Using Python in FileMaker

For MBS FileMaker Plugin 14.3 we add the possibility to evaluate Python code in FileMaker and run scripts. This enables you to integrate existing Python code and use it in FileMaker. That may be a quick calculation to calculate a checksum, process some text, JSON or XML data or to utilize an existing script by running it within FileMaker. Python comes with hundreds of available modules that you can import and use.

Initialization

Depending on the platform, you install Python libraries in different ways. For macOS you can use Homebrew command line tools to install the python package. For Windows you can download the official Python installer and run it. For Linux you use your package manager to get the python package installed. When done, you have on each system a folder with the libraries and you tell the plug-in where to find them.

For macOS you usually check look for the Python.framework:

/opt/homebrew/Cellar/python@3.12/3.12.3/Frameworks/Python.framework
or
/usr/local/homebrew/Cellar/python@3.12/3.12.3/Frameworks/Python.framework

For Windows the path to the dll may include the version number, so please adjust as needed. If you like use our Files.List function to check the names in the path in your script.

C:\Users\cs\AppData\Local\Programs\Python\Python312\python312.dll

For Linux the package is usually in the library path, so you can just refer to it by the name.

Let's put it all together into a sample script:

```
If [ MBS("IsMacOS") ]
	# We installed python via homebrew
	Set Variable [ $r ; Value: MBS("Python.Load"; "/opt/homebrew/Cellar/python@3.12/3.12.3/Frameworks/Python.framework") ]
Else If [ MBS("IsWindows") ]
	# We installed python via the installer from the python website
	Set Variable [ $r ; Value: MBS("Python.Load"; "C:\Users\cs\AppData\Local\Programs\Python\Python312\python312.dll") ]
Else If [ MBS("IsLinux") ]
	# We installed libpython3 packages
	Set Variable [ $r ; Value: MBS("Python.Load"; "libpython3.11.so") ]
Else
	Exit Script [ Text Result:    ]
End If
#
If [ MBS("IsError") ]
	Show Custom Dialog [ "Failed to load library." ; $r ]
Else
	Set Variable [ $p ; Value: MBS("Python.LibraryVersion") ]
	Set Field [ Python::LibraryVersion ; $p ]
End If
```

Check Version

Once you have Python library loaded, you can check the version: Python.LibraryVersion. You get back a string like "3.12.3 (main, Apr 9 2024, 08:09:14) [Clang 15.0.0 (clang-1500.3.9.4)]" to describe the version with compilation date and compiler used. For Windows the output may be "3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)]".

```
Show Custom Dialog [ "Python Version" ; MBS( "Python.LibraryVersion" ) ]
```

Print Hello World

Let us run our first script. We create a new environment with Python.New function. You can have multiple environments around and each of them has their own local variables. Inside that environment we can run script lines with Python.Run function and passing the script. The script may be as simple as this:

```
print("Hello World")
```

We only call print function and pass the text "Hello World". Print is very useful to output debug messages. Our plugin collects the print messages and return it with Python.PrintOutput function.

```
Set Variable [ $p ; Value: MBS("Python.New") ]
Set Variable [ $r ; Value: MBS("Python.Run"; $p; Python::Code) ]
Set Field [ Python::Result ; $r ]
Set Field [ Python::Print ; MBS("Python.PrintOutput"; $p) ]
Set Variable [ $r ; Value: MBS("Python.Release"; $p) ]
```

When you are done with a python environment, you can release it. But it is absolutely fine to create it once on start of the solution and use the same one throughout the solution. For server side scripts running in parallel, please have one environment per script to avoid accessing the same one from different threads.

When we run above Python script using the given FileMaker script, the output will be "Hello World" as returned by Python.PrintOutput. There is a Char(13) on the end to indicate the new line.

Process JSON in Python

Let's do something in Python like processing some JSON. For that we use the json module in Python to get the json.loads function to parse JSON and json.dumps to encode to JSON. The script will decode the given JSON in the variable InputValues, append a new number and then store the final JSON text in OutputValues variable.

```
import json

j = json.loads(InputValues)
j.append(5)
OutputValues = json.dumps(j)
```

In our FileMaker script we put in the variable value in the second line with Python.SetValue function and read the output using Python.GetValue function. Our plugin can pass numeric and text values directly. Passing a container value will pass it as bytes value to Python.

```
Set Variable [ $p ; Value: MBS("Python.New") ]
Set Variable [ $r ; Value: MBS("Python.SetValue"; $p; "InputValues"; "[1,2,3]") ]
Set Variable [ $r ; Value: MBS("Python.Run"; $p; Python::Code) ]
#
Set Field [ Python::Result ; MBS("Python.GetValue"; $p; "OutputValues") ]
Set Field [ Python::Print ; MBS("Python.PrintOutput"; $p) ]
#
Set Variable [ $r ; Value: MBS("Python.Release"; $p) ]
```

When you run this, you see the JSON [1, 2, 3, 5] in the result field.

Evaluate

If you have a Python environment setup, you can evaluate something at any time. Like evaluate() in FileMaker, just pass the text to evaluate to the Python.Evaluate function. Let's setup somewhere a global FileMaker variable pointing to a Python environment:

```
Set Variable [ $$python ; Value: MBS("Python.New") ]
Set Variable [ $r ; Value: MBS("Python.Run"; $$python; "import math") ]
```

Later we can use it to perform calculations:

```
MBS("Python.Evaluate"; $$Python; "1+2")
```

You may even use that in a custom function to do something in FileMaker. Since we imported math module directly above after creating the environment, we can now use the math module in expressions:

```
MBS("Python.Evaluate"; $$Python; "math.sqrt(5)")
```

Please try yourself in 14.3 pre-release of MBS FileMaker Plugin.