組み込み Python の概要

Version 2024.1
2024-06-03

InterSystems Corporation　1 Memorial Drive　Cambridge MA 02142　www.intersystems.com

組み込み Python の概要
InterSystems IRIS Data Platform　Version 2024.1　2024-06-03

インターシステムズ社の製品に関するサポートやご質問は、以下にお問い合わせください：

InterSystems Worldwide Response Center (WRC)
Tel:　+1-617-621-0700
Tel:　+44 (0) 844 854 2917
Email:　support@InterSystems.com

目次

組み込み Python の概要

組み込み Python によって、InterSystems IRIS データ・プラットフォームのネイティブのプログラミング言語である ObjectScript と並行して Python を使用できるようになります。組み込み Python を使用して InterSystems IRIS クラスでメソッドを記述すると、Python ソース・コードがコンパイルされた ObjectScript コードと共にサーバ上で実行されるオブジェクト・コードにコンパイルされます。これにより、ゲートウェイや Native SDK for Python を使用した場合よりもさらに緊密な統合が可能になります。また、カスタムであるか公開されているものであるかにかかわらず、Python パッケージをインポートして、ObjectScript コード内で使用できます。Python オブジェクトは、ObjectScript において第一級オブジェクトであり、その逆も同様です。

ここでは、組み込み Python の概要を提供すると共に、その使用方法のいくつかの例を示します。特に、このドキュメントでは以下のシナリオについて説明します。

- ObjectScript からの Python ライブラリの使用 ー このシナリオでは、読者が ObjectScript 開発者であり、Python 開発者コミュニティで利用可能なさまざまな Python ライブラリの機能を活用したいと考えていると想定しています。
- Python からの InterSystems IRIS API の呼び出し ー このシナリオでは、読者が InterSystems IRIS に慣れていない Python 開発者であり、InterSystems API へのアクセス方法を知りたいと考えていると想定しています。
- ObjectScript と Python の併用 ー このシナリオでは、読者が ObjectScript 開発者と Python 開発者が混在したチームに属しており、この 2 つの言語を併用する方法を知りたいと考えている場合を想定しています。

この項目を使用するには、実行中のバージョン 2021.2 以降の InterSystems IRIS インスタンスと、ご使用のオペレーティング・システムに応じていくらかの前提条件が必要です。また、ObjectScript シェル (InterSystems IRIS コマンド行ツール) にアクセスする方法も知っている必要があります。

このドキュメントの一部の例では、GitHub の Samples-Data リポジトリ (https://github.com/intersystems/Samples-Data) のクラスを使用しています。SAMPLES という名前の専用ネームスペースを作成し、そのネームスペースにサンプルをロードすることをお勧めします。サンプル・コードを表示または変更する場合は、統合開発環境 (IDE) を設定する必要があります。Visual Studio Code の使用が推奨されます。

この項目は、組み込み Python や InterSystems IRIS を使用したプログラミングの要旨を詳細に提示するものではありません。さらに探究するには、このドキュメントの最後に示しているソースを使用してください。

1 ObjectScript からの Python ライブラリの使用

組み込み Python を使用することで、ObjectScript 開発者は利用可能なさまざまな Python ライブラリ (通常パッケージとして知られる) をすべて InterSystems IRIS から直接簡単に使用できるようになり、既存の機能を再現するためにカスタム・ライブラリを開発する必要がなくなります。InterSystems IRIS は <installdir>/mgr/python ディレクトリでインストールされた Python パッケージを検索します。

ObjectScript から使用する Python パッケージの準備は、以下の 2 段階プロセスになります。

1. コマンド行で、Python Package Index (または別のインデックス) から希望のパッケージをインストールします。
2. ObjectScript で、インストールされたパッケージをインポートして、パッケージをロードし、それをオブジェクトとして返します。その後、インスタンス化された ObjectScript クラスと同じように、オブジェクトを使用することができます。

Python Package Index は、https://pypi.org で参照できます。

1.1 Python パッケージのインストール

組み込み Python で使用する前に、コマンド行から Python パッケージをインストールします。使用するコマンドは、Windows システムを使用しているか、UNIX ベースのシステムを使用しているかによって異なります。インターシステムズでは、パッケージを <installdir>/mgr/python にインストールすることをお勧めします。

Windows では、<installdir>/bin ディレクトリから irispip コマンドを使用します：`irispip install --target <installdir>\mgr\python <package>`

例えば、以下のようにして Windows マシンに numpy パッケージをインストールできます。

```
C:\InterSystems\IRIS\bin>irispip install --target C:\InterSystems\IRIS\mgr\python numpy
```

UNIX ベースのシステムでは（AIX を除く）、`python3 -m pip install --target <installdir>/mgr/python <package>` コマンドを使用します。

例えば、以下のようにして Linux マシンに numpy パッケージをインストールできます。

```
$ python3 -m pip install --target /InterSystems/IRIS/mgr/python numpy
```

注釈　まだインストールしていない場合は、まずシステムのパッケージ・マネージャでパッケージ `python3-pip` をインストールしてください。

InterSystems IRIS をコンテナで実行している場合は、"コンテナへの Python パッケージのインストール" を参照してください。

InterSystems IRIS を AIX で実行している場合は、"AIX への Python パッケージのインストール" を参照してください。

1.2 Python パッケージのインポート

%SYS.Python クラスには、ObjectScript から Python を使用するために必要な機能が含まれています。クラス、ターミナル・セッション、または SQL などの任意の ObjectScript コンテキストで、%SYS.Python を使用できます。メソッドの完全なリストは、クラス・リファレンスを参照してください。

Python パッケージまたはモジュールを ObjectScript からインポートするには、%SYS.Python.Import() メソッドを使用します。

例えば、ユーザが現在 USER ネームスペースにいると想定した場合、ターミナルで以下のコマンドを実行すると、math モジュールがインポートされます。

```
USER>set pymath = ##class(%SYS.Python).Import("math")
```

`math` モジュールは、標準の Python リリースに付属するため、インポートの前にそれをインストールする必要はありません。以下のように `pymath` オブジェクトで zwrite を呼び出すことで、これが組み込み `math` モジュールのインスタンスであることがわかります。

```
USER>zwrite pymath
pymath=1@%SYS.Python  ; <module 'math' (built-in)>  ; <OREF>
```

注釈　パッケージは Python モジュールのコレクションですが、パッケージをインストールすると、作成されるオブジェクトは常にモジュールのタイプになります。

これで、ObjectScript オブジェクトとほとんど同じように、math モジュールのプロパティとメソッドにアクセスできるようになりました。

```
USER>write pymath.pi
3.141592653589793116
USER>write pymath.factorial(10)
3628800
```

1.3 例

この例では geopy パッケージを使用して、OpenStreetMap の Nominatim ジオコーディング・ツールにアクセスします。ジオコーディングとは、住所や地名といった、ある場所のテキストベースの説明を取得し、緯度や経度などの地理座標を返して、地表上の場所を特定するプロセスです。

まず、この Windows の例のように、コマンド行から geopy をインストールします。

```
C:\InterSystems\IRIS\bin>irispip install --target C:\InterSystems\IRIS\mgr\python geopy
Collecting geopy
  Using cached geopy-2.2.0-py3-none-any.whl (118 kB)
Collecting geographiclib<2,>=1.49
  Using cached geographiclib-1.52-py3-none-any.whl (38 kB)
Installing collected packages: geographiclib, geopy
Successfully installed geographiclib-1.52 geopy-2.2.0
```

UNIX ベースのシステムでは (AIX を除く)、以下を使用します。

```
$ python3 -m pip install --target /InterSystems/IRIS/mgr/python geopy
```

InterSystems IRIS をコンテナで実行している場合は、"コンテナへの Python パッケージのインストール" を参照してください。

InterSystems IRIS を AIX で実行している場合は、"AIX への Python パッケージのインストール" を参照してください。

次に、ターミナルで以下のコマンドを実行し、モジュールをインポートして使用します。

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

USER>set args = { "user_agent": "Embedded Python" }

USER>set geolocator = geopy.Nominatim(args...)

USER>set flatiron = geolocator.geocode("175 5th Avenue NYC")

USER>write flatiron.address
Flatiron Building, 175, 5th Avenue, Flatiron District, Manhattan, New York County, New York, 10010,
United States
USER>write flatiron.latitude _ "," _ flatiron.longitude
40.74105919999999514,-73.98964162240997666
USER>set cityhall = geolocator.reverse("42.3604099,-71.060181")

USER>write cityhall.address
Government Center, Cambridge Street, Downtown Crossing, West End, Boston, Suffolk County, Massachusetts,
 02203, United States
```

この例では、geopy モジュールを ObjectScript にインポートします。次に、Nominatim モジュールを使用して、geolocator オブジェクトを作成します。この例では、geolocator の geocode() メソッドを使用して、文字列を指定して、地球上の場所を見つけます。次に、reverse() メソッドを呼び出して、緯度と経度を指定して、住所を見つけます。

注意すべき点として、Nominatim() は名前付きキーワード引数を取ります。これは、ObjectScript では直接サポートされていない構文です。このソリューションは、引数リスト (この場合は user_agent キーワードを "Embedded Python" の値に設定) を含む JSON オブジェクトを作成し、args... 構文を使用してそれをメソッドに渡すためのものです。

前の例でインポートされた math モジュールとは対照的に、geopy オブジェクトで zwrite オブジェクトを呼び出すことで、それが C:¥InterSystems¥iris¥mgr¥python にインストールされた geopy パッケージのインスタンスであることが示されます。

```
USER>zwrite geopy
geopy=2@%SYS.Python  ; <module 'geopy' from 'c:\\intersystems\\iris\\mgr\\python\\geopy\\__init__.py'>
  ; <OREF>
```

2 Python からの InterSystems IRIS API の呼び出し

組み込み Python を使用していて、InterSystems IRIS とやり取りする必要がある場合、Python シェルから、または InterSystems IRIS クラスの Python で記述されたメソッドから、iris モジュールを使用できます。このセッションの例に従うには、ターミナル・セッションから、ObjectScript コマンド do ##class(%SYS.Python).Shell() を使用して Python シェルを開始します。

ターミナル・セッションを開始すると、InterSystems IRIS の USER ネームスペースに配置され、プロンプト USER> が表示されます。ただし、GitHub からサンプル・クラスをロードしている場合は、それらにアクセスするには SAMPLES ネームスペースにいる必要があります。

ターミナルで、以下のように SAMPLES ネームスペースに変更してから、Python シェルを開始します。

```
USER>set $namespace = "SAMPLES"
SAMPLES>do ##class(%SYS.Python).Shell()

Python 3.9.5 (default, Jul 19 2021, 17:50:44) [MSC v.1927 64 bit (AMD64)] on win32
Type quit() or Ctrl-D to exit this shell.
>>>
```

ターミナル・セッションから Python シェルを開始する場合、Python シェルはターミナルと同じコンテキスト (現在のネームスペースとユーザなど) を継承します。ローカル変数は継承されません。

2.1 クラスの操作

Python から InterSystems IRIS クラスにアクセスするには、iris モジュールを使用して、使用するクラスをインスタンス化します。これで、Python クラスとほとんど同じように、そのプロパティとメソッドにアクセスできるようになります。

注釈　Python でモジュールを使用するには、まず、以下のようにそれをインポートする必要がある場合があります。

```
>>> import iris
```

ただし、%SYS.Python クラスの Shell() メソッドを使用した Python シェルの実行時に、iris モジュールを明示的にインポートする必要はありません。続行して、モジュールを使用してください。

以下の例では、システム・クラス %Library.File の ManagerDirectory() メソッドを使用して、InterSystems IRIS マネージャ・ディレクトリにパスを出力します。

```
>>> lf = iris.cls('%Library.File')
>>> print(lf.ManagerDirectory())
C:\InterSystems\IRIS\mgr\
```

この例では、システム・クラス %SYSTEM.CPU の Dump() メソッドを使用して、InterSystems IRIS インスタンスを実行するサーバについての情報を表示します。

```
>>> cpu = iris.cls('%SYSTEM.CPU')
>>> cpu.Dump()

-- CPU Info for node MYSERVER ----------------------------------------------
          Architecture: x86_64
                 Model: Intel(R) Core(TM) i7-7600U CPU @ 2.80GHz
                Vendor: Intel
          # of threads: 4
            # of cores: 2
            # of chips: 1
 # of threads per core: 2
   # of cores per chip: 2
          MT supported: 1
            MT enabled: 1
                   MHz: 2904
------------------------------------------------------------------------------
```

この例では、GitHub の Samples-Data リポジトリから Sample.Company クラスを使用します。任意のネームスペースからパーセント記号 (%) で始まるクラス (%SYS.Python や %Library.File など) を使用して Sample.Company クラスにアクセスできますが、前述のように SAMPLES ネームスペースにいる必要があります。

Sample.Company のクラス定義は以下のとおりです。

```
Class Sample.Company Extends (%Persistent, %Populate, %XML.Adaptor)
{

/// The company's name.
Property Name As %String(MAXLEN = 80, POPSPEC = "Company()") [ Required ];

/// The company's mission statement.
Property Mission As %String(MAXLEN = 200, POPSPEC = "Mission()");

/// The unique Tax ID number for the company.
Property TaxID As %String [ Required ];

/// The last reported revenue for the company.
Property Revenue As %Integer;

/// The Employee objects associated with this Company.
Relationship Employees As Employee [ Cardinality = many, Inverse = Company ];

}
```

このクラスは %Library.Persistent (多くの場合 %Persistent と省略される) を拡張します。これは、このクラスのオブジェクトが、InterSystems IRIS データベースで永続化できることを意味します。このクラスには、Name や TaxID などのいくつかのプロパティもあります。このどちらもオブジェクトを保存するために必要です。

クラス定義に表示されませんが、永続クラスには、%New()、%Save()、%Id()、および %OpenId() といった、このクラスのオブジェクトを操作するための多数のメソッドが備わっています。ただし、パーセント記号 (%) は Python のメソッド名では許可されないため、代わりにアンダースコア (_) が使用されます。

以下のコードは、新しい Company オブジェクトを作成し、必要な Name および TaxID プロパティを設定した後、データベース内にこの会社を保存します。

```
>>> myCompany = iris.cls('Sample.Company')._New()
>>> myCompany.Name = 'Acme Widgets, Inc.'
>>> myCompany.TaxID = '123456789'
>>> status = myCompany._Save()
>>> print(status)
1
>>> print(myCompany._Id())
22
```

上記のコードは _New() メソッドを使用してクラスのインスタンスを作成し、_Save() を使用してデータベースにインスタンスを保存します。_Save() メソッドはステータス・コードを返します。この場合、1 は保存が成功したことを示します。オブジェクトを保存すると、InterSystems IRIS は、後でストレージからそのオブジェクトを取得するために使用できる一意の ID を割り当てます。_Id() メソッドはオブジェクトの ID を返します。

以下のように、クラスの _OpenId() メソッドを使用して、永続ストレージからメモリにオブジェクトを取り込み、処理します。

```
>>> yourCompany = iris.cls("Sample.Company")._OpenId(22)
>>> print(yourCompany.Name)
Acme Widgets, Inc.
```

前述したように、InterSystems IRIS クラスの多くのメソッドはステータス・コードを返します。上の例で yourCompany の名前を変更し、オブジェクトを保存するとします。ステータス 1 は正常に保存されたことを示しますが、保存中にエラーが発生した場合は、ステータスにエンコードされたエラー文字列が含まれます。

```
>>> yourCompany.Name = 'The Best Company'
>>> status = yourCompany._Save()
>>> print(status)
0
0Sample.Company^SaveData=1^Sample.Company.1^USER^e^SaveData=1^Sample.Company.17^e^SerializeObject+7^Sample.Company.12^e^Save+8^Sample.Company.15^e^Shell+47^%SYS.Python.11^e^0
```

ステータスの内容を出力する代わりに、メソッド iris.check_status() を使用してステータスを確認し、エラーが含まれる場合は、以下の例のように Python 例外をスローできます。

```
>>> try:
...     iris.check_status(status)
... except Exception as ex:
...     print(ex)
...
ERROR #5803: Failed to acquire exclusive lock on instance of 'Sample.Company'
```

ここでは、誰かがこの会社を排他的にロックしたに違いないため、保存は失敗します。詳細は、"オブジェクト同時処理のオプション" を参照してください。

%SYSTEM.Status クラスに精通している場合は、以下の例のように、そのメソッドを使用してステータス・コードを操作することもできます。

```
>>> if iris.cls('%SYSTEM.Status').IsError(status):
...     iris.cls('%SYSTEM.Status').DisplayError(status)
...

ERROR #5803: Failed to acquire exclusive lock on instance of 'Sample.Company'1
```

IsError() メソッドはステータスにエラーが含まれるかどうかを確認し、DisplayError() はエラー・メッセージを出力します。

2.2 SQL の操作

InterSystems IRIS 内のクラスは SQL に投影されるため、クラス・メソッドの使用または直接グローバル・アクセスに加えて、クエリを使用してデータにアクセスできるようになります。iris モジュールは、Python から SQL ステートメントを実行するための 2 つの異なる方法を提供します。

以下の例は、iris.sql.exec() を使用して SQL SELECT 文を実行して、会社名が "Comp" で始まる Sample.Company クラスのすべてのインスタンスを検索し、各会社の名前とミッション・ステートメントをそれぞれ含む結果セットを返します。ここで、クラス Sample.Company は、同じ名前のテーブルとして SQL に投影します。

```
>>> rs = iris.sql.exec("SELECT Name, Mission FROM Sample.Company WHERE Name %STARTSWITH 'Comp'")
```

以下の例は、iris.sql.prepare() を使用して SQL クエリ・オブジェクトを準備してからクエリを実行し、"Comp" をパラメータとして渡します。

```
>>> stmt = iris.sql.prepare("SELECT Name, Mission FROM Sample.Company WHERE Name %STARTSWITH ?")
>>> rs = stmt.execute("Comp")
```

どちらの場合も以下のように結果セットを反復処理でき、出力は同じです。

```
>>> for idx, row in enumerate(rs):
...     print(f"[{idx}]: {row}")
...
[0]: ['CompuDynamics.com', 'Post-sale services for disruptive optical virtualized productivity tools
for our long-term clients.']
[1]: ['CompuCalc Holdings Inc.', 'Developers of enhanced seven-sigma forecasting technologies for the
 Entertainment industry.']
[2]: ['Compumo Gmbh.', 'Resellers of just-in-time database gaming for the Fortune 5.']
```

クエリ結果がエラーの場合、irisbuiltins.SQLError タイプのインスタンスである例外が返されます。以下の例は、新しい行を挿入しようとしたが、必須フィールドが欠落している場合を示しています。

```
>>> stmt = iris.sql.prepare("INSERT INTO Sample.Company (Mission, Name, Revenue, TaxID) VALUES (?, ?,
 ?, ?)")
>>> try:
...     rs = stmt.execute("We are on a mission", "", "999", "P62")
... except Exception as ex:
...     print(ex.sqlcode, ex.message)
...
-108 'Name' in table 'Sample.Company' is a required field
```

2.3 グローバルの操作

InterSystems IRIS データベースでは、すべてのデータはグローバルに格納されます。グローバルとは、永続的で (ディスクに格納されることを意味する)、多次元で (任意の数の添え字を使用できることを意味する)、スパースである (添え字が連続している必要がないことを意味する) 配列です。クラスのオブジェクトまたは行をテーブルに格納する場合、このデータは実際にはグローバルに格納されます。しかし、通常、これらにはメソッドまたは SQL を介してアクセスし、グローバルを直接操作することはありません。

クラスまたは SQL テーブルを設定せずに、グローバルに永続データを格納すると、便利な場合があります。InterSystems IRIS では、グローバルはその他の変数と同じように見えますが、名前の前にキャレット (^) を付記することによって示されます。以下の例は、現在のネームスペースのグローバル ^Workdays に平日の名前を格納します。

```
>>> myGref = iris.gref('^Workdays')
>>> myGref[None] = 5
>>> myGref[1] = 'Monday'
>>> myGref[2] = 'Tuesday'
>>> myGref[3] = 'Wednesday'
>>> myGref[4] = 'Thursday'
>>> myGref[5] = 'Friday'
>>> print(myGref[3])
Wednesday
```

コードの最初の行 `myGref = iris.gref('^Workdays')` は、^Workdays というグローバルへのグローバル参照 (または gref) を取得します。これは既に存在する場合も存在しない場合もあります。

2 行目の `myGref[None] = 5` は、添え字なしで、平日の数を ^Workdays に格納します。

3 行目の `myGref[1] = 'Monday'` は、文字列 Monday を ^Workdays(1) の場所に格納します。次の 4 行は、残りの平日を ^Workdays(2) から ^Workdays(5) の場所に格納します。

最後の行 `print(myGref[3])` は、gref を指定して、グローバルに格納された値にアクセスする方法を示しています。

詳細な例は、"グローバル" を参照してください。

3 ObjectScript と Python の併用

InterSystems IRIS により、ObjectScript プログラマと Python プログラマが混在したチームが協働することが容易になります。例えば、クラスの一部のメソッドを ObjectScript で記述し、その他のメソッドは Python で記述することができます。プログラマは、最も使い慣れた言語や、目下のタスクにより適した言語で記述することができます。

3.1 InterSystems IRIS の混在クラスの作成

以下のSample.Company クラスのバージョンには、Python で記述された Print() メソッドと、ObjectScript で記述された Write() メソッドが含まれますが、これらは機能的には同等で、どちらのメソッドも Python または ObjectScript から呼び出せます。

```
Class Sample.Company Extends (%Persistent, %Populate, %XML.Adaptor)
{

/// The company's name.
Property Name As %String(MAXLEN = 80, POPSPEC = "Company()") [ Required ];

/// The company's mission statement.
Property Mission As %String(MAXLEN = 200, POPSPEC = "Mission()");

/// The unique Tax ID number for the company.
Property TaxID As %String [ Required ];

/// The last reported revenue for the company.
Property Revenue As %Integer;

/// The Employee objects associated with this Company.
Relationship Employees As Employee [ Cardinality = many, Inverse = Company ];

Method Print() [ Language = python ]
{
    print(f"\nName: {self.Name} TaxID: {self.TaxID}")
}

Method Write() [ Language = objectscript ]
{
    write !, "Name: ", ..Name, " TaxID: ", ..TaxID, !
}
}
```

この Python コード・サンプルは、%Id が 2 である Company オブジェクトを開き、Print() メソッドと Write() メソッドの両方を呼び出す方法を示しています。

```
>>> company = iris.cls("Sample.Company")._OpenId(2)
>>> company.Print()

Name: IntraData Group Ltd. TaxID: G468
>>> company.Write()

Name: IntraData Group Ltd. TaxID: G468
```

この ObjectScript コード・サンプルは、同じ Company オブジェクトを開き、両方のメソッドを呼び出す方法を示しています。

```
SAMPLES>set company = ##class(Sample.Company).%OpenId(2)

SAMPLES>do company.Print()

Name: IntraData Group Ltd. TaxID: G468

SAMPLES>do company.Write()

Name: IntraData Group Ltd. TaxID: G468
```

3.2 Python と ObjectScript 間でのデータの受け渡し

Python と ObjectScript は多くの点で互換性がありますが、それぞれに独自のデータ型と構文も多くあるため、一方の言語から他方の言語にデータを渡す際にデータ変換が必要になる場合もあります。この 1 つの例は、ObjectScript から Python に名前付き引数を渡す前述の例に示しています。

%SYS.Python クラスの Builtins() メソッドは、Python の組み込み関数にアクセスする便利な方法を提供します。このメソッドを使用すると、Python メソッドで期待されるタイプのオブジェクトを作成できます。

以下の ObjectScript の例は、2 つの Python 配列、`newport` および `cleveland` を作成します。それぞれに、都市の緯度と経度が含まれています。

```
USER>set builtins = ##class(%SYS.Python).Builtins()

USER>set newport = builtins.list()

USER>do newport.append(41.49008)

USER>do newport.append(-71.312796)

USER>set cleveland = builtins.list()

USER>do cleveland.append(41.499498)

USER>do cleveland.append(-81.695391)

USER>zwrite newport
newport=11@%SYS.Python  ; [41.49008, -71.312796]  ; <OREF>

USER>zwrite cleveland
cleveland=11@%SYS.Python  ; [41.499498, -81.695391]  ; <OREF>
```

以下のコードは、geopy パッケージを使用します。このパッケージは前の例に出現したもので、ロード・アイランド州ニューポートと、オハイオ州クリーブランド間の距離を計算します。これは geopy.distance.distance() メソッドを使用して経路を作成し、その配列をパラメータとして渡して、経路の miles プロパティを出力します。

```
USER>set distance = $system.Python.Import("geopy.distance")

USER>set route = distance.distance(newport, cleveland)

USER>write route.miles
538.3904453677205311
```

注釈　geopy.distance.distance() メソッドは、実際にはパラメータが Python のタプル・データ型であることを期待していますが、配列も機能します。

3.3 ObjectScript からの任意の Python コマンドの実行

何かを開発またはテストしている場合は、ObjectScript から Python コードを 1 行実行して、それによって何が行われるのか、またはそれが機能するかどうかを確かめると役立つ場合があります。このようなケースには、以下の例に示すように、%SYS.Python.Run() メソッドを使用できます。

```
USER>do ##class(%SYS.Python).Run("print('hello world')")
hello world
```

3.4 組み込み Python からの任意の ObjectScript コマンドの実行

逆に、組み込み Python から ObjectScript コードを 1 行実行することが役立つ場合もあります。このようなケースには、以下の例に示すように、iris.execute() メソッドを使用できます。

```
>>> iris.execute('write "hello world", !')
hello world
```

4 組み込み Python の詳細

組み込み Python の詳細は、以下に示すリソースを参照してください。

- What Is Embedded Python? ー 組み込み Python についての短い紹介ビデオです。
- Parsing Images and Charting Data with Embedded Python ー アプリケーションを記述する際に ObjectScript と Python を切り替えながら使用する方法を示すやや長い演習です。
- 組み込み Python の使用法 ー 組み込み Python と ObjectScript を併用するための基本を含む、組み込み Python についての詳細な説明です。
- Python の組み込み関数 ー Python インタプリタに組み込みの関数のリスト。データを Python データ型に変換する際に役立つ関数が含まれます。

追加のドキュメントおよびオンライン・トレーニング資料が現在開発中で、このリストに追加される予定です。

5 InterSystems IRIS でのプログラミングの詳細

InterSystems IRIS でのプログラミングに関する一般的な情報は、以下に示すリソースを参照してください。

- Writing Python Applications with InterSystems ー pyodbc、組み込み Python、および外部言語サーバを含む、InterSystems IRIS で Python を使用するためのすべてのオプションを網羅する学習パス。
- グローバルの概要 ー 基礎となる InterSystems IRIS のストレージ・メカニズムの実践的な紹介です。
- ObjectScript チュートリアル ー ObjectScript プログラミング言語について実際の操作を通じて説明します。
- サーバ側プログラミングの入門ガイド ー インターシステムズ製品を使用してサーバ側コードを記述するプログラマ向けの基本的知識について説明します。ObjectScript と Python の両方の例が含まれます。