# 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. To execute code cells, 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 using the "Don't have a SAS Profile?" link on the ODA login page.)
2. We recommend enabling line numbers using the Tools menu: **Tools** -> **Settings** -> **Editor** -> **Show line numbers** -> **Save**
3. We also recommend enabling the Table of Contents using the View menu: **View** -> **Table of contents**
4. To save a copy of this notebook, along with any edits you make, please use the File menu: **File** -> **Save a copy in Drive**
5. 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 the value `Asia Pacific`, `Europe`, or `United States` next to your username in the upper-right corner. (For more information about Regions, please see the ODA documentation.)
2. If your ODA account is associated with a Region other than `United States`, comment out Line 11 by adding a number sign (`#`) at the beginning of the line, and then do the following:
    - If your ODA account is associated with the Region `Europe`, uncomment Line 14 by removing the number sign (`#`) at the beginning of the line.
    - If your ODA account is associated with the Region `Asia Pacific`, uncomment Line 17 by removing the number sign (`#`) at the beginning of the line.
3. Click anywhere in the code cell, and run the cell using Shift-Enter.
4. At the prompt `Please enter the IOM 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 IOM user`, enter the password for your SAS ODA account.

```
 1 !pip install saspy
 2
 3 import saspy
 4
 5 sas = saspy.SASsession(
 6     java='/usr/bin/java',
 7     iomport=8591,
 8     encoding='utf-8',
 9
10     # The following line should be uncommented if, and only if, your ODA account is associated with the Regio
11     iomhost = ['odaws01-usw2.oda.sas.com', 'odaws02-usw2.oda.sas.com', 'odaws03-usw2.oda.sas.com', 'odaws04-u
12
13     # The following line should be uncommented if, and only if, your ODA account is associated with the Regio
14     #iomhost = ['odaws01-euw1.oda.sas.com','odaws02-euw1.oda.sas.com'],
15
16     # The following line should be uncommented if, and only if, your ODA account is associated with the Regio
17     #iomhost = ['odaws01-apse1.oda.sas.com','odaws02-apse1.oda.sas.com'],
18
)
print(sas)
```

```
Collecting saspy
  Downloading saspy-4.3.0.tar.gz (9.9 MB)
     |████████████████████████████████| 9.9 MB 4.1 MB/s
Building wheels for collected packages: saspy
  Building wheel for saspy (setup.py) ... done
  Created wheel for saspy: filename=saspy-4.3.0-py3-none-any.whl size=9929656 sha256=750ea1b7d6300cd0b7aeab5
  Stored in directory: /root/.cache/pip/wheels/c3/b5/08/62c85da319a5178d19559f996ceefd7583b9bf31feeafbad8e
Successfully built saspy
Installing collected packages: saspy
Successfully installed saspy-4.3.0
Using SAS Config named: default
Please enter the IOM user id: isaiah.lankham@ucop.edu
Please enter the password for IOM user : ··········
SAS Connection established. Subprocess id is 137

Access Method         = IOM
SAS Config name       = default
SAS Config file       = /usr/local/lib/python3.7/dist-packages/saspy/sascfg.py
WORK Path             = /saswork/SAS_workFEDF00008895_odaws04-usw2.oda.sas.com/SAS_work416700008895_odaws04-
SAS Version           = 9.04.01M6P11072018
SASPy Version         = 4.3.0
Teach me SAS          = False
Batch                 = False
Results               = Pandas
SAS Session Encoding  = utf-8
Python Encoding value = utf-8
SAS process Pid value = 34965
```

**Note**: This establishes a connection from Python in Google Colab to a SAS session running in SAS ODA.

## Install and import additional packages

```
# Install the faker module for generating fake data
```

```python
!pip install faker

# Install the rich module for colorful printing
!pip install rich

# 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
```

```
Collecting faker
  Downloading Faker-13.11.1-py3-none-any.whl (1.5 MB)
     |████████████████████████████████| 1.5 MB 5.1 MB/s
Requirement already satisfied: typing-extensions>=3.10.0.2 in /usr/local/lib/python3.7/dist-packages (from fa
Requirement already satisfied: python-dateutil>=2.4 in /usr/local/lib/python3.7/dist-packages (from faker) (2
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.4-
Installing collected packages: faker
Successfully installed faker-13.11.1
Collecting rich
  Downloading rich-12.4.1-py3-none-any.whl (231 kB)
     |████████████████████████████████| 231 kB 3.0 MB/s
Requirement already satisfied: typing-extensions<5.0,>=4.0.0 in /usr/local/lib/python3.7/dist-packages (from
Collecting commonmark<0.10.0,>=0.9.0
  Downloading commonmark-0.9.1-py2.py3-none-any.whl (51 kB)
     |████████████████████████████████| 51 kB 6.1 MB/s
Requirement already satisfied: pygments<3.0.0,>=2.6.0 in /usr/local/lib/python3.7/dist-packages (from rich)
Installing collected packages: commonmark, rich
Successfully installed commonmark-0.9.1 rich-12.4.1
```

## 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?

- 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),
26     }
27     for _
28     in range(number_of_rows)
29   ]
30 )
31
32 display(additional_students_df)
```

```
      Name  Age  Height
0   Stacey   12    64.0
1     Lisa   14    66.5
2   Sophia   14    51.1
3    Tanya   12    58.5
4  Michael   17    51.0
```

**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: It's standard convention in the Python community to use an underscore (_) for a variable whose actual value isn't important.
- Fun Fact: 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    Stacey  12.0    64.0
8      Lisa  14.0    66.5
9    Sophia  14.0    51.1
10    Tanya  12.0    58.5
11  Michael  17.0    51.0
```

### 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

```
# TANGENT/ASIDE: We could have instead copied the DataFrame additional_students_df to SAS ODA and
# used PROC SQL to perform the union. Let's see how these two methods compare!

# Make a SAS dataset from additional_students_df.
sas.dataframe2sasdata(
    additional_students_df,
    table="additional_students_sds",
    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 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;
    '''
)

# 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 HTML results, 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))
```

```
24                                                     The SAS System
Saturday, May 21, 2022 05:03:00 AM

152          ods listing close;ods html5 (id=saspy_internal) file=_tomods1
options(bitmap_mode='inline') device=svg style=HTMLBlue;
152        ! ods graphics on / outputfmt=png;
153
154
155                    proc sql;
```

**Concept Check 3.4**

- Short Answer: List some others ways to vertically combine datasets in SAS.
- Fun Fact: Because the SAS dataset `Work.appended_sds` is created inside of the `submit` method, our Python session has no access to it. If we wanted to use the contents of `Work.appended_sds` in our Python session, we could create a `SASdata` object as follows: `sas.sasdata(table='appended_sds')`

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

```
25                                                     The SAS System
```

## Section 3.5. Create age_ranges_df

**The SAS System**

```
# 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)
    }
)

```

```
15 # Now add a column to age_ranges_df by "binning" integers in age_ranges_df['Age'] into the
16 # categories 10-12 (tween) and 13-19 (teen).
17
18 # In other words, use the pandas.cut method with these arguments:
19 # * bins=[10,12,19], which creates the values ranges [10,12] and (12,19]
20 # * labels=['tween','teen'], which specifies the label for each range, in order
21 age_ranges_df['Age_Range'] = pandas.cut(
22     age_ranges_df['Age'],
23     bins=[10,12,19],
24     labels=['tween','teen'],
25     include_lowest=True,
26 )
27 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

```
# Given the lookup table age_ranges_df, which has some (but not all) of its columns in common with
# appended_df, we can horizontally combine (aka merge or join) them to create a new, wider dateset.

# Specifically, starting with appended_df, we'll add an Age_Range column whose values are determined
# by matching values of Age in age_ranges_df as part of a left join, resulting in a DataFrame having
# the same height as appended_df.
merged_df = appended_df.merge(
    age_ranges_df,
    on='Age',
    how='left',
)

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

**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 dialetics, `pandas.merge` supports left, right, outer, inner, and cross joins.

11 Michael 17.0 teen

***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

```
# TANGENT/ASIDE: Since the DataFrame additional_students_df was already copied over to SAS ODA
# above, we could have instead copied over age_ranges_df and used PROC SQL to perform the left join.
# Let's see how these two methods compare!

# Make a SAS dataset from age_ranges_df.
sas.dataframe2sasdata(
    age_ranges_df,
    table="age_ranges_sds",
    libref="Work"
```

```python
)

# 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
Saturday, May 21, 2022 05:03:00 AM

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


                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;



        ods html5 (id=saspy_internal) close;ods listing;


37                                                          The SAS System
Saturday, May 21, 2022 05:03:00 AM

```

**The SAS System**

| Obs | Name | Age | Age_Range |
|---|---|---|---|
| 1 | James | 12 | tween |
| 2 | Tanya | 12 | tween |
| 3 | Jane | 12 | tween |
| 4 | Stacey | 12 | tween |
| 5 | Barbara | 13 | teen |
| 6 | Alice | 13 | teen |

|   |   |   |   |
|---|---|---|---|
| **7** | Alfred | 14 | teen |
| **8** | Henry | 14 | teen |
| **9** | Sophia | 14 | teen |

**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')`

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

## Section 3.8. Additional Exercises

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`.

```
# 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)

# 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)
```

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

|   | 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 |
| 9 | 59 | short |
| 10 | 60 | short |
| 11 | 61 | short |
| 12 | 62 | short |
| 13 | 63 | tall |
| 14 | 64 | tall |

| | | |
|---|---|---|
| **15** | 65 | tall |
| **16** | 66 | tall |
| **17** | 67 | tall |
| **18** | 68 | tall |
| **19** | 69 | tall |
| **20** | 70 | tall |
| **21** | 71 | tall |
| **22** | 72 | tall |

## Notes and Resources

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 comphrensions, we recomend 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