

Artículo

[Heloisa Paiva](#) · 23 sep, 2022 Lectura de 4 min

IRIS y Python en la práctica - ¡con ejemplos!

En este artículo vas a encontrar un sencillo programa con Python en un entorno IRIS y otro sencillo programa con ObjectScript en un entorno Python. Además, me gustaría compartir algunos de los errores que tuve cuando empecé la implementación de estos códigos.

Python en entorno IRIS

Supongamos, por ejemplo, que estás en un entorno IRIS y quieres resolver un problema que crees más fácil o más eficiente de resolver en Python.

Puedes simplemente cambiar el entorno: crea tu método como cualquier otro, y al final del nombre y sus especificaciones, añade [Language = python]:

```
Debug this method
ClassMethod Example(Arg As %String, OtherArg As Demo.Books.PD.Books) As %Status [ Language = python ]
{
    # Code anything you want in python here!
}
```

Puedes usar todo tipo de argumento en el método y, para acceder a ellos, haces lo mismo que harías en COS;

Supongamos que tenemos este argumento %String, llamado Arg, y un argumento OtherArg que viene de una clase custom. Esta clase puede tener propiedades como Title y Author. Puedes accederlas así:

```
Debug this method
ClassMethod Example(Arg As %String, OtherArg As Demo.Books.PD.Books) [ Language = python ]
{
    # Code anything you want in python here!

    # Get the Arg argument:
    firstArgument = Arg

    # Get the OtherArg properties:
    secondArgumentTitle = OtherArg.Title
    secondArgumentAuthor = OtherArg.Author

    print(firstArgument, "/", secondArgumentAuthor, ",", secondArgumentTitle)
}
```

Este método trae un output así:

```
PROBLEMS OUTPUT TERMINAL JUPYTER SQL CONSOLE DEBUG CONSOLE  
first argument / Tolkien , Lord of The Rings
```

Para acceder a los métodos de clase, es básicamente lo mismo. Si hay un método en Demo.Books.PD.Books (de OtherArg), llamado " CreateString " , que concatena el título y el autor en algo como "Title: <Title>; Author: <Author>", entonces, añadir esto al final de nuestro método Python:

```
methodString = OtherArg.CreateString(OtherArg)  
  
print(methodString)
```

proporcionará este output:

```
PROBLEMS OUTPUT TERMINAL JUPYTER SQL CONSOLE DEBUG CONSOLE  
first argument / Tolkien , Lord of The Rings  
Title: Lord of The Rings; Author: Tolkien
```

Para acceder al método, solo es necesario OtherArg.CreateString(), pero elegí usar los mismos valores en OtherArg (CreateString(OtherArg)) para que los outputs sean similares y el código, más simple.

ObjectScript en entorno Python

También hay situaciones en las que estás desarrollando en Python, pero quieres utilizar ObjectScript para ayudarte.

Para empezar, puede que quieras revisar esta lista para poder acceder a tus archivos COS desde un entorno Python de muchas maneras (nos las voy a usar todas aquí):

- ¿Tienes los requisitos previos? [Compruébalo aquí](#)
- ¿Puedes usar un servidor externo Python? [Compruébalo aquí](#)
- ¿Tienes todos los drivers necesarios? [Descárgalos aquí](#) o [Aprende más aquí](#)

Siempre puedes volver a esos enlaces o leer la lista de errores que encontré mientras creaba mis primeros programas Python con COS, por si te ayuda.

¡Entonces, empecemos a programar!

Antes de todo, hay que adaptar lo que viene de COS para Python. Por suerte, InterSystems ya lo hizo y

solo hay que escribir “ import iris ” al comienzo del código para acceder a todo!

```
1 # imports all python's methods and classes to adapt from COS
2 import iris
3
```

En el próximo paso, creamos una conexión al namespace deseado, usando una ruta con el host, port y namespace (host:port/namespace) y dándole usuario y contraseña:

```
# info for connecting into the namespace you want: mine is SAMPLE
connection_string = "localhost:1972/SAMPLE"
username = "_system"
password = "sys"

# create the connection
connection = iris.connect(connection_string, username, password)

# create an iris object
irispy = iris.createIRIS(connection)
```

Fíjate cómo creamos un OBJETO IRIS, para poder usar esta conexión para acceder a todo que necesitamos del namespace.

Finalmente, puedes “ programar ” todo lo que quieras!

```
15 # access your classes and methods with .classMethod and other functions
16 request = irispys.classMethodObject('Demo.Books.PD.Books', '%New')
17 request.set('Title', 'Lord of The Rings')
18 request.set('Author', 'Tolkien')
19
20 status = irispys.classMethodValue('Demo.Python.Test', 'Insert', request)
21
22 # while also coding in python anything you want
23 print(status)
```

Puedes acceder a métodos con `irispys.classMethodValue()`, dando el nombre de la clase y el nombre y los argumentos del método, puedes manipular objetos con `.set()` (para propiedades), y muchas otras posibilidades, mientras usas Python como quieras.

Para más informaciones sobre funciones de iris y cómo usarlas, echa un vistazo a [Introduction to the Native SDK for Python](#)

En este ejemplo, en la línea 16, he instanciado una clase %Persistent para, en las próximas líneas

darles los valores “ Lord of The Rings ” y “ Tolkien ” para las propiedades Title y Author.

En la línea 20, llamé un método de otra clase que guarda el objeto en una tabla y vuelve un status si funciona. Finalmente, en la línea 23, vuelve el status.

Mezcla!

En un entorno ObjectScript, puedes necesitar de las conocidas librerías de Python, o tus propios archivos con tus funciones y rutinas.

Puedes usar el comando “ import ” con NumPy, SciPy y cualquier otro que quieras (considerando que las tienes correctamente instaladas: [mira aquí cómo hacerlo](#))

Pero también, si quieres acceder a tus archivos locales hay muchas maneras de hacerlo y es fácil encontrar tutoriales para eso, ya que Python es muy popular.

A mí personalmente esta es la que más me gusta:

```
Debug this method
40  ClassMethod TesteImport() [ Language = python ]
41  {
42      import os
43      import sys
44      sys.path.append(os.path.abspath("C:/python/"))
45
46      import testesql
47
48      print(testesql.select())
49  }
50
```

Aquí importé todo del archivo testesql.py, situado en C:/python y dejé impreso todos los resultados de la función select()

Extras - problemas que encontré

- SHELLS: Cuando usas Shells, acuerdate de que Windows PowerShell funciona cómo un sistema UNIX-based (porque está basado en .NET) y el Command Prompt es el que funcionará con los ejemplos de Windows en la documentación oficial de InterSystems. Eso puede parecer simple para programadores con más experiencia, pero si no estás prestando mucha atención puedes perder un tiempo con este detalle.
- USUARIOS Y PRIVILEGIOS: El usuario actual para programar con ObjectScript en un entorno

Python necesita tener privilegios para los recursos del namespace. Recuerda que si no se selecciona ningún usuario, el actual es UnknownUser, y si no se selecciona ningún namespace, el actual es USER. Entonces, en el acceso más simple: Portal de Administración > Administración del Sistema > Seguridad > Usuarios > Unknown User > Roles, y selecciona %DBUSER, y guarda.

- NO SÉ QUÉ PASA: Para tener más información sobre los errores que estás teniendo, ve a: Portal de Administración > System Explorer > SQL y escribe “ SELECT * FROM %SYS.Audit ORDER BY UTCTimeStamp Desc ” , y ahí vas a tener las más recientes audits. Acordate de seleccionar el namespace correcto para la consulta de la query. Así, puedes leer las causas de errores como IRIS_ACCESSDENIED() y mucho más, que puedes tener hacia fuera del ambiente IRIS.
- ERROR COMPILANDO PYTHON: Evita llamar a Métodos con nombres como try() o con los de otras funciones de Python. El compilador no sabrá la diferencia entre el nombre del método y el nombre de la función.

¡Gracias por leer y, por favor, comparte tus comentarios, sugerencias o dudas!

[#Innovatium](#) [#Mejores prácticas](#) [#Python](#) [#InterSystems](#) [IRIS](#)

URL de
fuente: <https://es.community.intersystems.com/post/iris-y-python-en-la-pr%C3%A1ctica-%C2%A1con-ejemplos>