# SAS with Jupyter NB

# Agenda

- Why SAS with Jupyter?
- Installation steps
- Configuration
- Markdown
- Use Cases Demonstration

# Why Jupyter?

# Why SAS with Jupyter Notebook?

- Integrating SAS® into modern analytics development tools used by data scientists:
  - Allows collaboration among developers that are proficient in diverse tools
  - Entry point for launching into new analytic platforms such as SAS Viya Workbench, Python, and even Apache Spark etc., resulting in
  - Tackling data wrangling challenges without additional licensing (with just 9.4 license)

# What is Jupyter Notebook?

- Jupyter Notebook is a unique integrated development environment (IDE) which was mainly created for Python programming.
- It is ideal for analytics coding because it offers interactive programming, which simply means that the user can write and execute code within a single interface, as well as display many kinds of output directly inline with blocks of code.
- Jupyter is browser-based in that its user-interface application runs inside a standard web browser.
- Its most typical usage is entirely local to a single machine, which Jupyter orchestrates by creating a background Python process on your workstation that communicates with the web front-end via a local port.
- Shared deployments for running the background process on a remote server do exist (e.g., Google collab, Kaggle, Conda online etc.)

# What is Jupyter Notebook?

- While the Jupyter project grew out of Python, its notebooks can use a number of different languages, including R, Scala, Java, and as my demonstration shows today, even SAS®.
- To use additional languages, you must install a “kernel” that executes your code in another process, with Python acting as a bridge.
- Most kernels additionally provide syntax highlighting or other editing features like inline documentation and autocompletion.

# SAS Kernel for Jupyter

- It should be clear by now, that SAS Kernel for Jupyter Notebooks is needed to run SAS programs from within the Jupyter interface.
- To use the kernel, you must, of course, have an existing installation of SAS accessible to you, either on your local machine, or on the network (Workspace Server).
- The SAS kernel allows you to leverage all of the SAS products that are licensed on your device/network.
- After installing the SAS kernel, you can use a notebook and a SAS installation to write, document, and submit SAS programming statements.
- The Jupyter notebook interface allows sharing of results through JSON and the SAS kernel is no exception, you can share code and results in a static form through the Jupyter notebook.

# Deployment Strategy

- Prerequisites for the successful deployment of SAS Kernel for Jypyter
  - Python3 or higher
  - Jupyter version 4 or higher
  - SAS 9.4 or higher -- This includes SAS Viya.
  - The SAS kernel is compatible with any version of SAS released since July 2013.
  - SASPy -- The SAS kernel has a dependency on SASPy.
  - SASPy will be installed automatically but it must be configured to access your available SAS server.
  - SASPy must be configured before the SAS kernel can work successfully (we will go through this process of configuration) but first...

# What is SASPy?

- SASPy is an open source Python package that allows Python coders to access SAS data and analytics capabilities.
- Add SASPy to your Python environment, configure it to connect to your SAS environment, and begin using Python to access SAS.
- SASPy can be used directly (within Python code apart from the Jupyter environment, and has significant capabilities beyond the essential function of relaying SAS code and output.
- it is a powerful generator of SAS code, which means that it offers methods, objects, and syntax for use in Python that it can automatically convert to the appropriate SAS language statements for execution.
- It achieves this integration largely by creating references that act as interfaces between the two environments, using the popular Python package Pandas (facilitates coordination with Pandas DataFrame).

# Installation and Configuration Steps

# Starting Points

SAS Software Github Page
https://github.com/sassoftware/sas_kernel

SAS Kernel for Jupyter Documentation
https://sassoftware.github.io/sas_kernel

- There are different options for installing the kernel.
- These are given on the github site or in the documentation.
- In my case, I use the Anaconda ecosystem… so I installed anaconda install.

SAS Software Github Page:
https://github.com/sassoftware/saspy

SASPy Documentation
https://sassoftware.github.io/saspy

- One thing to note is things can change on a github repo.
- Sometimes essential changes are not reflected in the documentations in a timely way.

# Check Points

- Make sure you know where your installation resources are created.
  - If you install under a specific environment vs. the “Base” environment.
    - Sometimes c:\users\[acct]\AppData\local\anaconda3
    - Sometimes c:\users\[acct]\anaconda3
  - Make sure to note which Python package it is installed under and make sure it is not overridden during runtime.
    - Issue “where python” (Windows) or “which Python” (Linux)

# Configuration

- SAS kernel does not need configuration but SASPy does.
- Since the kernel depends on SASPy, before using the kernel make sure you configure SASPy
- Configuration is mainly about how the package connects to the SAS executable.
- SASPy can connect and start different kinds of SAS sessions.
    - It can connect to SAS on Unix, Mainframe, and Windows. It can connect to a local SAS session or remote session.
    - Because of the wide range of connection types, there are a number of different access methods which are used to connect to different kinds of SAS sessions.

# Configuration

- The current set of connection methods are as follows
  - STDIO: This connection method is available on the Linux platform only. This method enables you to connect to SAS on the same host as your Python process.
  - SSH: This connection method (STDIO over SSH) is available from Linux or Windows (or Mac), but only to SAS on the Linux platform. This method can connect to SAS that is installed on a remote host, if you have passwordless SSH configured or, new in version V4.3.0, you can use sshpass to provide user/pw.
  - IOM: The Integrated Object Method (IOM) connection method (IOM using Java) supports SAS on any platform. This method can make a local Windows connection and it is also the way to connect to a SAS Workspace Server on any SAS platform from any client. This includes SAS Grid deployments too.
  - HTTP:  This access method uses http[s] to connect to the Compute Service (a micro service) of a Viya installation. This does not connect to SAS 9.4 via http. The Compute Service will start a Compute Server using the SPRE image of MVA SAS that is installed in the Viya deployment. This is roughly equivalent to a Workspace server via IOM, but in Viya with no SAS 9.4.

# Use Case Demo

- Let’s walk through a quick demo:

# Thank you

Simon Geletta

515-343-9895

slg@kdsol.com

https://kdsol.com