InterSystems
Creative data technology

開発者向けオンラインセミナー

Pythonで データベースプログラミング

堀田 稔
インターシステムズジャパン株式会社

2023年4月26日

Pythonとは

- インタープリタ型の高水準汎用プログラミング言語 - Wikipedia
  - インタプリタ型の使い勝手とプリコンパイルによるパフォーマンスとの両立
  - 手続き型/オブジェクト指向/関数型といった複数のパラダイムをサポート
  - 読みやすく、シンプルに記述可能

- 豊富なライブラリ
  - 統計、機械学習、AI分野で人気化
  - 広大なエコシステム

www.python.org

プログラミング言語Pythonの特長（1）– 簡単な記述

- 変数の宣言は必要なく、弱い型付け
  - 実行時に変数の型がチェックされる
- 簡単で豊富な文字列処理
- 標準ライブラリで、数値計算、統計、ファイル処理、システム呼び出しなどをサポート

```
>>> number = 10
>>> print(number)
10
>>> string = '文字'
>>> print(string + number)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: can only concatenate str (not "int") to str
>>> print(string + str(number))
文字10
>>>
```

数値

文字列

文字列 + 数値はエラー

文字列 + 文字列は連結

```
>>> str = 'インターシステムズです'
>>> str[5]
'ス'
>>> str[2:5]
'ターシ'
>>> str[-3:]
'ズです'
>>> len(str)
11
```

(0から数えて) 5番目の文字

(0から数えて) 2番目の文字～4番目の文字

最後から3番目以降

文字列の長さ

```
>>> cities = '札幌 仙台 東京 名古屋 大阪 福岡'
>>> citylist = cities.split(' ')
>>> citylist
['札幌', '仙台', '東京', '名古屋', '大阪', '福岡']
>>> 'と'.join(citylist)
'札幌と仙台と東京と名古屋と大阪と福岡'
>>> 'and'.join(citylist)
'札幌and仙台and東京and名古屋and大阪and福岡'
```

‘ ‘で区切ってリストに変換

リストの要素を連結

プログラミング言語Pythonの特長 (2) – コレクション

- コレクション
  - リスト (List) :
    - 任意のデータを複数保持
    - 順序付き
    - 変更可能
  - タプル (Tuple) :
    - 変更できないリスト
  - 集合 (Set) :
    - 数学の集合
    - 重複を許さず、順序がない
  - 辞書 (Dictionary) :
    - キーと値のペアを複数保持
- **プログラミング = アルゴリズム + データ構造**

```
>>> lst1 = ['Cache', 'Ensemble']
>>> lst1
['Cache', 'Ensemble']
>>> lst2 = ['IRIS', 'HealthShare']
>>> lst2
['IRIS', 'HealthShare']
>>> lst1 + lst2
['Cache', 'Ensemble', 'IRIS', 'HealthShare']
```

リストの作成

リストの連結

```
>>> slist = list('インターシステムズ')
>>> slist
['イ', 'ン', 'タ', 'ー', 'シ', 'ス', 'テ', 'ム', 'ズ']
>>> slist[2:5]
['タ', 'ー', 'シ']
```

文字列から文字のリストを作成

リストの2番目～4番目の要素

```
>>> s1 = {'a', 'b', 'c'}
>>> s2 = {'b', 'e', 'e'}
>>> s1 & s2
{'b'}
>>> s1 | s2
{'e', 'c', 'b', 'a'}
>>>
>>> s1.add('b')
>>> s1
{'a', 'c', 'b'}
```

集合の作成

2つの集合のAND

2つの集合のOR

重複する要素は1つにまとめられる

プログラミング言語Pythonの特長 (2) – コレクション　（続き）

```
>>> tokyo = {'name': '東京', 'population': 14000000 }
>>> osaka = {'name': '大阪', 'population': 8800000 }
>>>
>>> cities = [tokyo, osaka]
>>> cities
[{'name': '東京', 'population': 14000000}, {'name': '大阪', 'population': 8800000}]
>>> for c in cities:
...     print(c['population'])
...
14000000
8800000
>>>
>>>
>>> pop = [ c['population'] for c in cities ]
>>> pop
[14000000, 8800000]
>>> sum(pop)
22800000
>>>
>>>
>>>
>>> json.dumps(cities, ensure_ascii=False)
'[{"name": "東京", "population": 14000000}, {"name": "大阪", "population": 8800000}]'
>>>
```

辞書の作成

2つの辞書を含むリストの作成

リストをループし、辞書の”population”を表示

内包表現で、辞書の”population”だけをリストに

populationの合計

JSON形式に変換

プログラミング言語Pythonの特長 (3) – Numpy

- NumPy (Numerical Python)は、数値計算や機械学習などで標準的に利用されているパッケージ

- ndarrayという数値の配列を扱う機能
  - 高速な演算
  - ループを書かずに配列を処理
  - 線形代数やフーリエ変換などの数学的計算を提供

```
>>> import numpy as np
>>>
>>> a = np.array([10, 2, 3.5])
>>> a
array([10. ,  2. ,  3.5])
>>> b = np.array([ [1, 3 ], [2, 4], [5, 6] ])
>>> b
array([[1, 3],
       [2, 4],
       [5, 6]])
>>> 3 * b
array([[ 3,  9],
       [ 6, 12],
       [15, 18]])
>>> a @ b
array([31.5, 59. ])
>>>
>>> np.where(b > 2, True, False)
array([[False,  True],
       [False,  True],
       [ True,  True]])
```

numpyをインポート

1x3行列(ベクトル)の作成

3x2行列の作成

行列 x 3

ベクトル x 行列

行列の要素に条件適用

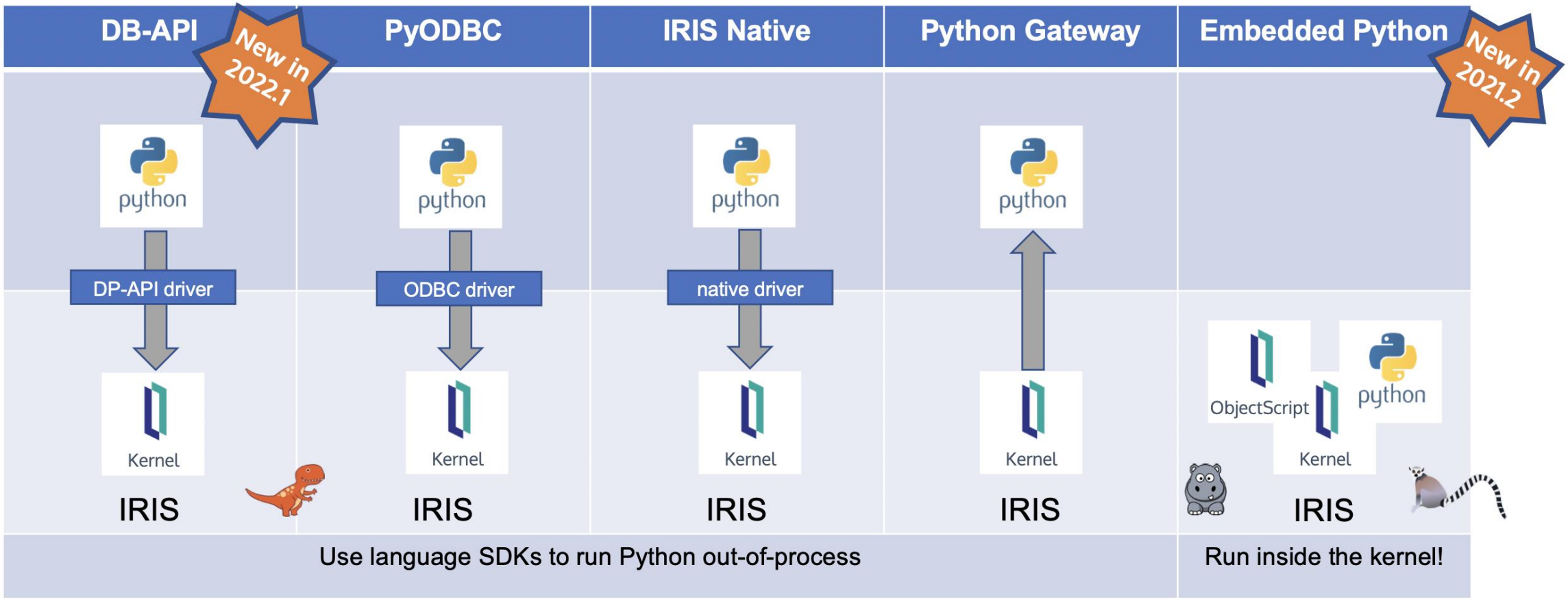
プログラミング言語Pythonの特長 (4) – Pandas

- Pandasは、データ分析を行うのに必要なデータ構造とデータ操作を提供するパッケージ

- 提供されるデータ構造
  - Series: あるデータ型の要素を複数保持する。テーブルのカラムに相当する。
  - DataFrame: テーブルのような構造。各カラムはSeries。

```
>>> import pandas as pd          pandasをインポート
>>>
>>> cities
[{'name': '東京', 'population': 14000000}, {'name': '大阪', 'population': 8800000},
{'name': '愛知', 'population': 7500000}]
>>> df = pd.DataFrame(cities)    辞書からDataFrameを作成
>>> df
  name  population
0   東京    14000000
1   大阪     8800000
2   愛知     7500000
>>>
>>> df['population']   “population’カラムだけを選択
0    14000000
1     8800000
2     7500000
Name: population, dtype: int64
>>>
>>> df[ df.population > 10000000 ]   “population’が10000000より大きい行を選択
  name  population
0   東京    14000000
>>>
>>> df.T    行と列を入れ替える
                0        1        2
name           東京       大阪       愛知
population  14000000  8800000  7500000
```

InterSystems IRISとPython



ユースケースの例

標準的なWebアプリケーションの開発。
データはリレーショナルモデルなので、
SQLを使用してデータを取得したい。

DB API
Or
PyODBC

Embedded
Python

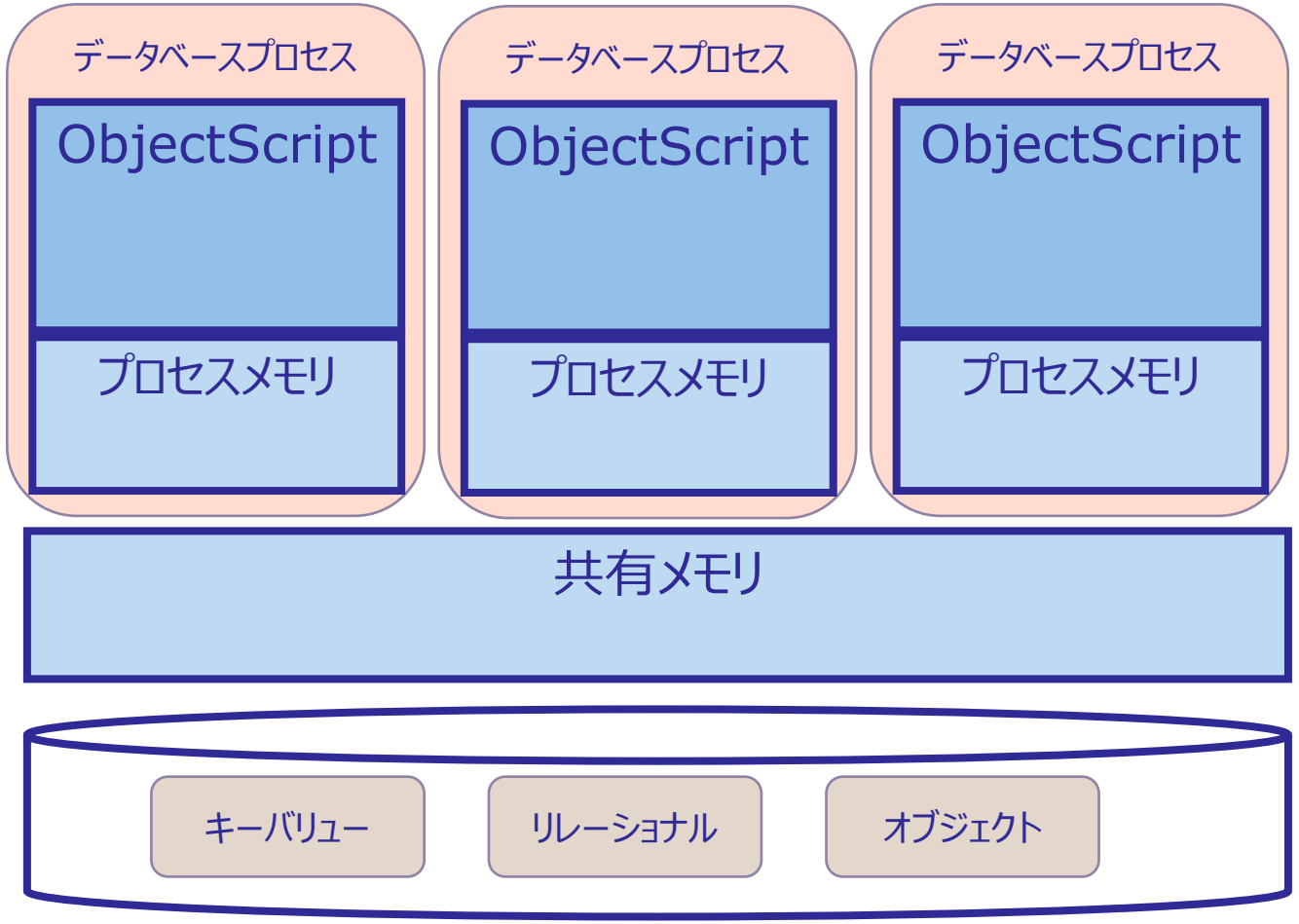
大量のデータを扱うPythonのプログラムがあり、高速かつ低いレイテンシーを実現したい。

ObjectScriptのアプリケーションがあり、新しい機能を追加する際、
Pythonで記述したい。
Pythonの豊富なライブラリを活用したい。

Embedded
Python

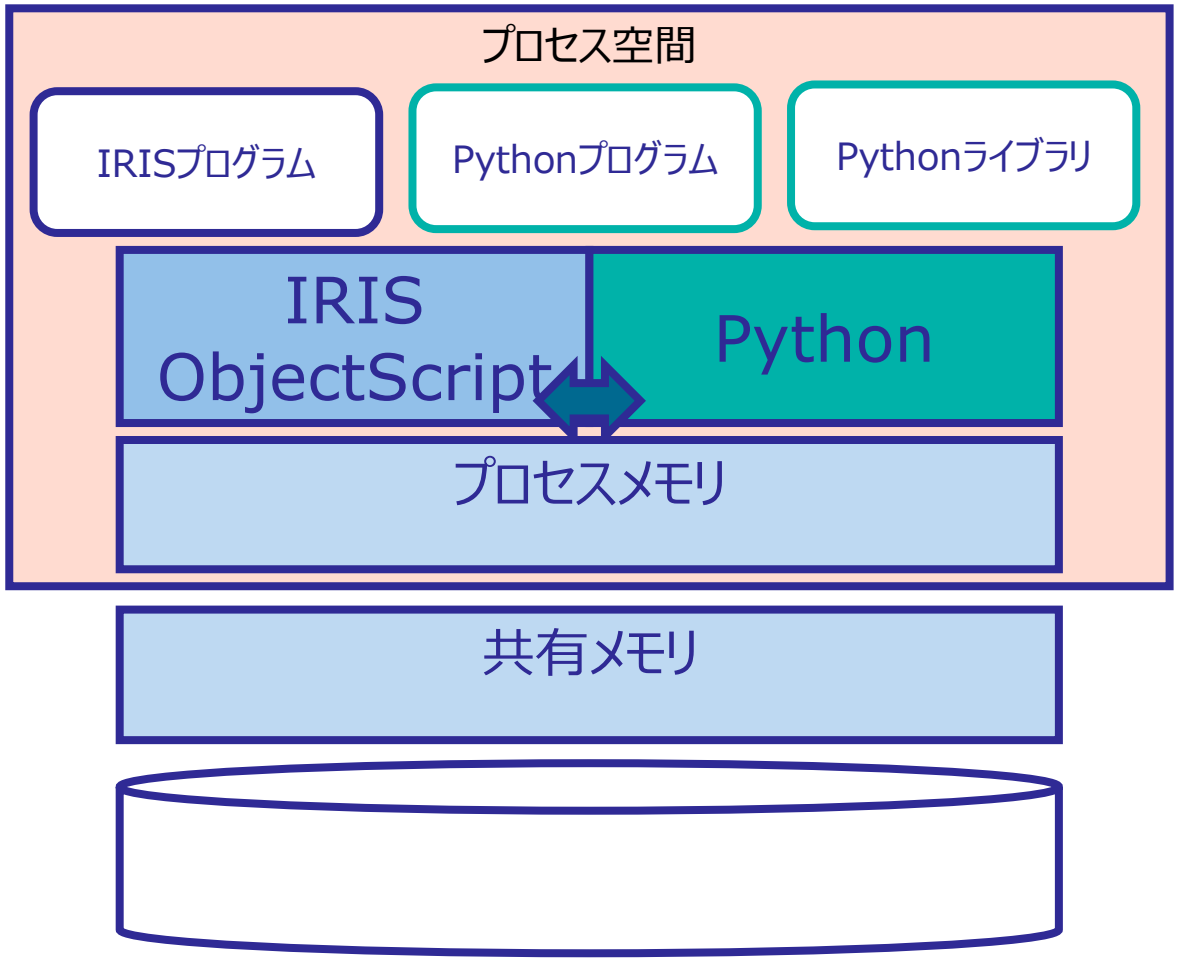
ObjectScript

- データベース内のプロセスに、ObjectScriptの処理系を持つ
- IRISのマルチモデルDBに、共有メモリを介してフルアクセス可能
  - オブジェクト
  - SQL
  - キーバリュー
  - ドキュメント(JSON)

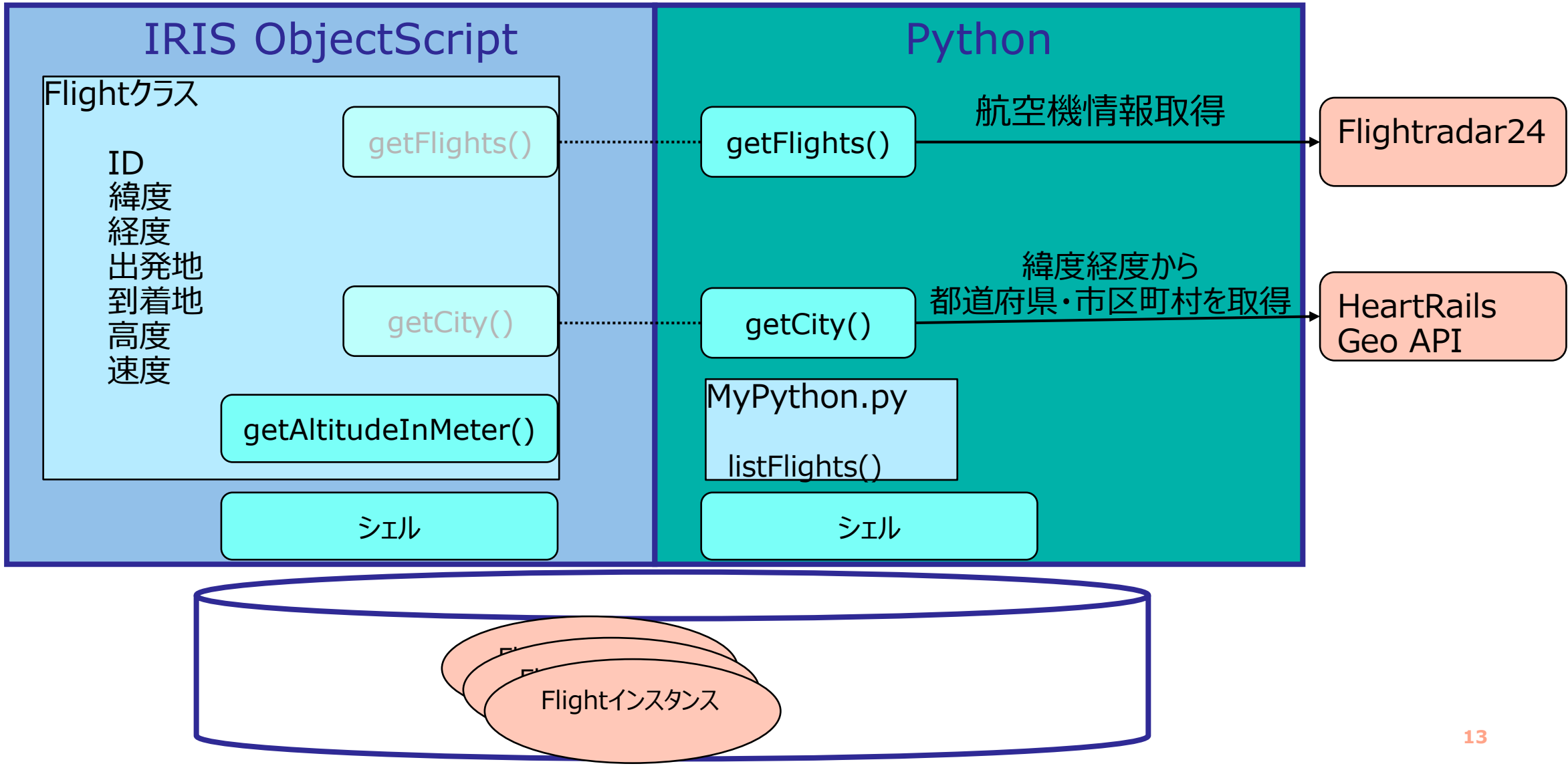


Embedded Python

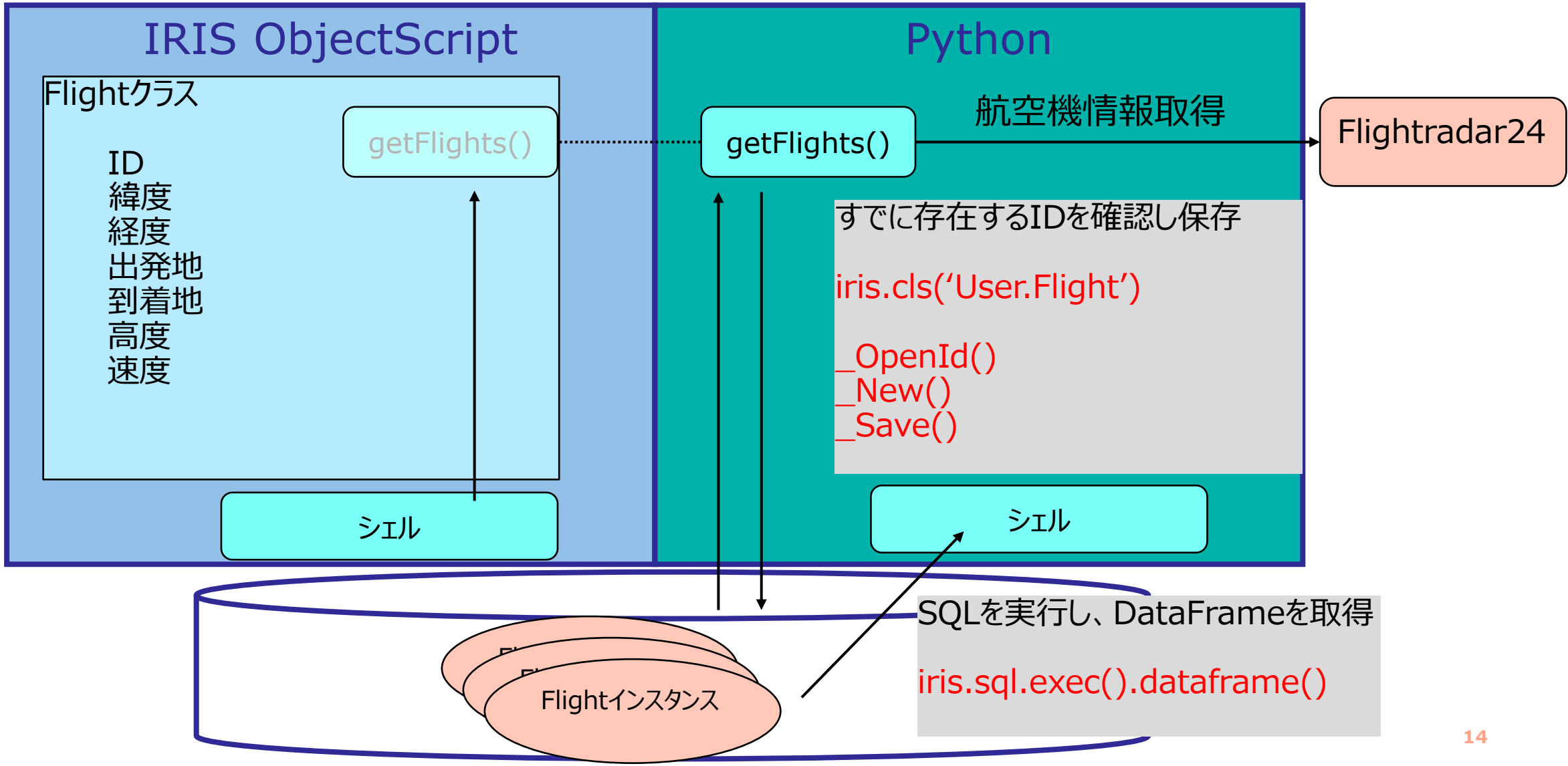
- PythonのランタイムをIRISの言語処理モジュールに組み込み
- 膨大なPythonの資産をIRISの「ネイティブ」コードとして利用可能
- PythonプログラマがIRISデータプラットフォームの機能を活用するのが容易
- 技術者の確保が容易



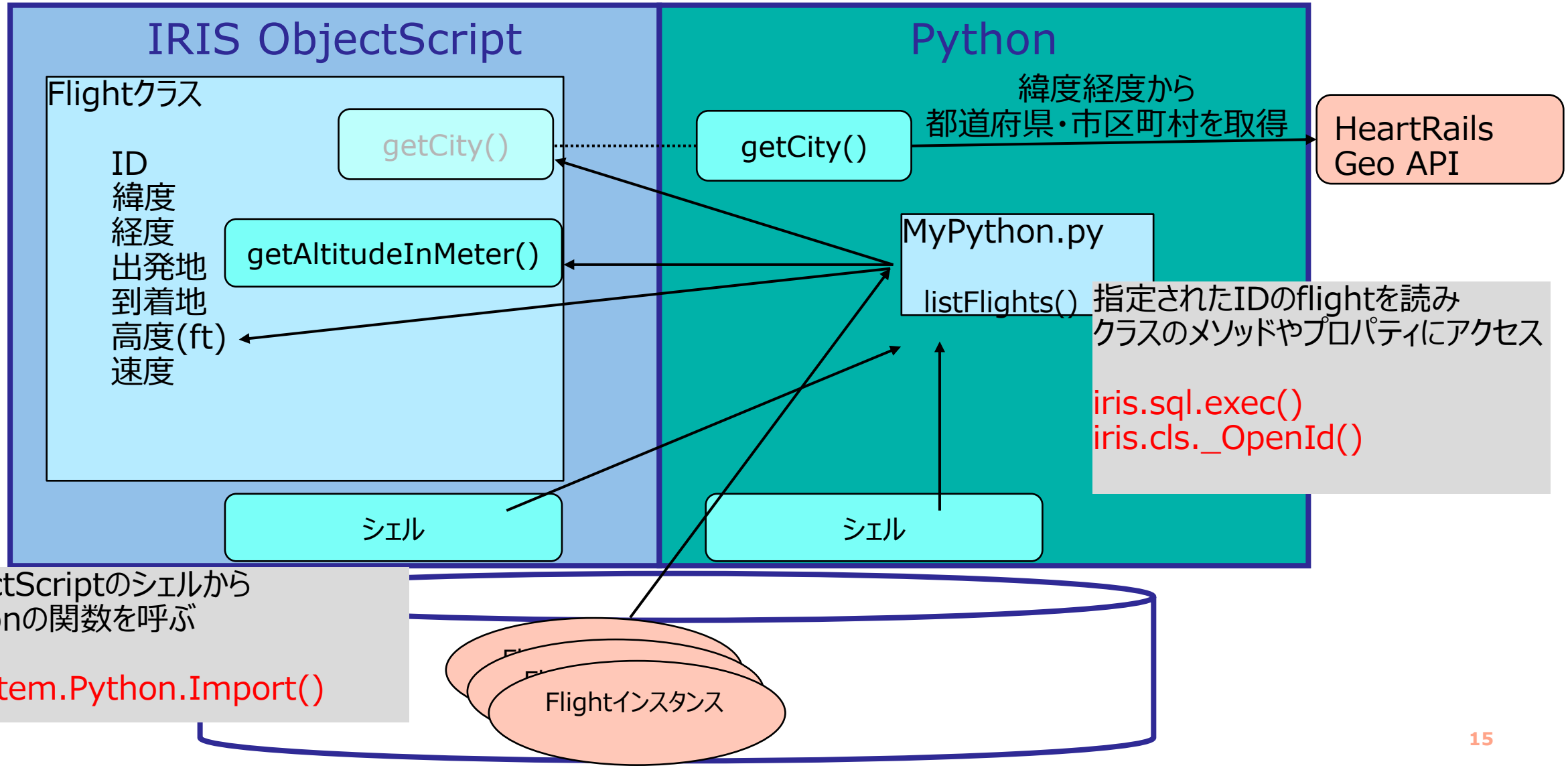
デモ：Embedded Pythonの機能



デモ：Embedded Pythonの機能



デモ：Embedded Pythonの機能



Embedded Python
セルフラーニングビデオ公開！

https://jp.community.intersystems.com/node/520751

公開中ビデオ（YouTubeプレイリスト）は以下の通りです。

- Embedded Python概要
- 利用前の準備（ご利用の前に少しだけ設定確認が必要です）
- Embedded Pythonでデータベースプログラミング：SQLアクセス編
- 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
- IRISでPythonを使ってみよう！

その他YouTubeに
たくさんの動画を公開しています　↓

インターシステムズ開発者コミュニティ

jp.community.intersystems.com

- **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**として役立つ記事がみつかります！
  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
- **最新機能・便利機能を確認できる**ウェビナーアーカイブ公開中！
  https://developer.intersystems.com/jp/#ウェビナー

本日の資料

https://jp.community.intersystems.com/node/538941