Web Scraping the SEC

The U.S. Securities and Exchange Commission (SEC) is an independent federal government agency responsible for protecting investors, maintaining fair and orderly functioning of securities markets and facilitating capital formation. Part of their job is to house financial disclosures that companies submit for a wide range of activities.

In this tutorial, we will explore how to web scrape the SEC using their public database. If you would like more information about the SEC and their data sources, I encourage you to visit the SEC's website directly. Here is the link for your reference: **https://www.sec.gov/edgar/searchedgar/accessing-edgar-data.htm (https://www.sec.gov/edgar/searchedgar/accessing-edgar-data.htm)**

The link above will give an overview of the EDGAR (Electronic Data Gathering, Analysis, and Retrieval system) which allows the complete access to filings of individuals, funds, and businesses. Additionally, my series on YouTube that covers this topic can be found here: **https://www.youtube.com/playlist?list=PLcFcktZ0wnNkOo9FQ2wrDcsV0jYqEYu1z (https://www.youtube.com/playlist?list=PLcFcktZ0wnNkOo9FQ2wrDcsV0jYqEYu1z)**

In [1]:
```
# import our libraries
import requests
import urllib
from bs4 import BeautifulSoup
```

Defining a URL Builder

To access the different archives and filings in EDGAR, we will need to request different URLs. However, while some of these URLs we can define beforehand others we will have to build from scratch. To make the process of building URLs consistent and straightforward, we will set a function that takes a base URL & a list of components and creates a URL from them.

In [2]:
```
# let's first make a function that will make the process of building a url eas
y.
def make_url(base_url , comp):

    url = base_url

    # add each component to the base url
    for r in comp:
        url = '{}/{}'.format(url, r)

    return url

# EXAMPLE
base_url = r"https://www.sec.gov/Archives/edgar/data"
components = ['886982','000156459019011378', '0001564590-19-011378-index-heade
rs.html']
make_url(base_url, components)
```

Out[2]:
```
'https://www.sec.gov/Archives/edgar/data/886982/000156459019011378/0001564590
-19-011378-index-headers.html'
```

Overview of the Archive Structure

Before we get to the actual pulling of data, it pays dividends to understand the overall structure of the archive. At this point, it only helps to know three of the main directories:

1. **Index**
2. **Feed**
3. **OldLoad**

Each of the directories has very similar information, but it organized differently. The first directory that we will talk about and use primarily in this tutorial is the **index** directory. The index directory indexes to all public filings are available from 1994Q3 through the present. It can further be broken down into two sub-directories:

1. **Archives/edgar/daily-index —** Daily index files through the current year.
2. **Archives/edgar/full-index —** Full indexes offer a "bridge" between quarterly and daily indexes, compiling filings from the beginning of the current quarter through the previous business day. At the end of the quarter, the full index is rolled into a static quarterly index.

The other directory we will talk about is the **Feed** directory which contains tar and gzips archive files (e.g., 20061207.nc.tar.gz) for each filing day. The third directory is the **OldLoad** directory which contains daily concatenated archive files of all public filing submissions complete with the filing header. Each directory and all child subdirectories contain three files to assist in automated crawling of these directories. **(Note that these are not visible through directory browsing.)**

1. **index.html** - The web browser would generally receive these.
2. **index.xml** - A XML structured version of the same content.
3. **index.json** - A JSON structured vision of the same material.

Pulling the documents for a single filing for a single company

If we want to pull all the records for a single filing, the process is simple. We pass through the company's CIK number; this will define the company we want to search. Once we establish a CIK number, we can request the filings for that company. Remember, that when we request the filings, we will get all the filings for that company sorted by the order in which they were filed. Older companies will have more filing than newer companies, and this can make the process seem a little daunting for older companies, but we have tools are our disposal to narrow the search.

If we need to, we can filter the filings to only a specific time range, but this will require that we filter the URLs that only contain those dates. Unfortunately, we will be able to select a particular year. If you look at the end of the file extension **(0001564590-19-011378)**, we come across this number which is called the accession number.

I'll be quoting directly from the SEC to define this number. In the example above, **"0001564590-19-011378"** is the "accession number," a unique identifier assigned automatically to an accepted submission by the EDGAR Filer System. The first set of numbers (0001564590) is the **CIK of the entity submitting the filing.**

This could be the company or a third-party filer agent. Some filer agents without a regulatory requirement to make disclosure filings with the SEC have a CIK but no searchable presence in the public EDGAR database. **The next two numbers (19) represent the year.** The last series of numbers represents a **sequential count of**

In [3]:
```
# define a base url, this would be the EDGAR data Archives
base_url = r"https://www.sec.gov/Archives/edgar/data"

# define a company to search (GOLDMAN SACHS), this requires a CIK number that
 is defined by the SEC.
cik_num = '886982'

# let's get all the filings for Goldman Sachs in a json format.
# Alternative is .html & .xml
filings_url = make_url(base_url, [cik_num, 'index.json'])

# Get the filings and then decode it into a dictionary object.
content = requests.get(filings_url)
decoded_content = content.json()

# Get a single filing number, this way we can request all the documents that w
ere submitted.
filing_number = decoded_content['directory']['item'][0]['name']

# define the filing url, again I want all the data back as JSON.
filing_url = make_url(base_url, [cik_num, filing_number, 'index.json'])

# Get the documents submitted for that filing.
content = requests.get(filing_url)
document_content = content.json()

# get a document name
for document in document_content['directory']['item']:
    if document['type'] != 'image2.gif':
        document_name = document['name']
        filing_url = make_url(base_url, [cik_num, filing_number, document_name
])
        print(filing_url)
```

```
https://www.sec.gov/Archives/edgar/data/886982/000156459019011587/0001564590-
19-011587-index-headers.html
https://www.sec.gov/Archives/edgar/data/886982/000156459019011587/0001564590-
19-011587-index.html
https://www.sec.gov/Archives/edgar/data/886982/000156459019011587/0001564590-
19-011587.txt
https://www.sec.gov/Archives/edgar/data/886982/000156459019011587/gs-424b2.ht
m
```

Pulling all the documents for all the filings for a single company

If we want to pull all the records for all the filings, the process is very similar. All we are going to do is a loop through all the filings instead of just grabbing one. Remember, that there can be many filings for a single company so you may get back more than you intend.

In [24]:
```
# define a base url, this would be the EDGAR data Archives
base_url = r"https://www.sec.gov/Archives/edgar/data"

# define a company to search (GOLDMAN SACHS), this requires a CIK number that
 is defined by the SEC.
cik_num = '886982'

# let's get all the filings for Goldman Sachs in a json format.
# Alternative is .html & .xml
filings_url = make_url(base_url, [cik_num, 'index.json'])

# Get the filings and then decode it into a dictionary object.
content = requests.get(filings_url)
decoded_content = content.json()

# Get a filing number, this way we can request all the documents that were sub
mitted.
# HERE I AM JUST GRABBING CERTAIN FILINGS FOR READABILITY REMOVE [3:5] TO GRAB
THEM ALL.
for filing_number in decoded_content['directory']['item'][3:5]:

    filing_num = filing_number['name']
    print('-'*100)
    print('Grabbing filing : {}'.format(filing_num))

    # define the filing url, again I want all the data back as JSON.
    filing_url = make_url(base_url, [cik_num, filing_num, 'index.json'])

    # Get the documents submitted for that filing.
    content = requests.get(filing_url)
    document_content = content.json()

    # get a document name
    for document in document_content['directory']['item']:
        document_name = document['name']
        filing_url = make_url(base_url, [cik_num, filing_num, document_name])
        print(filing_url)
```

```
--------------------------------------------------------------------------------
----------------------
Grabbing filing : 000156459019011533
https://www.sec.gov/Archives/edgar/data/886982/000156459019011533/0001564590-
19-011533-index-headers.html
https://www.sec.gov/Archives/edgar/data/886982/000156459019011533/0001564590-
19-011533-index.html
https://www.sec.gov/Archives/edgar/data/886982/000156459019011533/0001564590-
19-011533.txt
https://www.sec.gov/Archives/edgar/data/886982/000156459019011533/gbtfcyx5zdi
f000001.jpg
https://www.sec.gov/Archives/edgar/data/886982/000156459019011533/gbtfcyx5zdi
f000002.jpg
https://www.sec.gov/Archives/edgar/data/886982/000156459019011533/gbtfcyx5zdi
f000003.jpg
https://www.sec.gov/Archives/edgar/data/886982/000156459019011533/gbtfcyx5zdi
f000004.jpg
https://www.sec.gov/Archives/edgar/data/886982/000156459019011533/gbtfcyx5zdi
f000005.jpg
https://www.sec.gov/Archives/edgar/data/886982/000156459019011533/gs-424b2.ht
m
--------------------------------------------------------------------------------
----------------------
Grabbing filing : 000156459019011530
https://www.sec.gov/Archives/edgar/data/886982/000156459019011530/0001564590-
19-011530-index-headers.html
https://www.sec.gov/Archives/edgar/data/886982/000156459019011530/0001564590-
19-011530-index.html
https://www.sec.gov/Archives/edgar/data/886982/000156459019011530/0001564590-
19-011530.txt
https://www.sec.gov/Archives/edgar/data/886982/000156459019011530/gbabbnbzlvv
a000001.jpg
https://www.sec.gov/Archives/edgar/data/886982/000156459019011530/gbabbnbzlvv
a000002.jpg
https://www.sec.gov/Archives/edgar/data/886982/000156459019011530/gbabbnbzlvv
a000003.jpg
https://www.sec.gov/Archives/edgar/data/886982/000156459019011530/gbabbnbzlvv
a000004.jpg
https://www.sec.gov/Archives/edgar/data/886982/000156459019011530/gbabbnbzlvv
a000005.jpg
https://www.sec.gov/Archives/edgar/data/886982/000156459019011530/gbabbnbzlvv
a000006.jpg
https://www.sec.gov/Archives/edgar/data/886982/000156459019011530/gs-424b2.ht
m
```

Pulling the daily index filings

The Daily-Index endpoint will return all the filings for a given year. When I work with the daily-index archive I try to break the process into a few key steps:

- Define the Year we want.
- Define the Quarter we want.
- Define the day we want.

All this process does is assemble a simple URL into a more complex URL. Notice how I take the `base_url` and tack on the different folder names? I get the folder names by looping through each of the items in the `['directory']['item']` list and grabbing the value assigned to the `name` key.

There are a couple of things I want you to take away from the original file URLs. The first thing you will notice is that there are four main types of file and for the most part all the information is identical in the data, just sorted differently. Each of them can be defined in the following way:

- **Company —** Sorted by company name
- **Form —** Sorted by form type
- **Master —** Sorted by CIK number
- **XBRL —** List of submissions containing XBRL financial files, sorted by CIK number; these include Voluntary Filer Program submissions

I Also want to let you know that there is a **crawler** file and a **sitemap** file. Out of these two files, the sitemap is the only valuable one because it contains a more structured list of all the files submitted for any given day. Where each file can be found in a `url` tag:

```
<url>
    <loc>http://www.sec.gov/Archives/edgar/data/4977/0000004977-19-000003-index.htm</loc>
    <lastmod>2019-01-02</lastmod>
    <changefreq>never</changefreq>
    <priority>0.5</priority>
</url>
```

For more information, please visit the documentation provided by the SEC.
https://www.sec.gov/edgar/searchedgar/accessing-edgar-data.htm
(https://www.sec.gov/edgar/searchedgar/accessing-edgar-data.htm)

In [5]:
```
# define the urls needed to make the request, let's start with all the daily f
ilings
base_url = r"https://www.sec.gov/Archives/edgar/daily-index"

# The daily-index filings, require a year and content type (html, json, or xm
l).
year_url = make_url(base_url, ['2019', 'index.json'])

# Display the new Year URL
print('-'*100)
print('Building the URL for Year: {}'.format('2019'))
print("URL Link: " + year_url)

# request the content for 2019, remember that a JSON strucutre will be sent ba
ck so we need to decode it.
content = requests.get(year_url)
decoded_content = content.json()

# the structure is almost identical to other json requests we've made. Go to t
he item list.
# AGAIN ONLY GRABBING A SUBSET OF THE FULL DATASET
for item in decoded_content['directory']['item'][0:1]:

    # get the name of the folder
    print('-'*100)
    print('Pulling url for Quarter: {}'.format(item['name']))

    # The daily-index filings, require a year, a quarter and a content type (h
tml, json, or xml).
    qtr_url = make_url(base_url, ['2019', item['name'], 'index.json'])

    # print out the url.
    print("URL Link: " + qtr_url)

    # Request, the new url and again it will be a JSON structure.
    file_content = requests.get(qtr_url)
    decoded_content = file_content.json()

    print('-'*100)
    print('Pulling files')

    # for each file in the directory items list, print the file type and file
 href.
    # AGAIN DOING A SUBSET
    for file in decoded_content['directory']['item'][0:10]:

        file_url = make_url(base_url, ['2019', item['name'], file['name']])
        print("File URL Link: " + file_url)
```

```
--------------------------------------------------------------------------------
----------------------
Building the URL for Year: 2019
URL Link: https://www.sec.gov/Archives/edgar/daily-index/2019/index.json
--------------------------------------------------------------------------------
----------------------
Pulling url for Quarter: QTR1
URL Link: https://www.sec.gov/Archives/edgar/daily-index/2019/QTR1/index.json
--------------------------------------------------------------------------------
----------------------
Pulling files
File URL Link: https://www.sec.gov/Archives/edgar/daily-index/2019/QTR1/compa
ny.20190102.idx
File URL Link: https://www.sec.gov/Archives/edgar/daily-index/2019/QTR1/compa
ny.20190103.idx
File URL Link: https://www.sec.gov/Archives/edgar/daily-index/2019/QTR1/compa
ny.20190104.idx
File URL Link: https://www.sec.gov/Archives/edgar/daily-index/2019/QTR1/compa
ny.20190107.idx
File URL Link: https://www.sec.gov/Archives/edgar/daily-index/2019/QTR1/compa
ny.20190108.idx
File URL Link: https://www.sec.gov/Archives/edgar/daily-index/2019/QTR1/compa
ny.20190109.idx
File URL Link: https://www.sec.gov/Archives/edgar/daily-index/2019/QTR1/compa
ny.20190110.idx
File URL Link: https://www.sec.gov/Archives/edgar/daily-index/2019/QTR1/compa
ny.20190111.idx
File URL Link: https://www.sec.gov/Archives/edgar/daily-index/2019/QTR1/compa
ny.20190114.idx
File URL Link: https://www.sec.gov/Archives/edgar/daily-index/2019/QTR1/compa
ny.20190115.idx
```

Parsing the master IDX file

Out of all the files to parse, I find the Master.idx file the easiest because it is possible to separate each field by a delimiter. Whereas the other files do not offer such a delimiter or they lack the additional detail that is provided by master file. With that being said, if I had to choose a second file to parse, it would probably be the sitemap file because of the provided structure in it.

The first thing is to load the information to a text file so that way you don't have to make a second request and not burden the server. After we create a new text file with the content, we can reload it into by opening the text file. From here, I usually encourage people to explore the data before they perform any parsing. We will notice right away that getting the info may be a little challenging, but it can be done.

The approach that I laid out below worked for most files I encountered, but I cannot guarantee it will work for all of them. As time goes on you, have more detailed data so parsing the dataset will become more comfortable.

In [6]:
```
# define a url, in this case I'll just take one of the urls up above.
file_url = r"https://www.sec.gov/Archives/edgar/daily-index/2019/QTR2/master.20190401.idx"

# request that new content, this will not be a JSON STRUCTURE!
content = requests.get(file_url).content

# we can always write the content to a file, so we don't need to request it again.
with open('master_20190102.txt', 'wb') as f:
    f.write(content)
```

In [7]:
```
# let's open it and we will now have a byte stream to play with.
with open('master_20190102.txt','rb') as f:
     byte_data = f.read()

# Now that we loaded the data, we have a byte stream that needs to be decoded
 and then split by double spaces.
data = byte_data.decode("utf-8").split('  ')

# We need to remove the headers, so look for the end of the header and grab i
t's index
for index, item in enumerate(data):
    if "ftp://ftp.sec.gov/edgar/" in item:
        start_ind = index

# define a new dataset with out the header info.
data_format = data[start_ind + 1:]

master_data = []

# now we need to break the data into sections, this way we can move to the fin
al step of getting each row value.
for index, item in enumerate(data_format):

    # if it's the first index, it won't be even so treat it differently
    if index == 0:
        clean_item_data = item.replace('\n','|').split('|')
        clean_item_data = clean_item_data[8:]
    else:
        clean_item_data = item.replace('\n','|').split('|')

    for index, row in enumerate(clean_item_data):

        # when you find the text file.
        if '.txt' in row:

            # grab the values that belong to that row. It's 4 values before an
d one after.
            mini_list = clean_item_data[(index - 4): index + 1]

            if len(mini_list) != 0:
                mini_list[4] = "https://www.sec.gov/Archives/" + mini_list[4]
                master_data.append(mini_list)

# grab the first three items
master_data[:3]
```

```
Out[7]: [['1236397',
          'BRADBURY DANIEL',
          '4',
          '20190401',
          'https://www.sec.gov/Archives/edgar/data/1236397/0000886744-19-000047.tx
        t'],
         ['1236458',
          'WILLIAMS PAUL S',
          '4',
          '20190401',
          'https://www.sec.gov/Archives/edgar/data/1236458/0001227654-19-000074.tx
        t'],
         ['1237789',
          'BLAIR DONALD W',
          '4',
          '20190401',
          'https://www.sec.gov/Archives/edgar/data/1237789/0001127602-19-013788.tx
        t']]
```

Creating our Document Dictionary

An extra step we can take is converting our master list of data into a list of dictionaries, where each dictionary represents a single filing document. This way we can quickly iterate over the master list to grab the data we need. This structure will help us down the road when we need to access only some aspects of information. I encourage individuals to put the time up in front to ensure a quick and easy process for the bulk of the parsing.

In [8]:
```
# loop through each document in the master list.
for index, document in enumerate(master_data):

    # create a dictionary for each document in the master list
    document_dict = {}
    document_dict['cik_number'] = document[0]
    document_dict['company_name'] = document[1]
    document_dict['form_id'] = document[2]
    document_dict['date'] = document[3]
    document_dict['file_url'] = document[4]

    master_data[index] = document_dict
```

Filtering by File Type

Naturally, we might not need all the files that we have scraped, so let's explore how to filter the data. If we want to grab all the 10-K filings from the dataset, we loop through our `master_data` list and only print the ones where the `form_id` has a value of `10-K`. In the example below, I only loop through the first 100 dictionaries for readability.

Finally, I do some extra transformation on the final document URL to set the stage for the next tutorial. The beautiful thing about these document URLs is that with a few other transformations we can now get to that particular company filing archive.

If you would like more info on the different financial forms that we have access to, I encourage you to visit **https://www.sec.gov/forms (https://www.sec.gov/forms)** for more info.

In [20]:
```
# by being in a dictionary format, it'll be easier to get the items we need.
for document_dict in master_data[0:100]:

    # if it's a 10-K document pull the url and the name.
    if document_dict['form_id'] == '10-K':

        # get the components
        comp_name = document_dict['company_name']
        docu_url = document_dict['file_url']

        print('-'*100)
        print(comp_name)
        print(docu_url)

# Create a url that takes us to the Detail filing landing page
file_url_adj = docu_url.split('.txt')
file_url_archive = file_url_adj[0] + '-index.htm'

print('-'*100)
print('The Filing Detail can be found here: {}'.format(file_url_archive))

# Create a url that will take us to the archive folder
archive_url = docu_url.replace('.txt','').replace('-','')

print('-'*100)
print('The Archive Folder can be found here: {}'.format(archive_url))

# Create a url that will take us the Company filing Archive
company_url =docu_url.rpartition('/')

print('-'*100)
print('The Company Archive Folder can be found here: {}'.format(company_url[0
]))
```

```
--------------------------------------------------------------------------------
----------------------
GENERAL STEEL HOLDINGS INC
https://www.sec.gov/Archives/edgar/data/1239188/0001144204-19-017485.txt
--------------------------------------------------------------------------------
----------------------
COMMONWEALTH INCOME & GROWTH FUND V
https://www.sec.gov/Archives/edgar/data/1253347/0001654954-19-003881.txt
--------------------------------------------------------------------------------
----------------------
Joway Health Industries Group Inc
https://www.sec.gov/Archives/edgar/data/1263364/0001213900-19-005388.txt
--------------------------------------------------------------------------------
----------------------
The Filing Detail can be found here: https://www.sec.gov/Archives/edgar/data/
1263364/0001213900-19-005388-index.htm
--------------------------------------------------------------------------------
----------------------
The Archive Folder can be found here: https://www.sec.gov/Archives/edgar/dat
a/1263364/000121390019005388
--------------------------------------------------------------------------------
----------------------
The Company Archive Folder can be found here: https://www.sec.gov/Archives/ed
gar/data/1263364
```