WPS Python procedure
user guide and reference

Version: 4.1.1

www.worldprogramming.com

Contents

PYTHON procedure

The PYTHON procedure enables WPS programs written in the SAS language to include code written in the Python language.

Combining the Python language and the SAS language enables the bulk of a data processing and analytics solution to be written in the industrial strength and high-performing SAS language, while also exploiting features present in the Python language.

Introduction

The Python procedure enables SAS language programs to include code written in the Python language.

By combining Python and the SAS language you can:

- Use the SAS language to bulk process and prepare data, and pass the processed data to Python.
- Use Python packages you have previously developed for data analysis.
- Use Python data analysis packages or solutions that may not be available in the SAS language.

Data is passed between the SAS language and Python language environments using the `EXPORT` statement. Once data has been transferred, that data is made available as a pandas DataFrame to a Python program. On completion of the Python program data can, if required, be returned to the SAS language environment using the `IMPORT` statement.

Setup and configuration

Setting the Python environment for WPS.

When using Python with WPS:

- The 32-bit Python interpreter is required for the 32-bit version of WPS, and the 64-bit Python interpreter is required for the 64-bit version of WPS.
- The `pandas` package must be installed with the Python interpreter. This can be checked using the pip utility by running `pip list` on the command line.

The WPS distribution does not include either the Python interpreter or pandas package. If you do not have Python installed, you can obtain a copy of the Python interpreter from *https://www.python.org* or install a packaged Python environment that includes the necessary modules.

The `PYTHON` procedure can be used with Python version 3.5.0 and later, and is currently supported on Microsoft Windows, Linux-based systems, and macOS.

The procedure is not currently supported on IBM mainframe.

Python environment variables

You must set the *PYTHONHOME* environment variable for WPS to locate and use the Python interpreter. This variable must reference the folder where the main Python library is located – for example, `python3.dll` on Microsoft Windows.

Standard output and error streams

The Python standard output stream (`sys.stdout`) and standard error stream (`sys.stderr`) are redirected to WPS output when the procedure is run:

- `sys.stderr` is redirected to the WPS log file.
- `sys.stdout` is redirected to the WPS listing file when WPS is run on the command line, or to all specified ODS output destinations when Workbench is used.

If you use the Python `print()` function to put output to the WPS listing file, the number of characters appearing in the listing output is determined by the WPS `LINESIZE` system option. The `LINESIZE` option can be used to print strings of up to 256 characters; the use of the `print()` function should therefore be for limited information such as log statements.

If you want to return large volumes of Python-generated content to WPS, create a DataFrame containing the required content and import the DataFrame into WPS using the `IMPORT` statement.

The following example creates a list of functions available in the Python pandas package. The list is printed using `print (fnList)` and also converted to a DataFrame and imported into WPS.

```
PROC PYTHON;
SUBMIT;
import inspect
fnPandas = inspect.getmembers(pandas, inspect.isfunction)
fnList = [fn[0] for fn in fnPandas]
print (fnList)
fnTable = pandas.DataFrame({'function': fnList})
ENDSUBMIT;
IMPORT DATA=PANDAFN PYTHON=fnTable;
RUN;
```

The printed output truncates the list of functions:

```
['Expr', 'Term', 'bdate_range', 'concat', 'crosstab',
 'cut', 'date_range', 'eval', 'factorize',
```

When converted to a DataFrame and imported as a WPS dataset, all available functions are listed.

Using Python with WPS

Using Python in a SAS language program enables you to make use of specialist Python packages such as `Scikit-learn` or `Tensorflow`.

The first time the Python procedure is invoked in a SAS language program, WPS automatically imports the `pandas` and `numpy` packages. You can access the functionality in the `pandas` and `numpy` packages in an in-line Python language program – written between the `SUBMIT` and `ENDSUBMIT` statements – by referencing the fully-qualified package, class or function name, for example:

```
PROC PYTHON;
  SUBMIT;
content = pandas.read_csv('file:///C:/project/sourcedata/wps.csv')
...
  ENDSUBMIT;
 RUN;
```

Alternatively, you can use the `import ... as ...` statement to alias either the `pandas` or `numpy` package name, for example:

```
PROC PYTHON;
  SUBMIT;
import pandas as pd
content = pd.read_csv('file:///C:/project/sourcedata/wps.csv')
...
  ENDSUBMIT;
 RUN;
```

Other Python packages can be imported and used within the in-line Python code, for example:

```
PROC PYTHON;
  EXPORT DATA=source;
  SUBMIT;
import statsmodels.formula.api as lm
result = lm.ols(formula='x ~ y + z', data=source).fit()
...
  ENDSUBMIT;
RUN;
```

Each subsequent use of the PYTHON procedure in a SAS language program can use the same Python environment. This means any global variables or imported packages used in a PYTHON procedure invocation are available to all subsequent PYTHON procedure invocations.

Each PYTHON procedure invocation can include multiple blocks of in-line Python language code, and use a combination of in-line Python language code, and Python programs run using the `EXECUTE` statement.

Data type conversion

Describes the correspondence between SAS language formats and pandas data types.

This section describes the correspondence between WPS formats and Python pandas data types. WPS has many formats that affect the output and display of data. When you write data to a pandas DataFrame using the Python procedure `EXPORT` statement, formatted data is converted to an equivalent and pandas or numpy data type.

Many formats only affect the layout of data output, such as adding currency symbols or comma separators, and these formats have no effect when writing data.

SAS language unformatted data to Python

Unformatted data is converted to a pandas DataFrame type as follows:

WPS format	Python data type	Notes
Unformatted number	`float64`	
Unformatted string	`object`	The maximum object length for a variable is not known as part of the dataset metadata.

SAS language formatted data to Python – numbers

The core numeric formats are converted to a pandas DataFrame type as follows:

WPS format	Python data type	Notes
w.d	Float64	
BEST. and BEST*w*.	float64	
FLOAT*w.d*	float64	

SAS language formatted data to Python – strings

The core character formats are converted to a pandas DataFrame type as follows:

WPS format	Python data type	Notes
$*w*. $CHAR*w*. $F*w*.	object	The maximum object length for a variable is not known as part of the dataset metadata.

SAS language formatted data to Python – dates and times

Date, datetime and time formats are converted to a pandas DataFrame type as follows:

WPS format	Python data type	Notes
DATE*w*.	datetime64[ns]	Numpy datetime type.
DDMMYY*w*. and all variants (such as DDMMYYB*w*., MMDDYYS*w*., and YYMM*w*.)	datetime64[ns]	Numpy datetime type.
DTDATE*w*. and all variants (such as DTMONYY*w*. and DTWKDATX*w*.	datetime64[ns]	Numpy datetime type.
TIME*w*., HOUR*w*., HHMM*w*. and all similar time formats.	float64	
JULIAN*w*. and all similar date formats.	datetime64[ns]	Numpy datetime type.

Python pandas data types to SAS language dataset

SAS language datasets only contain numeric and character data. Formats might be applied to the data in the dataset to more closely represent the source data from Python. Importing a pandas DataFrame using the IMPORT statement converts data as follows:

Pandas data type	WPS format	Notes
Object	Character	The maximum object length is calculated before importing.
int64	Numeric	
bool	Numeric	*True* is converted to 1; *False* is converted to zero (0).

Pandas data type	WPS format	Notes
Float64	Numeric	
datetime64[ns]	Numeric	Formatted as `DATETIME19.`

Some values in Python cannot be represented in the same manner is a SAS language datasets. The following table shows how these values are converted:

Pandas data value	WPS value	Notes
null string (`''`)	`' '`	Missing character value.
`True`	`1`	Numeric value
`False`	`0` (zero)	Numeric value
`NaN` Defined with `float('nan')`	`.`	Missing numeric value
Infinity Defined with `float('inf')` Defined with `float('-inf')`	`.`	Missing numeric value

Import custom Python modules

How to use your own packages and modules in Python.

To use custom packages, they must be accessible to Python. The paths to locations containing custom packages are specified by the Python *sys.path* variable. The *sys.path* variable constructs a list of locations to search using the *PYTHONHOME* and *PYTHONPATH* environment variables, or the locations can be specified by modifying the value of the variable during the execution of a program.

- `sys.path` contains a list of folders searched for packages. The variable is constructed from the values in *PYTHONHOME* and *PYTHONPATH*.
- *PYTHONHOME* specifies the location of the Python interpreter and standard libraries, including packages installed into *site-packages* using a package manager such as *PIP*. The variable must be specified for WPS to interact with Python.
- *PYTHONPATH* specifies a list of folders to search for packages. This list is prepended to the search list defined in `sys.path`.

The first item in `sys.path` is either the directory containing the python program or an empty string, interpreted by Python as the current directory. When run from WPS, `sys.path[0]` contains the first search path specified in either *PYTHONPATH* or *PYTHONHOME*. If you program references packages in the current directory, you must modify `sys.path` when `PROC PYTHON` is running. See Modifying the *sys.path* variable (page 9)

Set *PYTHONPATH* before running WPS

The *PYTHONPATH* variable can be defined as a system variable, and used by all applications on your device that run Python.

If you have multiple installations of Python, for example Python 2 and Python 3, setting *PYTHONPATH* using a system variable might cause your program to attempt to import the incorrect version of a package. In these circumstances, you should set the variable as part of the SAS language program you run in WPS.

Set *PYTHONPATH* in a SAS language program

The *PYTHONPATH* can be set in a SAS language program using the `SET` system option. For example, to use packages stored in `C:\temp\python` folder, the following can be added to a program before the `PROC PYTHON` statement:

```
OPTIONS SET = PYTHONPATH 'C:\temp\python';
```

If *PYTHONHOME* is specified as `C:\python3`, the content of the Python `sys.path` variable is:

```
['C:\\temp\\python', 'C:\\python3\\python37.zip',
 'C:\\python3\\DLLs', 'C:\\python3\\lib', 'C:\\python3',
 'C:\\python3\\lib\\site-packages']
```

Modifying the *sys.path* variable

The Python *sys.path* variable can be modified programmatically by adding the following to a Python language program:

```
import os, sys
sys.path.append(os.getcwd())
```

If a Python language program run using the `EXECUTE` statement includes other Python files, add the above as an in-line program before the executed program to enable WPS to locate the included files:

```
PROC PYTHON;
  SUBMIT;
import os, sys
sys.path.append(os.getcwd())
  ENDSUBMIT;
  EXECUTE 'programs/dataCollect.py';
RUN;
```

Using graphics created by Python

Any graphics generated using functionality in Python are captured and can be included in the WPS session's standard ODS output.

WPS creates a temporary variable, *wpsgloc* pointing to a temporary folder that is used to store graphics for inclusion in ODS output. The variable must be added to the name of image file every time you create an image. This can be done using, for example, the `os.path.join()` Python function:

```
PROC PYTHON;
SUBMIT;
import os
import matplotlib.pyplot as plt
...
plt.savefig(os.path.join(wpsgloc, 'my_image.png'))
ENDSUBMIT;
RUN;
```

The *wpsgloc* variable cannot be modified in the SAS language program, and is either:

- The `WORK` location when run in Workbench, for example:

  ```
  C:\Users\user-id\AppData\Local\Temp\WPS Temporary Data
  ```

- The working directory when WPS Analytics in run on the command line.

```
ODS PDF FILE='scatter_plot.pdf';
PROC PYTHON;
  EXPORT DATA=STATS PYTHON=df;
  SUBMIT;
import os
import matplotlib.pyplot as plt
df.plot(kind='scatter', x='i', y='j')
plt.savefig(os.path.join(wpsgloc, 'scatter.png'))
  ENDSUBMIT;
RUN;
ODS PDF CLOSE;
```

wpsgloc is a temporary location for graphics images used in Workbench.

When run on the command line, the graphics are created in the current directory. The graphics are referenced in HTML output and copied into PDF output.

Example

Demonstrates how to use a SAS language dataset in PROC PYTHON to create a scatter plot diagram.

The following example creates a dataset in a SAS language DATA step, and then uses the EXPORT statement to pass that dataset to the Python environment. The dataset is converted to a pandas DataFrame as part of the export, and the DataFrame is used to create a scatter plot using Matplotlib.

An output PDF file destination is created using the SAS language Output Delivery System (ODS). Adding PDF to the output destinations includes the returned scatter plot image file in the PDF output. The PDF is saved and the output can be viewed in a PDF viewer.

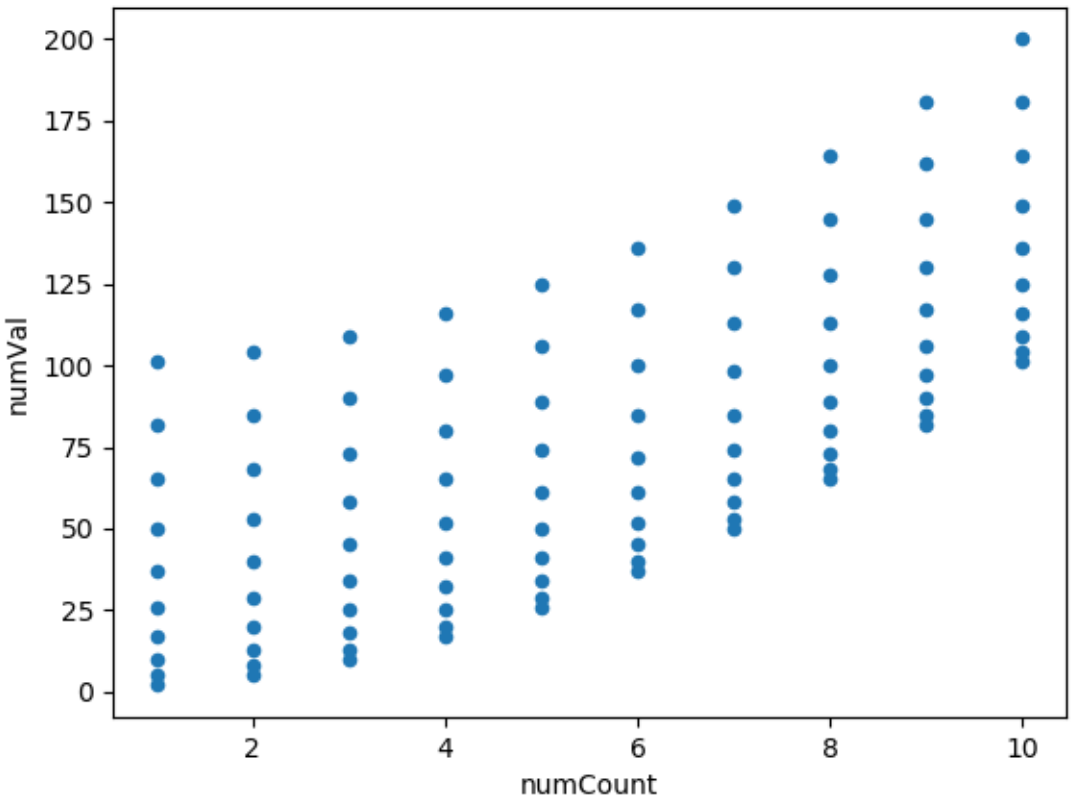
This example requires the following Python packages:

- SciPy
- Matplotlib

```
ODS PDF FILE='scatter_plot.pdf';
DATA stats (drop = count);
  DO count=1 TO 10;
    DO numCount=1 TO 10;
      numVal = (numCount*numCount)+(count*count);
      OUTPUT;
    END;
  END;
RUN;

PROC PYTHON;
  EXPORT DATA=STATS PYTHON=df;
  SUBMIT;
import os
import matplotlib.pyplot as plt
df.plot(kind='scatter', x='numCount', y='numVal')
plt.savefig(os.path.join(wpsgloc, 'scatter.png'))
  ENDSUBMIT;
RUN;
ODS PDF CLOSE;
```

This creates the following scatter plot in the ODS PDF output:



Python procedure reference

Describes the syntax and options for `PROC PYTHON` and its contained statements.

PROC PYTHON

Invokes the Python environment that enables the execution of in-line or external Python language programs.

```
PROC PYTHON [ options ] ;
```

Datasets created in WPS can be made available to the Python program using the EXPORT statement, and a dataset imported from the Python program into WPS using the IMPORT statement.

A Python program can be either written in-line in the PYTHON procedure, or run from a separate file:

- To run an in-line Python program, use the SUBMIT and ENDSUBMIT statements.
- To run a Python program stored in an external file use the EXECUTE statement.

The Python environment is exited using a RUN statement.

When the Python environment is invoked, the pandas and numpy packages are automatically loaded. These packages can be used either by quoting the full package name in code, or by using the import ... as ... statement to alias the package name.

Options

The following options are available with the PROC R statement.

KEEP

Specifies that the current Python environment is not terminated when the procedure exits.

KEEP

When specified, the current Python environment is kept active when the current procedure exits, and the environment is used in the next invocation of the Python procedure in the same program. If that invocation does not specify KEEP, the environment is terminated when the procedure exits.

The default behaviour is to terminate the Python environment at the end of the procedure. Specifying KEEP keeps the current Python environment, including any modules loaded during the execution of a Python program, to be used in the next invocation of PROC PYTHON.

You can specify the PYTHONKEEP system option to use the same Python environment for the duration of the execution of the SAS language program.

LIB

Specifies the default library location for the procedure step. The default location is the WORK library.

LIB = *library-reference*

The LIB location is used when *libname* is not specified as part of the path for the DATA option of either the EXPORT or IMPORT statements.

TERMINATE

Specifies that the Python environment is terminated when the procedure exits.

[TERMINATE | TERM]

Specifying TERMINATE stops the current Python environment even if the PYTHONKEEP system option has been specified. A subsequent invocation of PROC PYTHON in the same program only has the default pandas and numpy packages loaded.

Example

The following example shows how to invoke the PYTHON procedure to print hello world to ODS output.

```
PROC PYTHON;
SUBMIT;
print ('Hello World')
ENDSUBMIT;
RUN;
```

EXECUTE

Runs a Python program stored in a separate file.

EXECUTE " *filename* " [{ *cmd-argument* ... }] ;

The EXECUTE statement is an alternative to using the SUBMIT statement. It enables Python code placed in a separate file to be run WPS Analytics.

Execute options

The following options are available with the EXECUTE statement.

filename

A string, in quotation marks, containing the path of the Python program file. *filename* can be either an absolute pathname or a relative pathname.

In Workbench, the path for relative file pathnames is the Workspace. For example, to run a file named myProgram.py from a project named python, the relative path is /python/myProgram.py.

cmd-argument

Specifies a command line argument passed to the Python program. Arguments are accessed in the programing from sys.argv.

All command line arguments are passed to the Python program as strings. Numeric arguments must be in quotation marks, and the Python program must convert the arguments to the required numeric type.

Basic example

In this example, a Python program stored in an external file is executed in the PYTHON procedure. The file is referenced using the absolute path:

```
PROC PYTHON;
  EXECUTE 'C:\temp\printDS.py';
RUN;
```

Example – pass an argument to a Python program

In this example, an external Python program multiple.py is executed that returns the square of a specified value. The value is specified in a variable.

```
import sys

def multiply (value):
  return value*value

print(multiply(int(sys.argv[0])))
```

multiple.py is executed from the PYTHON procedure in a SAS language program, and the required value to multiply is passed to the Python program as an argument to the EXECUTE statement. The numeric argument *6* is passed as a string. The Python program uses the built-in int() function to return an integer before calculating the square.

```
PROC PYTHON;
  EXECUTE 'C:/temp/multiple.py' '6';
RUN;
```

This produces the following in ODS output:

```
36
```

EXPORT

Converts a SAS language dataset to a pandas DataFrame.

```
[ EXPORT | SEND ] DATA = [ libname . ] dataset
  [   PYTHON = dataframe-name
  ] ;
```

Dataset preparation should be completed before exporting the data to Python. This enables you to use the dataset processing capability of the SAS language to create an export dataset containing only the required data for the Python language program.

Export options

The following options are available with the EXPORT statement.

DATA

Specifies the dataset location and name of the SAS language dataset to be converted.

The library location can be specified using either *libname* in the DATA option, or the LIB option of the PROC PYTHON statement.

- If *libname* is specified, that location is always used.
- If the LIB option of the PROC PYTHON statement is specified and *libname* is not specified, the location in the LIB option is used.
- If neither *libname* nor the LIB option on the PROC PYTHON statement are specified, the WORK location is used.

PYTHON

Specifies the name of the pandas DataFrame as used in the Python language program.

The pandas DataFrame name in Python language code is case sensitive, and the *dataframe-name* specified must match the case used of the Python variable name.

If this option is not specified, the *dataframe-name* default is the *dataset* name specified in the DATA option. If you use the default *dataset* name in an in-line Python language program, the variable name must match the case used in the DATA option.

Example – export a SAS language dataset to a pandas DataFrame

The following example creates a dataset in a SAS language DATA step. The dataset is then exported to a pandas DataFrame and the column types printed out.

```
DATA TESTDATA;
INFILE CARDS DLM='#';
INPUT num char $;
CARDS;
1 # Hello
2 # World
;
RUN;

PROC PYTHON;
EXPORT DATA=TESTDATA PYTHON=dframe;
SUBMIT;
print (dframe)
ENDSUBMIT;
RUN;
```

Which writes the following to ODS output:

```
   num      char
0  1.0      Hello
1  2.0      World
```

IMPORT

Enables a pandas `DataFrame` to be converted to a SAS language dataset and referenced in a SAS language program.

[`IMPORT` | `RECV` **]** `PYTHON` = *dataframe-name* **{** `DATA` = **[** *libname* `.` **]** *dataset* **...** **}** `;`

Import options

The following options are available with the `IMPORT` statement.

DATA

Specifies the dataset location and name as used in the WPS Analytics SAS language environment.

The library location can be specified using either *libname* in the `DATA` option, or the `LIB` option of the `PROC PYTHON` statement.

- If *libname* is specified, that location is always used.
- If the `LIB` option of the `PROC PYTHON` statement is specified and *libname* is not specified, the location in the `LIB` option is used.
- If neither *libname* nor the `LIB` option on the `PROC PYTHON` statement are specified, the `WORK` location is used.

PYTHON

Specifies the name of the pandas `DataFrame` as used in the Python environment. Must be specified.

dataframe-name is case sensitive and must match the case used for the imported pandas DataFrame in the Python program.

SUBMIT

Specifies the start of an in-line Python language program.

```
SUBMIT ;
```

An in-line program is defined as part of the PYTHON procedure in a SAS language program. The `SUBMIT` statement marks the start of the program, and the `ENDSUBMIT` statement marks the end.

The Python language program must start on a new line after the SUBMIT statement, and the ENDSUBMIT statement must appear at the beginning of a line on its own. The first line of the Python program code must not start with any white space and any subsequent statements must follow the normal Python requirements for indentation, for example:

```
PROC PYTHON;
  SUBMIT;
fruits = ['apple', 'banana', 'cherry', 'damson', 'elderberry', 'fig']
for fruit in fruits:
    print(fruit)
  ENDSUBMIT;
RUN;
```

Multiple in-line Python language programs can exist in a single PYTHON procedure. Each Python language program is executed as it is encountered. Variables defined in one in-line language program can be used in subsequent in-line programs in the same Python procedure.

ENDSUBMIT

Specifies the end of an in-line Python language program.

```
ENDSUBMIT ;
```

The ENDSUBMIT statement must be entered at the start of a new line after the Python language program.

Legal Notices

Trademarks

WPS and World Programming are registered trademarks or trademarks of World Programming Limited in the European Union and other countries. (r) or ® indicates a Community trademark.

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

All other trademarks are the property of their respective owner.

General Notices

World Programming Limited is not associated in any way with the SAS Institute.

WPS is not the SAS System.

The phrases "SAS", "SAS language", and "language of SAS" used in this document are used to refer to the computer programming language often referred to in any of these ways.

The phrases "program", "SAS program", and "SAS language program" used in this document are used to refer to programs written in the SAS language. These may also be referred to as "scripts", "SAS scripts", or "SAS language scripts".

The phrases "IML", "IML language", "IML syntax", "Interactive Matrix Language", and "language of IML" used in this document are used to refer to the computer programming language often referred to in any of these ways.

WPS includes software developed by third parties. More information can be found in the THANKS or acknowledgments.txt file included in the WPS installation.