**A Quick Start to Running Python Code in SQL Server**

[Emmitt Albright](https://www.sqlservercentral.com/author/emmittalbright), 2019-04-18

FROM: <https://www.sqlservercentral.com/articles/a-quick-start-to-running-python-code-in-sql-server>

SQL Server 2016 introduced the idea of running R language scripts from inside SQL Server. In SQL Server 2017, this capability was renamed Machine Learning Services and enhanced to include the ability to run Python code. This article gives a quick start in how you can execute Python code inside SQL Server and transform data in new ways.

The Setup

The ability to run Python code is not allowed by default in SQL Server. There are a couple of stages that you need to complete before you can run code. First, you need to install Machine Learning Services (MLS) as part of your SQL Server set up. If you haven’t done this, you can run set up and choose to add the MLS option to your installation. You can read more about installation in [our Stairway](https://www.sqlservercentral.com/steps/installing-machine-learning-services-level-1-of-the-stairway-to-ml-services) or [in Books Online](https://docs.microsoft.com/en-us/sql/advanced-analytics/install/sql-machine-learning-services-windows-install?view=sql-server-2017), but be sure you choose the “Machine Learning Services (In-Database)” option and then select Python support. You can add R support if you desire.

Once you have installed this, you must now enable the instance to allow execution of scripts. We do this with a configuration change. Run this code on your instance (as an admin).

EXEC sp\_configure 'external scripts enabled', 1;

RECONFIGURE;

GO

This will enable script execution using the Launchpad service and the sp\_execute\_external\_script stored procedure. You can read more about how to get started in the [Stairway to Machine Learning Services](http://www.sqlservercentral.com/stairway/171757/).

Your First Python Program

Once we have configured SQL Server, we can start running Python code. We submit our code as a string variable, so we won’t be working with files. Instead, we declare a variable, set a string value containing the code, and then execute it. We use [sp\_execute\_external\_script](https://docs.microsoft.com/en-us/sql/relational-databases/system-stored-procedures/sp-execute-external-script-transact-sql?view=sql-server-2017), which takes a number of parameters. In this section we will only use two of these, and cover others later. We will use:

* @language – the language of the code in our script
* @script – the text of our program

To begin, let’s create a database that we can use on a SQL Server 2017 instance. I’ll do this with the following code:

CREATE DATABASE MLDemo

GO

USE MLDemo

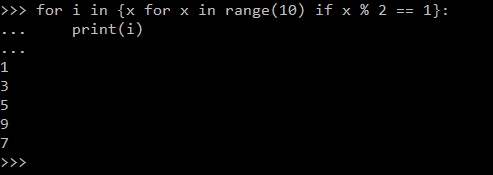
GO

Once we have this, let’s create a script. I will use this simple Python script from a recent Question of the Day.

for i in {x for x in range(10) if x % 2 == 1}

print(i)

If I run this in the Python REPL, I see this:



Now let’s run this in SQL Server. I’ll take my script and include as the value for the script parameter. That means I’ll be using this code:

EXEC sys.sp\_execute\_external\_script

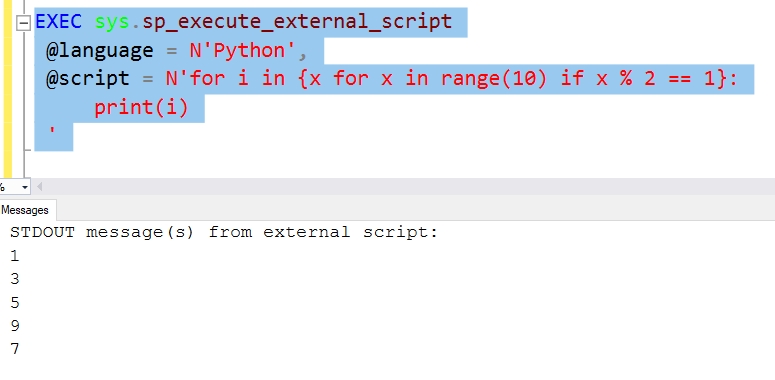
@language = N'Python',

@script = N'for i in {x for x in range(10) if x % 2 == 1}:

print(i)

'

When I run this, I get these results.



Hurray, we’ve executed Python code in SQL Server.

Passing in Data

Running a script without data isn’t very useful. Let’s now look at how we can pass data into a script. The sp\_execute\_external\_script procedure includes a parameter, called @input\_data\_1, that provides input for our Python code. This is query that we pass in, or essentially the result set from that query. We can use a name for this query with another parameter, @input\_data\_1\_name, to create a variable that our Python code uses.

This gets confusing, and it means we’re coupling both the T-SQL code and the Python code together. Let’s see how this works. I’ll write a short query here to get a simple scalar value. I’ll do this without a name, so we will access it inside of the script as InputDataSet. The corresponding default output is OutputDataSet. We’ll just repeat back what goes in to verify things are correct.

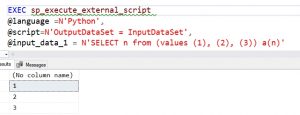
exec sp\_execute\_external\_script

@language =N'Python',

@script=N'OutputDataSet = InputDataSet',

@input\_data\_1 = N'SELECT n from (values (1), (2), (3)) a(n)'

When we run this, we get the following results:



Note that we don’t have any metadata for the results, and we’ve just repeated back the input as output. This shows we can pass data into our script as a query. This must be an NVARCHAR type query and a best practice is to define the query separately, as I have here:

DECLARE @InputQuery NVARCHAR(2000) = N'

SELECT n from (values (1), (2), (3)) a(n)

'

INSERT dbo.Result1

EXEC sp\_execute\_external\_script

@language =N'Python',

@script=N'OutputDataSet = InputDataSet',

@input\_data\_1 = @InputQuery

Getting Specific Results

We had no columns listed above, but we can change this by including the WITH RESULT SETS clause on our code. This allows us to describe the results with metadata about a column name and data type. When we do this, we specify the result set in parenthesis and then each column as we would for a table definition. In my case, I have one column above, but I’ll add a second below to show this.

In this case, I will input two columns and define those with new names as the output.

DECLARE @InputQuery NVARCHAR(2000) = N'

SELECT n, o

FROM (values (1,10), (2,20), (3,30)) a(n, o)

'

EXEC sp\_execute\_external\_script

@language =N'Python',

@script=N'OutputDataSet = InputDataSet',

@input\_data\_1 = @InputQuery

WITH RESULT SETS

( ( INT NOT NULL,

[iplusTen] INT NULL)

)

The WITH RESULT SETS clause is a part of [the EXEC() statement](https://docs.microsoft.com/en-us/sql/t-sql/language-elements/execute-transact-sql?view=sql-server-2017), and allows us to define multiple result sets of specific shapes. The reason there are two sets of parenthesis is for multiple sets. The outer set of parenthesis is for the WITH RESULT SETS clause itself. The inner set is for the first result set.

Getting results as output is good, but sometimes I want to save that data elsewhere and use it in another set of code. I can do that with INSERT .. EXEC. To show this, I’ll create a table to store my data in. In this case, I’ll use the two column result set from above.

CREATE TABLE TwoNumbers

( LowNumber INT

, HighNumber NUMERIC(4, 1));

Now, I’ll use this code to insert data from my procedure call into this table:

DECLARE @InputQuery NVARCHAR(2000) = N'

SELECT n, cast(o as float)

FROM (values (1,10.1), (2,20.2), (3,30.3), (4, 40.4)) a(n, o)

WHERE n < 4

';

INSERT dbo.TwoNumbers

EXEC sp\_execute\_external\_script

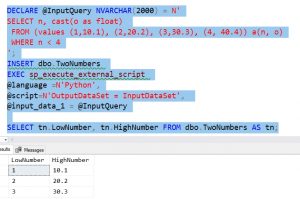
@language =N'Python',

@script=N'OutputDataSet = InputDataSet',

@input\_data\_1 = @InputQuery

SELECT tn.LowNumber, tn.HighNumber FROM dbo.TwoNumbers AS tn;

Note that we don’t use the WITH RESULT SETS clause with an INSERT. This gives us these results:



There are other ways of getting results into a table, but this seems like the simplest to me.

Using Python Packages

One of the things we often want to do with data is process the data with Python functions. Let’s see a short example of this. I’ll call the SUM() function from the numpy package. To do this, I need to use the same type of code structure I do in Python to get the function.

DECLARE @InputQuery NVARCHAR(2000) = N'

SELECT n

FROM (values (1), (2), (3), (4)) a(n)

';

EXEC sp\_execute\_external\_script

@language =N'Python',

@script=N'

import numpy as np

import pandas as pd

OutputDataSet = pd.DataFrame(np.sum(InputDataSet))',

@input\_data\_1 = @InputQuery

In this case, we need two packages: numpy and pandas. numpy gets us the sum function that works with a dataframe. Pandas converts the scalar result back to a dataframe. If we don’t use these packages, we’ll get other errors.

You can use any of the packages that come with the Anaconda 3.5.2 Python distribution in SQL Server 2017, or you can add your own. That’s a topic for another day.

Summary

This is a short look at how to get started executing Python code inside of your SQL Server with SQL Server 2017+. We examined the basics of enabling and using the sp\_execute\_external\_script function that will proxy the call to the python environment on the host instance. We looked briefly at some of the options for input and dealing with results.

There is more ways to work with your Python code and certainly more to consider when executing code inside Machine Learning Services. If there are specific things you would like to see, leave a comment in the discussion. I will look to write more articles in the future covering other ways of working with Python in SQL Server.