PharmaSUG 2023 - Paper QT-165

Running Python Code Inside a SAS® Program

Jim Box, SAS Institute, Cary NC

ABSTRACT

Did you know that you can execute Python code inside a SAS® Program? With the SAS Viya Platform, you can call PROC PYTHON and pass variables and datasets easily between a Python call and a SAS program. In this paper, we will look at ways to integrate Python in your SAS Programs.

INTRODUCTION

SAS has a long-term relationship with open-source programs. PROC IML (Interactive Matrix Language) has enabled programmers to run R code in SAS for years. The new cloud-based SAS environment, SAS Viya, has opened the door for several ways to integrate SAS with Python. We will focus here on using Python code inside a SAS program, but there are multiple other ways to integrate Python code.

All SAS code shown in the paper was run in SAS Studio.

PROC PYTHON

PROC PYTHON in the SAS Viya environment allows you to submit python code directly in a SAS program. It also permits the passage of variables and values between Python and SAS code blocks. The SAS Administrator enables the connection between SAS and Python and manages all packages; inside the PROC PYTHON the user can import libraries and execute any code. PROC PYTHON also provides a module called SAS that facilitates communication between the SAS session and the Python subprocess. (Note that since these methods are called inside a Python code block, they are case-sensitive). These methods fall into three basic groups:

INTERACTING WITH SAS

- **SAS.pyplot()**: allows you to write a PNG file of a matplotlib.pyplot object.
- **SAS.sasfnc()**: allows you to call any SAS or PROC FCMP function. You can assign the results of the function call to a Python variable.
- **SAS.submit("")**: submits all the SAS code contained inside the quotation marks.
- **SAS.symget()**: returns the value of a macro variable that had been assigned in the SAS code. This can be assigned to a Python variable.
- **SAS.symput()**: assigns a value to a SAS macro variable that will be available in the SAS code outside of the Python session.

TRANSFER DATA WITHIN THE PYTHON PROCEDURE

- **SAS.df2ds()**: Transfers data from a Pandas DataFrame to a SAS dataset. This action will overwrite an already existing SAS dataset
- **SAS**.**sd2df**(): Transfers data from a SAS dataset/view to a Pandas DataFrame

CONTROL LOG OUTPUT

- **SAS.hideLOG(True|False):** "True" directs the Python output to a sas2py.log file in the work directory via PROC PRINTTO. "False" sends it back to the SAS log.
- **SAS.printLOG**(): displays in the SAS log all accumulated Python output since the SAS.hideLOG method was called.

EXAMPLES

The easiest way to understand how SAS and Python interact is by examples.

CODING EXAMPLE

```
1
2   %* Define a SAS macro variable in SAS code;
3   %let language = 'python';
4
5   proc python;
6   submit;
7
8   print("Python in the SAS Log:")
9
10  # %*use symget to read a SAS macro variable into a python variable;
11  lang = SAS.symget('language')
12  ver = 3.8
13
14  # %* Submit SAS code inside python, using python syntax.  This dataset will live in WORK library;
15  SAS.submit("data work.test; language={}; version={}; run;".format(lang,ver))
16
17
18  # %* Execute SAS functions with sasfnc;
19  var3 = SAS.sasfnc("upcase","hello world")
20  print( var3)
21
22  # %*Use symput to assign the value of a Python variable to a SAS macro;
23  py_var = 'Inside python'
24  SAS.symput('macrovar', py_var)
25
26
27  endsubmit;
28  run;
29
30  %* Show that the SAS macro variable persists and is populated;
31  %put &=macrovar;
32
33
34  %* Show that the test dataset created in the python code lives in SAS;
35  proc print data=test;
36  run;
```

Display 1. PROC PYTHON call within a SAS program

Everything between the submit statement (Display 1, line 6) and the endsubmit statement (line 27) is Python code – note the lack of semicolons at the end of the lines, and that all comments must be started with a "#". The syntax highlighting currently only applies to the SAS keywords, which can be a little confusing. Keep in mind that Python is case sensitive, and that line indentations mean something specific (used in functions, loops, if..then statements, and the like), so you have to follow Python code formatting inside the Python blocks.

Let's take a closer look at what is happening in the code, then we'll examine the logs.

Code Line	Description
3	Assigns a value to the SAS macro variable **language**, which will be available in the Python code
5-6	Required syntax to initiate the Python code block
8	Prints a line directly into the SAS log (default behavior, can be changed with SAS.hideLOG(True) to redirect messages)
11	Uses the SAS method to assign the value of the SAS macro **language** to the Python variable ***lang***

15	Uses the SAS method to submit the code in quotes to the SAS engine. Notice that the string uses a Python formatting method, so the actual code being submitted is
	`data work.test;`
	`language = 'python';`
	`version = 3.8;`
	`run;`

19	Uses the SAS method to execute the upcase function and assign the results to a Python variable
20	Prints the value of var3 to the SAS log
24	Uses the SAS method to create a SAS marco variable called **macrovar** that assigns the value in the Python variable py_var
27-28	Required syntax to close and execute the Python block
31	Puts the value of the SAS macro variable **macrovar** into the SAS log
35-36	Prints the dataset that was generated in line 15.

Table 1. Explanation of SAS/Python Code

Now let's examine the SAS log to see the results of this code.

```
83   proc python;
84   submit
NOTE: Python initialized.
Python 3.8.5 (default, Sep  4 2020, 07:30:14)
[GCC 7.3.0] :: Anaconda, Inc. on linux
```

Display 2. SAS Log for lines 5-6

Here you can see that Python was initialized, and the version number and source are identified.

```
108  data work.test; language='python'; version=3.8; run;
NOTE: The data set WORK.TEST has 1 observations and 2 variables.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.01 seconds
```

Display 3. SAS Log for line 15

Here you see the actual code submitted to the SAS engine from line 15.

```
>>>
Python in the SAS Log:
HELLO WORLD
>>>
>>>
NOTE: PROCEDURE PYTHON used (Total process time):
      real time           2.36 seconds
      cpu time            0.03 seconds
```

Display 4. SAS Log for lines 8 & 20

All the results of the various print statements in the log will come out in one area at the bottom of the log file, so it is important to be mindful of printing line breaks and using spacing for clarity.

```
110  %* Show that the SAS macro variable persists and is populated;
111  %put &=macrovar;
MACROVAR=Inside python
```

Display 5. SAS Log for line 31

We see that the macro variable assigned in line 24 persists at the end of the Python session as a SAS macro variable.

GRAPHICS EXAMPLE

```
1   %let inputTable = SASHELP.HEART;
2   %let histX = AgeAtStart;
3   %let histGroup = Sex;
4
5   title "Overlay Histograms with Python Seaborn";
6   proc python;
7   submit;
8
9   import seaborn as sns
10  import matplotlib.pyplot as plt
11  import pandas as pd
12
13  df = SAS.sd2df(SAS.symget('inputTable'))
14  Xvar=SAS.symget('histX')
15  groups=SAS.symget('histGroup')
16
17  plt.clf()
18  sns.histplot(data=df, x=Xvar, hue=groups,alpha=0.5,bins=15,kde=False,palette='Set2',multiple="stack")
19  SAS.pyplot(plt)
20
21  endsubmit;
22  run;
```

Display 6. SAS / Python Code to Make a Histogram Plot

Code Line	Description
1-3	Creates SAS macro variables to identify data and graph elements
6-7	Required syntax to initiate the Python code block
9-11	Imports Python packages of interest
13	Uses the SAS method to read a SAS dataset into a Pandas Dataframe
14-15	Uses the SAS method to read SAS macro variables into Python variables
17	Pyplot function to clear the current figure
18	Seaborn function for creating a histogram
19	Uses the SAS method to print the Python plt object to the SAS Studio Results tab. We could have directed the output to a PNG file and saved it to a file location, but with just the defaults selected the graph appears in the Results tab.
21-22	Required syntax to close and execute the Python block

Table 2. Explanation of Code

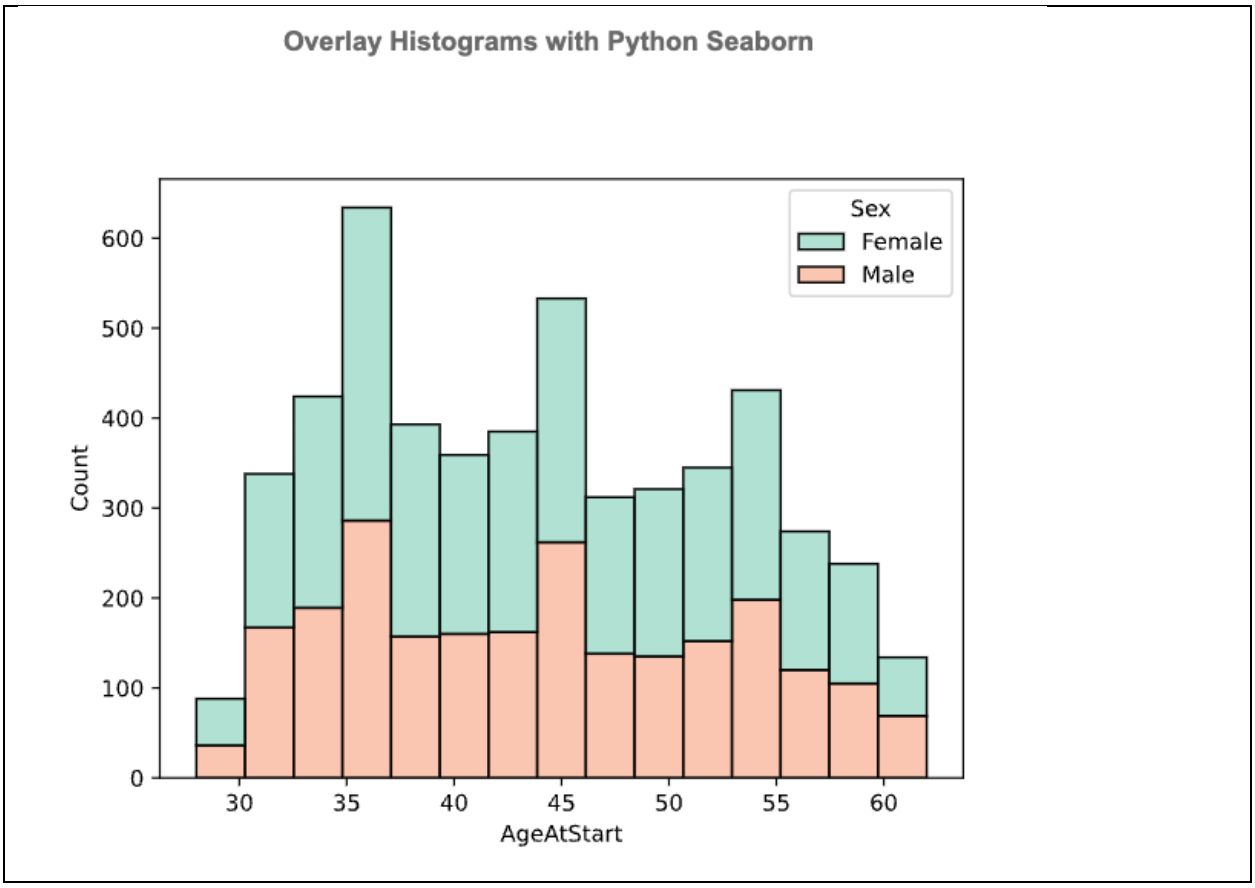

Figure 1. Histogram produced from PROC PYTHON code

CONCLUSION

PROC PYTHON is a powerful tool to add to your programming toolkit. With it, you can produce different graphics, utilize new methods and tools, and combine Python code with SAS code to use the right approach for different parts of your workstreams. There are several ways to use Python (and R) together with SAS, I have included some papers of interest in the recommended reading section.

ACKNOWLEDGMENTS

The author would like to Laura Watson and Brittany Shiver for their editing contributions.

RECOMMENDED READING

SAS DOCUMENTATION

- PROC PYTHON documentation: https://go.documentation.sas.com/doc/en/pgmsascdc/v_017/proc/n0asd2rsj9aedgn1828aptww56of.htm
- SAS & Open Source Integration: https://www.sas.com/en_us/software/viya/open.html
- SASpy Library for Developers: https://developer.sas.com/guides/saspy.html

USER PAPERS

- Foreman, Carrie. SWAT's it all about? SAS Viya ® for Python Users. *SAS Global Forum 2019 Paper 3610-2019*. Available at https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2019/3610-2019.pdf
- Lankham, Isiah & Slaughter, Matthew. Everything is better with friends: Executing SAS ® code in Python scripts with SASpy. *SAS Global Forum 2019. Paper 3189-2019*. Available at https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2019/3189-2019.pdf
- Matise, Joe. Connecting to Datasets through Python and SAS ®. MWSUG Paper 47-2019. Available at https://www.mwsug.org/proceedings/2019/AL/MWSUG-2019-AL-047.pdf
- Nakajima, Yuichi. Utilization of Python in clinical study by SASPy. *SAS Global Forum 2019 Paper 3191-2019*. Available at https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2019/3191-2019.pdf
- Phillips, Jason. A Complete Introduction to SASPy and Jupyter Notebooks. *SAS Global Forum 2019 Paper 3238-2019*. Available at https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2019/3238-2019.pdf
- Vickery, John. Integrate Python with SAS ® using SASPy for a Simpler, More Effective Script. *SESUG Paper 152-2019*. Available at https://www.lexjansen.com/sesug/2019/SESUG2019_Paper-152_Final_PDF.pdf
- Weber, Tom. The History and Evolution of SASPy, Including an Overview of What It Can Do and How to Use It. *SAS Global Forum 2020 Paper SAS4141-2020*. Available at https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2020/4141-2020.pdf

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Jim Box
SAS Institute
Jim.Box@sas.com

APPENDIX: SAMPLE CODE

PROC PYTHON

```
%* Define a SAS macro variable in SAS code;
%let language = 'python';

proc python;
submit;

print("Python in the SAS Log:")

# %*use symget to read a SAS macro variable into a python variable;
lang = SAS.symget('language')
ver = 3.8

# %* Submit SAS code inside python, using python syntax.  This dataset will live in WORK library;
SAS.submit("data work.test; language={}; version={}; run;".format(lang,ver))

# %* Execute SAS functions with sasfnc;
var3 = SAS.sasfnc("upcase","hello world")
print( var3)

# %*Use symput to assign the value of a Python variable to a SAS macro;
py_var = 'Inside python'
SAS.symput('macrovar', py_var)

endsubmit;
run;

%* Show that the SAS macro variable persists and is populated;
%put &=macrovar;

%* Show that the test dataset created in the python code lives in SAS;
proc print data=test;
run;
```

Table 3. SAS / Python Code Interactions

HISTOGRAM IN PROC PYTHON

```
%let inputTable = SASHELP.HEART;
%let histX = AgeAtStart;
%let histGroup = Sex;

title "Overlay Histograms with Python Seaborn";
proc python;
submit;

import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

df = SAS.sd2df(SAS.symget('inputTable'))
Xvar=SAS.symget('histX')
groups=SAS.symget('histGroup')

plt.clf()
sns.histplot(data=df, x=Xvar, hue=groups,alpha=0.5,bins=15,kde=False,palette='Set2',
multiple="stack")
SAS.pyplot(plt)

endsubmit;
run;
```

Table 4. SAS / Python Code to Make a Histogram Plot