Everything is Better with Friends

Using SAS in Python Applications with SASPy and Open-Source Tooling (Getting Started)

Section 0. Setup and Connect to SAS OnDemand for Academics (ODA)

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.)

      Note: Please go to https://gmail.com/ and send an email to isaiah.lankham@gmail.com from your Gmail account with the subject `PharmaSUG 2022 HoT`. Isaiah will add you to a Google Space for tech support during this class.
    - 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 a value like `Asia Pacific 1`, `Asia Pacific 2`, `Europe 1`, `United States 1`, or `United States 2` next to your username in the upper-right corner. (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 1`, comment out Line 11 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 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.

```
!pip install saspy

import saspy

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.

    # 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)
```

```
Collecting saspy
  Downloading saspy-4.3.0.tar.gz (9.9 MB)
     |████████████████████████████████| 9.9 MB 6.3 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=561db35c887870d527b
  Stored in directory: /root/.cache/pip/wheels/c3/b5/08/62c85da319a5178d19559f996ceefd7583b9bf31feeafbad
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 134

Access Method         = IOM
SAS Config name       = default
SAS Config file       = /usr/local/lib/python3.7/dist-packages/saspy/sascfg.py
WORK Path             = /saswork/SAS_work75A600014928_odaws02-usw2.oda.sas.com/SAS_work735000014928_odaws
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 = 84264
```

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 rich module for colorful printing
!pip install rich

# 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'll use the platform platform to get information about our Python environment.
import platform

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

# We're also setting the maximum line width of rich.print to be a bit wider (to avoid line wrapping)
from rich import get_console
console = get_console()
console.width = 165
```

```
    Collecting rich
      Downloading rich-12.4.1-py3-none-any.whl (231 kB)
        |████████████████████████████████| 231 kB 5.1 MB/s
    Requirement already satisfied: typing-extensions<5.0,>=4.0.0 in /usr/local/lib/python3.7/dist-packages (f
    Requirement already satisfied: pygments<3.0.0,>=2.6.0 in /usr/local/lib/python3.7/dist-packages (from ric
    Collecting commonmark<0.10.0,>=0.9.0
      Downloading commonmark-0.9.1-py2.py3-none-any.whl (51 kB)
        |████████████████████████████████| 51 kB 7.5 MB/s
    Installing collected packages: commonmark, rich
    Successfully installed commonmark-0.9.1 rich-12.4.1
```

Section 1. Setup and Connect to SAS OnDemand for Academics (ODA)

Example 1.1. Meet the Python environment

Type the following into the code cell immediately below, and then run that cell using Shift-Enter:

```
print(platform.dist())
print('\n')
print(platform.sys.version)
print('\n')
print(sorted(list(platform.sys.modules)))
```

```
print(platform.dist())
print('\n')
print(platform.sys.version)
print('\n')
print(sorted(list(platform.sys.modules)))
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: DeprecationWarning: dist() and linux_dist
  """Entry point for launching an IPython kernel.
('Ubuntu', '18.04', 'bionic')

3.7.13 (default, Apr 24 2022, 01:04:09)
[GCC 7.5.0]

[
    'IPython',
    'IPython.core',
    'IPython.core.alias',
    'IPython.core.application',
    'IPython.core.autocall',
    'IPython.core.builtin_trap',
    'IPython.core.compilerop',
    'IPython.core.completer',
    'IPython.core.completerlib',
    'IPython.core.crashhandler',
    'IPython.core.debugger',
    'IPython.core.display',
    'IPython.core.display',
    'IPython.core.display_trap',
    'IPython.core.displayhook',
    'IPython.core.displaypub',
    'IPython.core.error',
    'IPython.core.events',
    'IPython.core.excolors',
    'IPython.core.extensions',
    'IPython.core.formatters',
    'IPython.core.getipython',
    'IPython.core.history',
    'IPython.core.hooks',
    'IPython.core.inputsplitter',
    'IPython.core.inputtransformer',
    'IPython.core.interactiveshell',
    'IPython.core.latex_symbols',
    'IPython.core.logger',
    'IPython.core.macro',
    'IPython.core.magic',
    'IPython.core.magic_arguments',
    'IPython.core.magics',
    'IPython.core.magics.auto',
    'IPython.core.magics.basic',
    'IPython.core.magics.code',
    'IPython.core.magics.config',
    'IPython.core.magics.display',
    'IPython.core.magics.execution',
    'IPython.core.magics.extension',
    'IPython.core.magics.history',
    'IPython.core.magics.logging',
    'IPython.core.magics.namespace',
    'IPython.core.magics.osm',
    'IPython.core.magics.pylab',
    'IPython.core.magics.script',
    'IPython.core.oinspect',
    'IPython.core.page',
    'IPython.core.payload',
    'IPython.core.payloadpage',
    'IPython.core.prefilter',
    'IPython.core.profiledir',
    'IPython.core.pylabtools',
    'IPython.core.release',
    'IPython.core.shadowns',
    'IPython.core.shellapp',
```

Notes about Example 1.1.

1. Assuming a Python 3 kernel is associated with this Notebook, the following should be printed, separated by blank lines:

    - operating-system information
    - the Python version
    - a sorted list of python modules currently installed

2. This example illustrates three ways Python syntax differs from SAS:

    - We don't need semicolons at the end of each statement.
    - The code `PLATFORM.DIST()` would produce an error because capitalization matters.
    - The code `platform.sys.version` uses object-oriented dot-notation to have the `platform` object module invoke the sub-module object `sys` nested inside of it, and then have `sys` invoke the object `version` nested inside of it. (Think Russian nesting dolls or turduckens.)

3. Python comes with a large standard library because of its "batteries included" philosophy, and numerous third-party modules are also actively developed and made freely available through sites like https://github.com/ and https://pypi.org/. For the examples in this notebook, we'll need these third-party modules:

    - `IPython`, which stands for "Interactive Python." Google Colab is built on top of JupyterLab, and JupyterLab is built on top of `IPython`, so `IPython` is already available in Google Colab.

    - `pandas`, which provided `DataFrame` objects. DataFrames can be found in other languages, like R, and are similar to SAS datasets. Because `pandas` is a fundamental package for working with data in Python, it's already available in Google Colab.

    - `saspy`, which is a Python package developed by the SAS Institute for connecting to a SAS kernel. Because `saspy` doesn't come pre-installed in Google Colab sessions, we had to manually install it in Section 0 above.

4. To increase performance, only a small number of modules in Python's standard library are available by default, which is why we needed to explicitly load the modules we'll be using in Section 0 above.

5. For extra credit, try the following:

- Run the Python code `help(saspy)` to get the built-in help for the package `saspy` we'll be using in Sections 2-4 below.

```
     'IPython.utils.module_paths',
     'IPython.utils.openpy'
```

```
help(saspy)
```

```
Help on package saspy:

NAME
    saspy

DESCRIPTION
    # Copyright SAS Institute
    #
    #  Licensed under the Apache License, Version 2.0 (the License);
    #  you may not use this file except in compliance with the License.
    #  You may obtain a copy of the License at
    #
    #      http://www.apache.org/licenses/LICENSE-2.0
    #
    #  Unless required by applicable law or agreed to in writing, software
    #  distributed under the License is distributed on an "AS IS" BASIS,
    #  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
    #  See the License for the specific language governing permissions and
    #  limitations under the License.
    #

PACKAGE CONTENTS
    SASLogLexer
    autocfg
    sasViyaML
    sas_magic
    sasbase
    sascfg
    sasdata
    sasdecorator
    sasets
    sasexceptions
    sasiocom
    sasiohttp
    sasioiom
```

```
        sasioiom
        sasiostdio
        sasml
        sasproccommons
        sasqc
        sasresults
        sasstat
        sastabulate
        sasutil
        version

    FUNCTIONS
        isnotebook()

    DATA
        SAScfg = '/usr/local/lib/python3.7/dist-packages/saspy/sascfg.py'
        absolute_import = _Feature((2, 5, 0, 'alpha', 1), (3, 0, 0, 'alpha', 0...
        division = _Feature((2, 2, 0, 'alpha', 2), (3, 0, 0, 'alpha', 0), 8192...
        logger = <Logger saspy (INFO)>
        print_function = _Feature((2, 6, 0, 'alpha', 2), (3, 0, 0, 'alpha', 0)...

    VERSION
        4.3.0

    FILE
        '_pydev_bundle._pydev_filesystem_encoding',
        '_pydev_bundle._pydev_imports_tipper',
```

Example 1.2. Python lists and indexing

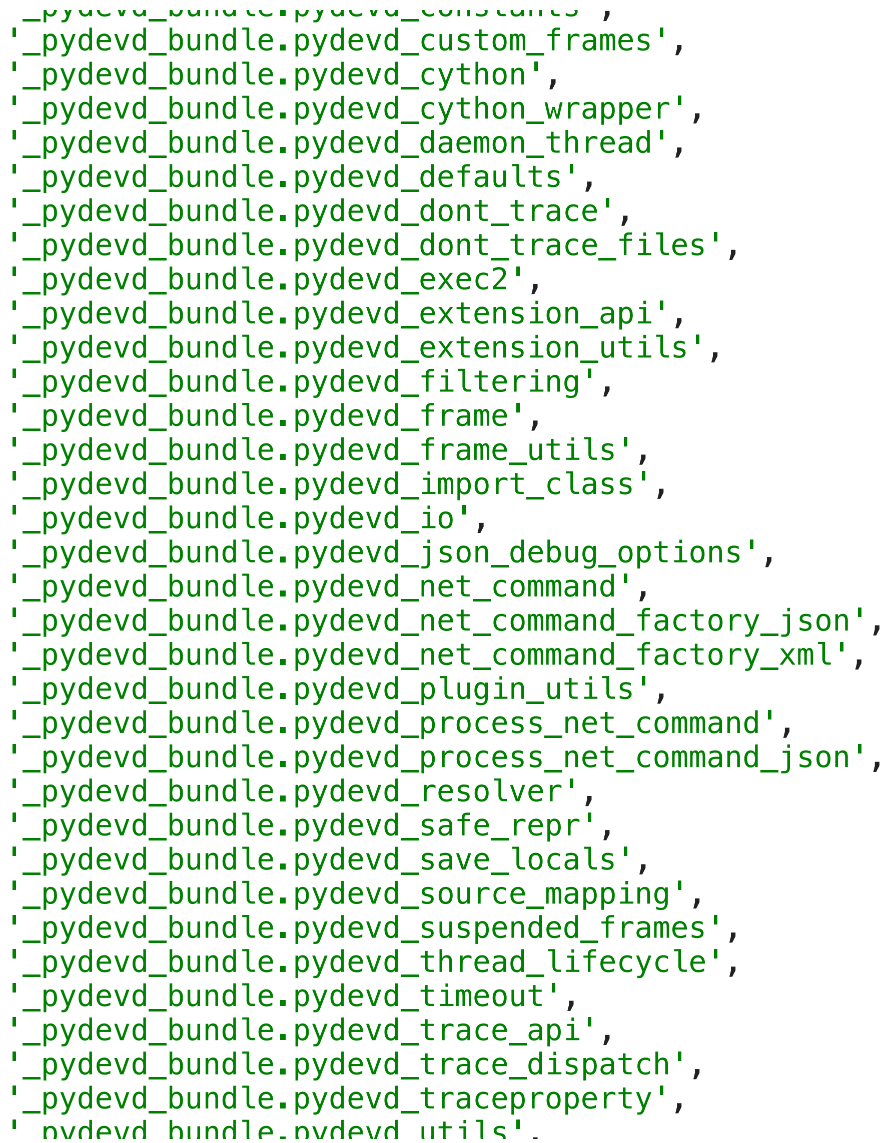
Type the following into the code cell immediately below, and then run that cell using Shift-Enter:

```
hello_world_list = ['Hello', 'list']
print(hello_world_list)
print('\n')
print(type(hello_world_list))
```

```
        '_pydevd_bundle._debug_adapter',
        '_pydevd_bundle._debug_adapter.pydevd_base_schema',
        '_pydevd_bundle._debug_adapter.pydevd_schema',
hello_world_list = ['Hello', 'list']
print(hello_world_list)
print('\n')
print(type(hello_world_list))
```

```
['Hello', 'list']

<class 'list'>
```



```
 '_pydevd_bundle.pydevd_constants',
 '_pydevd_bundle.pydevd_custom_frames',
 '_pydevd_bundle.pydevd_cython',
 '_pydevd_bundle.pydevd_cython_wrapper',
 '_pydevd_bundle.pydevd_daemon_thread',
 '_pydevd_bundle.pydevd_defaults',
 '_pydevd_bundle.pydevd_dont_trace',
 '_pydevd_bundle.pydevd_dont_trace_files',
 '_pydevd_bundle.pydevd_exec2',
 '_pydevd_bundle.pydevd_extension_api',
 '_pydevd_bundle.pydevd_extension_utils',
 '_pydevd_bundle.pydevd_filtering',
 '_pydevd_bundle.pydevd_frame',
 '_pydevd_bundle.pydevd_frame_utils',
 '_pydevd_bundle.pydevd_import_class',
 '_pydevd_bundle.pydevd_io',
 '_pydevd_bundle.pydevd_json_debug_options',
 '_pydevd_bundle.pydevd_net_command',
 '_pydevd_bundle.pydevd_net_command_factory_json',
 '_pydevd_bundle.pydevd_net_command_factory_xml',
 '_pydevd_bundle.pydevd_plugin_utils',
 '_pydevd_bundle.pydevd_process_net_command',
 '_pydevd_bundle.pydevd_process_net_command_json',
 '_pydevd_bundle.pydevd_resolver',
 '_pydevd_bundle.pydevd_safe_repr',
 '_pydevd_bundle.pydevd_save_locals',
 '_pydevd_bundle.pydevd_source_mapping',
 '_pydevd_bundle.pydevd_suspended_frames',
 '_pydevd_bundle.pydevd_thread_lifecycle',
 '_pydevd_bundle.pydevd_timeout',
 '_pydevd_bundle.pydevd_trace_api',
 '_pydevd_bundle.pydevd_trace_dispatch',
 '_pydevd_bundle.pydevd_traceproperty',
 '_pydevd_bundle.pydevd_utils',
```

Notes about Example 1.2.

1. A list object named `hello_world_list` with two string values is created, and the following are printed with a blank line between them:
   - the value of the list
   - its type (which is `<class 'list'>`)
2. Lists are a fundamental Python data structure and are similar to SAS DATA step arrays. Values in lists are always kept in insertion order, meaning the order they appear in the list's definition, and they can be individually accessed using numerical indexes within bracket notation:
   - `hello_world_list[0]` returns `'Hello'`
   - `hello_world_list[1]` returns `'list'`
3. This example illustrates another way Python syntax differs from SAS: The left-most element of a list is always at index `0`. Unlike SAS, customized indexing is only available for more sophisticated Python data structures, like the dictionaries and DataFrames we'll be using in the following examples.
4. For extra credit, try any or all of the following:
   - Print out the initial element of the list.
   - Print out the final element of the list.
   - Create a list of length five, and print its middle elements.

```
 'astor.op_util',
 'astor.source_repr',
 'astor.string_repr',
 'astor.tree_walk',
 'asyncio',
 'asyncio.base_events',
 'asyncio.base_futures',
 'asyncio.base_subprocess',
 'asyncio.base_tasks',
 'asyncio.constants',
 'asyncio.coroutines',
 'asyncio.events',
print(hello_world_list[0])
print('\n')
print(hello_world_list[1])
print('\n')
list_with_five_elements = ['a','b','c','d','e']
print(list_with_five_elements[2])
```

```
Hello

list

c
```

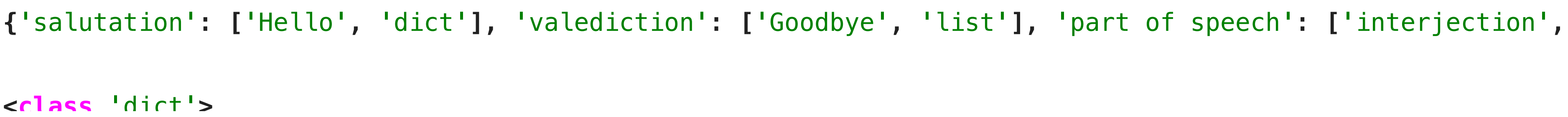
```
        'atexit',
```

Example 1.3 Python dictionaries

Type the following into the code cell immediately below, and then run that cell using Shift-Enter:

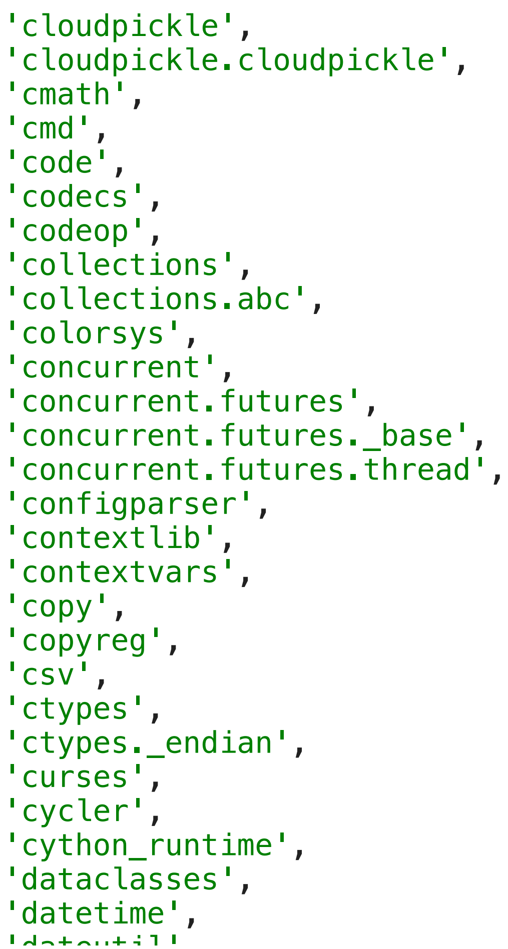
```
hello_world_dict = {
        'salutation'     : ['Hello'       , 'dict'],
        'valediction'    : ['Goodbye'     , 'list'],
        'part of speech' : ['interjection', 'noun'],
}
print(hello_world_dict)
print('\n')
print(type(hello_world_dict))
```

```
        'bottleneck._version',
        'bottleneck.benchmark',
        'bottleneck.benchmark.autotimeit',
        'bottleneck.benchmark.bench',
        'bottleneck.benchmark.bench_detailed',
        'bottleneck.move',
        'bottleneck.nonreduce',
        'bottleneck.nonreduce_axis',
        'bottleneck.reduce',
hello_world_dict = {
        'salutation'     : ['Hello'        , 'dict'],
        'valediction'    : ['Goodbye'      , 'list'],
        'part of speech' : ['interjection', 'noun'],
}
print(hello_world_dict)
print('\n')
print(type(hello_world_dict))
```



```
{'salutation': ['Hello', 'dict'], 'valediction': ['Goodbye', 'list'], 'part of speech': ['interjection',


<class 'dict'>
```



```
'cloudpickle',
'cloudpickle.cloudpickle',
'cmath',
'cmd',
'code',
'codecs',
'codeop',
'collections',
'collections.abc',
'colorsys',
'concurrent',
'concurrent.futures',
'concurrent.futures._base',
'concurrent.futures.thread',
'configparser',
'contextlib',
'contextvars',
'copy',
'copyreg',
'csv',
'ctypes',
'ctypes._endian',
'curses',
'cycler',
'cython_runtime',
'dataclasses',
'datetime',
'dateutil'
```

Notes about Example 1.3.

1. A dictionary (`dict` for short) object named `hello_world_dict` with three key-value pairs is created, and the following are printed with a blank line between them:
    - the value of the dictionary
    - its type (which is `<class 'dict'>`)
2. Dictionaries are another fundamental Python data structure and are related to SAS formats and DATA step hash tables. Dictionaries are more generally called *associative arrays* or *maps* because they map keys (appearing before the colons) to values (appearing after the colons). In other words, the value associated with each key can be accessed using bracket notation:
    - `hello_world_dict['salutation']` returns `['Hello', 'dict']`
    - `hello_world_dict['valediction']` returns `['Goodbye', 'list']`
    - `hello_world_dict['part of speech']` returns `['interjection', 'noun']`
3. Whenever indexable data structures are nested in Python, indexing methods can be combined. Here's an example combining dictionary indexing with list indexing: `hello_world_dict['salutation'][0] == ['Hello', 'dict'][0] == 'Hello'`.
4. For extra credit, try any or all of the following:
    - Print out the list with key `'salutation'`.
    - Print out the initial element in the list associated with key `'valediction'`.
    - Print out the final element in the list associated with key `'part of speech'`.

```
 'email._encoded_words',
 'email._parseaddr',
 'email._policybase',
 'email.base64mime',
 'email.charset',
 'email.encoders',
 'email.errors',
 'email.feedparser',
 'email.header',
print(hello_world_dict['salutation'])
print('\n')
print(hello_world_dict['valediction'][0])
print('\n')
print(hello_world_dict['part of speech'][1])
```

```
['Hello', 'dict']

Goodbye

noun
    'errno'.
```

Example 1.4 Pandas DataFrames

Type the following into the code cell immediately below, and then run that cell using Shift-Enter:

```
hello_world_df = pandas.DataFrame(
    {
        'salutation'     : ['Hello'      , 'DataFrame'],
        'valediction'    : ['Goodbye'    , 'dict'],
        'part of speech' : ['exclamation', 'noun'],
    }
)
display(hello_world_df)
print('\n')
print(hello_world_df.shape)
print('\n')
hello_world_df.info()
```

```
    'google.colab._inspector',
    'google.colab._installation_commands',
    'google.colab._interactive_table_helper',
    'google.colab._interactive_table_hint_button',
    'google.colab._ipython',
hello_world_df = pandas.DataFrame(
    {
        'salutation'     : ['Hello'      , 'DataFrame'],
        'valediction'    : ['Goodbye'    , 'dict'],
        'part of speech' : ['exclamation', 'noun'],
    }
)
display(hello_world_df)
print('\n')
print(hello_world_df.shape)
print('\n')
hello_world_df.info()
```

	salutation	valediction	part of speech
0	Hello	Goodbye	exclamation
1	DataFrame	dict	noun

```
(2, 3)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2 entries, 0 to 1
Data columns (total 3 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   salutation      2 non-null      object
 1   valediction     2 non-null      object
 2   part of speech  2 non-null      object
dtypes: object(3)
memory usage: 176.0+ bytes
```

```
     http.client ,
```

Notes about Example 1.4.

1. A DataFrame (`df` for short) object named `hello_world_df` with 2 rows and 3 columns is created, and the following are printed with blank lines between them:

    - the values in the DataFrame
    - the number of rows and columns in `hello_world_df`
    - some information about the DataFrame, which is obtained by having `hello_world_df` calling its `info` method

2. Since DataFrames aren't built into Python, we had to import the `pandas` module at the start of this notebook. Like their R counterpart, DataFrames are two-dimensional arrays of values comparable to SAS datasets. However, while SAS datasets are typically accessed from disk and processed row-by-row, DataFrames are loaded into memory all at once. This means values in DataFrames can be randomly accessed, but it also means the size of DataFrames can't grow beyond available memory.

3. The dimensions of the DataFrame are determined as follows:

    - The keys `'salutation'`, `'valediction'`, and `'part of speech'` of the dictionary passed to the `DataFrame` constructor function become column labels.
    - Because each key maps to a list of length two, each column will be two elements tall. (Note: An error will occur if the lists are not the same length).

4. This example gives one option for building a DataFrame, but the constructor function accepts many object types, including nested lists or another DataFrame. See https://pandas.pydata.org/docs/

5. For extra credit, try any or all of the following:

    - Print out the column with key `'salutation'`.

    - Print out the initial element in the column with key `'valediction'`.

    - Print out the final element in the column with key `'part of speech'`.

      Hint: DataFrame columns can be indexed just like dictionaries, and their rows can be indexed numerically like lists.

```
'ipywidgets.widgets.domwidget',
'ipywidgets.widgets.interaction',
'ipywidgets.widgets.trait_types',
'ipywidgets.widgets.util',
'ipywidgets.widgets.valuewidget',
'ipywidgets.widgets.widget',
'ipywidgets.widgets.widget_bool'
```

```
print(hello_world_df['salutation'])
print('\n')
print(hello_world_df['valediction'][0])
print('\n')
print(hello_world_df['part of speech'][1])
```

```
0        Hello
1    DataFrame
Name: salutation, dtype: object

Goodbye

noun
```

```
    'ipywidgets.widgets.widget_string'
```

Section 2. SASPy Data Round Trip

```
    'itertools',
    'json',
    'json.decoder',
    'json.encoder',
    'json.scanner',
    'jupyter_client',
    'jupyter_client._version',
    'jupyter_client.adapter',
    'jupyter_client.blocking',
    'jupyter_client.blocking.channels',
    'jupyter_client.blocking.client',
    'jupyter_client.channels',
    'jupyter_client.channelsabc',
    'jupyter_client.client',
    'jupyter_client.clientabc',
    'jupyter_client.connect',
    'jupyter_client.jsonutil',
    'jupyter_client.kernelspec',
    'jupyter_client.launcher',
    'jupyter_client.localinterfaces',
    'jupyter_client.manager',
    'jupyter_client.managerabc',
    'jupyter_client.multikernelmanager',
    'jupyter_client.session',
```

Example 2.1. Load a SAS dataset into a pandas DataFrame

Type the following into the code cell immediately below, and then run that cell using Shift-Enter:

```
fish_df_smelt_only = sas.sasdata2dataframe(
    table='fish',
    libref='sashelp',
    dsopts={
        'where' : ' Species = "Smelt" ',
        'obs'   : 10,
    },
)
print(type(fish_df_smelt_only))
print('\n')
print(fish_df_smelt_only.shape)
print('\n')
display(fish_df_smelt_only.head())
```

```
        'matplotlib.artist',
        'matplotlib.axes',
        'matplotlib.axes._axes',
        'matplotlib.axes._base',
        'matplotlib.axes._secondary_axes',
        'matplotlib.axes._subplots',
        'matplotlib.axis',
        'matplotlib.backend_bases',
        'matplotlib.backend_tools',
        'matplotlib.backends',
        'matplotlib.backends._backend_agg',
        'matplotlib.backends.backend_agg',
        'matplotlib.bezier',
        'matplotlib.blocking_input',
        'matplotlib.category',
        'matplotlib.cbook',
        'matplotlib.cbook.deprecation',
fish_df_smelt_only = sas.sasdata2dataframe(
    table='fish',
    libref='sashelp',
    dsopts={
        'where' : ' Species = "Smelt" ',
        'obs'   : 10,
    },
)
print(type(fish_df_smelt_only))
print('\n')
print(fish_df_smelt_only.shape)
print('\n')
display(fish_df_smelt_only.head())
```

```
<class 'pandas.core.frame.DataFrame'>

(10, 7)
```

	Species	Weight	Length1	Length2	Length3	Height	Width
0	Smelt	6.7	9.3	9.8	10.8	1.7388	1.0476
1	Smelt	7.5	10.0	10.5	11.6	1.9720	1.1600
2	Smelt	7.0	10.1	10.6	11.6	1.7284	1.1484
3	Smelt	9.7	10.4	11.0	12.0	2.1960	1.3800
4	Smelt	9.8	10.7	11.2	12.4	2.0832	1.2772

```
 'matplotlib.streamplot',
 'matplotlib.style',
 'matplotlib.style.core',
 'matplotlib.table',
 'matplotlib.texmanager',
 'matplotlib.text',
 'matplotlib.textpath',
 'matplotlib.ticker',
 'matplotlib.tight_bbox',
 'matplotlib.tight_layout',
```

Notes about Example 2.1.

1. A DataFrame object named `fish_df_smelt_only` is created from the first 10 rows of the SAS dataset `fish` in the `sashelp` library satisfying `Species = "Smelt"`, and the following are printed with a blank line between them:
    - the type of object `fish_df_smelt_only` (which is `<class 'pandas.core.frame.DataFrame'>`)
    - the number of rows and columns in `fish_df_smelt_only`
    - the first five rows of `fish_df_smelt_only`, which are at row indices 0 through 4 since Python uses zero-based indexing
2. The `sas` object represents a connection to a SAS session and was created in Section 0 above. Here, `sas` uses its `sasdata2dataframe` method to access the SAS library `sashelp` and load the contents of `sashelp.fish(obs=10 where=(Species = "Smelt"))` into `fish_df_smelt_only`.
3. For extra credit, try any or all of the following:
    - Pass a numerical parameter to the `head` method to see a different number of rows (e.g., `fish_df_smelt_only.head(7)`).
    - Change the `head` method to `tail` to see a different part of the dataset.
    - To view other portions of `fish_df_smelt_only`, explore the more advanced indexing methods `loc` and `iloc` explained at https://brohrer.github.io/dataframe_indexing.html

```
       numpy.__config__ ,
display(fish_df_smelt_only.head(7))
print('\n')
display(fish_df_smelt_only.tail(3))
print('\n')
display(fish_df_smelt_only.loc[:,['Species', 'Weight']])
```

	Species	Weight	Length1	Length2	Length3	Height	Width
0	Smelt	6.7	9.3	9.8	10.8	1.7388	1.0476
1	Smelt	7.5	10.0	10.5	11.6	1.9720	1.1600

2	Smelt	7.0	10.1	10.6	11.6	1.7284	1.1484
3	Smelt	9.7	10.4	11.0	12.0	2.1960	1.3800
4	Smelt	9.8	10.7	11.2	12.4	2.0832	1.2772
5	Smelt	8.7	10.8	11.3	12.6	1.9782	1.2852
6	Smelt	10.0	11.3	11.8	13.1	2.2139	1.2838

	Species	Weight	Length1	Length2	Length3	Height	Width
7	Smelt	9.9	11.3	11.8	13.1	2.2139	1.1659
8	Smelt	9.8	11.4	12.0	13.2	2.2044	1.1484
9	Smelt	12.2	11.5	12.2	13.4	2.0904	1.3936

	Species	Weight
0	Smelt	6.7
1	Smelt	7.5
2	Smelt	7.0
3	Smelt	9.7
4	Smelt	9.8
5	Smelt	8.7
6	Smelt	10.0
7	Smelt	9.9
8	Smelt	9.8
9	Smelt	12.2

```
       'numpy.lib.scimath',
```

Example 2.2. Manipulate a DataFrame

Type the following into the code cell immediately below, and then run that cell using Shift-Enter:

```
fish_df     = sas.sasdata2dataframe(table='fish',libref='sashelp')
fish_df_g   = fish_df.groupby('Species')
fish_df_gs  = fish_df_g['Weight']
fish_df_gsa = fish_df_gs.agg(['count', 'std', 'mean', 'min', 'max'])
display(fish_df_gsa)
```

```
        'numpy.polynomial',
```

```
fish_df     = sas.sasdata2dataframe(table='fish',libref='sashelp')
fish_df_g   = fish_df.groupby('Species')
fish_df_gs  = fish_df_g['Weight']
fish_df_gsa = fish_df_gs.agg(['count', 'std', 'mean', 'min', 'max'])
display(fish_df_gsa)
```

	count	std	mean	min	max
Species					
Bream	34	206.604585	626.000000	242.0	1000.0
Parkki	11	78.755086	154.818182	55.0	300.0
Perch	56	347.617717	382.239286	5.9	1100.0
Pike	17	494.140765	718.705882	200.0	1650.0
Roach	20	88.828916	152.050000	0.0	390.0
Smelt	14	4.131526	11.178571	6.7	19.9
Whitefish	6	309.602972	531.000000	270.0	1000.0

```
         'packaging.__about__',
        'packaging._structures'
```

Notes about Example 2.2.

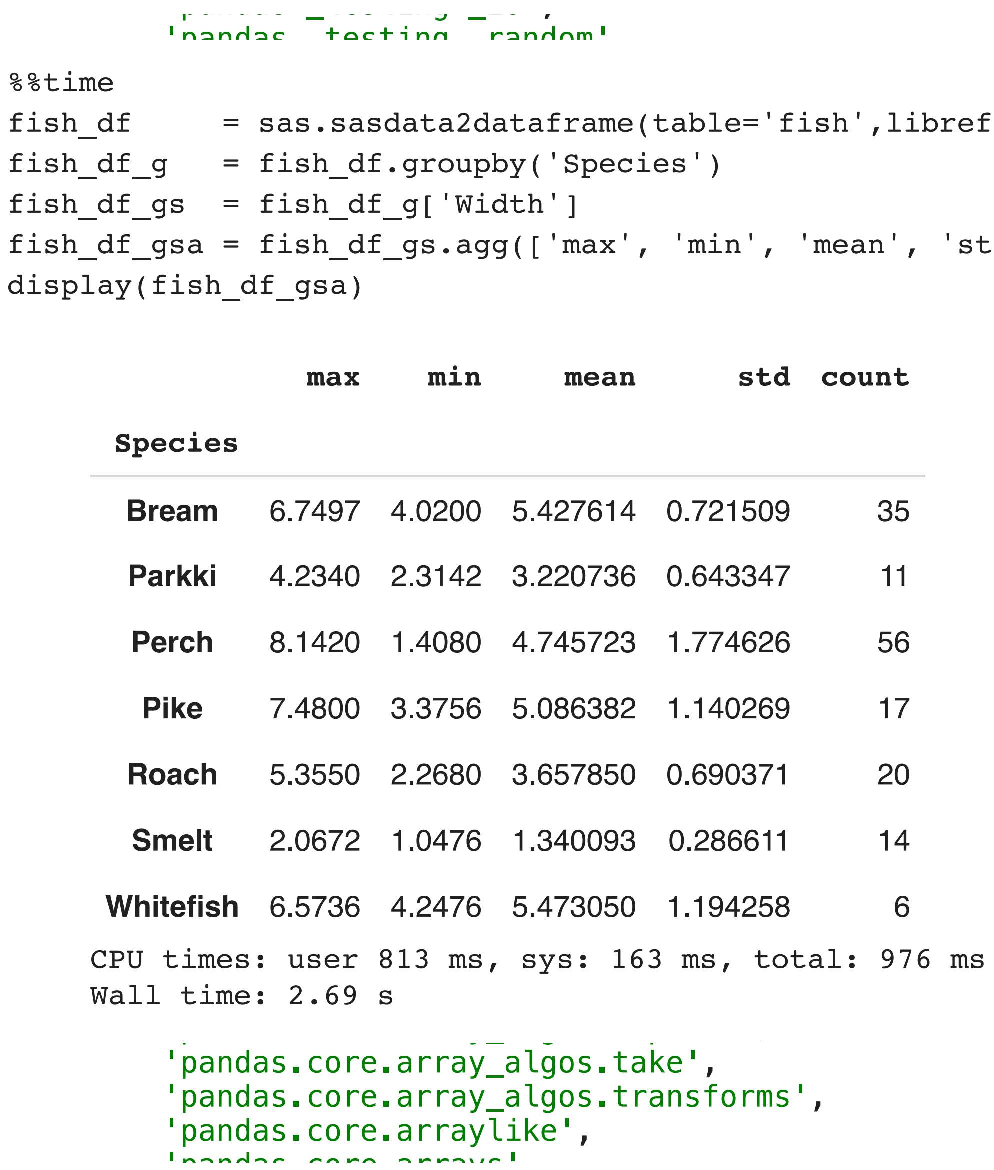
1. The DataFrame `fish_df` is created from the SAS dataset `sashelp.fish`, and this time all 159 rows are included since no dataset options were used. After some `pandas` operations are performed on `fish_df`, the following is printed:
    - a table giving the number of rows, standard deviation, mean, min, and max of `Weight` in `fish_df` when aggregated by `Species`
2. This is accomplished by creating a series of new DataFrames:
    - The DataFrame `fish_df_g` is created from `fish_df` using the `groupby` method to group rows by values in column `'Species'`.
    - The DataFrame `fish_df_gs` is created from `fish_df_g` by extracting the `'Weight'` column using bracket notation.
    - The DataFrame `fish_df_gsa` is created from `fish_df_gs` using the `agg` method to aggregate by the functions in the list `['count', 'std', 'mean', 'min', 'max']`.
3. Identical results could be obtained using the following SAS code:

```
PROC MEANS DATA=sashelp.fish STD MEAN MIN MAX;
     CLASS species;
     VAR weight;
RUN;
```

However, while PROC MEANS operates on SAS datasets row-by-row from disk, DataFrames are stored entirely in main memory. This allows any number of DataFrame operations to be combined for on-the-fly reshaping using "method chaining." In other words, `fish_df_gsa` could instead be created with the following one-liner, which avoids the need for intermediate DataFrames (and executes much more quickly):

```
fish_df.groupby('Species')['Weight'].agg(['count', 'std', 'mean', 'min', 'max'])
```

4. For extra credit, try any or all of the following:
    - Move around and/or remove functions used for aggregation, and see how the output changes.
    - Change the variable whose values are summarized to `'Width'`.
    - Obtain execution time for the one-liner version by included the JupyerLab magic `%%time` at the start of a code cell (on a line by itself).



```
    'pandas._testing._random'
```

```
%%time
fish_df      = sas.sasdata2dataframe(table='fish',libref='sashelp')
fish_df_g    = fish_df.groupby('Species')
fish_df_gs   = fish_df_g['Width']
fish_df_gsa = fish_df_gs.agg(['max', 'min', 'mean', 'std', 'count'])
display(fish_df_gsa)
```

	max	min	mean	std	count
Species					
Bream	6.7497	4.0200	5.427614	0.721509	35
Parkki	4.2340	2.3142	3.220736	0.643347	11
Perch	8.1420	1.4080	4.745723	1.774626	56
Pike	7.4800	3.3756	5.086382	1.140269	17
Roach	5.3550	2.2680	3.657850	0.690371	20
Smelt	2.0672	1.0476	1.340093	0.286611	14
Whitefish	6.5736	4.2476	5.473050	1.194258	6

```
CPU times: user 813 ms, sys: 163 ms, total: 976 ms
Wall time: 2.69 s
```

```
    'pandas.core.array_algos.take',
    'pandas.core.array_algos.transforms',
    'pandas.core.arraylike',
    'pandas.core.arrays'
```

```
%%time
fish_df.groupby('Species')['Weight'].agg(['count', 'std', 'mean', 'min', 'max'])
```

```
CPU times: user 3.11 ms, sys: 10 µs, total: 3.12 ms
Wall time: 3.03 ms
```

	count	std	mean	min	max
Species					
Bream	34	206.604585	626.000000	242.0	1000.0
Parkki	11	78.755086	154.818182	55.0	300.0
Perch	56	347.617717	382.239286	5.9	1100.0
Pike	17	494.140765	718.705882	200.0	1650.0
Roach	20	88.828916	152.050000	0.0	390.0
Smelt	14	4.131526	11.178571	6.7	19.9
Whitefish	6	309.602972	531.000000	270.0	1000.0

```
    'pandas.core.computation.align',
    'pandas.core.computation.api',
    'pandas.core.computation.check',
    'pandas.core.computation.common',
    'pandas.core.computation.engines',
    'pandas.core.computation.eval',
    'pandas.core.computation.expr',
    'pandas.core.computation.expressions',
    'pandas.core.computation.ops',
    'pandas.core.computation.parsing',
    'pandas.core.computation.pytables',
    'pandas.core.computation.scope',
    'pandas.core.config_init',
    'pandas.core.construction',
    'pandas.core.describe',
    'pandas.core.dtypes',
    'pandas.core.dtypes.api',
    'pandas.core.dtypes.base',
    'pandas.core.dtypes.cast',
```

Example 2.3. Load a DataFrame into a SAS dataset

Type the following into the code cell immediately below, and then run that cell using Shift-Enter:

```
from IPython.display import display, HTML

sas.dataframe2sasdata(
    fish_df_gsa.reset_index(),
    table="fish_sds_gsa",
    libref="Work"
)
sas_submit_return_value = sas.submit(
    '''
        PROC PRINT DATA=fish_sds_gsa;
        RUN;
    '''
)
sas_submit_log = sas_submit_return_value['LOG']
print(sas_submit_log)
sas_submit_results = sas_submit_return_value['LST']
display(HTML(sas_submit_results))
```

```
        'pandas.core.indexes.range',
```

```
from IPython.display import display, HTML

sas.dataframe2sasdata(
    fish_df_gsa.reset_index(),
    table="fish_sds_gsa",
    libref="Work"
)
sas_submit_return_value = sas.submit(
    '''
        PROC PRINT DATA=fish_sds_gsa;
        RUN;
    '''
)
sas_submit_log = sas_submit_return_value['LOG']
print(sas_submit_log)
sas_submit_results = sas_submit_return_value['LST']
display(HTML(sas_submit_results))
```

```
42                                                     The SAS System                          Mond

280          ods listing close;ods html5 (id=saspy_internal) file=_tomods1 options(bitmap_mode='inline') d
280        ! ods graphics on / outputfmt=png;
281
282
283                  PROC PRINT DATA=fish_sds_gsa;
284                  RUN;
285
286
287
288          ods html5 (id=saspy_internal) close;ods listing;
289

43                                                     The SAS System                          Mond

290
```

The SAS System

Obs	Species	max	min	mean	std	count
1	Bream	6.7497	4.0200	5.42761	0.72151	35
2	Parkki	4.2340	2.3142	3.22074	0.64335	11
3	Perch	8.1420	1.4080	4.74572	1.77463	56
4	Pike	7.4800	3.3756	5.08638	1.14027	17
5	Roach	5.3550	2.2680	3.65785	0.69037	20
6	Smelt	2.0672	1.0476	1.34009	0.28661	14
7	Whitefish	6.5736	4.2476	5.47305	1.19426	6

```
      'pandas.io.parsers',
      'pandas.io.parsers.base_parser',
      'pandas.io.parsers.c_parser_wrapper',
      'pandas.io.parsers.python_parser',
      'pandas.io.parsers.readers',
      'pandas.io.pickle',
      'pandas.io.pytables',
      'pandas.io.sas',
      'pandas.io.sas.sasreader'
```

Notes about Example 2.3.

1. The DataFrame `fish_df_gsa`, which was created in Example 2.2 from the SAS dataset `sashelp.fish`, is used to create the new SAS dataset `Work.fish_sds_gsa`. The SAS PRINT procedure is then called, and the following is displayed:
    - the SAS log returned by PROC PRINT
    - the output returned by PROC PRINT
2. The `sas` object, which was created in Section 0, is a persistent connection to a SAS session, and two of its methods are used as follows:
    - The `dataframe2sasdata` method writes the contents of the DataFrame `fish_df_gsa` to the SAS dataset `fish_sds_gsa` stored in the `Work` library. (Note: The row indexes of the DataFrame `fish_df_gsa` are lost when the SAS dataset `fish_sds_gsa` is created.)
    - The `submit` method is used to submit the PROC PRINT step to the SAS kernel, and a dictionary is returned with the following two key-value pairs:
        - `sas_submit_return_value['LOG']` is a string comprising the plain-text log resulting from executing PROC PRINT
        - `sas_submit_return_value['LST']` is a string comprising the results from executing PROC PRINT, which will be in HTML by default.
3. Python strings surrounded by single quotes (e.g., `'Hello, World!'`) cannot be written across multiple lines of code, whereas strings surrounded by triple quotes (e.g., the argument to the `submit` method) can.
4. For extra credit, try any or all of the following:
    - Change the SAS procedure used to interact with SAS dataset `Work.fish_sds_gsa` (e.g., try PROC CONTENTS).
    - Change the format of the SAS output by adding the argument `results='TEXT'` to the `sas.submit` call, and display it with the `print` function instead.

```
                'pkg_resources.extern.packaging',
               'pkg_resources.extern.packaging._compat'
```

```
sas_submit_return_value = sas.submit(
    '''
        PROC CONTENTS DATA=fish_sds_gsa;
        RUN;
    ''',
    results="TEXT"
)
sas_submit_log = sas_submit_return_value['LOG']
print(sas_submit_log)
sas_submit_results = sas_submit_return_value['LST']
print(sas_submit_results)
```

```
46                                                         The SAS System                                Mond

304
305
306                   PROC CONTENTS DATA=fish_sds_gsa;
307                   RUN;
308
309
310
311

47                                                         The SAS System                                Mond

312

                                                           The SAS System                            Monday, 

                                                       The CONTENTS Procedure

                  Data Set Name        WORK.FISH_SDS_GSA                                        Observation
                  Member Type          DATA                                                     Variables
                  Engine               V9                                                       Indexes
                  Created              05/23/2022 04:12:30                                      Observation
                  Last Modified        05/23/2022 04:12:30                                      Deleted Obs
                  Protection                                                                    Compressed
                  Data Set Type                                                                 Sorted
                  Label
                  Data Representation  SOLARIS_X86_64, LINUX_X86_64, ALPHA_TRU64, LINUX_IA64
                  Encoding             utf-8  Unicode (UTF-8)
```

```
                                        Engine/Host Dependent Information

   Data Set Page Size          131072
   Number of Data Set Pages    1
   First Data Page             1
   Max Obs per Page            2334
   Obs in First Data Page      7
   Number of Data Set Repairs  0
   Filename                    /saswork/SAS_work75A600014928_odaws02-usw2.oda.sas.com/SAS_work735000014928_
                               odaws02-usw2.oda.sas.com/fish_sds_gsa.sas7bdat
   Release Created             9.0401M6
   Host Created                Linux
   Inode Number                1936147
   Access Permission           rw-r--r--
   Owner Name                  isaiah.lankham
   File Size                   256KB
   File Size (bytes)           262144


                                     Alphabetic List of Variables and Attributes

                                          #    Variable    Type    Len

                                          1    Species     Char      9
                                          6    count       Num       8
                                          2    max         Num       8
                                          4    mean        Num       8
                                          3    min         Num       8
                                          5    std         Num       8
```

```
     'prompt_toolkit.styles.from_dict'
```

Section 3. Executing SAS Procedures with Convenience Methods

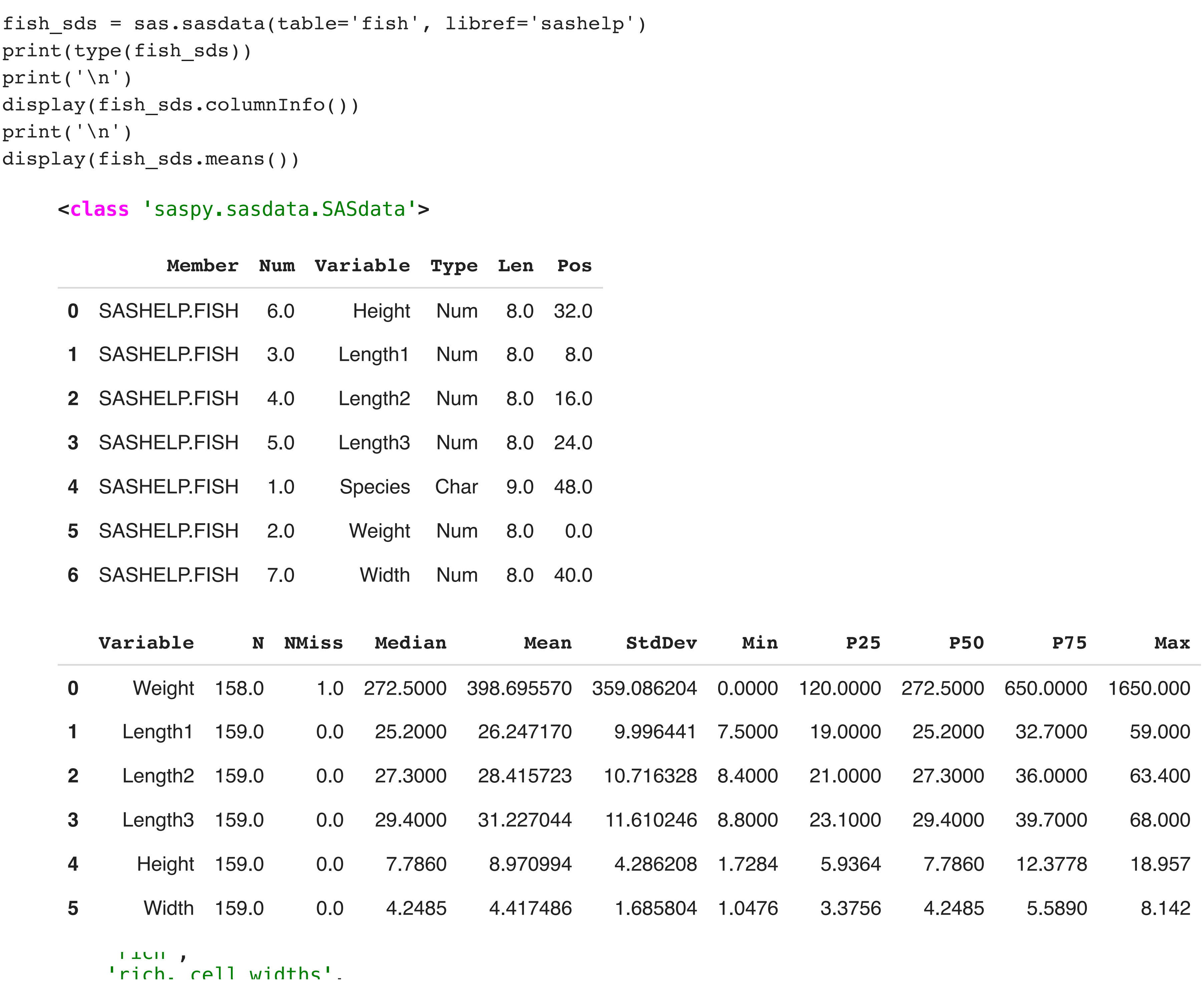
```
     'prompt_toolkit.terminal.vt100_input',
     'prompt_toolkit.terminal.vt100_output',
     'prompt_toolkit.token',
     'prompt_toolkit.utils',
     'prompt_toolkit.validation',
     'pstats',
     'psutil',
```

Example 3.1. Connect directly to a SAS dataset

Type the following into the code cell immediately below, and then run that cell using Shift-Enter:

```
fish_sds = sas.sasdata(table='fish', libref='sashelp')
print(type(fish_sds))
print('\n')
display(fish_sds.columnInfo())
print('\n')
display(fish_sds.means())
```

```
        'pyarrow.filesystem',
        'pyarrow.hdfs',
        'pyarrow.ipc',
        'pyarrow.lib',
        'pyarrow.serialization',
        'pyarrow.types',
        'pyarrow.util',
        'pydev_ipython',
        'pydevconsole',
        'pydevd',
        'pydevd_concurrency_analyser',
        'pydevd_concurrency_analyser.pydevd_concurrency_logger',
        'pydevd_concurrency_analyser.pydevd_thread_wrappers',
        'pydevd_file_utils',
        'pydevd_plugins',
        'pydevd_plugins.django_debug',
        'pydevd_plugins.extensions',
        'pydevd_plugins.extensions.types',
        'pydevd_plugins.extensions.types.pydevd_helpers',
        'pydevd_plugins.extensions.types.pydevd_plugin_numpy_types',
        'pydevd_plugins.extensions.types.pydevd_plugins_django_form_str',
        'pydevd_plugins.jinja2_debug',
        'pydevd_tracing',
        'pydoc',
        'pyexpat',
        'pyexpat.errors',
        'pyexpat.model',
```



```
fish_sds = sas.sasdata(table='fish', libref='sashelp')
print(type(fish_sds))
print('\n')
display(fish_sds.columnInfo())
print('\n')
display(fish_sds.means())
```

```
<class 'saspy.sasdata.SASdata'>
```

```
        Member  Num  Variable  Type  Len   Pos
0  SASHELP.FISH  6.0    Height   Num  8.0  32.0
1  SASHELP.FISH  3.0   Length1   Num  8.0   8.0
2  SASHELP.FISH  4.0   Length2   Num  8.0  16.0
3  SASHELP.FISH  5.0   Length3   Num  8.0  24.0
4  SASHELP.FISH  1.0   Species  Char  9.0  48.0
5  SASHELP.FISH  2.0    Weight   Num  8.0   0.0
6  SASHELP.FISH  7.0     Width   Num  8.0  40.0
```

```
   Variable      N  NMiss    Median        Mean      StdDev     Min       P25       P50       P75       Max
0    Weight  158.0    1.0  272.5000  398.695570  359.086204  0.0000  120.0000  272.5000  650.0000  1650.000
1   Length1  159.0    0.0   25.2000   26.247170    9.996441  7.5000   19.0000   25.2000   32.7000    59.000
2   Length2  159.0    0.0   27.3000   28.415723   10.716328  8.4000   21.0000   27.3000   36.0000    63.400
3   Length3  159.0    0.0   29.4000   31.227044   11.610246  8.8000   23.1000   29.4000   39.7000    68.000
4    Height  159.0    0.0    7.7860    8.970994    4.286208  1.7284    5.9364    7.7860   12.3778    18.957
5     Width  159.0    0.0    4.2485    4.417486    1.685804  1.0476    3.3756    4.2485    5.5890     8.142
```

```
  'rich',
 'rich, cell widths',
```

Notes about Example 3.1.

1. The SASdata object `fish_sds` (meaning a direct connection to the disk-based SAS dataset `sashelp.fish` in our remote SAS ODA session, not an in-memory DataFrame in our local Python session) is created, and the following are printed with a blank line between them:
    - the type of object `fish_sds` (which is `<class 'saspy.sasdata.SASdata'>`)
    - a portion of PROC CONTENTS applied to the SAS dataset `sashelp.fish`
    - summary information about the numeric columns in `sashelp.fish`
2. The `sas` object, which was created in Section 0, is a persistent connection to a SAS session, and its `sasdata` method is used to create the connection to `sashelp.fish`.
3. The `fish_sds` object calls its *convenience method* `means`, which implicitly invokes PROC MEANS on `sashelp.fish`.
4. For extra credit, try the following:
    - Explore the additional convenience methods listed at https://sassoftware.github.io/saspy/api.html#sas-data-object.

```
'rich.errors',
```

Example 3.2 Display generated SAS code

Type the following into the code cell immediately below, and then run that cell using Shift-Enter:

```
sas.teach_me_SAS(True)
fish_sds.means()
sas.teach_me_SAS(False)
```

```
'rich.scope',
'rich.screen',
'rich.segment',
'rich.style',
'rich.styled',
'rich.table',
'rich.terminal_theme',
sas.teach_me_SAS(True)
fish_sds.means()
sas.teach_me_SAS(False)
```

```
proc means data=sashelp.'fish'n  stackodsoutput n nmiss median mean std min p25 p50 p75 max;run;
```

'saspy.sasViyaML'.

Notes about Example 3.2

1. The SASdata object `fish_sds`, which was created in Example 3.1 as a direct connection to the SAS dataset `sashelp.fish`, calls its *convenience method* `means` within a "Teach Me SAS" sandwich, and the following is printed:
    - the SAS code for the PROC MEANS step implicitly generated by the `means` convenience method
2. The `sas` object, which was created in Section 0, is a persistent connection to a SAS session, and its `teach_me_SAS` method is used as follows:
    - When called with argument `True`, SAS output is suppressed for all subsequent `saspy` convenience methods, and the SAS code that would be generated by the convenience method is printed instead.
    - When `teach_me_SAS` is called with argument `False`, this behavior is turned off.
3. `True` and `False` are standard Python objects. Like their SAS equivalents, they are interchangeable with the values `1` and `0`, respectively.
4. One benefit of this process is being able to extract and modify the SAS code. For example, if a convenience method doesn't offer an option like a class statement for PROC MEANS, we can manually add it to the code generated by the `teach_me_SAS` method and then execute the modified SAS code using the `submit` method (as in Example 2.3 above).
5. For extra credit, try any or all of the following:
    - Change `means` to a different convenience method, such as `columnInfo`.
    - Submit the generated SAS code by `teach_me_SAS` using the `submit` method.

'sqlite3'.

```
sas.teach_me_SAS(True)
```

```
fish_sds.columnInfo()
sas.teach_me_SAS(False)

sas_submit_return_value = sas.submit(
    '''
        proc contents data=sashelp.'fish'n ;ods select Variables;run;
    '''
)
sas_submit_log = sas_submit_return_value['LOG']
print(sas_submit_log)
sas_submit_results = sas_submit_return_value['LST']
display(HTML(sas_submit_results))
```

```
proc contents data=sashelp.'fish'n ;ods select Variables;run;
```

```
82                                                         The SAS System                                    Mond

529        ods listing close;ods html5 (id=saspy_internal) file=_tomods1 options(bitmap_mode='inline') d
529      ! ods graphics on / outputfmt=png;
530
531
532               proc contents data=sashelp.'fish'n ;ods select Variables;run;
533
534
535
536        ods html5 (id=saspy_internal) close;ods listing;
537

83                                                         The SAS System                                    Mond

538
```

The SAS System

The CONTENTS Procedure

Alphabetic List of Variables and Attributes			
#	Variable	Type	Len
6	Height	Num	8
3	Length1	Num	8
4	Length2	Num	8
5	Length3	Num	8
1	Species	Char	9
2	Weight	Num	8
7	Width	Num	8

```
    'zmq.backend.cython._device',
    'zmq.backend.cython._poll',
    'zmq.backend.cython._proxy_steerable',
    'zmq.backend.cython._version',
```

Example 3.3 Adding variables to a SAS dataset

Type the following into the code cell immediately below, and then run that cell using Shift-Enter:

```
class_sds = sas.sasdata(
    table='class',
    libref='sashelp'
)
class_bmi_sds = sas.sasdata(
    table='class_bmi',
    libref='work'
)
class_sds.add_vars(vars = {'bmi':'(Weight/Height**2)*703'}, out = class_bmi_sds)
display(class_bmi_sds.head())
print('\n')
display(class_bmi_sds.means())
```

```
        zmq.utils.strtypes
```

```
class_sds = sas.sasdata(
    table='class',
    libref='sashelp'
)
class_bmi_sds = sas.sasdata(
    table='class_bmi',
    libref='work'
)
class_sds.add_vars(vars = {'bmi':'(Weight/Height**2)*703'}, out = class_bmi_sds)
display(class_bmi_sds.head())
print('\n')
display(class_bmi_sds.means())
```

```
Table work.class_bmi does not exist. This SASdata object will not be useful until the data set is createc

94                                                          The SAS System                          Monda

597
598          data work.'class_bmi'n ; set sashelp.'class'n ;
599          bmi = (Weight/Height**2)*703;
600          ; run;
601
602
603

95                                                          The SAS System                          Monda

604
```

	Name	Sex	Age	Height	Weight	bmi
0	Alfred	M	14.0	69.0	112.5	16.611531
1	Alice	F	13.0	56.5	84.0	18.498551
2	Barbara	F	13.0	65.3	98.0	16.156788
3	Carol	F	14.0	62.8	102.5	18.270898
4	Henry	M	14.0	63.5	102.5	17.870296

	Variable	N	NMiss	Median	Mean	StdDev	Min	P25	P50	P75	Max
0	Age	19.0	0.0	13.000000	13.315789	1.492672	11.000000	12.000000	13.000000	15.000000	16.00000
1	Height	19.0	0.0	62.800000	62.336842	5.127075	51.300000	57.500000	62.800000	66.500000	72.00000
2	Weight	19.0	0.0	99.500000	100.026316	22.773933	50.500000	84.000000	99.500000	112.500000	150.00000
3	bmi	19.0	0.0	17.804511	17.863252	2.092619	13.490001	16.611531	17.804511	20.094369	21.42966

Notes about Example 2.2

Notes about Example 3.3:

1. The SASdata object `class_sds` (meaning a direct connection to the disk-based SAS dataset `sashelp.class` in our remote SAS ODA session) is created, the results of adding a new column to `class_sds` are then output into the new SASdata object `class_bmi_sds`, and the following are printed with a blank line between them:
    - the first few rows of `class_bmi_sds`
    - summary information about the numeric columns in `class_bmi_sds`
2. The `sas` object, which was created in Section 0, is a persistent connection to a SAS session, and two of its methods are used as follows:
    - The `sasdata` method is used twice, first to create a pointer to `sashelp.class` and then to create a pointer to the not-yet-created SAS dataset `work.class_bmi`.
    - The `add_vars` method is used to creata a new column in `sashelp.class`, but to output the results of creating this new column as `work.class_bmi`. (If no `out=` argument has been specified, SAS would have attempted to modify `sashelp.class` in place.)
3. Identical results could be obtained using the following SAS code:

```
DATA Work.class_bmi;
     SET sashelp.class;
     bmi = (Weight/Height**2)*703;
RUN;
PROC PRINT DATA=Work.class_bmi(obs=5);
RUN;
PROC MEANS DATA=Work.class_bmi;
RUN;
```

4. For extra credit, try any or all of the following:

- Print only the first three rows of `class_bmi_sds` by adding a numeric argument to the `head` method call (just like the corresponding `pandas` method).
- Add a `dsopts=` parameter to either use of the `sasdata` method above. (For example, you could use similar syntax as in Example 2.1 to limit the columns with `where` and/or `obs` options.)

  Note: Just like the `sasdata2dataframe` method used in Example 2.1, the `sasdata` method has a `dsopts` argument, which allows dataset options to specified. The underlying SAS dataset itself will not be modified unless `dsopts` is specified for an output dataset.

```
class_sds = sas.sasdata(
    table='class',
    libref='sashelp',
    dsopts={
        'where' : ' Age > 13',
        'keep'  : ['Name','Age','Height','Weight'],
    }
)
class_bmi_sds = sas.sasdata(
    table='class_bmi',
    libref='work',
    dsopts={'keep' : ['Name','bmi']}
)
class_sds.add_vars(vars = {'bmi':'(Weight/Height**2)*703'}, out = class_bmi_sds)
display(class_bmi_sds.head(3))
display(class_bmi_sds.means())
```

```
134                                                        The SAS System                          Monda

843
844          data work.'class_bmi'n (keep=Name bmi ); set sashelp.'class'n (where=( Age > 13) keep=Name Age
845          bmi = (Weight/Height**2)*703;
846          ; run;
847
848
849

135                                                        The SAS System                          Monda

850
```

	Name	bmi
0	Alfred	16.611531
1	Carol	18.270898
2	Henry	17.870296

	Variable	N	NMiss	Median	Mean	StdDev	Min	P25	P50	P75	Max
0	bmi	9.0	0.0	17.870296	18.342336	1.831753	15.302976	17.804511	17.870296	20.2464	20.82847

Section 4. Staying D.R.Y. (aka "Don't Repeat Yourself!")

Example 4.1. Imitate the SAS Macro Processor

Type the following into the code cell immediately below, and then run that cell using Shift-Enter:

```
sas_code_fragment = 'PROC MEANS DATA=sashelp.{data}; RUN;'
for dsn in ['fish', 'class']:
    sas_submit_return_value = sas.submit(
        sas_code_fragment.format(data=dsn)
    )
    print(sas_submit_return_value['LOG'])
    display(HTML(sas_submit_return_value['LST']))
```

```
sas_code_fragment = 'PROC MEANS DATA=sashelp.{data}; RUN;'
for dsn in ['fish', 'class']:
    sas_submit_return_value = sas.submit(
        sas_code_fragment.format(data=dsn)
    )
    print(sas_submit_return_value['LOG'])
    display(HTML(sas_submit_return_value['LST']))
```

```
164                                                    The SAS System                          Mond

1029        ods listing close;ods html5 (id=saspy_internal) file=_tomods1 options(bitmap_mode='inline') d
1029      ! ods graphics on / outputfmt=png;
1030
1031        PROC MEANS DATA=sashelp.fish; RUN;
1032
1033
1034        ods html5 (id=saspy_internal) close;ods listing;
1035

165                                                    The SAS System                          Mond

1036
```

The SAS System

The MEANS Procedure

Variable	N	Mean	Std Dev	Minimum	Maximum
Weight	158	398.6955696	359.0862037	0	1650.00
Length1	159	26.2471698	9.9964412	7.5000000	59.0000000
Length2	159	28.4157233	10.7163281	8.4000000	63.4000000
Length3	159	31.2270440	11.6102458	8.8000000	68.0000000
Height	159	8.9709937	4.2862076	1.7284000	18.9570000
Width	159	4.4174855	1.6858039	1.0476000	8.1420000

```
166                                                          The SAS System                          Mond

1039        ods listing close;ods html5 (id=saspy_internal) file=_tomods1 options(bitmap_mode='inline') d
1039      ! ods graphics on / outputfmt=png;
1040
1041        PROC MEANS DATA=sashelp.class; RUN;
1042
1043
1044        ods html5 (id=saspy_internal) close;ods listing;
1045

167                                                          The SAS System                          Mond

1046
```

The SAS System

The MEANS Procedure

Variable	N	Mean	Std Dev	Minimum	Maximum
Age	19	13.3157895	1.4926722	11.0000000	16.0000000
Height	19	62.3368421	5.1270752	51.3000000	72.0000000
Weight	19	100.0263158	22.7739335	50.5000000	150.0000000

Notes about Example 4.1.

1. A string object named `sas_code_fragment` is created with templating placeholder `{data}`, which will be filled using other strings in subsequent uses of `sas_code_fragment`.
2. The output of PROC MEANS applied to SAS datasets `sashelp.fish` and `sashelp.class` is then displayed.
3. The `sas` object represents a connection to a SAS session and was created in Section 0. Here, `sas` calls its `submit` method for each value of the for-loop indexing variable `dsn`, and the `{data}` portion of `sas_code_fragment` is replaced by the value of `dsn`. In other words, the following SAS code is submitted to the SAS kernel:

   ```
   PROC MEANS DATA=sashelp.fish; RUN;
   PROC MEANS DATA=sashelp.class; RUN;
   ```

4. The same outcome could also be achieved with the following SAS macro code:

   ```
   %MACRO loop();
        %LET dsn_list = fish class;
        %DO i = 1 %TO 2;
            %LET dsn = %SCAN(&dsn_list.,&i.);
            PROC MEANS DATA=sashelp.&dsn.;
            RUN;
        %END;
   %MEND;
   %loop()
   ```

   However, note the following differences:

   - Python allows us to concisely repeat an arbitrary block of code by iterating over a list using a for-loop. In other words, the body of the for-loop (meaning everything indented underneath it, since Python uses indentation to determine scope) is repeated for each string in the list `['fish','class']`.

- The SAS macro facility only provides do-loops based on numerical index variables (the macro variable `i` above), so clever tricks like implicitly defined arrays (macro variable `dsn_list` above) need to be used together with functions like `%scan` to extract a sequence of values.

5. For extra credit, try any or all of the following:
    - Add additional SASHELP datasets to the list being iterated over by the for-loop (e.g., iris or cars).
    - Change the `sas_code_fragment` to run a different SAS procedure (e.g., PROC PRINT).

```
sas_code_fragment = 'PROC PRINT DATA=sashelp.{data}(obs=3); RUN;'
for dsn in ['fish', 'class', 'iris', 'cars']:
    sas_submit_return_value = sas.submit(
        sas_code_fragment.format(data=dsn)
    )
    print(sas_submit_return_value['LOG'])
    display(HTML(sas_submit_return_value['LST']))
```

```
168                                                          The SAS System                      Mond

1049        ods listing close;ods html5 (id=saspy_internal) file=_tomods1 options(bitmap_mode='inline') d
1049      ! ods graphics on / outputfmt=png;
1050
1051        PROC PRINT DATA=sashelp.fish(obs=3); RUN;
1052
1053
1054        ods html5 (id=saspy_internal) close;ods listing;
1055

169                                                          The SAS System                      Mond

1056
```

The SAS System

Obs	Species	Weight	Length1	Length2	Length3	Height	Width
1	Bream	242	23.2	25.4	30.0	11.5200	4.0200

1	Bream	242	23.2	25.4	30.0	11.5200	4.0200
2	Bream	290	24.0	26.3	31.2	12.4800	4.3056
3	Bream	340	23.9	26.5	31.1	12.3778	4.6961

```
170                                                         The SAS System                          Mond

1059         ods listing close;ods html5 (id=saspy_internal) file=_tomods1 options(bitmap_mode='inline') d
1059       ! ods graphics on / outputfmt=png;
1060
1061         PROC PRINT DATA=sashelp.class(obs=3); RUN;
1062
1063
1064         ods html5 (id=saspy_internal) close;ods listing;
1065

171                                                         The SAS System                          Mond

1066
```

The SAS System

Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0

```
172                                                         The SAS System                          Mond

1069         ods listing close;ods html5 (id=saspy_internal) file=_tomods1 options(bitmap_mode='inline') d
1069       ! ods graphics on / outputfmt=png;
1070
1071         PROC PRINT DATA=sashelp.iris(obs=3); RUN;
1072
1073
1074         ods html5 (id=saspy_internal) close;ods listing;
1075

173                                                         The SAS System                          Mond
```

Wrapping Up: Call to Action!

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

1. Continue learning Python.
   - For general programming, we recommend starting with these:
     - Automate the Boring Stuff with Python, a free online book with numerous beginner-friendly hands-on projects
     - Fluent Python, which provided a deep dive into Intermediate to Advanced Python concepts
   - For data science, we recommend starting with these:
     - A Whirlwind Tour of Python, a free online book with coverage of essential Python features commonly used in data science projects
     - Python for Data Analysis, which provided a deep dive into the `pandas` package by its creator, Wes McKinney
   - For web development in Python, we recommend starting with this:
     - The Flask Mega-Tutorial, a freely accessible series of blog posts covering essential features of developing dynamic websites with the `flask` web framework
2. Try using SASPy outside of Google Colab. For example, if you're interested in using a local SASPy environment, with Python talking to a commercial SAS installation, you're welcome to follow the setup instructions for the demo application https://github.com/saspy-bffs/dataset-explorer

1. Keep in touch for follow-up questions/discussion (one of our favorite parts of teaching!) using isaiah.lankham@gmail.com and matthew.t.slaughter@gmail.com
2. 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

In addition, you might also find the following documentation useful:

1. For more about the `pandas` package, including the methods used above, see the following:
   - https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.agg.html
   - https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.groupby.html
   - https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.head.html
   - https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.info.html
   - https://pandas.pydata.org/docs/reference/api/pandas.Index.shape.html
2. For more about the `platform` package, see https://docs.python.org/3/library/platform.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.sasdata.SASdata.add_vars
   - https://sassoftware.github.io/saspy/api.html#saspy.sasdata.SASdata.columnInfo
   - https://sassoftware.github.io/saspy/api.html#saspy.sasdata.SASdata.head
   - https://sassoftware.github.io/saspy/api.html#saspy.sasdata.SASdata.means
   - 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
   - https://sassoftware.github.io/saspy/api.html#saspy.SASsession.teach_me_SAS