Amanda Devine
25 July 2019
SI Carpentries Brown Bag
https://github.com/amdevine/cbb-python-requests (https://github.com/amdevine/cbb-python-requests)

Harvest data from Web APIs using the Python Requests library

This Jupyter notebook walks through the process of using the Python Requests library, in conjunction with the Pandas library, to download data from website REST API services.

As an example dataset, this notebook uses data on the US National Parks available via the United States National Park Service Data API.

Definitions

- **API**: Application Programming Interface. A special page on a website that provides structured data for other programs and applications.
  - **REST**: Newer API format, easily accessed via URL, returns data in a variety of formats including JSON
  - **SOAP**: Older API format, more complex, accepts queries and returns data in XML only
- **GET Request**: An HTTP command to retrieve code and data from a website. GET requests can be made in a variety of ways; the Requests library offers a very easy way to make GET requests from Python.
- **JSON**: JavaScript Object Notation. A common format of structuring data, analogous to a Python dictionary.
- **Base URL**: The website URL for all API data. A variety of endpoints can be added to the base URL.

  > NPS Base URL: `https://developer.nps.gov/api/v1`

- **Endpoint**: The specific URL where the API page can be found. Each website might have multiple endpoints that return different kinds of data.

  > Parks Endpoint: `https://developer.nps.gov/api/v1/parks`

- **Parameter**: An additional criterion that is added to the endpoint to filter data returned. Parameters are usually added to the endpoint with a `?` character, and are in the format of `field=value`. Multiple parameters can be added to an endpoint, separated by a &.

  > parkCode, stateCode, and limit parameters: `https://developer.nps.gov/api/v1/parks?parkCode=yell&stateCode=WY&limit=5`

- **API Key**: A string of characters assigned by the website to identify the user requesting data via the API. For many API services, an API Key is required when making a request.

  > National Parks API Key: `https://developer.nps.gov/api/v1/parks?api_key=1mdaBewB37R0kUA2ZtfA6URe7PeUsig6jLQmSXyx` (not a real key!)

NPS Data API

The National Park Service (NPS) Data API is a service provided by the United States National Park Service to supply data about natural areas managed by the National Park Service. Data available through the API include

- park information
- campground information
- alerts, events, news, and educational resources.

All users of the NPS Data API are required to have an API Key. Keys can be obtained here: https://www.nps.gov/subjects/developer/get-started.htm (https://www.nps.gov/subjects/developer/get-started.htm)

API documentation is available here: https://www.nps.gov/subjects/developer/api-documentation.htm (https://www.nps.gov/subjects/developer/api-documentation.htm)
This documentation shows a list of possible endpoints. Clicking on an endpoint shows the parameters that can be supplied to that endpoint to filter results, as well as an example of the JSON data that will be returned from a GET request.

The NPS Data API GitHub Repository contains examples of using Python and/or PHP to retrieve data. It is available here: https://github.com/nationalparkservice/nps-api-samples (https://github.com/nationalparkservice/nps-api-samples)

Python `requests` library

The Python Requests library provides a simple way to query and retrieve JSON data from API services. It is a wrapper that calls other Python libraries such as `urllib`.

Requests can be download via Terminal/Command Line with `conda install requests` (if you are running Anaconda/Miniconda) or `pip install requests` (if you are running base Python).

Making a GET request to retrieve JSON data with Requests is usually done in the following way:

```
import requests
url = 'https://baseurl.com/endpoint'
params = {
    'field1': 'value1',
    'field2': 'value2',
}
r = requests.get(url, params).json()
```

This creates a dictionary, `r`, from which data can be accessed via key-value pairs.

Parameters supplied to a GET request are automatically encoded to HTML specifications.

> e.g. the parameter `'q': 'César E. Chávez'` is automatically encoded to `q=C%C3%A9sar%20E.%20Ch%C3%A1vez` by Requests

Requests has significantly more functionality and many more options beyond simple GET requests. Quickstart documentation is available here: https://2.python-requests.org/en/master/user/quickstart/ (https://2.python-requests.org/en/master/user/quickstart/)

Setup

Import the `requests` library to retrieve data from the NPS Data API. Import the `pandas` library to work with retrieved data and export as tabular files.

In [1]:
```
import requests
import pandas as pd
import pprint   # Prints dictionaries/JSON in a more human-readable format
```

Assign the API Key to a constant. If the code will be available on GitHub or other public sites, avoid assigning it directly and import it from a local file instead. (And don't add this file to GitHub!)

In [2]:
```
# API_KEY = '1mdaBewB37R0kUA2ZtfA6URe7PeUsig6jLQmSXyx'
with open('api_key_file.txt', 'r') as f:
    API_KEY = f.read().strip()
print("API Key: {}".format("API_KEY")) # Remove quotes to display actual API_KEY
```

```
API Key: API_KEY
```

Make a GET request to the API to retrieve data

Use `requests.get()` to make an **HTTP GET Request** to the API. Any parameters can be provided to requests.get() as a dictionary.

The following request will return data on up to 100 parks in Washington DC, Maryland, and Virginia. `api_key` is a required parameter for all NPS Data API requests. `stateCode` filters parks based on two-letter US state abbreviations. `fields` specifies additional fields to return in addition to the default fields. `limit` specifies the maximum number of results to return.

In [3]:
```
url = 'https://developer.nps.gov/api/v1/parks'
params = {
    'api_key': API_KEY,
    'stateCode': 'DC,MD,VA', # Per the API documentation, separate multiple values with commas
    'fields': 'entranceFees',
    'limit': 100
}
r = requests.get(url, params)
```

`requests.get()` returns a variety of information about the web page retrieved. This info can be useful for troubleshooting.

In [4]:
```
print("The response code is: {}".format(r.status_code))
print("\nThe retrieved URL is: {}".format("r.url")) # Remove quotes to display URL
print("\nThe first 300 characters of the retrieved text are:\n{}".format(r.text[:300]))
```

```
The response code is: 200

The retrieved URL is: r.url

The first 300 characters of the retrieved text are:
{"total":"80","data":[{"states":"DC","entranceFees":[{"cost":"0.0000","description":"No Entrance Fee
to enter park site.","title":"No Entrance Fee"}],"directionsInfo":"The memorial is located at the cor
ner of Vermont Avenue, 10th St, and U Street NW, near the U Street\/African-American Civil War Mem
```

Work with retrieved data

Convert GET request object to dictionary

When the API data are supplied in the **JSON** format, they can easily be turned into a Python dictionary using the Requests `.json()` method.

In [5]:
```
parks_data = r.json()

print("First item in 'data':\n")
pprint.pprint(parks_data['data'][0])
```

```
First item in 'data':

{'description': 'Over 200,000 African-American soldiers and sailors served in '
                'the U.S. Army and Navy during the Civil War. Their service '
                'helped to end the war and free over four million slaves. The '
                'African American Civil War Memorial honors their service and '
                'sacrifice.',
 'designation': '',
 'directionsInfo': 'The memorial is located at the corner of Vermont Avenue, '
                   '10th St, and U Street NW, near the U '
                   'Street/African-American Civil War Memorial/Cardozo Metro '
                   'Station.',
 'directionsUrl': 'http://www.nps.gov/afam/planyourvisit/directions.htm',
 'entranceFees': [{'cost': '0.0000',
                   'description': 'No Entrance Fee to enter park site.',
                   'title': 'No Entrance Fee'}],
 'fullName': 'African American Civil War Memorial',
 'id': '1A47416F-DAA3-4137-9F30-14AF86B4E547',
 'latLong': 'lat:38.916554, long:-77.025977',
 'name': 'African American Civil War Memorial',
 'parkCode': 'afam',
 'states': 'DC',
 'url': 'https://www.nps.gov/afam/index.htm',
 'weatherInfo': 'Washington DC gets to see all four seasons. Humidity will '
                'make the temps feel hotter in summer and colder in winter.\n'
                '\n'
                'Spring (March - May) Temp: Average high is 65.5 degrees with '
                'a low of 46.5 degrees\n'
                '\n'
                'Summer (June - August) Temp: Average high is 86 degrees with '
                'a low of 68.5 degrees\n'
                '\n'
                'Fall (September - November) Temp: Average high is 68 degrees '
                'with a low of 51.5 degrees\n'
                '\n'
                'Winter (December - February) Temp: Average high is 45 degrees '
                'with a low of 30 degrees\n'
                '\n'
                '(Source: www.usclimatedata.com)'}
```

Create a Pandas DataFrame

Pandas DataFrames make it easy to work with the retrieved data in a tabular format.

This code filters the retrieved data to states and associated lat/long coordinate for each park.

In [6]:
```
parks_df = pd.DataFrame(parks_data['data'])
locations_df = parks_df[['parkCode', 'fullName', 'designation', 'states', 'latLong']]
locations_df.head(10)
```

Out[6]:

	parkCode	fullName	designation	states	latLong
0	afam	African American Civil War Memorial		DC	lat:38.916554, long:-77.025977
1	anac	Anacostia Park	Park	DC	lat:38.89644397, long:-76.96314236
2	anti	Antietam National Battlefield	National Battlefield	MD	lat:39.46763452, long:-77.73828017
3	appa	Appalachian National Scenic Trail	National Scenic Trail	CT,GA,MA,MD,ME,NC,NH,NJ,NY,PA,TN,VA,VT,WV	lat:40.41029575, long:-76.4337548
4	apco	Appomattox Court House National Historical Park	National Historical Park	VA	lat:37.38022164, long:-78.79856982
5	arho	Arlington House, The Robert E. Lee Memorial		VA	lat:38.8822021484375, long:-77.0734786987305
6	asis	Assateague Island National Seashore	National Seashore	MD,VA	lat:38.05593172, long:-75.24524611
7	balt	Baltimore National Heritage Area	National Heritage Area	MD	lat:39.2904968261719, long:-76.6284027099609
8	bawa	Baltimore-Washington Parkway	Parkway	MD	lat:39.02604289, long:-76.85410921
9	bepa	Belmont-Paul Women's Equality National Monument	National Monument	DC	lat:38.89231541, long:-77.00381882

Restructure/flatten data

If the results contain nested data that need to be flattened (e.g. multiple `entranceFees` for each park), or the results could be otherwise restructured in a more "tidy" format, a new list of dictionaries can be created by iterating through the data. This list can then be converted to a DataFrame.

JSON data for an individual park's multiple entrance fees:

In [7]:
```
pprint.pprint(parks_data['data'][2])
```

```
{'description': '23,000 soldiers were killed, wounded or missing after twelve '
                'hours of savage combat on September 17, 1862. The Battle of '
                "Antietam ended the Confederate Army of Northern Virginia's "
                'first invasion into the North and led Abraham Lincoln to '
                'issue the preliminary Emancipation Proclamation.',
 'designation': 'National Battlefield',
 'directionsInfo': 'Ten miles south of I-70 on Maryland Route 65',
 'directionsUrl': 'http://www.nps.gov/anti/planyourvisit/directions.htm',
 'entranceFees': [{'cost': '7.0000',
                   'description': '3 day pass - $7.00 per bike or motorcycle \n'
                                  'This is the entry fee to the battlefield '
                                  'proper, museum, movie, and ranger programs.',
                   'title': 'Antietam National Battlefield Entrance Fee'},
                  {'cost': '15.0000',
                   'description': '3 day vehicle pass.  This pass covers '
                                  'everyone in a vehicle, ie. family.  The '
                                  'pass covers entry to the battlefield '
                                  'proper, museum, movie, and ranger programs.',
                   'title': 'Antietam National Battlefield Entrance Fee'}],
 'fullName': 'Antietam National Battlefield',
 'id': '8415526C-C932-4236-A634-2D89DF718936',
 'latLong': 'lat:39.46763452, long:-77.73828017',
 'name': 'Antietam',
 'parkCode': 'anti',
 'states': 'MD',
 'url': 'https://www.nps.gov/anti/index.htm',
 'weatherInfo': 'The weather is fairly mild.  Summers can be very warm and '
                'humid and winters cold and snowy.  We have four distinct '
                'seasons with the fall and spring being the best times to '
                'visit the battlefield.'}
```

For each park in the dataset, and for each entrance fee in that park, add some park and fee values as a dictionary to a new `entry_fee_data` list.

In [8]:
```
entry_fees_data = []

for park in parks_data['data']:
    for fee in park['entranceFees']:
        entry_fees_data.append({
            'parkCode': park['parkCode'],
            'fullName': park['fullName'],
            'designation': park['designation'],
            'fee_usd': fee['cost'],
            'fee_type': fee['title'],
            'fee_description': fee['description']
        })

pprint.pprint(entry_fees_data[2:4])
```

```
[{'designation': 'National Battlefield',
  'fee_description': '3 day pass - $7.00 per bike or motorcycle \n'
                     'This is the entry fee to the battlefield proper, museum, '
                     'movie, and ranger programs.',
  'fee_type': 'Antietam National Battlefield Entrance Fee',
  'fee_usd': '7.0000',
  'fullName': 'Antietam National Battlefield',
  'parkCode': 'anti'},
 {'designation': 'National Battlefield',
  'fee_description': '3 day vehicle pass.  This pass covers everyone in a '
                     'vehicle, ie. family.  The pass covers entry to the '
                     'battlefield proper, museum, movie, and ranger programs.',
  'fee_type': 'Antietam National Battlefield Entrance Fee',
  'fee_usd': '15.0000',
  'fullName': 'Antietam National Battlefield',
  'parkCode': 'anti'}]
```

Convert `entry_fee_data` to a DataFrame

In [9]:
```
entry_fees_df = pd.DataFrame(entry_fees_data)
entry_fees_df = entry_fees_df[['parkCode', 'fullName', 'designation', 'fee_usd', 'fee_type']]
entry_fees_df['fee_usd'] = entry_fees_df['fee_usd'].astype(float)
entry_fees_df.head(10)
```

Out[9]:

	parkCode	fullName	designation	fee_usd	fee_type
0	afam	African American Civil War Memorial		0.0	No Entrance Fee
1	anac	Anacostia Park	Park	0.0	Entrance Fees
2	anti	Antietam National Battlefield	National Battlefield	7.0	Antietam National Battlefield Entrance Fee
3	anti	Antietam National Battlefield	National Battlefield	15.0	Antietam National Battlefield Entrance Fee
4	appa	Appalachian National Scenic Trail	National Scenic Trail	0.0	Appalachian National Scenic Trail Entrance Fee
5	apco	Appomattox Court House National Historical Park	National Historical Park	0.0	Entrance Fee
6	arho	Arlington House, The Robert E. Lee Memorial		0.0	No Fee
7	asis	Assateague Island National Seashore	National Seashore	20.0	Assateague 7 day per vehicle pass
8	asis	Assateague Island National Seashore	National Seashore	20.0	Chincoteague National Wildlife Refuge Weekly Pass
9	balt	Baltimore National Heritage Area	National Heritage Area	0.0	Baltimore National Heritage Area

Export data as a tabular file

Pandas DataFrames have a method, `.to_csv()`, that allows them to be exported as a CSV or TSV file. This file can be imported into another program for further analysis.

CSV file: `df_name.to_csv('output_file_name.csv', index=False)`

TSV file: `df_name.to_csv('output_file_name.tsv', sep='\t', index=False)`

In [10]:
```
locations_df.to_csv('parks_data.tsv', sep='\t', index=False)
entry_fees_df.to_csv('parks_entry_fees.tsv', sep='\t', index=False)
```

Additional API Resources

Full **Requests documentation**: https://2.python-requests.org/en/master (https://2.python-requests.org/en/master)

data.gov is a repository of data produced by US Federal and State Government departments. In addition to data files, data.gov also provides a list of API services for Federal data (https://catalog.data.gov/dataset?res_format=API (https://catalog.data.gov/dataset?res_format=API)).

Programmable Web (https://www.programmableweb.com/ (https://www.programmableweb.com/)) is a website that aggregates information on APIs provided by governmental and private organizations. This can be a good resource for locating APIs of interest.