PythonエンジニアがIRISを使う場合の2つの方法と今後の計画

InterSystems
Creative data technology

Speaker

飯島 美穂子
教育サービス担当
mihoko.iijima@intersystems.com

トレーニングを担当しています（オンラインもやってます）。

Agenda & Benefits

ご紹介内容

- PythonからIRISへアクセスする方法と計画中の新機能についてご紹介します。

このセッションの主な目標

- 現在サポートしているPythonからIRISへアクセスする方法をご確認いただけます。
- IRISからPythonを実行できる新機能の予告もご覧いただけます。

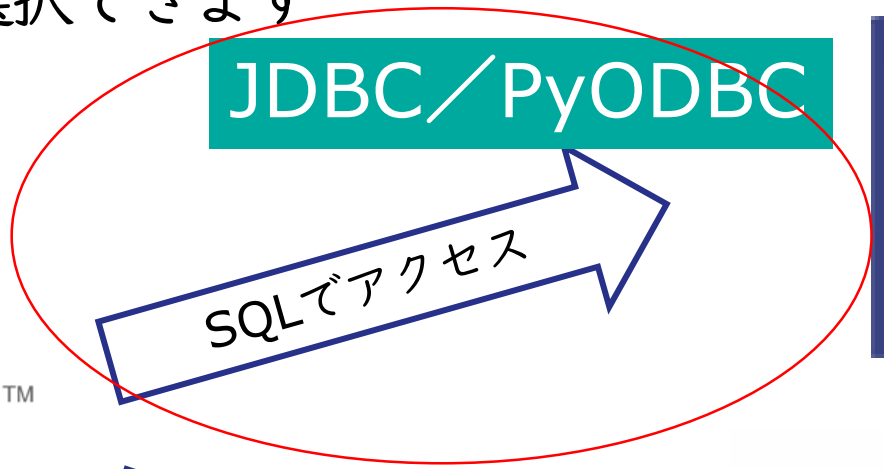
2つのアクセス方法

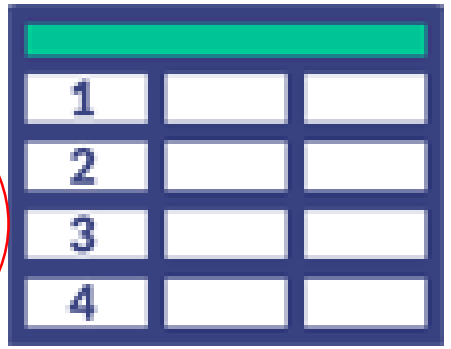
現在利用できる2種類の方法をご紹介します。

- SQLを使う場合の方法（JDBC／PyODBC）
- Native APIを使う方法（キーバリュー形式のデータにPythonからアクセスするためのAPI）

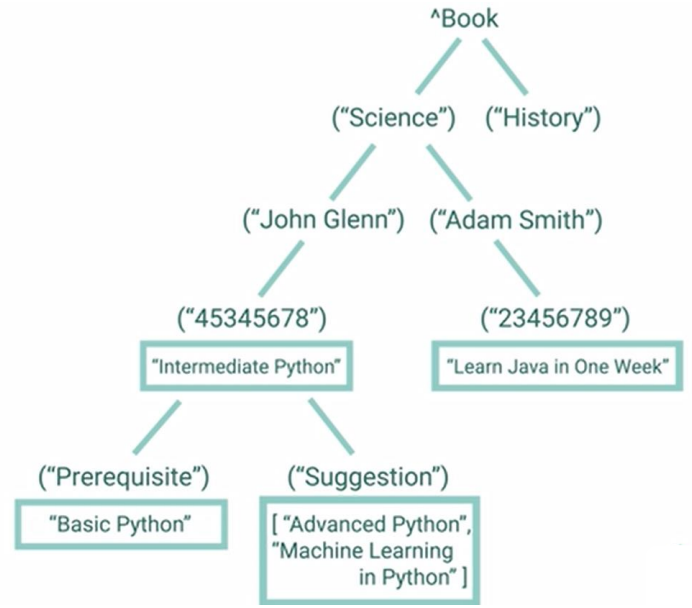
2つのアクセス方法

SQLかキーバリューを選択できます





Create
Update
Delete
Read
Call APIs



グローバル変数へのアクセス

クラスメソッドルーチン呼出

SQLでアクセスする場合

JDBCとPyODBCを利用できます。

（IRISのインストールでそれぞれのドライバは用意されます。）

- JDBC
  - コード例では、JayDeBeApi を利用してJDBC経由でIRISに接続しています。
- PyODBC
  - IRIS 2019.2から PyODBCをサポートしています。
    - PyODBCは、オープンソースの Python モジュールで ODBC を利用してデータベースに接続します。

SQLその1：JDBCの準備

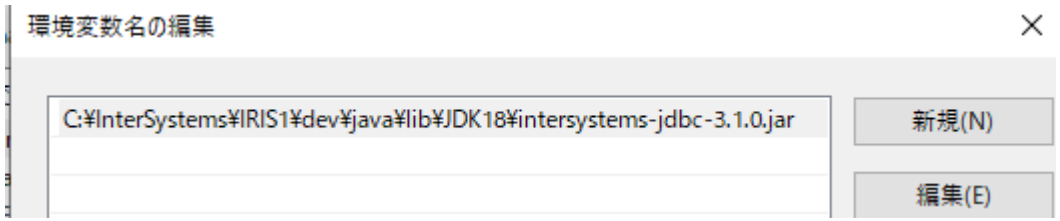
JDBCドライバをローカルのCLASSPATHに設定します。

IRISのインストール環境では、以下ディレクトリに用意があります。

<インストールディレクトリ>/dev/java/lib/JDK18

CLASSPATH設定は以下の通りです。

- インストールディレクトリ： c:¥InterSystems¥IRIS1




```
>SET CLASSPATH
ClassPath=C:¥InterSystems¥IRIS1¥dev¥java¥lib¥JDK18¥intersystems-jdbc-3.1.0.jar;
```

※WindowsにインストールしたIRISを使用しています

SQLその1：JDBCでアクセス

例では、Python の JayDeBeApi モジュールを利用して JDBC 経由で IRIS に接続しています。

```
import jaydebeapi
import sys
driver = "com.intersystems.jdbc.IRISDriver"
url = "jdbc:IRIS://localhost:1972/USER"
user="_system"
password="SYS"
jarfile="C:¥InterSystems¥IRIS1¥dev¥java¥lib¥JDK18¥intersystems-jdbc-3.1.0.jar"
connection = jaydebeapi.connect(driver, url, [user, password], jarfile)
```

jaydebeapi.connect()　関数の引数に接続文字列を指定し、サーバ接続用オブジェクトを作成します。接続文字列の指定内容詳細は以下の通りです。

ドライバ名	**com.intersystems.jdbc.IRISDriver**
URL	jdbc:IRIS://<ホスト名>:<ポート番号>/<ネームスペース名>
ユーザ名、パスワード	[ユーザ名,パスワード]
JARファイル	例：<IRISインストールディレクトリ>¥dev¥java¥lib¥JDK18¥intersystems-jdbc-3.1.0.jar

SQLその2：PyODBCの準備

- PyODBCドライバをインストールします。
  - ローカルに IRIS をインストールしている場合は、手続き不要です（インストール時に行っています）。
  - ローカルに IRIS をインストールしていない場合は、以下URLからファイルをダウンロードし、環境に合わせ、インストールコマンドを実行してください。

https://github.com/intersystems/quickstarts-python

Operating System	Command
Local instance	InterSystems IRIS PyODBC driver is installed. You can skip this step.
Windows	`pyodbc_wheel\ODBC-2018.1.1.635.0-win_x64.exe`
Mac	`odbcinst -i -d -f pyodbc_wheel/mac/odbcinst.ini`
Linux	`odbcinst -i -d -f pyodbc_wheel/linux/odbcinst.ini`

表の情報詳細は以下URLをご参照ください。
https://github.com/intersystems/quickstarts-python/#connect-to-intersystems-via-pyodbc

SQLその2：PyODBCで接続

```
import pyodbc
connection=pyodbc.connect("DRIVER={InterSystems IRIS ODBC35};
        SERVER=localhost;PORT=1972;DATABASE=user;UID=_system;PWD=SYS")
connection.setencoding(encoding='utf-8')
```

- **pyodbc.connect()**　関数の引数に接続文字列を指定し、サーバ接続用オブジェクトを作成します。
  - 接続文字列の区切りにセミコロンを指定します。指定内容詳細は以下の通りです。

ドライバ名	**DRIVER={InterSystems IRIS ODBC35}**
サーバ名	SERVER=サーバ名（またはIPアドレス）
ポート番号	PORT=スーパーサーバポート番号
ネームスペース名	DATABASE=ネームスペース名
ユーザ名	UID=ユーザ名
パスワード	PWD=パスワード

connection.setdecoding()　関数を利用して、返送データをutf-8でエンコードします。

接続後の操作は同じ

接続後は DB-API 2.0 に準拠した方法で操作します。

- JayDeBeApi（https://pypi.org/project/JayDeBeApi/）
- PyODBC（https://pypi.org/project/pyodbc/）

SELECTの結果をpandasのDataFrameへ

コネクションオブジェクト（connection）の **cursor()** 関数を使用して、カーソル（**cursor**）を作成し、**execute()** 関数を利用して、SQL文を実行し、**fetchall()** 関数で全結果を取得しています。

- 変数 **rows** は検索結果行が含まれるリストで、各行の結果はタプルで格納されています。
- タプルが格納されたリストから DataFrameを作成するため **rows** を numpy の行列に変換しています（np.array(**rows**)）。

```
import numpy as np
import pandas as pd
sql="SELECT Name,Email from Training.Person"
cursor = connection.cursor()
cursor.execute(sql)
rows=cursor.fetchall()
data = pd.DataFrame(np.array(rows))
data.columns = ['Name','Email']
```

この他、pandas の **read_sql()** 関数を利用して DataFrame にクエリの結果を格納する方法も利用できます。

```
sql_result = pd.read_sql(sql, connection)
```

INSERTの実行

引数は？で指定します。引数の値は、引数を指定した順序通りにタプルに格納し **execute()** 関数の第2引数に指定します。

```
sql="INSERT INTO Training.Person (Name,Email) VALUES(?,?)"
cursor = connection.cursor()
print("名前を入力")
p1 = input()
print("Emailを入力")
p2=input()
cursor.execute(sql,(p1,p2))
connection.commit()
```

1回の操作で複数行をINSERTする場合は、**executemany()** 関数を使用します。引数は複数のタプルをリストに格納して指定します。

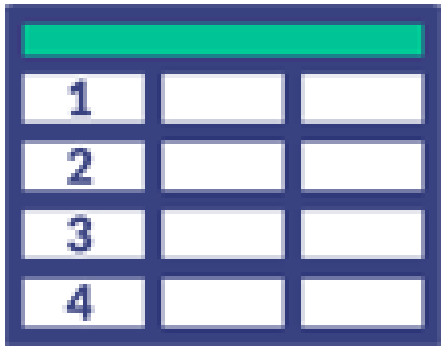
```
cursor.executemany(sql,[('引数1-1','引数1-2'),('引数2-1','引数2-2')])
```

2つのアクセス方法

SQLかキーバリューを選択できます

JDBC／PyODBC

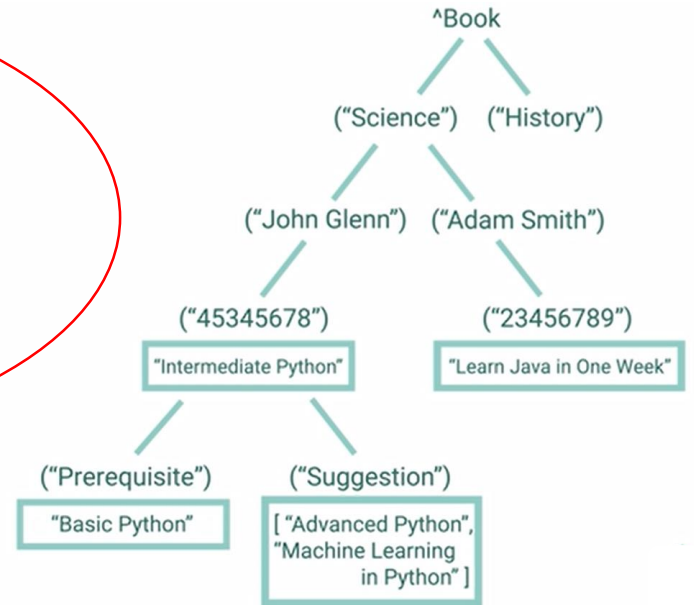
SQLでアクセス



Create
Update
Delete
Read
Call APIs

キーバリューでアクセス

Native API



グローバル変数へのアクセス

クラスメソッドルーチン呼出

Native APIでアクセスする場合

Native APIとは？

- レコードでもオブジェクトでもない **キーバリュー形式のIRISネイティブなデータ構造（グローバル）にアクセスする方法**です。
- グローバルは多次元配列でサーバ側では、SET／KILLコマンドでデータの設定／削除を行います。

都道府県別、**年別**のアルコール消費量をグローバルに格納した場合の例は以下の通りです。

※酒税の課税関係等状況表（4月～3月）https://www.nta.go.jp/taxes/sake/tokei/kanen.htm

^Alcohol(**都道府県名**,**年別**)=”種類1消費量,種類2消費量,・・・”

清酒,合成清酒,連続式蒸留焼酎,単式蒸留焼酎,ビール,発泡酒

- グローバル変数名の先頭に ^ が付きます。
- サブスクリプトには数字、文字列を指定できます。
- サブスクリプトは多次元で指定できます。
- サブスクリプトは **UNICODE 昇順で自動的にソートされます。**

```
1: ^Alcohol("大阪",2015) = "35652,1938,17138,36888,221896,68103,7289"
2: ^Alcohol("大阪",2016) = "33493,1728,15427,33322,213450,66450,7290"
3: ^Alcohol("大阪",2017) = "34615,1882,15516,33098,230974,65607,7293"
4: ^Alcohol("大阪",2018) = "33480,1782,15101,32398,227512,61287,7318"
5: ^Alcohol("大阪",2019) = "32734,1778,15325,31727,219077,56917,7330"
6: ^Alcohol("東京",2015) = "68265,4670,64819,43844,484347,72687,11258"
7: ^Alcohol("東京",2016) = "69694,4486,65155,44494,493290,76404,11328"
8: ^Alcohol("東京",2017) = "71864,4281,65155,46830,507098,73520,11390"
9: ^Alcohol("東京",2018) = "67036,4065,62657,43300,506951,76385,11531"
```

Native APIを利用するための準備

ご参考：https://github.com/intersystems/quickstarts-python

Native API 用パッケージ（**irisnative**）のインストールが必要です。

Native API 用パッケージのインストール

- ローカルにIRISをインストールしている場合は、以下ディレクトリにインストールされています。
  ＜インストールディレクトリ＞/dev/python
- ローカルにIRISをインストールしていない場合は、以下URLよりダウンロードします。
  github.com/intersystems/quickstarts-python/tree/master/Solutions/nativeAPI_wheel

irisnativeパッケージをインストールします。

- ご利用いただくPythonのバージョンとOSに合わせてインストールファイルを決定します。
- Windows上で Python 3 を使用している場合のコマンド実行例は以下の通りです。
  - OS毎のインストール方法については https://github.com/intersystems/quickstarts-python もご参照ください。

pip install *irisnative-1.0.0-cp34.cp35.cp36.cp37-none-win_amd64.whl*

Native APIでIRISに接続する

```
import irisnative
connection =
irisnative.createConnection("localhost",1972,"user","_system","SYS")
IrisObject = irisnative.createIris(connection)
```

- **irisnative.createConnection()**　メソッドを利用して、サーバ接続用オブジェクトを作成します。
  - 第1引数： ホスト名
  - 第2引数： ポート番号（スーパーサーバポート番号）
  - 第3引数： ネームスペース名
  - 第4引数： ユーザ名
  - 第5引数： パスワード
- **irisnative.createIris()**　メソッドを利用して新規でIRISオブジェクトを作成します。
  - 第1引数： createConenction()　メソッドで作成したサーバ接続用オブジェクト

グローバルに値を設定／参照する

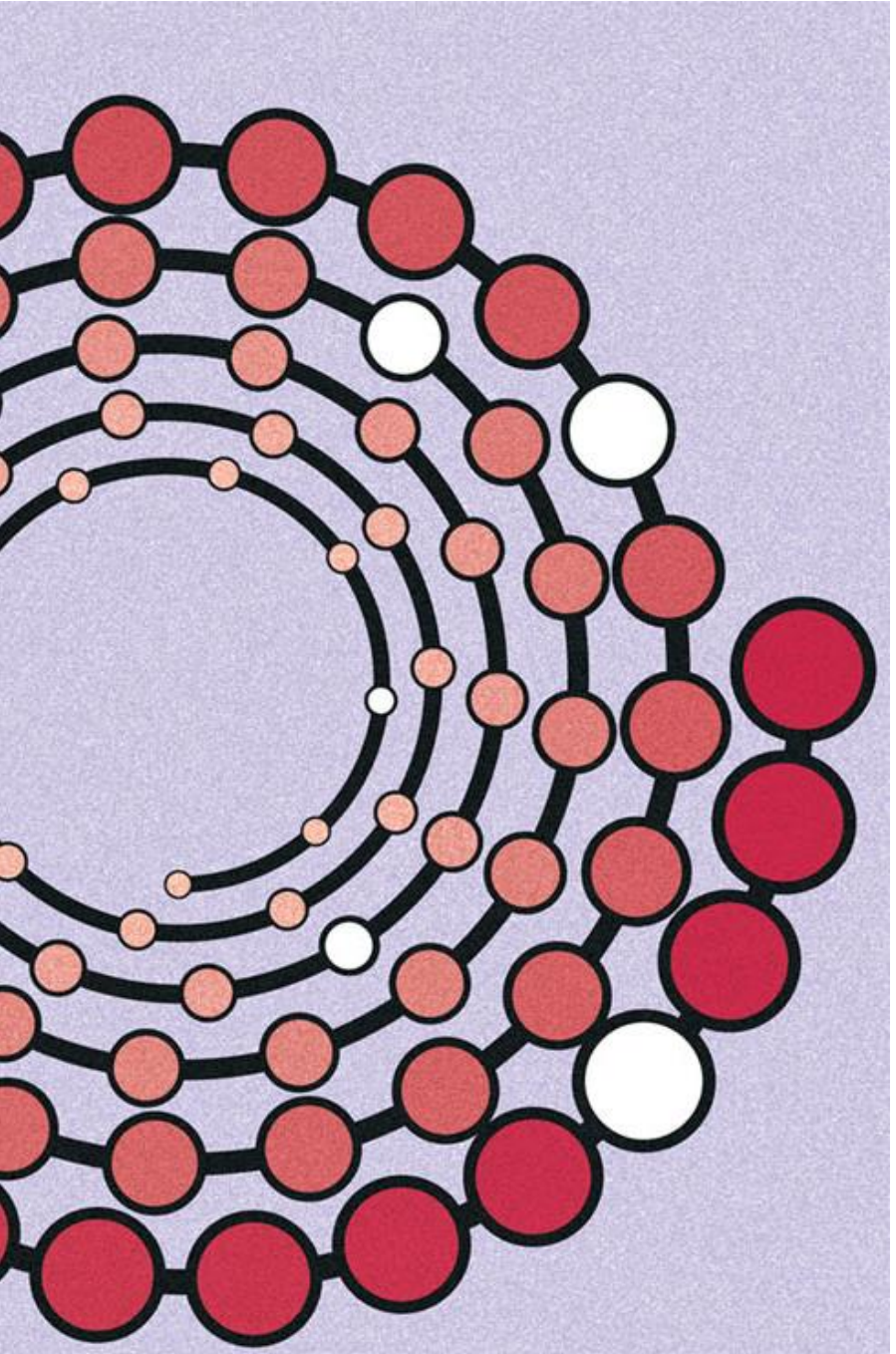
```
IrisObject.set("山田","employee","EMP010")
```

```
^employee("EMP010")="山田"
```

- **set**()メソッドを利用して指定のグローバル変数に値を設定します。
  - 第1引数　　：　設定したいデータ
  - 第2引数　　：　グローバル変数名
  - 第3引数以降　：　サブスクリプト

```
IrisObject.get("employee","EMP010")
```

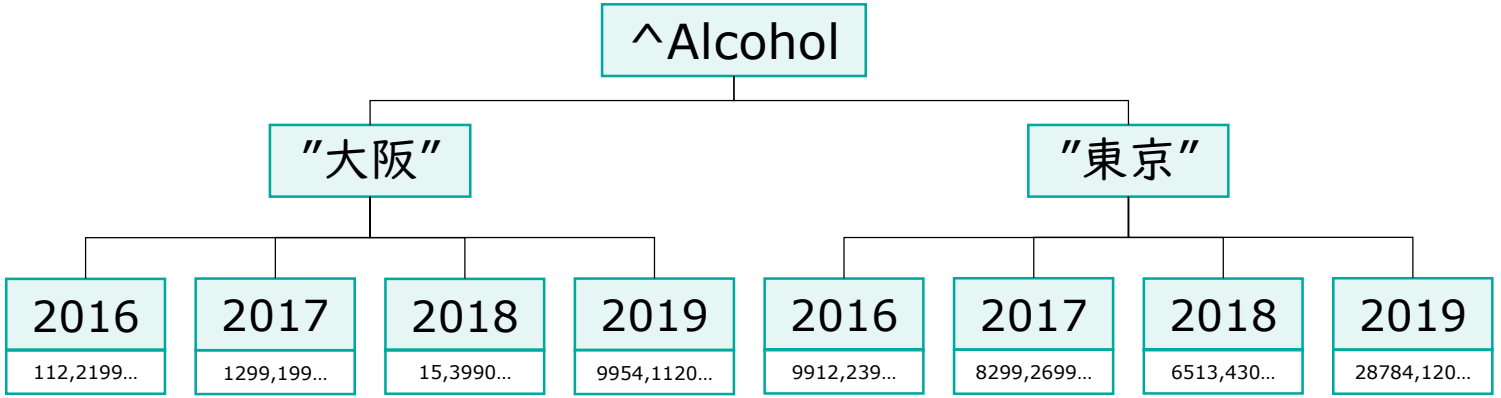
- **get**()メソッドを利用して指定のグローバル変数、サブスクリプトに格納された値を取得します。
  - 第1引数　　：　グローバル変数名
  - 第2引数以降　：　サブスクリプト



実演1

Native API の操作例をご覧いただきます。

シンプルなグローバル変数の操作例をご覧いただいた後、下図のような構造のグローバル変数を操作します（都道府県別、年別の酒類消費量の登録）。



Embedded Python

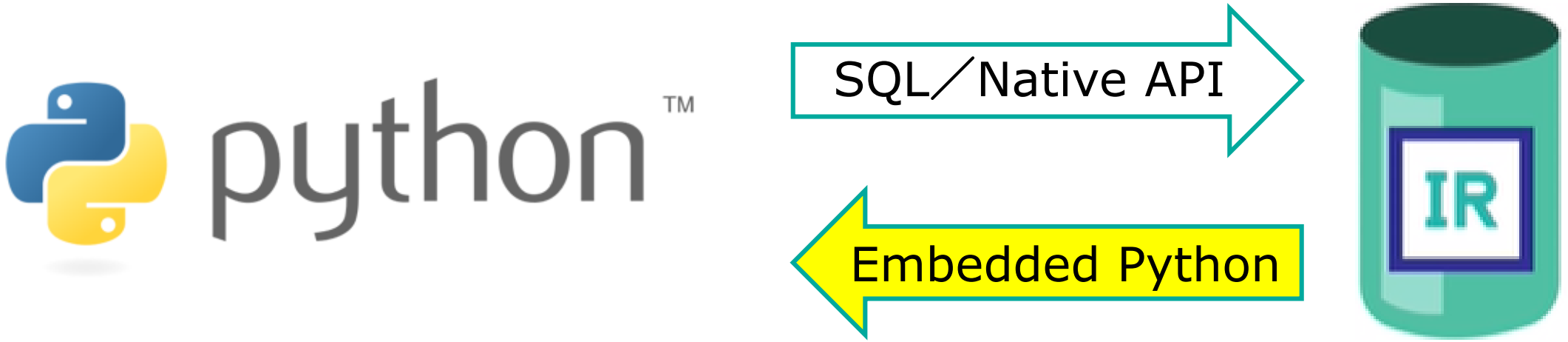
計画中機能のご紹介

IRISのサーバ側スクリプト（ObjectScript）からPythonを実行できる方法です。
ObjectScriptの開発者の視点でご紹介します。

- 開発中機能のため、操作方法や名称が変更される場合もあります。予めご了承ください。

Embedded Python

IRISのプロセス内で Python を実行できます。



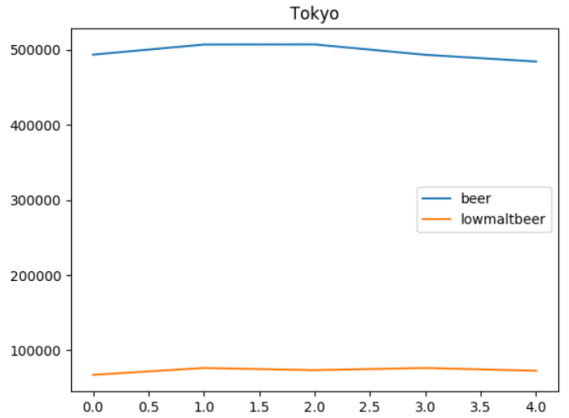
実行例1

ObjectScript で Pythonの datetime をインポート

```
USER>set dt=##class(%SYS.Python).Import("datetime")

USER>set today=dt.datetime.now()

USER>write today.year
2021
USER>write today.month
1
USER>write today.day
12
USER>write today.hour
16
USER>write today.minute
37
```

```
USER>set b=##class(%SYS.Python).Import("builtins")

USER>do b.print(today.year)
2021

USER>do b.print(today)
2021-01-12 16:37:19.798288
```

実行例2



SELECTの結果をDataFrameに格納しPythonに渡す。

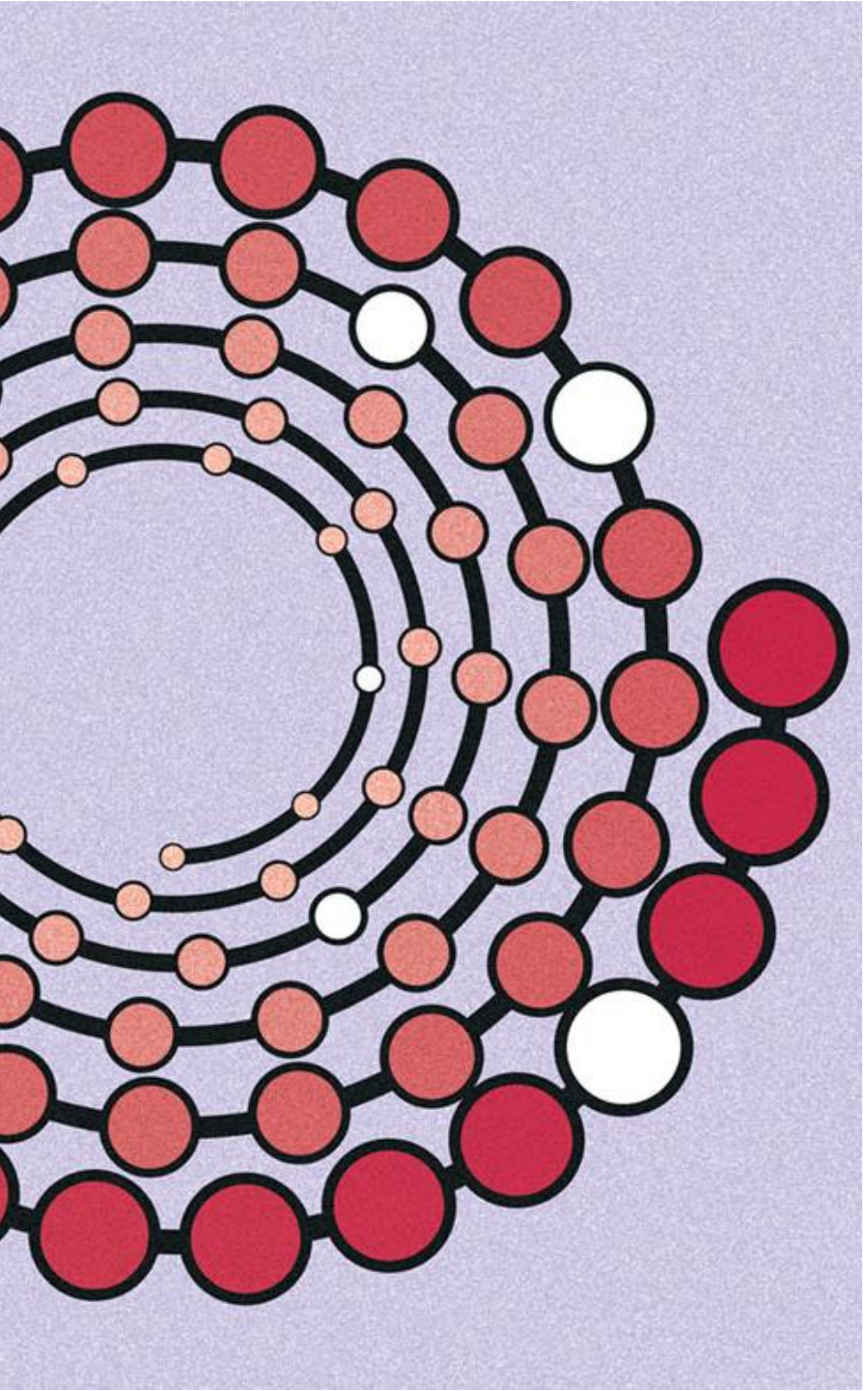
```
ClassMethod GetDF() As %SYS.Python
{
    set stmt=##class(%SQL.Statement).%New()
    set st=stmt.%Prepare("select Year,Beer,LowMaltBeer from VS2021.Alcohol")
    set rset=stmt.%Execute()
    set provider=##class(%ML.AutoML.Provider).%New()
    set status=provider.%ResultSetToDataFrame(rset,.info,.df,.count)
    return df
}
```

SELECTの結果セットを
DataFrameに変換できる
ユーティリティクラスを使用

```
//実行するbarchart.pyをインポート
set bar=##class(%SYS.Python).Import("barchart")
//ObjectScriptでメソッドを呼出しDataFrame作成
set df=##class(VS2021.Alcohol).GetDF()
//buildGraph()をにDataFrameを渡し実行
do bar.buildGraph(df,"/ISC/src/test2.png","beer","lowmaltbeer")
```

barchart.py

```
import pandas as pd
import matplotlib.pyplot as plt

def buildGraph(df,filename,xSeriesName,ySeriesName):
    df[[xSeriesName,ySeriesName]].plot(title="Tokyo")
    plt.savefig(filename)
```



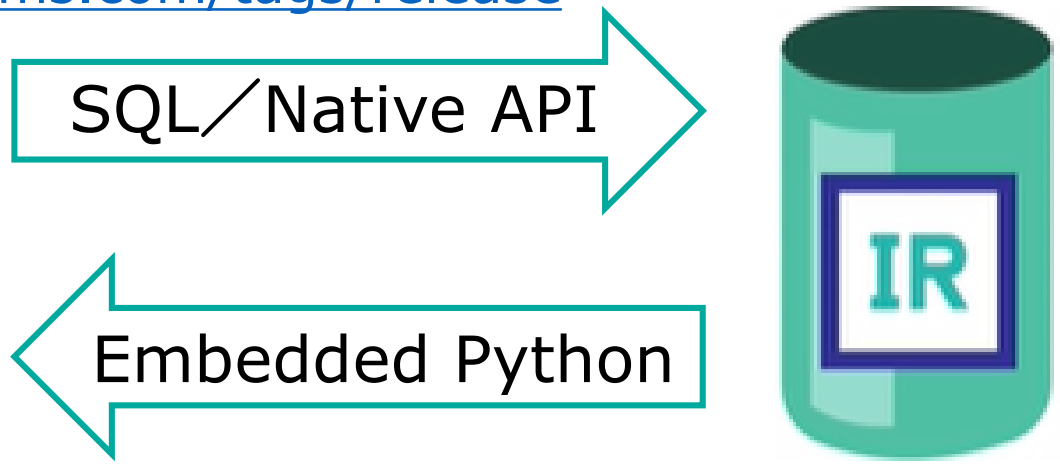
実演

Embedded Python

グローバル変数から、PythonのDataFrameオブジェクトを作成するコードをご覧いただきます。

Key Takeaway

- Python から IRIS へ SQL／キーバリューを利用してアクセスできることが確認できました。
  - SQL：JDBC／PyODBCでアクセスします。
  - キーバリュー： Native API でアクセスします。
- 将来のバージョン（2021.1）に含まれる予定の Embedded Python では、IRIS の ObjectScript から直接 Python を呼び出せることが確認できました。
  - 開発者コミュニティのリリース情報をぜひチェックしてください！（releaseタグ）
    https://jp.community.intersystems.com/tags/release



Next Steps

お勧めのビデオ

Embedded Python に関連する日本語字幕付きビデオ

- OD01[Embedded Python: Bring the Python Ecosystem to your ObjectScript App]
- OD05 [Embedding Python in SQL: Write Your Stored Procedures in Python]

Next Steps

お勧めの開発者コミュニティの記事

- Python Native APIでNoSQLデータベースにアクセス
  https://jp.community.intersystems.com/node/480707
- 【はじめての InterSystems IRIS】セルフラーニングビデオ：アクセス編：Python から **PyODBC** を使って IRIS に接続してみよう
  https://jp.community.intersystems.com/node/478616
- 【はじめての InterSystems IRIS】セルフラーニングビデオ：アクセス編：Python の **NativeAPI** に挑戦
  https://jp.community.intersystems.com/node/478611

jp.community.intersystems.com**/tags/python**

InterSystems 開発者コミュニティ

jp.community.intersystems.com

ぜひ、ご活用ください！

- **開発者同士の交流の場**として
  技術的な質問＆回答が行えます！
- **ヒントを探す場所**として
  jp.community.intersystems.com/tags/tips-tricks
- **学びの場**として
  セルフラーニングビデオ公開中！
  jp.community.intersystems.com/tags/beginner

InterSystems
Developer Community