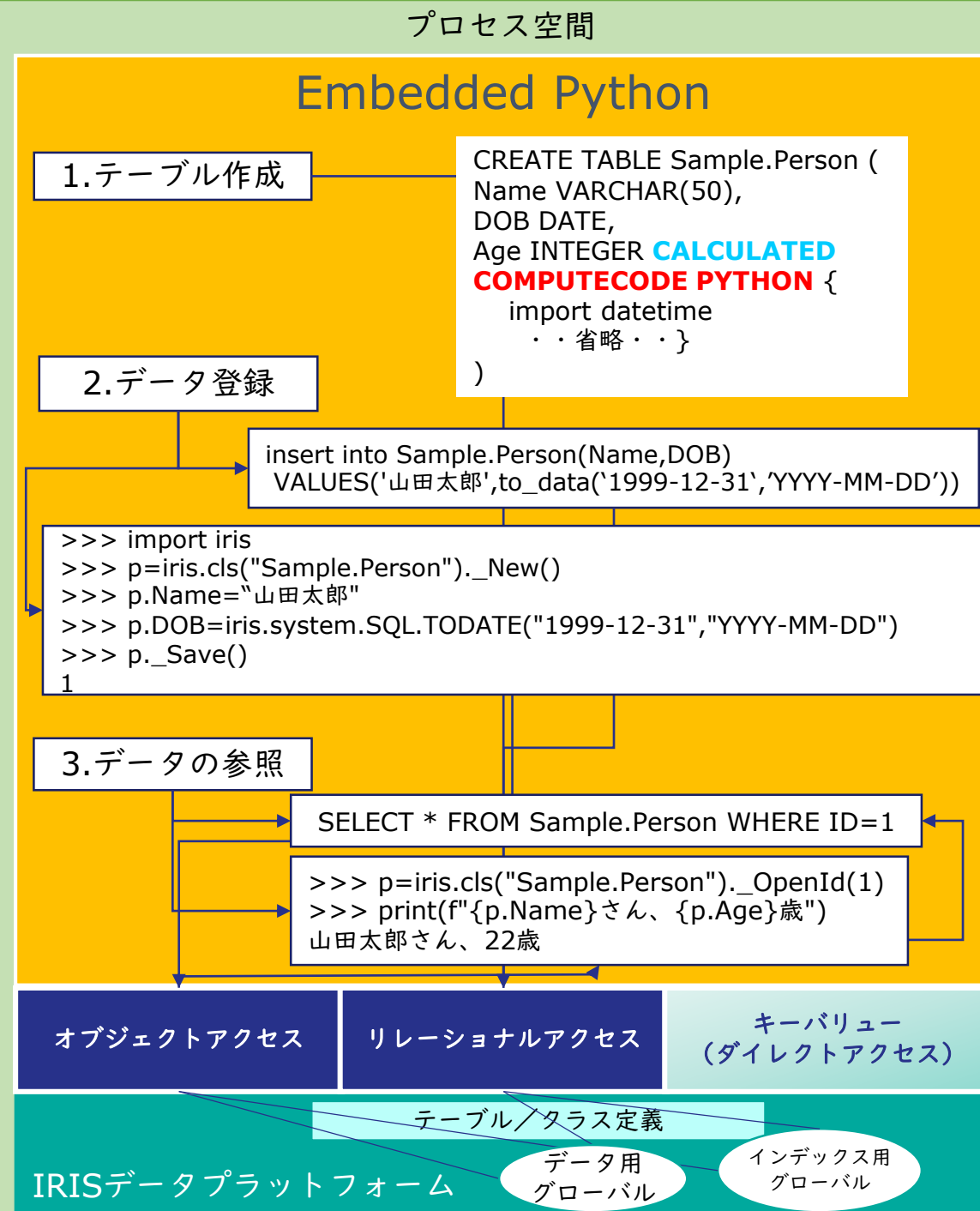
```
2 Class Sample.Person Extends %Persistent [ ClassType = persistent, DdlA
3 {
4
5 Property Name As %Library.String(MAXLEN = 50) [ SqlColumnNumber = 2 ];
6
7 Property DOB As %Library.Date [ SqlColumnNumber = 3 ];
8
9 Property Age As %Library.Integer(MAXVAL = 2147483647, MINVAL = -214748
10
11 Method Print() [ Language = python ]
12 {
13     print(f"名前は : {self.Name}、年齢は : {self.Age}")
14 }
15
```

```
>>> import iris
>>> obj=iris.cls("Sample.Person")._OpenId(1)
>>> obj.Print()
名前は：山田太郎、年齢は：22
```

# Python開発者からみた Embedded Python

データへのアクセスにはSQLを使用し、サーバー側ロジックにPythonを使用します。  
(お好みで、オブジェクト操作でデータにアクセスすることもできます。)

1. Pythonシェルやスクリプトファイル、管理ポータルでCREATE TABLE実行
  - 計算フィールドやストアドの処理の記述にPython使用可
2. データ登録
  - SQLで登録
  - オブジェクトで登録
3. データの参照
  - SELECT実行
  - オブジェクトオープン



# Embedded Pythonでデータベースプログラミング 【オブジェクトアクセス編】

---

## <コースの主な目標>

Embedded Pythonを使用して、IRIS の永続クラスの定義方法、オブジェクト操作方法、メソッドの記述と実行方法を習得し、1つのデータに対してSQLでもオブジェクトでもアクセスできる「マルチモデルデータベース」の特徴をご理解いただきます。

## <対象者>

- Pythonのプログラミング経験がある方
- IRIS／Cachéで開発を行っている方

## <コース内容>

- Embedded Python概要（サーバ側に組み込まれたPythonについて）
- 利用前の準備
- IRISのクラス定義作成練習
- PythonからIRIS内クラスに対するインスタンス操作練習
- language=pythonのメソッド記述と実行練習
  - 引数・戻り値のデータタイプについて
  - ObjectScriptのシステムクラスによくある%Statusの扱い

# その他情報について

---

コミュニティにも利用例が掲載されています。

- [1対多のリレーションシップを使った例（レシートの中身をIRISに登録する例）](#)
- [Excel のデータを IRIS グローバルに格納する方法](#)

最新情報については、以下タグにアクセスしてください。

- [#Embedded Python](#)

より良い方法を発見した場合、エラーが出た場合、使い方が不明な場合は、ぜひコミュニティへ投稿してください！