Everything is Better with Friends

Using SAS in Python Applications with SASPy and Open-Source Tooling (Beyond the Basics)

Setup for Part 3

Getting setup to use Google Colab with SAS OnDemand for Academics (ODA)

1. Please enable line numbers using the Tools menu: **Tools** -> **Settings** -> **Editor** -> **Show line numbers** -> **Save**
2. To execute code examples, you'll need credentials for the following accounts:
   - Google. (If you're not already signed in, you should see a **Sign In** button in the upper right corner. You can also visit https://accounts.google.com/signup to create an account for free.)
   - SAS OnDemand for Academics. (You can create an account for free at https://welcome.oda.sas.com/ using an existing SAS Profile account. If you don't already have a SAS Profile account, you can create one for free by clicking on the link near the bottom of the ODA login page under the heading "Get Started".)
3. To save a copy of this notebook, along with any edits you make, please use the File menu: **File** -> **Save a copy in Drive**
4. We also recommend enabling the Table of Contents using the View menu: **View** -> **Table of contents**
5. Some useful Zoom Reactions:
   - 👍 (Thumbs Up) when you're done with a section
   - ✋ (Raise Hand) when you need tech support
   - ☕ (I'm Away) to let us know you've stepped away
6. Looking for "extra credit"? Please let us know if you spot any typos!

Connect to SAS OnDemand for Academics (ODA) and start a SAS session

Instructions:

1. Determine the Region for your ODA account by logging into https://welcome.oda.sas.com/. You should see a value like `Asia Pacific 1`, `Asia Pacific 2`, `Europe 1`, `United States 1`, or `United States 2` at the top of screen near your username. (For more information about Regions and using Python in Jupyter Notebooks, please see the ODA documentation at https://support.sas.com/ondemand/caq_new.html#region and https://support.sas.com/ondemand/saspy.html.)
2. If your ODA account is associated with a Region other than `United States 2`, comment out Line 40 by adding a number sign (`#`) at the beginning of the line, and then uncomment the list of servers corresponding to your Region.

   Note: As of the time of creation of this Notebook, only the Regions listed below were available. If your SAS ODA account is associated with a Region that's not listed, you will need to manually add the appropriate servers.
3. Click anywhere in the code cell, and run the cell using Shift-Enter.
4. At the prompt `Please enter the OMR user id`, enter either your SAS ODA user ID or the email address associated with your ODA account.
5. At the prompt `Please enter the password for OMR user`, enter the password for your SAS ODA account.

```
 1 !pip install saspy
 2
 3 # import standard library packages
 4 import io
 5 import pathlib
 6 import zipfile
 7
 8 # import third-party libraries
 9 import requests
10 import saspy
11
12 # because of recent changes to SAS ODA, we may need to install some files in our Colab session
13 zip_file_url = 'https://drive.google.com/uc?id=1vQ6oVgky8UcLAvhct7CL8Oc5I9Mctiw5&export=download'
```

```
expected_zip_file_contents = {'sas.rutil.jar', 'sas.rutil.nls.jar', 'sastpj.rutil.jar'}
jar_file_installation_path = '/usr/local/lib/python3.8/dist-packages/saspy/java/iomclient/'

# check the JAVA config files currently available in the SASPy installation of our Colab session
current_saspy_jar_files = {
    file.name
    for file
    in pathlib.Path(jar_file_installation_path).glob('*.jar')
}

# if any of three specific .jar files aren't found, download and install them in our Colab session
if not expected_zip_file_contents.issubset(current_saspy_jar_files):
  zip_file_url_response = requests.get(zip_file_url)
  zip_file_contents = zipfile.ZipFile(io.BytesIO(zip_file_url_response.content))
  zip_file_contents.extractall('/usr/local/lib/python3.8/dist-packages/saspy/java/iomclient/')

# with the preliminaries out of the way, we can now establish a connection from Colab to SAS ODA
sas = saspy.SASsession(
    java='/usr/bin/java',
    iomport=8591,
    encoding='utf-8',

    # For Region "United States 1", uncomment the line below.
    #iomhost = ['odaws01-usw2.oda.sas.com','odaws02-usw2.oda.sas.com','odaws03-usw2.oda.sas.com','odaws04-usw2.oda.sas.com'],

    # For Region "United States 2", uncomment the line below.
    iomhost = ['odaws01-usw2-2.oda.sas.com','odaws02-usw2-2.oda.sas.com'],

    # For Region "Europe 1", uncomment the line below.
    #iomhost = ['odaws01-euw1.oda.sas.com','odaws02-euw1.oda.sas.com'],

    # For Region "Asia Pacific 1", uncomment the line below.
    #iomhost = ['odaws01-apse1.oda.sas.com','odaws02-apse1.oda.sas.com'],

    # For Region "Asia Pacific 2", uncomment the line below.
    #iomhost = ['odaws01-apse1-2.oda.sas.com','odaws02-apse1-2.oda.sas.com'],

)
print(sas)
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting saspy
  Downloading saspy-4.7.0.tar.gz (9.9 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9.9/9.9 MB 29.6 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: saspy
  Building wheel for saspy (setup.py) ... done
  Created wheel for saspy: filename=saspy-4.7.0-py3-none-any.whl size=9938036 sha256=0d03a5523d3fef550e9cbdcff98ebd4acb44ddda
  Stored in directory: /root/.cache/pip/wheels/c1/ab/9f/ffe82d792f6a8314e07be05a27f8fa0b8459e0110a682cf8c6
Successfully built saspy
Installing collected packages: saspy
Successfully installed saspy-4.7.0
Using SAS Config named: default
Please enter the OMR user id: isaiah.p.lankham@kpchr.org
Please enter the password for OMR user : ··········
SAS Connection established. Subprocess id is 304

Access Method         = IOM
SAS Config name       = default
SAS Config file       = /usr/local/lib/python3.8/dist-packages/saspy/sascfg.py
WORK Path             = /saswork/SAS_work2F9B0000BCDF_odaws01-usw2-2.oda.sas.com/SAS_work39350000BCDF_odaws01-usw2-2.oda.sas.
SAS Version           = 9.04.01M7P08062020
SASPy Version         = 4.7.0
Teach me SAS          = False
Batch                 = False
Results               = Pandas
SAS Session Encoding  = utf-8
Python Encoding value = utf-8
SAS process Pid value = 48351
```

Notes:

- This installs the SASPy package and establishes a connection from Python in Google Colab to a SAS session running in SAS ODA.

- If an error is displayed, an incompatible kernel has been chosen. This Notebook was developed using the Python 3.8 kernel provided in Google Colab as of February 2023.
- ODA was recently upgraded to SAS version 9.40M7, which is why SASPy needs the contents of a .zip file to be downloaded and installed inside Colab before SASPy can connect to SAS ODA. If Python and SAS were installed on the same machine, the contents of the .zip file would already be available as part of the SAS installation, per https://sassoftware.github.io/saspy/configuration.html
- If your SAS session times out or terminates (e.g., by closing this notebook or using the `sas.endsas()` command), you'll need to run this cell again and re-enter your ODA login credentials.

Install and import additional packages

```
# Install the faker module for generating fake data
!pip install faker

# Install the rich module for colorful printing, limiting the version for compatibility with Colab
!pip install 'rich<13.3'

# Initialize a Faker session called fake
from faker import Faker
fake = Faker()

# We'll use IPython to display DataFrames or HTML content
from IPython.display import display, HTML

# We'll use the pandas package to create and manipulate DataFrame objects
import pandas

# We're overwriting the default print function with rich.print
from rich import print
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting faker
  Downloading Faker-17.3.0-py3-none-any.whl (1.7 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.7/1.7 MB 22.6 MB/s eta 0:00:00
Requirement already satisfied: python-dateutil>=2.4 in /usr/local/lib/python3.8/dist-packages (from faker) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.8/dist-packages (from python-dateutil>=2.4->faker) (1.15.0
Installing collected packages: faker
Successfully installed faker-17.3.0
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting rich<13.3
  Downloading rich-13.2.0-py3-none-any.whl (238 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 238.9/238.9 KB 6.4 MB/s eta 0:00:00
Collecting markdown-it-py<3.0.0,>=2.1.0
  Downloading markdown_it_py-2.2.0-py3-none-any.whl (84 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 84.5/84.5 KB 12.3 MB/s eta 0:00:00
Requirement already satisfied: pygments<3.0.0,>=2.6.0 in /usr/local/lib/python3.8/dist-packages (from rich<13.3) (2.6.1)
Requirement already satisfied: typing-extensions<5.0,>=4.0.0 in /usr/local/lib/python3.8/dist-packages (from rich<13.3) (4.5
Collecting mdurl~=0.1
  Downloading mdurl-0.1.2-py3-none-any.whl (10.0 kB)
Installing collected packages: mdurl, markdown-it-py, rich
Successfully installed markdown-it-py-2.2.0 mdurl-0.1.2 rich-13.2.0
```

Part 3. Merging and appending datasets in SAS and Python

Section 3.1. Create class_df

```
# Let's start by importing and displaying the dataset sashelp.class from SAS ODA.

# Use the sas.sasdata2dataframe method to copy the contents of sashelp.class into DataFrame
# class_df, and use dataset options to get only the first few rows and specific columns.
class_df = sas.sasdata2dataframe(
    table='class',
    libref='sashelp',
    dsopts={
        'keep': ['Name','Age','Height'],
        'obs': 7
    },
)
```

```

# Display the resulting DataFrame.
display(class_df)
```

	Name	Age	Height
0	Alfred	14.0	69.0
1	Alice	13.0	56.5
2	Barbara	13.0	65.3
3	Carol	14.0	62.8
4	Henry	14.0	63.5
5	James	12.0	57.3
6	Jane	12.0	59.8

Concept Check 3.1

- Short Answer: What are some other dataset options we might consider including?
- Fun Fact: The DataFrame `class_df` is a data structure kept in memory in our local Google Colab session, having copied the contents of the dataset `sashelp.class` over the wire.

Solution: An overview of the dataset options available in SASPy can be found at https://sassoftware.github.io/saspy/api.html#saspy.SASsession.sasdata

Section 3.2. Create additional_students_df

```python
# Now imagine we want to create additional example student records to append to sashelp.class, but
# we don't feel like being creative.

# Instead, let's use a "standard recipe" to have Python make a DataFrame with fake values for us!

# Set the number of rows of mock data to generate.
number_of_rows = 5

# Define a function that returns a fake first name that's not already in the Name column of the
# DataFrame class_df defined above.
def create_distinct_first_name():
    random_first_name = fake.unique.first_name()
    while random_first_name in class_df['Name'].unique():
        random_first_name = fake.unique.first_name()
    return random_first_name

# Generate and display a DataFrame called additional_students_df with the specified number_of_rows
# Note: This example uses more advanced Python features, including list comprehensions. There's no
# need to focus on anything other than the overall intent of creating a DataFrame with mock data.
additional_students_df = pandas.DataFrame(
  [
    {
        'Name': create_distinct_first_name(),
        'Age': fake.pyint(min_value=10,max_value=19),
        'Height': fake.pyfloat(right_digits=1,min_value=50,max_value=75),
    }
    for _
    in range(number_of_rows)
  ]
)

display(additional_students_df)
```

	Name	Age	Height
0	Katrina	15	67.2
1	Kevin	14	50.6
2	Daniel	13	65.0
3	Ashley	13	58.7
4	Bill	13	64.8

Concept Check 3.2

- Try this, and see what happens: Change the underscore (_) on Line 27 to any valid Python variable name, and then rerun the code cell above.
- True or False: The code runs just as well, whether an underscore (_) or an actual variable name is used.
- Fun Facts:
    - It's standard convention in the Python community to use an underscore (_) for a variable whose actual value isn't important.
    - You might remember the "Fun Fact" from Part 2 about DataFrame operations being slow when embedded in a for-loop. Here, we've effectively turned this on its head by using a list comprehension to embed a for-loop inside of a DataFrame operation, which is a fairly standard (and highly efficient) Python practice.

Solution: True! The single underscore (_) is commonly used as a general-purpose "throwaway" variable in Python.

Section 3.3. Create appended_df

```
1 # Now that we have two DataFrames with identical columns, we can vertically combine (aka append or
2 # union) them to create a new, taller dataset.
3
4 # Starting with class_df, append each row from additional_students_df, and display the result.
5 appended_df = class_df.append(additional_students_df, ignore_index=True)
6 display(appended_df)
```

	Name	Age	Height
0	Alfred	14.0	69.0
1	Alice	13.0	56.5
2	Barbara	13.0	65.3
3	Carol	14.0	62.8
4	Henry	14.0	63.5
5	James	12.0	57.3
6	Jane	12.0	59.8
7	Stephanie	17.0	54.5
8	Steven	11.0	59.8
9	Corey	16.0	51.4
10	Vanessa	19.0	56.8
11	Christopher	17.0	60.7

Concept Check 3.3

- Try this, and see what happens: Delete the option `ignore_index=True` on Line 5.
- True or False: The option `ignore_index=True` on Line 5 has no affect on the contents of the resulting DataFrame.
- Fun Fact: When a pandas DataFrame is created, index values default to row numbers. However, rows tend to keep their index value, even when row order is changed. To see an example of this, try the following: `appended_df.sort_values('Name')`

Solution: False! Deleting `ignore_index=True` would cause each row to retain its original index.

Section 3.4. Create appended_sds

```
1 # TANGENT/ASIDE: We could have instead copied the DataFrame additional_students_df to SAS ODA and
2 # used PROC SQL to perform the union. Let's see how these two methods compare!
3
4 # Make a SAS dataset from additional_students_df.
5 sas.dataframe2sasdata(
6     additional_students_df,
7     table="additional_students_sds",
8     libref="Work"
)

# Use the sas.submit method to submit a PROC SQL step directly to SAS ODA
sas_submit_return_value = sas.submit(
    '''
        proc sql;
            create table appended_sds as
                select Name, Age, Height from sashelp.class(obs=7)
                union all corr
                select * from additional_students_sds
            ;
        quit;
        proc print data=appended_sds;
        run;
    '''
)

# But what type of object exactly do we get back from sas.submit?
print(f'The type of sas_submit_return_value: {type(sas_submit_return_value)}')

# And what are the keys in this dict?
print(f'\n\nThe keys in sas_submit_return_value: {list(sas_submit_return_value.keys())}')

# Output the SAS log, which corresponds to the key 'LOG' in the dict returned by sas.submit.
print('\n\n\nThe LOG component of sas_submit_return_value:')
sas_submit_log = sas_submit_return_value['LOG']
print(sas_submit_log)

# Render and display the SAS HTML results, which corresponds to the key 'LST' in the dict returned
# by sas.submit.
print('\n\n\nThe LST (aka results) component of sas_submit_return_value:')
sas_submit_results = sas_submit_return_value['LST']
display(HTML(sas_submit_results))
```

```
The type of sas_submit_return_value: <class 'dict'>

The keys in sas_submit_return_value: ['LOG', 'LST']


The LOG component of sas_submit_return_value:
24                                                         The SAS System                     Tuesday, February 28,
2023 02:43:00 PM

166        ods listing close;ods html5 (id=saspy_internal) file=_tomods1 options(bitmap_mode='inline') device=svg
```

Concept Check 3.4

- Short Answer: List some others ways to vertically combine datasets in SAS.
- Fun Facts:
  - Because the SAS dataset `Work.appended_sds` is created inside of the `submit` method, our Python session has no access to it. To use the contents of `Work.appended_sds` in our Python session, create a `SASdata` object as follows:

    `sas.sasdata(table='appended_sds')`
  - The `submit` method always returns a dictionary with the same two keys, `LOG` and `LST`.

179

Solution: Options include PROC APPEND and a SET statement in a DATA step.

182

Section 3.5. Create age_ranges_df

```
# Now suppose we want to add a column describing the age range for each student in our full
# DataFrame.

# We'll start by building a lookup table:

# As a first step, let's make a DataFrame with a single column called Age, using the built-in range
# function to populate this column with the integer values 10 through 19. (Note: The range function
# always stops at one less than the specified upper bound.)
age_ranges_df = pandas.DataFrame(
    {
        'Age': range(10,20)
    }
)

# Now add a column to age_ranges_df by "binning" integers in age_ranges_df['Age'] into the
# categories 10-12 (tween) and 13-19 (teen).

# In other words, use the pandas.cut method with these arguments:
# * bins=[10,12,19], which creates the values ranges [10,12] and (12,19]
# * labels=['tween','teen'], which specifies the label for each range, in order
age_ranges_df['Age_Range'] = pandas.cut(
    age_ranges_df['Age'],
    bins=[10,12,19],
    labels=['tween','teen'],
    include_lowest=True,
)
display(age_ranges_df)
```

	Age	Age_Range
0	10	tween
1	11	tween
2	12	tween
3	13	teen
4	14	teen
5	15	teen
6	16	teen
7	17	teen
8	18	teen
9	19	teen

Concept Check 3.5

- Try this, and see what happens: Comment out the option `include_lowest=True` on Line 25, and then rerun the code cell above.
- True or False: Commenting out the option `include_lowest=True` on Line 25 won't affect the contents of the resulting DataFrame.
- Fun Fact: The `pandas.cut` method could have instead been used directly on DataFrame `appended_df` to bin values of `Age`.

Solution: False! Deleting `include_lowest=True` will cause the value 10 to not have a corresponding label.

Section 3.6. Create merged_df

```
 1 # Given the lookup table age_ranges_df, which has some (but not all) of its columns in common with
 2 # appended_df, we can horizontally combine (aka merge or join) them to create a new, wider dataset.
 3
 4 # Specifically, starting with appended_df, we'll add an Age_Range column whose values are determined
 5 # by matching values of Age in age_ranges_df as part of a left join, resulting in a DataFrame having
 6 # the same height as appended_df.
 7 merged_df = appended_df.merge(
 8     age_ranges_df,
 9     on='Age',
10     how='left',
11 )
12
13 # Since we're not using the Height column, drop it from merged_df, and display the result.
14 merged_df.drop(columns=['Height'], inplace=True)
15 display(merged_df)
```

	Name	Age	Age_Range
0	Alfred	14.0	teen
1	Alice	13.0	teen
2	Barbara	13.0	teen
3	Carol	14.0	teen
4	Henry	14.0	teen
5	James	12.0	tween
6	Jane	12.0	tween
7	Stephanie	17.0	teen
8	Steven	11.0	tween
9	Corey	16.0	teen
10	Vanessa	19.0	teen
11	Christopher	17.0	teen

Concept Check 3.6

- Try this, and see what happens: Delete the option `inplace=True` on Line 14, and then rerun the code cell above.
- True or False: Deleting the option `inplace=True` on Line 14 won't affect the contents of the resulting DataFrame.
- Fun Fact: Just like many SQL dialects, `pandas.merge` supports left, right, (full) outer, inner, and cross joins.

Solution: False! The column will not be dropped successfully unless you use `inplace=True`, or unless you assign the result `merged_df.drop(columns=['Height']` to a variable.

Section 3.7. Create merged_sds

```
1 # TANGENT/ASIDE: Since the DataFrame additional_students_df was already copied over to SAS ODA
2 # above, we could have instead copied over age_ranges_df and used PROC SQL to perform the left join.
3 # Let's see how these two methods compare!
4
5 # Make a SAS dataset from age_ranges_df.
6 sas.dataframe2sasdata(
7     age_ranges_df,
8     table="age_ranges_sds",
9     libref="Work"
)

# Use the sas.submit method to submit a PROC SQL step directly to ODA, and capture the resulting
# dict in sas_submit_return_value.
sas_submit_return_value = sas.submit(
    '''
        proc sql;
            create table merged_sds as
                select
                     A.Name
                    ,A.Age
                    ,B.Age_Range
                from
                    appended_sds as A
                    left join
                    age_ranges_sds as B
                    on A.Age = B.Age
            ;
        quit;
        proc print data=merged_sds;
        run;
    '''
)

# Output the SAS log, which corresponds to the key 'LOG' in the dict returned by sas.submit.
sas_submit_log = sas_submit_return_value['LOG']
print(sas_submit_log)

# Render and display the SAS results HTML, which corresponds to the key 'LST' in the dict returned
# by sas.submit.
sas_submit_results = sas_submit_return_value['LST']
display(HTML(sas_submit_results))
```

```
36                                                          The SAS System                  Tuesday, February 28,
2023 02:43:00 PM

275        ods listing close;ods html5 (id=saspy_internal) file=_tomods1 options(bitmap_mode='inline') device=svg
style=HTMLBlue;
275      ! ods graphics on / outputfmt=png;
```

Concept Check 3.7

- Short Answer: List some others ways to get the same result in SAS without a join.
- Fun Fact: Just like in Section 3.4, because the SAS dataset `Work.merged_sds` is created inside of the `submit` method, our Python session has no access to it unless we create a `SASdata` object as follows: `sas.sasdata(table='merged_sds')`

```
284                                from
```

Solution: Options include a MERGE or UPDATE statements in a DATA step, or creating age ranges using PROC FORMAT.

```
288                                    on A.Age = B.Age
```

Section 3.8. Additional Exercises

```
292                  run;
```

For practice, we recommend the following:

1. Run the code cell below to convert the `Height` column of `class_df` to integer values and create a reference table called `height_ranges_df`.
2. Then add new code, repeating the steps in Sections 3.6 to merge `height_ranges_df` with `class_df` on `Height`.

For additional practice, you might also try applying `pandas.cut` directly to `class_df`.

```python
# Make sure the values of height are formatted as integers.
class_df['Height'] = class_df['Height'].apply(int)
display(class_df)

# Create a range of possible heights (in inches, per the contents of sashelp.class).
height_ranges_df = pandas.DataFrame(
    {
        'Height': range(50,75)
    }
)

# Bin the possible values of height in the reference dataset.
height_ranges_df['Height_Range'] = pandas.cut(
    height_ranges_df['Height'],
    bins=[50,62,75],
    labels=['short','tall'],
    include_lowest=True,
)
display(height_ranges_df)
```

	Name	Age	Height
0	Alfred	14.0	69
1	Alice	13.0	56
2	Barbara	13.0	65
3	Carol	14.0	62
4	Henry	14.0	63
5	James	12.0	57
6	Jane	12.0	59

	Height	Height_Range
0	50	short
1	51	short
2	52	short
3	53	short
4	54	short
5	55	short
6	56	short
7	57	short
8	58	short

```python
# Merge in the binned values of height from the reference dataset.
merged_height_df = class_df.merge(
    height_ranges_df,
    on='Height',
    how='left',
)

# Since we're not using the Age column, drop it from merged_df, and display the result.
merged_height_df.drop(columns=['Age'], inplace=True)
display(merged_height_df)

# Or just apply pandas.cut directly to class_df (without dropping the Age column).
class_df['Height_Range'] = pandas.cut(
    class_df['Height'],
    bins=[50,62,75],
    labels=['short','tall'],
    include_lowest=True,
)
display(class_df)
```

Notes and Resources

1 Alice 56 short

Want some ideas for what to do next? Here are our suggestions:

1. For more about the `faker` package, which can generate many other types of fake data, see https://faker.readthedocs.io/
2. For more about the `pandas` package, including the methods used above, see the following:
    - https://pandas.pydata.org/docs/reference/api/pandas.cut.html
    - https://pandas.pydata.org/docs/reference/api/pandas.unique.html
    - https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.append.html
    - https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.drop.html
    - https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.merge.html
3. For more about the `rich` package, see https://rich.readthedocs.io/
4. For more about the `saspy` package, including the methods used above, see the following:
    - https://sassoftware.github.io/saspy/api.html#saspy.SASsession.dataframe2sasdata
    - https://sassoftware.github.io/saspy/api.html#saspy.SASsession.sasdata2dataframe
    - https://sassoftware.github.io/saspy/api.html#saspy.SASsession.submit
5. For more about some of the Python features used, such as functions and list comprehensions, we recommend the following chapters of A Whirlwind Tour of Python:
    - https://jakevdp.github.io/WhirlwindTourOfPython/08-defining-functions.html
    - https://jakevdp.github.io/WhirlwindTourOfPython/11-list-comprehensions.html
6. We welcome follow-up conversations. You can connect with us on LinkedIn or email us at isaiah.lankham@gmail.com and matthew.t.slaughter@gmail.com
7. If you have a GitHub account (or don't mind creating one), you can also chat with us on Gitter at https://gitter.im/saspy-bffs/community