



Wilbram Hazejager  
Product Management  
at SAS Institute

# Agenda

- Python Code editor in SAS Studio
- The PYTHON procedure
- Python Program step in SAS Studio Flow
- Integration with SAS Lineage
- How/where does SAS Viya find Python packages?

# Python Code editor in SAS Studio

Open Save All

Python.py x +

Run Cancel | | | | | Copy to My Snippets | + Code to Flow | | Clear Log

Code

```

1 input_table='sashelp.class'
2 output_table='work.class_transposed'
3
4 dfin = SAS.sd2df(input_table)
5 print("input data shape is:",dfin.shape)
6 dfout = dfin.transpose()
7
8 # Use row 0 for column names
9 dfout.columns=dfout.iloc[0]
10 # Remove row 0
11 dfout=dfout[1:]
12
13 print("output data shape is:",dfout.shape)
14 SAS.df2sd(dfout, output_table)

```

Log Output Data (1)

```

1 %studio_hide_wrapper;
77
78 proc python;
79 submit
NOTE: Python initialized.
Python 3.8.11 (default, Oct 21 2021, 07:21:37)
[GCC 8.4.1 20200928 (Red Hat 8.4.1-1)] on linux
Type "help", "copyright", "credits" or "license" for more in
>>>
79 !      ;
80
81 input_table='sashelp.class'
82 output_table='work.class_transposed'
83
84 dfin = SAS.sd2df(input_table)
85 print("input data shape is:",dfin.shape)

```

Allows SAS user to  
test/embed Python code in  
their SAS jobs

TRANSPOSED

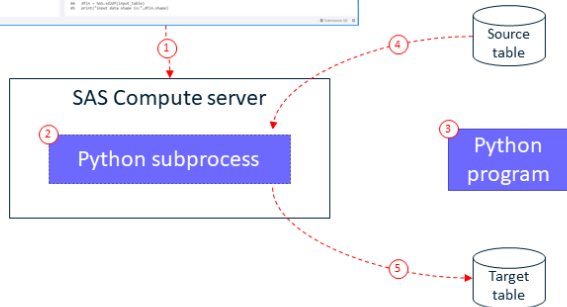
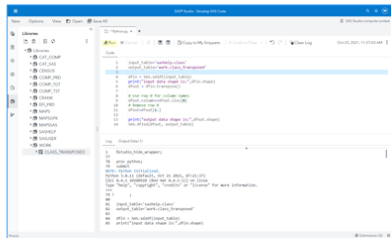
# Python procedure

Starts Python subprocess

Adds constructs to exchange data and information between originating SAS process and Python subprocess

Also constructs to submit SAS code from Python subprocess

SAS Studio Python Code Editor



```
proc python; ②  
submit;
```

```
input_table='sashelp.class'  
output_table='work.class_transposed'  
  
# get input data from SAS into Pandas DataFrame  
dfin = SAS.sd2df(input_table) ④  
print("input data shape is:",dfin.shape)  
dfout = dfin.transpose()  
  
# Use row 0 for column names  
dfout.columns=dfout.iloc[0]  
# Remove row 0  
dfout=dfout[1:]  
  
print("output data shape is:",dfout.shape)  
# Write Pandas DataFrame to SAS  
SAS.df2sd(dfout, output_table) ⑤
```

```
endsubmit;  
quit;
```

# Python Program step in SAS Studio Flow

# Python Program step - Studio Flow

SAS Studio - Develop SAS Code

Flow 1 - Python Program step.flow

Python Program

Name: Python Program

Description: Enter a description

Input Ports and Macro Variables

Input Port: Input tables 1

Macro Variable:

Output Ports and Macro Variables

Output Port: Output tables 1

Macro Variable:

SAS Studio - Develop SAS Code

Flow 1 - Python Program step.flow

```
1 dfin = SAS.sdsd( _input1 )
2 print("input data shape is:",dfin.shape)
3 dfout = dfin.transpose()
4
5 # Use row 0 for column names
6 dfout.columns=dfout.iloc[0]
7 # Remove row 0
8 dfout=dfout[1:]
9
10 print("output data shape is:",dfout.shape)
11 SAS.df2sd(dfout, _output1)
```

- Input port(s) and output port(s) are available as variables in Python process
- Those variables contain names of connected tables (libref.tablename notation)

# Displaying Python graphs in Studio



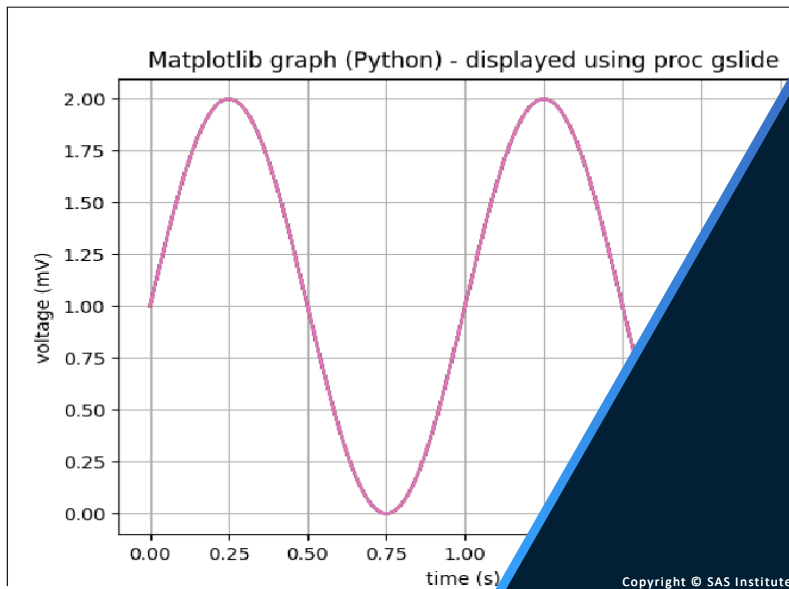
Python - ex2 - show Python graph in SAS results window.py x +

Run Cancel Copy to My Snippets + Code to Flow Clear Log

Code

```
12 plt.ylabel('voltage (mV)')
13 plt.title('Matplotlib graph (Python) - displayed using proc gslide')
14 plt.grid(True)
15 plt.show()
16
17 # Use SAS.workpath, it is a supported object attribute
18 graphfile=SAS.workpath+'matplotlib-ex42.png'
19 plt.savefig(graphfile)
20
21 # As proc gslide is an interactive proc, best practice is to use quit; to stop the proc
22 SAS.submit("proc gslide iframe='{ }' imagestyle=fit; run;quit;".format(graphfile))
```

Log Results

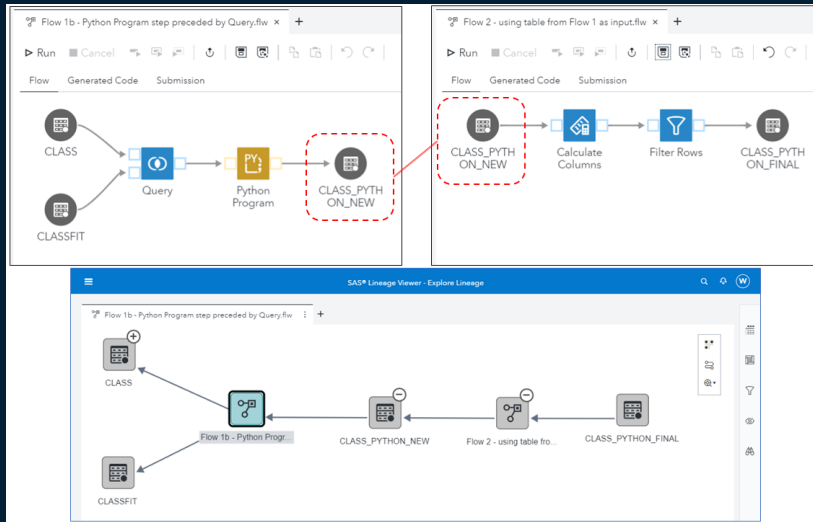


1. Save plot produced by Python package in a graphics file

2. Then display in SAS Studio using proc gslide in a SAS.submit() block

# Integration with SAS Lineage

# Integration with SAS Lineage



- Where is data used?
- Where did my data come from?
- Which Flows touched my data?

And all of this also when *Python Program* steps are being used

**How/where does proc python find  
python.exe and its packages?**

## PROC PYTHON

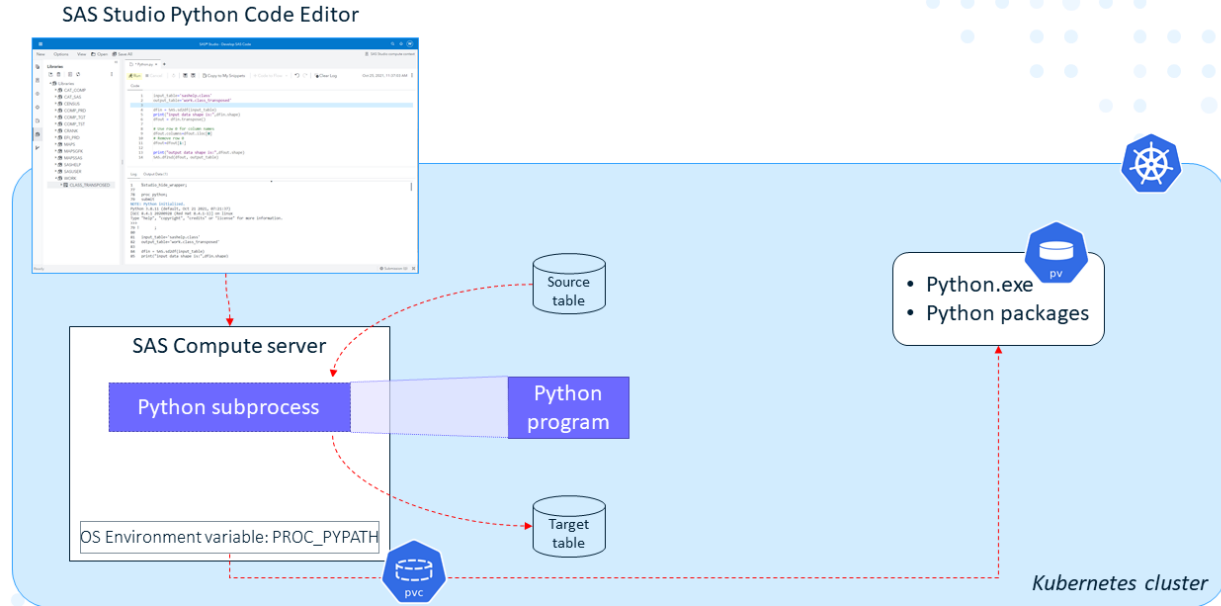
Uses configuration setting specified by Kubernetes/SAS Administrator

SAS Administrator can allow users to override that location from SAS code

Python 3.x is supported

**Note:** Additional SAS Compute Contexts can be created to each point to a specific conda/venv environment

# Location of Python packages



## Wrap up

- Python Code Editor and Python Program steps allow programmers and data scientists to code, execute and schedule Python scripts
- Allows a mix of Python steps \*and\* SAS steps in a single Studio Flow to perform data preparation and analytics
- By using SAS Lineage Viewer, you can quickly see which tables are used where, even when your Studio Flows contain Python Program steps.