Python and .NET Integration Reference

Combine Python and .NET in scripts.

Overview

The platform includes a unique set of features:

- Multiple Python Versions
- In-Solution Designer Editing and Testing
- Tag Direct Access
- .NET and Python Interoperability
- Extensibility Toolkit

Overview
Installation and Setup
Using Python in the Solution
Designer Python Code Editor
.NET Interoperability
Python and .NET Integration Links

Installation and Setup

The Python Shell integration requires Python releases 3.7 onwards and the installation of Python.NET.

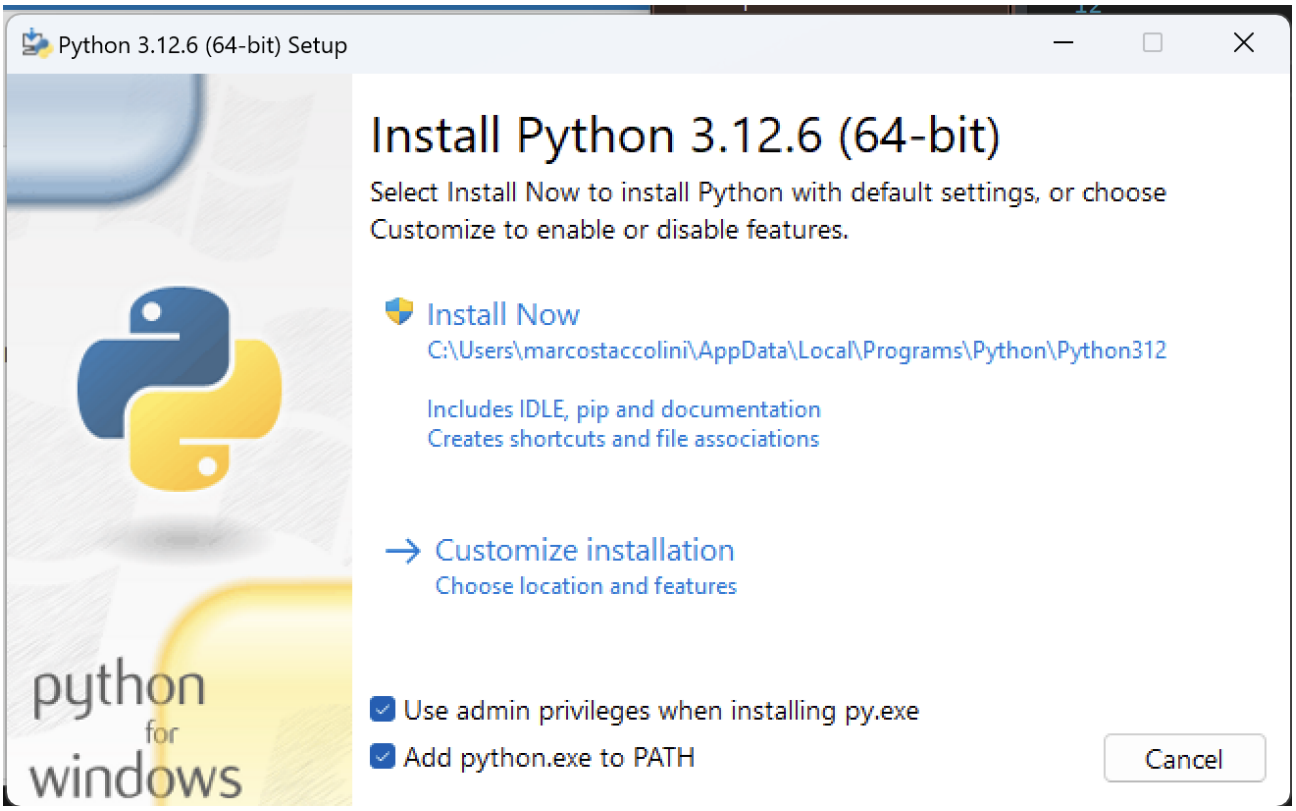
Setup Steps:

1. Install Python and Python.NET in the machines where you run the application and on those using the FrameworX AI Designer.
2. Configure the Solution Settings to select the appropriate Python shell folder.

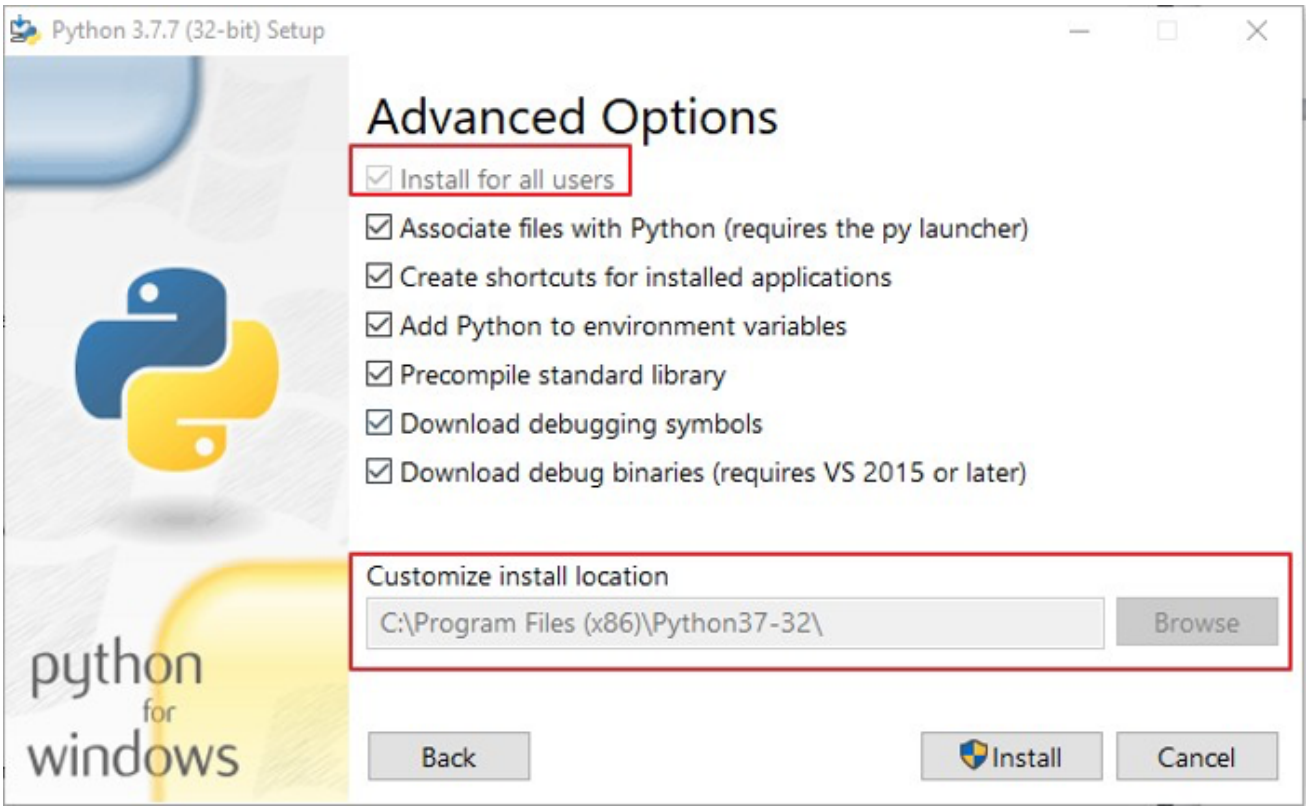
Each solution can specify its own Python interpreter, facilitating the development and maintenance of different solutions that may require different Python versions.

You can download Python here.

Check the Use admin privileges checkbox, and select Customize installation.



During the installation of the Python Engine,select the option to “Install for all users”.



If You install Python under your local user, you may have issues when running the solution as a service, or when deploying the solution for production.

This image may be slightly different for your Python version, but always enable the options equivalent to:

- Add Python.exe to PATH

Especially in the production server.

- Install with admin rights to all users,
- Customize the installation to installation in the Program Files, instead of Users folder

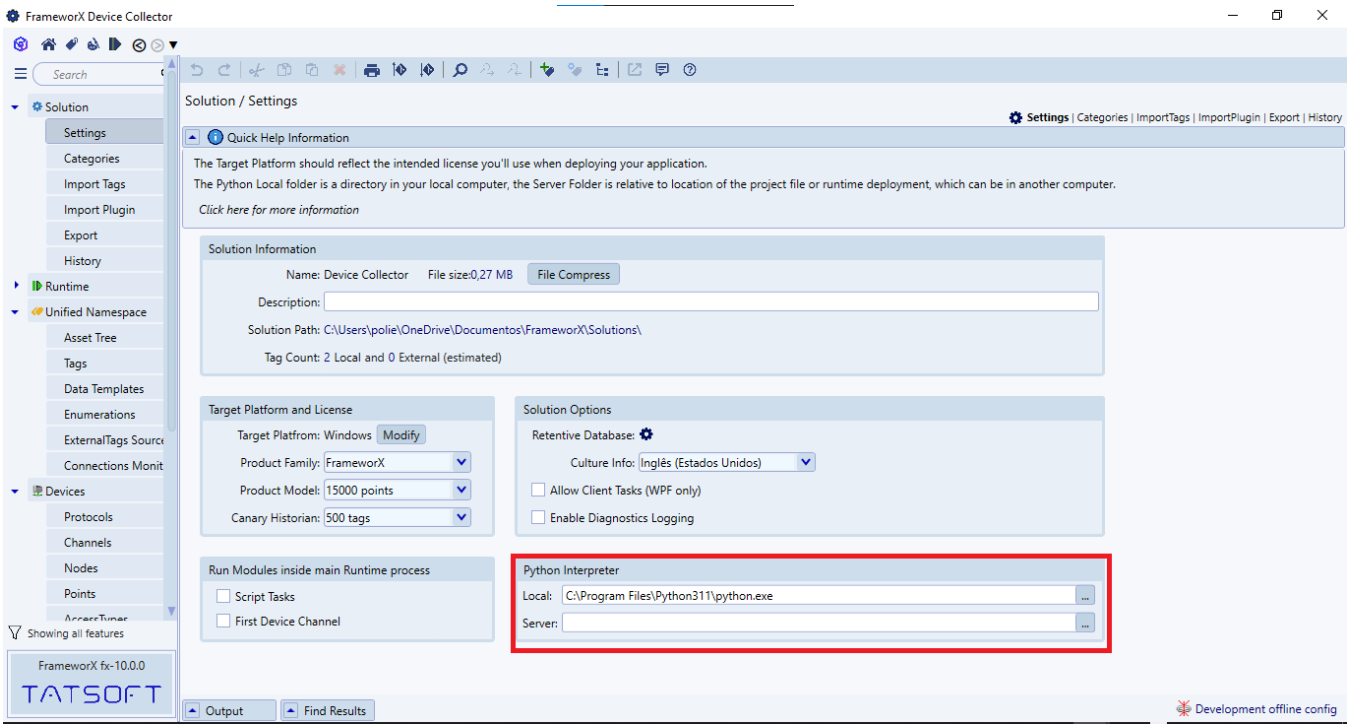
Once you've downloaded, open the command prompt as an ADMINISTRATOR and type “pip install pythonnet”. For this work, you'll need internet access. If internet is not available, download and install manually.

Once Python.NET has been installed, you can start using Python in your solution.

```
PS C:\Windows\system32> pip install pythonnet
Collecting pythonnet
  Using cached pythonnet-3.0.3-py3-none-any.whl.metadata (6.6 kB)
```

Note: For Python versions above 3.12, Pythonnet is not available. For more information, visit their documentation.

Go to **Solution / *Settings*** and locate the Python Interpreter box. Click the "**...**" button, navigate to find the installed Python Engine, and select the *pytho n.exe* file.



Python Interpreter	
Local	Defines the path of the Python installation folder on the local machine.
Server	Defines the path of the Python installation folder on the remote server.

Using Python in the Solution

The Python Example illustrates how to use the Python and .NET languages for the Scripts.

This section presents a summary of that information, listing the modules in the solution where Python code can be used.

Scripts Tasks and Classes

- **Scripts / Tasks**: Event driven or periodical tasks, running on the server computer. They can be written in Python, C# or VB.NET.
- **Scripts / Classes:** library of methods and functions that can be used on both server and client machines. They can be written in Python, C# or VB.NET.

Displays CodeBehind and Graphical Elements

The Code Behind for the displays can be written in C#, VB.NET or JavaScript, not Python directly. But from the CodeBehind you can call tasks and methods, which can be written in Python.

Shell Integration

- **Shell Integration:** allows you to execute Python code from external files.

This integration allows you to execute Python code from external files. Just create a Script Task with the following code:

Shell Integration example

```
# This code call the execution of the external file using Python Shell
# with the optional args defined in this initial section
#
# The macro _ExecutionPath_ is replaced by the path where the solution is set to execute
# Replace that macro by a specific path, or use other built-in macros as needed
#

arg1 = @Tag.Tag1
arg2 = @Tag.Tag2
result = TK.ExecutePythonShell("_ExecutionPath_ExternalSum.py", [arg1, arg2])
@Tag.Result = result
```

Python Shell Version and Tests

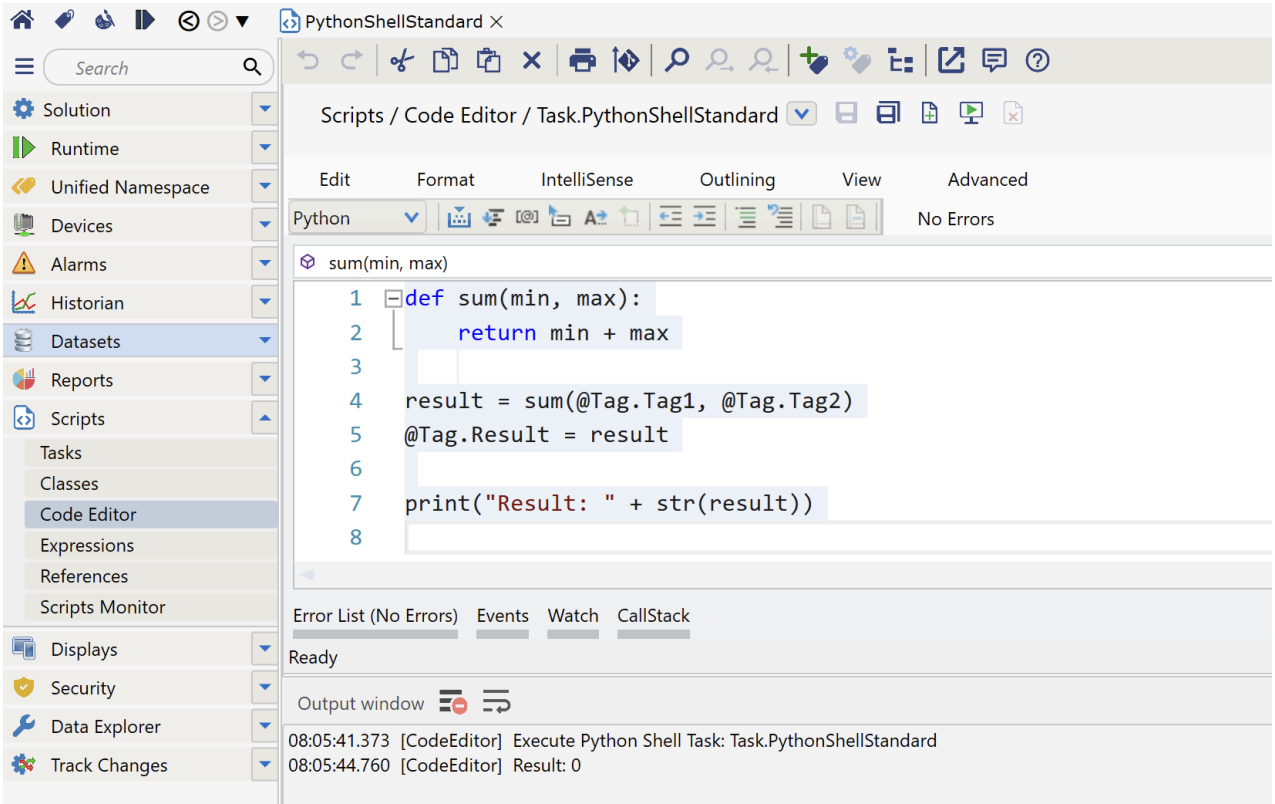
Any Python scripts you create in **Scripts / Tasks** can be edited and tested in **Scripts / Code Editor**. To test your script, click on [Monitor Icon] in the top toolbar. This will run your code and display messages in the Output Window, under the script editor.

This integration will work if your Python version is 3.7 or newer, with Python.NET installed, and the executable is correctly mapped in **Soluti on / Settings**.

Designer Python Code Editor

Code Editor User Interface

The code editor can be accessed through **Scripts / Code Editor**.



Testing the code

When the editor is used to edit Script Tasks, you can test the scripts directly, using the Play button at the Task name line. The results of your code will show in the standard Designer output.

Accessing Tags and Modules Properties

The access to tags and other properties for other models is exactly the same as C# or VB.NET; just use the '@' symbol to access the properties directly, as shown in the image: '@Tag.Tag1'.

.NET Interoperability

Calling ScriptClass Methods

In the Script, or in any place in your solution where Expressions are allowed, you can use the following syntax to call classes.

```
@Script.Class.TestClass.ExampleMethod(param1, param2)
```

This activation is independent of the language the class was created in.

This means C# and VB.NET can call Python classes, and Python code can call .NET classes.

Python and .NET Integration Links

Explanation - to understand concepts

Platform / Technology Foundation / Python and .NET Integration

Tutorials - to learn by doing

Tutorials / Technology Learning / Python and .NET Integration

Reference - technical details

Technical Reference / Programming and APIs Reference / Python and .NET Integration

---

In this section...
