



algogenericdev

Seguir

30 de septiembre de 2021 · 4 minutos de lectura · Escucha



Ahorrar

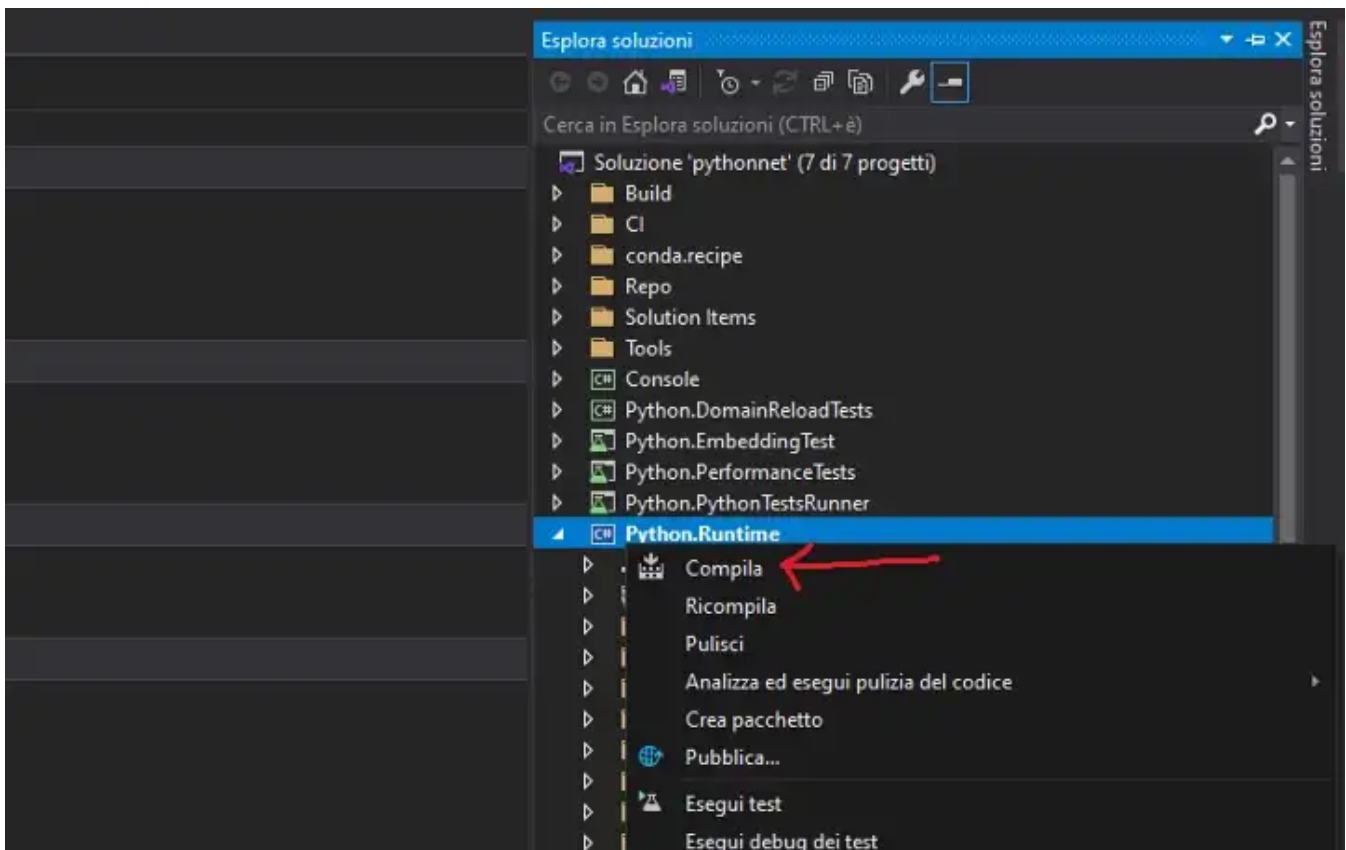


# Llamar a Python desde C#: una introducción a PythonNET

Recientemente, tuve que profundizar en PythonNET porque tenía que encontrar una forma de llamar a los scripts de Python desde mis proyectos de C# en el trabajo. Sorprendentemente, encontré muy poco material en línea sobre el tema y la mayor parte era contradictorio o bastante confuso. La documentación oficial de PythonNET tampoco fue particularmente útil, así que decidí escribir esta guía para aquellos que se encontrarán en la misma situación que yo en el futuro.

## Compilar la dll de PythonNET

Lo primero que haremos será compilar la dll de PythonNET. Descarga el código fuente de PythonNET desde [Github](#), ábrelo con Visual Studio y compila el proyecto llamado “Python.Runtime”. Si la compilación es exitosa, generará un archivo .dll en pythonnet-master\pythonnet\runtime\Python.Runtime.dll



Simplemente haga clic en compilar y espere a que termine. ¡Eso es!

## Requisitos previos de Python

En el momento en que se escribió esta guía (septiembre de 2021), PythonNET solo es compatible con Python 3.6–3.8. La última versión disponible de Python es la 3.9.7. Compruebe si tiene una versión más nueva de Python instalada en su máquina y, si ese es el caso, **asegúrese de desinstalarla** para evitar problemas. A continuación, puede descargar Python 3.8 desde [aquí](#).

Si tiene una versión como "3.8.10" o algo así, *debería* estar bien, a PythonNET no parece importarle hasta que llegue a 3.9 y más, según mi experiencia personal. También deberá instalar el módulo de PythonNET iniciando `pip install pythonnet` desde el CMD.

## Crear una aplicación de prueba

Vamos a crear una aplicación de consola llamada "TestProject" y agreguemos una referencia a la dll de PythonNET (Python.Runtime.dll) en ella.

## Hola Mundo

Vamos a crear una clase llamada "PythonInterop". Vamos a crear un método de inicialización, donde vamos a configurar el dll de Python como una variable de

entorno (asegúrese de reemplazar mi ruta con la suya) e inicializar el motor de PythonNET. Puede encontrar la dll de Python en la ruta de instalación de Python. De forma predeterminada, debería ser C:\Users\YourUserNameHere\AppData\Local\Programs\Python\Python38\python38.dll. para Python 3.8.0.

*NOTA: Si planea ejecutar PythonNET en Linux, ¡debe buscar el archivo .so de Python en su lugar! Puede ser bastante difícil de encontrar ya que no siempre se encuentra en el mismo directorio. Puede intentar ejecutar este script para obtener su ubicación:*

wget

[https://gist.githubusercontent.com/tkf/d980eee120611604c0b9b5fef5b8dae6/raw/9f074cd233f83180676b4421212ed33c257968af/find\\_libpython.py](https://gist.githubusercontent.com/tkf/d980eee120611604c0b9b5fef5b8dae6/raw/9f074cd233f83180676b4421212ed33c257968af/find_libpython.py)  
/usr/bin/python3 find\_libpython.py --list-all

También vamos a crear un método para ejecutar nuestro código Python con PythonNET.

```
1      public static void Initialize()
2      {
3          string pythonDll =
4              @"C:\Users\YourUserNameHere\AppData\Local\Programs\Python\Python38\python38.dll";
5          Environment.SetEnvironmentVariable("PYTHONNET_PYDLL", pythonDll);
6          PythonEngine.Initialize();
7      }
8
9      public static void RunPythonCode(string pycode)
10     {
11         Initialize();
12         using (Py.GIL())
```

Abrir en la aplicación ↗

Inscribirse

Registrarse



PythonInterop.cs hosted with ❤ by GitHub

view raw

Ahora vayamos a Program.cs y lancemos nuestro hola mundo.

```
1      static void Main(string[] args)
2      {
3          PythonInterop.RunPythonCode(@"print("""hello world from python!""");
4      }
```

Program.cs hosted with ❤️ by GitHub

[view raw](#)

## Usar clases de C# en Python

Para importar clases, variables y métodos de C# en Python, necesitamos importar un módulo llamado clr.

Vamos a crear una clase llamada "MyClass" con una propiedad de cadena llamada "MyVar".

```
1      public class MyClass
2      {
3          public static string MyVar = "this is a C# variable";
4      }
```

MyClass.cs hosted with ❤️ by GitHub

[view raw](#)

En nuestro archivo Program.cs, ejecutemos

```
1      static void Main(string[] args)
2      {
3          PythonInterop.RunPythonCode(
4      @"import clr
5      clr.AddReference("""TestProject""");
6      from TestProject import MyClass;
7      csharpVariable=MyClass.MyVar
8      csharpVariable=csharpVariable + "" and this part was added by Python"";
9      print(csharpVariable);
10
11  ");
12      }
```



4



1

Program.cs hosted with ❤️ by GitHub

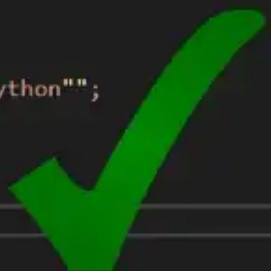
[view raw](#)

Si todo salió bien, deberíamos ver nuestra variable impresa en la Consola.

```

6
7
8      0 riferimenti
9      static void Main(string[] args)
10     {
11         PythonInterop.RunPythonCode(
12             @"import clr
13             clr.AddReference(""TestProject"");
14             from TestProject import MyClass;
15             csharpVariable=MyClass.MyVar
16             csharpVariable=csharpVariable + "" and this part was added by Python"";
17             print(csharpVariable);
18         );
19     }
20
21

```




Tenga cuidado cuando escriba la cadena que contiene el código de Python. Hacer esto

```

8
9
10     0 riferimenti
11     static void Main(string[] args)
12     {
13         PythonInterop.RunPythonCode(
14             @"import clr
15             clr.AddReference(""TestProject"");
16             from TestProject import MyClass;
17             csharpVariable=MyClass.MyVar
18             csharpVariable=csharpVariable + "" and this part was added by Python"";
19             print(csharpVariable);
20         );
21     }
22
23

```



¡y NO esto! Python usa sangría para definir el alcance, por lo que incluir tabulaciones y espacios innecesarios en su cadena probablemente generará errores.

De esta manera, podemos importar cualquier clase, incluidas las clases del sistema de C#, y usar sus métodos también.

```

1      static void Main(string[] args)
2      {
3          PythonInterop.RunPythonCode(
4      @"import clr
5      clr.AddReference(""System"");
6      from System.Diagnostics import Debug;
7      Debug.WriteLine(""this line was printed by Python in the Debug window!"");
8      ");}

```

Program.cs hosted with  by GitHub

[view raw](#)

```

1  using System;
2
3  namespace TestProject
4  {
5      0 riferimenti
6      class Program
7      {
8          0 riferimenti
9          static void Main(string[] args)
10         {
11             PythonInterop.RunPythonCode(
12                 @"import clr
13                 clr.AddReference(""System"");
14                 from System.Diagnostics import Debug;
15                 Debug.WriteLine(""this line was printed by Python in the Debug window!"");
16                 ");
17         }
18     }

```

Output:

```

TestProject.exe (CoreCLR: clrhost): caricamento di 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\5.0.9\System.Collections...'
TestProject.exe (CoreCLR: clrhost): caricamento di 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\5.0.9\System.Linq.Expre...'
TestProject.exe (CoreCLR: clrhost): caricamento di 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\5.0.9\System.Reflection...'
TestProject.exe (CoreCLR: clrhost): caricamento di 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\5.0.9\System.Threading...'
TestProject.exe (CoreCLR: clrhost): caricamento di '__CodeGenerator_Assembly' completato.
TestProject.exe (CoreCLR: clrhost): caricamento di 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\5.0.9\System.Runtime.In...'
TestProject.exe (CoreCLR: clrhost): caricamento di 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\5.0.9\System.Runtime.In...'
TestProject.exe (CoreCLR: clrhost): caricamento di 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\5.0.9\Microsoft.Win32.F...'
Eccezione generata: 'System.MissingMethodException' in Python.Runtime.dll
Eccezione generata: 'System.MissingMethodException' in Python.Runtime.dll
TestProject.exe (CoreCLR: clrhost): caricamento di 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\5.0.9\System.Collection...'
TestProject.exe (CoreCLR: clrhost): caricamento di 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\5.0.9\System.ObjectMode...'
TestProject.exe (CoreCLR: clrhost): caricamento di 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\5.0.9\System.IO.FileSys...'
TestProject.exe (CoreCLR: clrhost): caricamento di 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\5.0.9\System.dll' compi...
TestProject.exe (CoreCLR: clrhost): caricamento di 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\5.0.9\System.Threading...'
this line was printed by Python in the Debug window!

```

iFuncional!

## Usar tipos de C# de Python

De la misma manera, PythonNET nos permite ver los tipos de C# desde Python también.

```

1  PythonInterop.RunPythonCode(
2  @import clr
3  clr.AddReference(""System"");
4  from System import DateTime
5  mydate=DateTime(2010, 5, 11)
6  print(mydate)
7  ");

```

Program.cs hosted with ❤ by GitHub

[view raw](#)

## Importar variables con los métodos de PythonNET

Una forma alternativa de importar variables de C# a Python es usar los métodos de PythonNET.

Vamos a crear una sobrecarga del método RunPythonCode en nuestra clase PythonInterop

```
1  public static void RunPythonCode(string pycode, object parameter, string parameterName) {  
2      Initialize();  
3      using (Py.GIL())  
4      {  
5          using (PyScope scope = Py.CreateScope())  
6          {  
7              scope.Set(parameterName, parameter.ToPython());  
8              scope.Exec(pycode);  
9          }  
10     }  
11 }  
12 }
```

PythonInterop.cs hosted with ❤ by GitHub

[view raw](#)

*parámetro* es la variable que queremos inyectar en Python, *parámetroNombre* es el nombre que queremos que tenga la variable dentro de Python (generalmente, el mismo que C#).

Creemos una clase Person y probemos nuestro método.

## **Devolver valores de Python a C#**

Let's suppose now that we want to return a variable from Python into C# to perform some other operations on it.

To do so, let's go back into our PythonInterop class and create a method.



*returnedVariableName* is the name of the Python variable that we want to get returned in C#.

Let's try to edit the Age property's value, to assign it to a variable and to return it to C#.

## **Import from external projects**

You can also easily import from external libraries using the `clr` module. Let's create a class library called "MyLibrary" in our solution and let's reference it from our main Console Application method.

After that, let's add a class called "LibClass" in our library.

Now, let's just try to print our string from Python.

## Final notes

Esto fue pensado como una breve introducción a PythonNET para principiantes absolutos, ya que fue muy difícil para mí encontrar material como este en línea. De ahora en adelante, probablemente tendrá un conjunto de herramientas decente que puede usar para investigar y perseguir sus objetivos con PythonNET por su cuenta. ¡Buena suerte!

Voy a dejar el código fuente completo de la clase PythonInterop a continuación.



Punto net

c sostenido

pythonnet

Pitón

aspnet

Acerca de

Ayuda

Términos

Privacidad

## Obtén la aplicación mediana

