Everything is better with friends: Executing SAS® code in Python scripts with SASPy

PYTHON SYNTAX OVERVIEW FOR SAS PROGRAMMERS

Python is a versatile programming language with syntax resembling DATA steps in SAS, but with the following important differences.

Capitalization is significant

These are **not** equivalent:

```
print('Hello, World!')
```

```
PRINT('Hello, World!')
```

Semicolons are only used to separate multiple statements on the same line

These are equivalent:

```
message = 'Hi!'
print(message)
```

```
message = 'Hi!'; print(message)
```

There are multiple, (mostly) interchangeable quoting styles

These are (mostly) equivalent:

```
'Hi!'
```

```
"Hi!"
```

```
'''Hi!'''
```

```
"""Hi!"""
```

[Strings surrounded by triple quotes can contain embedded line breaks.]

Assignment (=) and equality testing (==) use different operators

These are **not** equivalent:

```
fireworks = 'Yes!'
```

```
fireworks == 'Yes!'
```

White space is significant (and used to determine scope)

These are **not** equivalent:

```
if fireworks == 'Yes!':
    print('🎆')
```

```
if fireworks == 'Yes!':
print('🎆')
```

[The non-indented version produces an error because the if-statement has no body.]

REPLICATING HANDS-ON WORKSHOP EXAMPLES

Step 1: Download and install the freely available VirtualBox (see https://www.virtualbox.org/) and SAS University Edition (see https://www.sas.com/en_us/software/university-edition/download-software.html).

Step 2: Start SAS University Edition, and access JupyterLab within a web browser (see https://support.sas.com/software/products/university-edition/faq/jn_runvirtualbox.htm).

Step 3: Download the repo https://github.com/saspy-bffs/wuss-2019-how (e.g., using the "Clone or Download" button to "Download ZIP").

Step 4: Load the file *WUSS2019-HOW-Everything_Is_Better_With_Friends-examples.ipynb* into JupyterLab (see https://jupyterlab.readthedocs.io/en/stable/user/files.html).

Step 5: Follow the instructions in the examples file.

Step 6: See https://jakevdp.github.io/WhirlwindTourOfPython/ for a free, concise overview of Python.

REPLICATING HANDS-ON WORKSHOP DEMOS

Because Python[1] is community-driven, there is immense flexibility in how it can be used. These instructions use SASPy inside the popular Integrated Development Environment (IDE) PyCharm[2] and were developed under Windows 10 with SAS 9.4, Java SE version 8 update 201, Python 3.7.3, Git 2.21.0, and PyCharm Professional Edition 2019.1.1 installed. All default installation options are recommended.

Step 1: Setup development environment.

a. Download and install Java SE (to be used by SASPy to invoke SAS) from https://www.java.com/
b. Download and install Python (i.e., the CPython implementation) from https://www.python.org/
c. Download and install Git (to be used by PyCharm's GitHub integration) from https://git-scm.com/
d. Download and install PyCharm from https://www.jetbrains.com/pycharm/

Step 2: Setup project in PyCharm.

a. Start PyCharm, and when prompted, choose **Check out from Version Control → Git**, enter URL https://github.com/saspy-bffs/wuss-2019-how, choose a directory to copy the files to, and click **Clone**.
b. Use the menu commands **File → Settings → Project: wuss-2019-how → Project Interpreter**. Then click the **Gear Icon** (⚙) in the upper-right corner of the dialog box and select **Add…** This should prompt you to create a new virtual environment[3] as a subfolder named *venv* in your project folder. Once setup, click **OK** and, once processing has finished, click **OK** again to exit the dialog box.
c. Use the menu command **View → Tool Windows → Terminal** to open a terminal window. (Alternatively, click **Terminal** at the bottom of the PyCharm window.) Then type the following at the command prompt and press Enter: **`pip install -r requirements.txt`**
d. Use the project-navigation area in the left-hand panel to open the file *sascfg_personal-example.py* (i.e., double-click its name), and copy its contents to the system clipboard. Then use the menu command **File → New → Python File** to create a new file named *sascfg_personal.py*, paste the contents of the system clipboard into it, and update to match your SAS installation setup per the instructions at https://sassoftware.github.io/saspy/install.html. (Warning: This is **not** straightforward.)

Step 3: Run the example file.

a. Use the menu command **Run → Run … → WUSS2019-HOW-Everything_Is_Better_With_Friends-examples.py**, and then watch the output scroll by in the Run portion of bottom panel. (Alternatively, open the file *WUSS2019-HOW-Everything_Is_Better_With_Friends-examples.py* by double-clicking its name in the project-navigation area in the left-hand panel, right-click anywhere inside the code editor window, and choose the command **Run 'WUSS2019-HOW-Everythi…'**)

Step 4: Repeat with a full-application example using the popular Python web framework Flask.

a. Use the menu command **VCS → Checkout from version control → Git** to create a new project from https://github.com/saspy-bffs/dataset-explorer and repeat Step 2(b-d). Then run *app.py*, visit the URL http://127.0.0.1:8000/ in a web browser, and follow the instructions. (If SAS was installed with default options, try the directory **C:\Program Files\SASHome\SASFoundation\9.4\core\sashelp**)

---

[1] When most people talk about the Python language, they mean the C-based *CPython* reference implementation hosted at https://github.com/python/cpython. Other implementations of the Python language specification include Java-based *Jython* (https://www.jython.org/) and .NET-based *IronPython* (https://ironpython.net/). An alternative CPython implementation for data-science applications can also be installed as part of the Anaconda distribution (https://www.anaconda.com/distribution/), which includes many popular data-science packages like NumPy, pandas, scikit-learn, and TensorFlow; however, the Anaconda distribution has its own separate form of virtual environment called a *conda environment*.

[2] You can choose between PyCharm Community Edition, which is free and open-source, and PyCharm Professional Edition, which has many additional features but requires a commercial license.

[3] A *virtual environment* (aka *venv*) is essentially a completely separate installation of Python, which is cloned from the version of Python installed as Step 1(b). It's considered best practice to create a new venv (or conda environment, if using the Anaconda distribution of Python instead of the one from https://www.python.org/) for each project in order to keep its dependencies isolated from other projects and their dependencies.