Everything is Better with Friends

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

Setup for Part 2

Getting setup to use Google Colab

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.)
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. Looking for "extra credit"? Please let us know if you spot any typos!

Install and import 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 requests package to call a web API
import requests

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

```
# 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
from rich import get_console
console = get_console()
console.width = 165
```

```
Collecting rich
  Downloading rich-10.15.2-py3-none-any.whl (214 kB)
     |████████████████████████████████| 214 kB 5.1 MB/s
Requirement already satisfied: pygments<3.0.0,>=2.6.0 in /usr/local/lib/python3.
Collecting commonmark<0.10.0,>=0.9.0
  Downloading commonmark-0.9.1-py2.py3-none-any.whl (51 kB)
     |████████████████████████████████| 51 kB 4.2 MB/s
Requirement already satisfied: typing-extensions<5.0,>=3.7.4 in /usr/local/lib/p
Collecting colorama<0.5.0,>=0.4.0
  Downloading colorama-0.4.4-py2.py3-none-any.whl (16 kB)
Installing collected packages: commonmark, colorama, rich
Successfully installed colorama-0.4.4 commonmark-0.9.1 rich-10.15.2
```

Part 2. Rectangularizing unstructured data in Python applications

Section 2.1. Create defns_of_set_response

```
# I was obsessed with The Guinness Book of Records as a kid, and remember reading
# "set" has the most distinct meanings of any word in the English language. Let's

# Use an open web API to get definitions of "set".
defns_of_set_response = requests.get('https://api.dictionaryapi.dev/api/v2/entries

# Check the resulting status code to make sure the API call was successful, with 2
http_status = defns_of_set_response.status_code
http_status_info = f'https://httpstatuses.com/{defns_of_set_response.status_code}'
if defns_of_set_response.status_code == 200:
    print('API call successful!\n')
print(f'See {http_status_info} for more information about HTTP status code {http_s
```

```
API call successful!

See https://httpstatuses.com/200 for more information about HTTP status code 200
```

Concept Check 2.1

1. True or False: Unindenting Line 11 would produce identical behavior.

2. True or False: Single-equals (`=`) and double-equals (`==`) can be used interchangeable in

1. *False. Removing this white space would produce an error since the `if` statement would have no body.*
2. *False. In Python, `=` is only used for assignment, and `==` is only used to test for equality.*

Section 2.2. Create defns_json

```
# Extract and print the JSON-formatted definitions of "set".
defns_json = defns_of_set_response.json()
print(defns_json)
```

```
[
    {
        'word': 'set',
        'phonetic': 'sɛt',
        'phonetics': [{'text': 'sɛt', 'audio': '//ssl.gstatic.com/dictionary/sta
        'origin': 'Old English settan, of Germanic origin; related to Dutch zett
        'meanings': [
            {
                'partOfSpeech': 'verb',
                'definitions': [
                    {
                        'definition': 'put, lay, or stand (something) in a speci
                        'example': 'Delaney set the mug of tea down',
                        'synonyms': [
                            'put',
                            'place',
                            'put down',
                            'lay',
                            'lay down',
                            'deposit',
```

Concept Check 2.2

1. True or False: In Python, it's common to work with deeply nested objects (like a Russian nested doll, or a Turducken).
2. Short Answer: What types of standard Python objects appear in the output of `defns_json`?

1. *True. Python's object-oriented design frequently makes use of deeply nested objects.*
2. *Both `list` and `dict` objects appear, both containing `str` objects.*

Section 2.3. Create defns_list

```
# Because the API returns usage patterns for the word "set" as a deeply nested col
# and dicts, we'll need to match this structure by recursively using
# (a) for-loops to loop over lists and
# (b) dict-indexing to get values corresponding to specific keys.
# This is typically the least straightforward part of using web APIs.

# Accumulate definitions in a list of lists called defns_list.
defns_list = []
for usage_pattern in defns_json:
    for meaning in usage_pattern['meanings']:
        part_of_speech = meaning['partOfSpeech']
        for defn in meaning['definitions']:
            defns_list.append(
                [
                    part_of_speech,
                    defn['definition'],
```

```
                    defn.get('example',''),
                ]
            )

# And then print defns_list
print(defns_list)
```

```
[
    ['verb', 'put, lay, or stand (something) in a specified place or position.',
    ['verb', 'put or bring into a specified state.', 'the Home Secretary set in
    ['verb', 'adjust (a clock or watch), typically to show the right time.', 'se
    ['verb', 'harden into a solid or semi-solid state.', 'cook for a further thi
    [
        'verb',
        "(of the sun, moon, or another celestial body) appear to move towards ar
        'the sun was setting and a warm red glow filled the sky'
    ],
    [
        'verb',
        '(of a tide or current) take or have a specified direction or course.',
        "a fair tide can be carried well past Land's End before the stream sets
    ],
    ['verb', 'start (a fire).', 'the school had been broken into and the fire ha
    ['verb', '(of blossom or a tree) form into or produce (fruit).', 'wait until
    ['verb', 'sit.', 'the rest of them people just set there goggle-eyed for a m
    ['noun', 'a group or collection of things that belong together or resemble c
    ['noun', 'the way in which something is set, disposed, or positioned.', 'the
    ['noun', 'a radio or television receiver.', 'a TV set'],
    ['noun', 'a collection of scenery, stage furniture, and other articles used
    ['noun', 'an arrangement of the hair when damp so that it dries in the requi
    ['noun', 'a cutting, young plant, or bulb used in the propagation of new pla
    ['noun', 'the last coat of plaster on a wall.', ''],
    ['noun', 'the amount of spacing in type controlling the distance between let
    ['noun', 'variant spelling of sett.', ''],
    ['noun', 'another term for plant (sense 4 of the noun).', ''],
    ['verb', 'group (pupils or students) in sets according to ability.', ''],
    ['adjective', 'fixed or arranged in advance.', 'try to feed the puppy at set
    ['adjective', 'ready, prepared, or likely to do something.', 'the first fami
    ['noun', 'the den or burrow of a badger.', ''],
    ['noun', 'a granite paving block.', ''],
    ['noun', 'the particular pattern of stripes in a tartan.', '']
]
```

Concept Check 2.3

1. True or False: The dictionary key `'partOfSpeech'` can be used interchangeably with `'partofspeech'` (all lower case).
2. Short Answer: What types of standard Python objects appear in the definition of `defns_list`?

1. *False. Dictionary key-lookups are case-sensitive, as are most operations in Python.*

2. *`defns_list` is a nested `list`. Each of the component `list` objects contains `str` objects.*

Section 2.4. Create defns_df

```
# Now that we've finish looping, we can put the definitions in a DataFrame called
defns_df = pandas.DataFrame(defns_list, columns = ['part_of_speech', 'definition',

# We can also inspect the size of defns_df before printing it.
print(f'The size of defn_df: {defns_df.shape}')
print('\n')
display(defns_df)

# Finally, let's look at how often each part of speech occurs.
print('\n')
print('The frequency of each part of speech in defn_df:')
display(defns_df['part_of_speech'].value_counts())
```

Concept Check 2.4

1. True or False: Instead of bothering with a list of lists, `defns_df` could have instead been built row-by-row in Section 2.3.
2. Multiple Choice: If `x = ['partOfSpeech', 'definition', 'example']`, which of these would produce `'example'`?
    A. `x[0]`
    B. `x[1]`
    C. `x[2]`
    D. `x[-1]`

1. *While technically True, this is not recommended for performance reasons. In general, it's best to avoid doing anything with a `DataFrame` inside a loop.*
2. *Options C and D are both correct.*

Section 2.5. Additional Exercises

For practice, we recommend the following:

- Run the code cell below to see example output from the API endpoint https://httpbin.org/json
- Repeat the steps in Sections 2.3-4 to extract the list of items returned by https://httpbin.org/json

```
# Let's call a commonly used open web API for testing web scrapers.
httpbin_org_response = requests.get('https://httpbin.org/json')

# Check the resulting status code to make sure the API call was successful, with 2
if httpbin_org_response.status_code == 200:
    print('API call successful!\n')

# Finally, let's extract and print the JSON-formatted return value.
httpbin_org_json = httpbin_org_response.json()
print('Here\'s the resulting data structure:')
print(httpbin_org_json)
```

```
API call successful!

Here's the resulting data structure:
{
    'slideshow': {
        'author': 'Yours Truly',
        'date': 'date of publication',
        'slides': [
            {'title': 'Wake up to WonderWidgets!', 'type': 'all'},
            {'items': ['Why <em>WonderWidgets</em> are great', 'Who <em>buys</em>
        ],
        'title': 'Sample Slide Show'
    }
}
```

```
httpbin_org_list = []
slideshow = httpbin_org_json['slideshow']
author = slideshow['author']
date = slideshow['date']
slides = slideshow['slides']
for slide in slides:
    httpbin_org_list.append(
        [
            author,
            date,
            slide['title'],
            slide['type'],
        ]
    )

print(httpbin_org_list)
```

```
[['Yours Truly', 'date of publication', 'Wake up to WonderWidgets!', 'all'], ['Y
```

```
httpbin_org_df = pandas.DataFrame(httpbin_org_list, columns = ['author', 'date', '

print(f'The size of defn_df: {httpbin_org_df.shape}')
```

```
print('\n')
display(httpbin_org_df)
```

```
The size of defn_df: (2, 4)
```

	author	date	title	type
0	Yours Truly	date of publication	Wake up to WonderWidgets!	all
1	Yours Truly	date of publication	Overview	all

Notes and Resources

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

1. For more about the `pandas` package, including the methods used above, see the following:
    - https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.shape.html
    - https://pandas.pydata.org/docs/reference/api/pandas.Series.value_counts.html
2. For more about the `requests` package, see https://docs.python-requests.org/
3. For more about some of the Python features used, such as dictionaries, lists, and control flow with if-then-else conditionals and for-loops, we recommend the following chapters of A Whirlwind Tour of Python:
    - https://jakevdp.github.io/WhirlwindTourOfPython/06-built-in-data-structures.html
    - https://jakevdp.github.io/WhirlwindTourOfPython/07-control-flow-statements.html
4. For more information on f-strings (i.e., Python strings like `f'https://httpstatuses.com/{defns_of_set_response.status_code}'`), see https://realpython.com/python-f-strings/.
5. For background on the HTTP Request/Response Cycle, we recommend the following:
    - Brief Overview: https://backend.turing.edu/module2/lessons/how_the_web_works_http
    - Deeper Overview: https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview
    - Summary of HTTP Status Codes: https://httpstatuses.com/
    - Google's Implementation of HTTP Status Code 418: https://www.google.com/teapot
6. For more practice with open web APIs, we recommend looking through https://github.com/public-apis/public-apis and trying to parse the output from http://deckofcardsapi.com/
7. For more about the complexity of parsing JSON in SAS, see https://blogs.sas.com/content/sasdummy/2016/12/02/json-libname-engine-sas/

8. 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
9. 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