3.THE SCRAPING TOOL

The primary objective of the project was to create a list of companies operating in a market and prepare an automated scraping tool to extract real-time key data from 10-K and 10-Q filings from EDGAR, Bloomberg Terminal, PROWESS, MCA etc. For this objective, we did a thorough research of various sectors that Industry ARC deals in and finally opted for the Chemical Sector for this. And as our source for financial data, we chose EDGAR, it being the most popular and more user friendly compared to the other platforms. So this tool can be used to extract data from EDGAR. We have made it more universal in the sense that it can be used to extract data from any sector, by entering the **Standard Industrial Classification(SIC) Code**.

3.1 Prerequisites

- Latest (stable) Python version installed into the system.
- Jupyter Notebook installed in the same directory as Python.
- Required Python libraries – requests, beautifulsoup, pandas, re, urllib

3.2 Installation

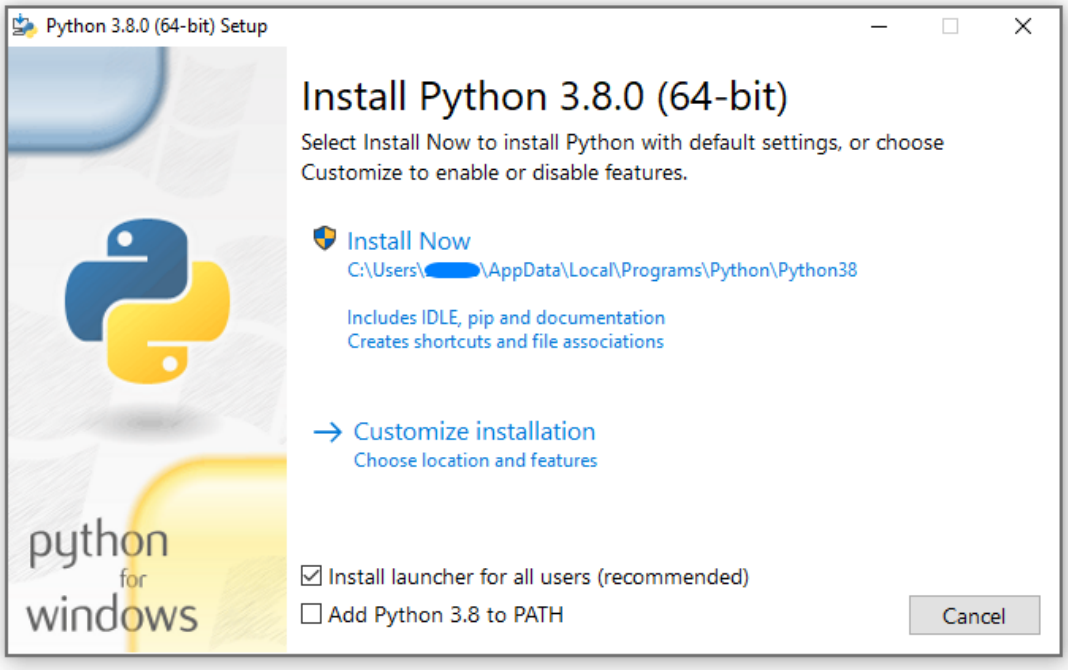
3.2.1 PYTHON (Latest Stable Version)

Python 3.8.3 is the latest running version of python for both Windows and UNIX based operating system. Since we have built the project in Windows and it is more popular, we'll provide the step by step installation guide for Windows.

(In case someone wants to check out how to install python in UNIX systems such as Ubuntu, refer : https://docs.python.org/3.8/using/unix.html)

Go to https://www.python.org/downloads/ and click on “Download Python 3.8.3”

After starting the installer, one of two options may be selected:



If you select “Install Now”:

- You will *not* need to be an administrator(unless a system update for the C Runtime Library is required or you install the Python Launcher for Windows for all users)

- Python will be installed into your user directory

- The Python Launcher for Windows will be installed according to the option at the bottom of the first page
- The standard library, test suite, launcher and pip will be installed
- If selected, the install directory will be added to your PATH
- Shortcuts will only be visible for the current user

Selecting "Customize installation" will allow you to select the features to install, the installation location and other options or post-install actions. To install debugging symbols or binaries, you will need to use this option.

To perform an all-users installation, you should select "Customize installation". In this case:

- You may be required to provide administrative credentials or approval
- Python will be installed into the Program Files directory
- The Python Launcher for Windows will be installed into the Windows directory
- Optional features may be selected during installation
- The standard library can be pre-compiled to bytecode
- If selected, the install directory will be added to the system PATH
- Shortcuts are available for all users

It is recommended to install normally as suggested by the installer. Make sure to **select the option "Add Python 3.8 to PATH"** for avoiding usage issues in the future.
Python 3.8 will be installed to your default user directory.

For an in-depth tutorial to use Python, visit https://docs.python.org/3.8/tutorial/index.html

3.2.2 JUPYTER NOTEBOOK

For an easy installation guide, please visit :
https://www.youtube.com/watch?v=DVahWSqQaAc

If you use pip, you can install it with:

```
pip install notebook
```

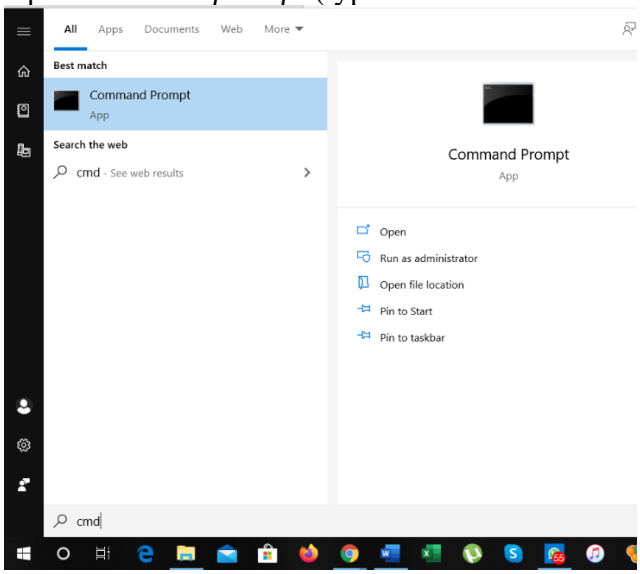
Congratulations, you have installed Jupyter Notebook! To run the notebook, run the following command at the Terminal (Mac/Linux) or Command Prompt (Windows):

```
jupyter notebook
```

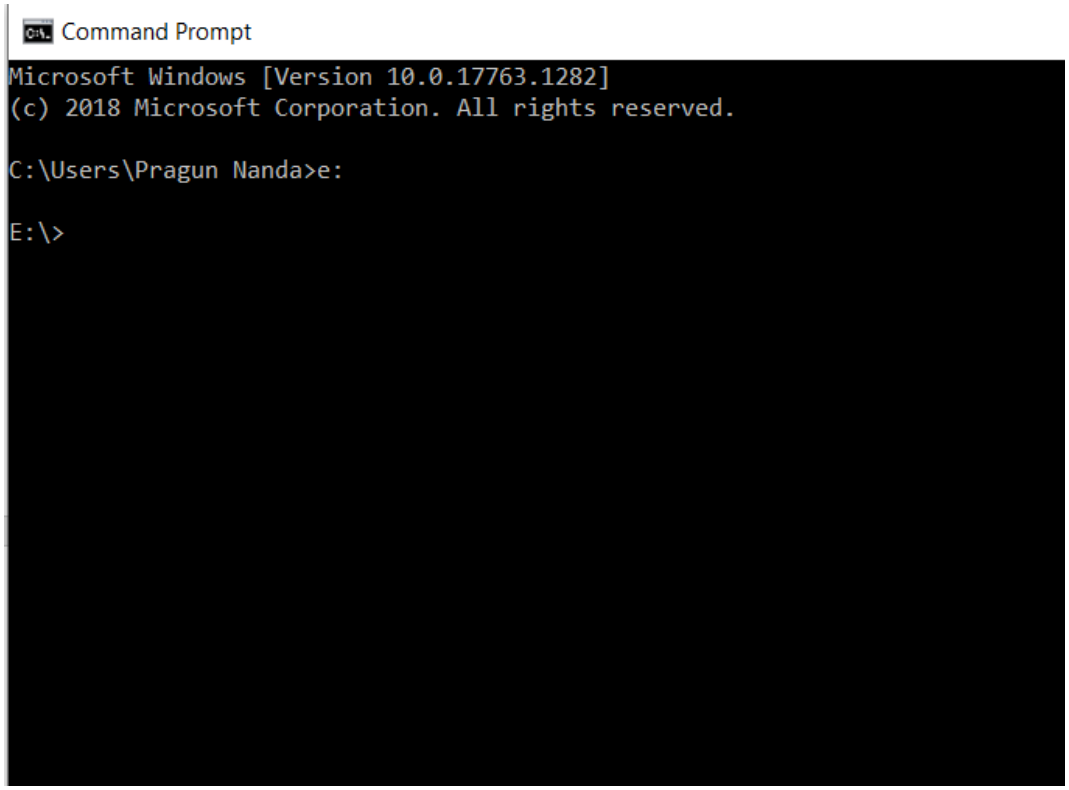
See Running the Notebook for more details.

Whenever you want to use the scraping tool

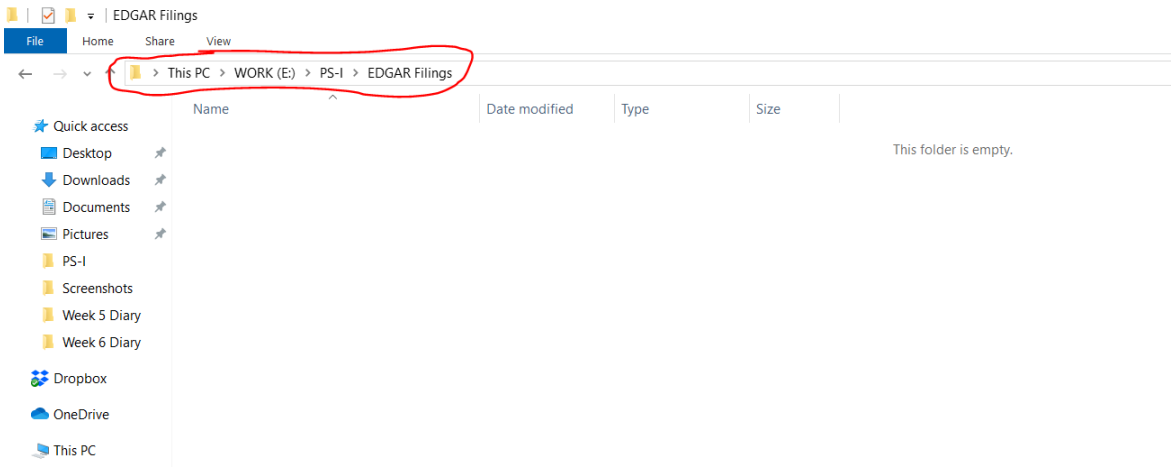
1. Open *command prompt* (type “cmd” in Start Menu).



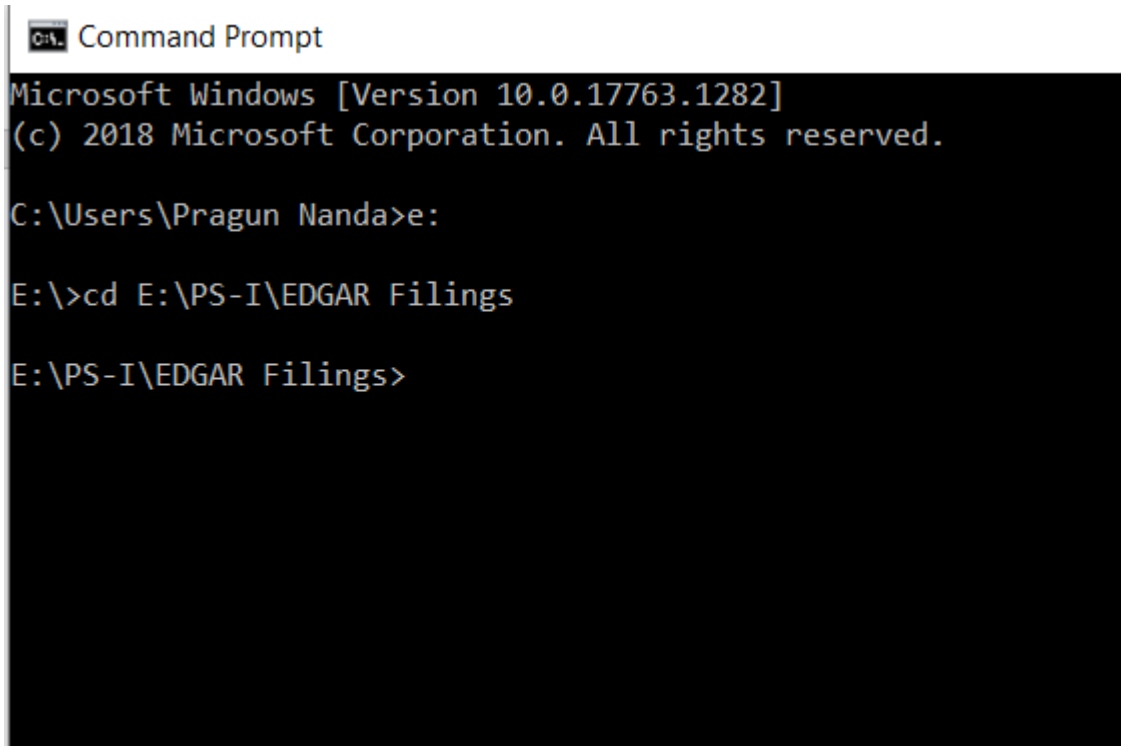
2. Select the drive in which you want to open jupyter notebook and save the files. For this, type the name of the drive and press enter. For example “e:”.



3. Open the folder where you want to download the filings. Click on the address bar and select and copy the address of the folder. For example : E:\PS-I\EDGAR Filings



4. Now open the command prompt and type “cd” and “right-click” from your mouse to paste the copied address there and press enter. NOTE: Do not use keyboard shortcuts, as they have other functions in command prompt and can close your application. Right click and paste.



5. Now to open the jupyter notebook, simply type “jupyter notebook” and press enter.

Starting the Notebook Server

After you have installed the Jupyter Notebook on your computer, you are ready to run the notebook server. You can start the notebook server from the command line (using Terminal on Mac/Linux, Command Prompt on Windows) by running:

```
jupyter notebook
```

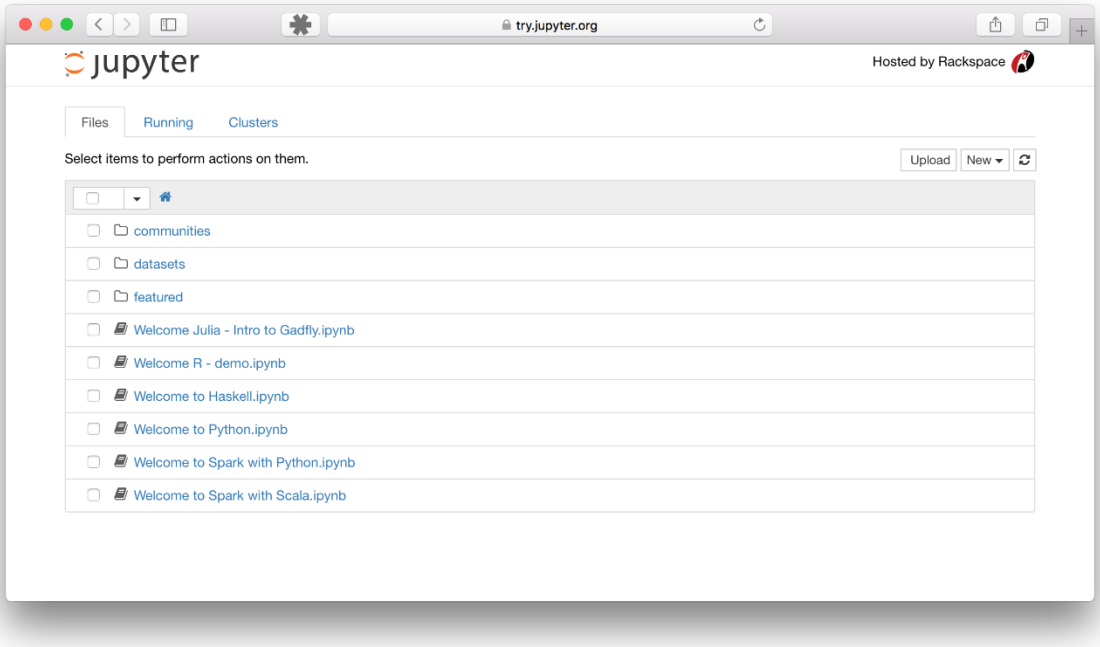
This will print some information about the notebook server in your terminal, including the URL of the web application (by default, http://localhost:8888):

```
$ jupyter notebook

[I 08:58:24.417 NotebookApp] Serving notebooks from local directory: /Users/catherine

[I 08:58:24.417 NotebookApp] 0 active kernels

[I 08:58:24.417 NotebookApp] The Jupyter Notebook is running at: http://localhost:8888/

[I 08:58:24.417 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
```

It will then open your default web browser to this URL.

When the notebook opens in your browser, you will see the Notebook Dashboard, which will show a list of the notebooks, files, and subdirectories in the directory where the notebook server was started. Most of the time, you will wish to start a notebook server in the highest level directory containing notebooks. Often this will be your home directory.

Notebook Dashboard



To open a specific Notebook

The following code should open the given notebook in the currently running notebook server, starting one if necessary. Example : for a file named “notebook.ipynb”

```
jupyter notebook notebook.ipynb
```

3.3 PARSING COMPANY 10Ks and 10Qs from the SEC

3.3.1 Import the libraries

This module will require five libraries- the first is the requests library for making the URL requests; bs4 to parse the files and content; pandas which will be used for taking our cleaned data and giving it structure; re libraries to take care of regular expression; and finally urllib library to fetch, read, open and download from URLs.

```
# import our libraries
import requests
import pandas as pd
from bs4 import BeautifulSoup
import re
import urllib
```

3.3.2 Web Scraping from the SEC Query Page

1. The company search page allows us to make a specific query for a single company and their filings, and this page will then return all the documents related to that company. From here, we can filter all of their documents to the ones that meet our criteria.

2. This includes being able to filter by a specific date or even a particular type of form. Once, we've filtered the results we can go directly to the document or if we want we can go to the filing folder containing that document. One thing to keep in mind is the scope of your search. If you search for a company name, you can get back more than one company back.

3. This usually doesn't present a problem, but it does mean you may have to look through multiple companies to find the documents you want. It might make more sense to search by the CIK number, to get to the company you want.

4. Link to the company search page: https://www.sec.gov/edgar/searchedgar/companysearch.html

3.3.3 Define the Parameters of the Search

To create a search we need to "build" a URL that takes us to a valid results query, this requires taking our base endpoint and attaching on different parameters to help narrow down our search. I'll do my best to explain how each of these parameters works, but unfortunately, there is no formal documentation on this.

Endpoint The endpoint for our EDGAR query is https://www.sec.gov/cgi-bin/browse-edgar if you go to this link without any additional parameters it will be an invalid request.

3.3.4 Parameters

action: (required) By default should be set to getcompany. CIK: (required) Is the CIK number of the company you are searching. type: (optional) Allows filtering the type of form. For example, if set to 10-k only the 10-K filings are returned. dateb: (optional) Will only return the filings before a given date. The format is as follows YYYYMMDD owner: (required) Is set to exclude by default and specifies ownership. You may also set it to include and only. start: (optional) Is the starting index of the results. For example, if I have 100 results but want to start at 45 of 100, I would pass 45. state: (optional) The company's state. filenum: (optional) The filing number. sic: (optional) The company's SIC (Standard Industry Classification) identifier output: (optional) Defines returned data structure as either xml (atom) or normal html. count: (optional) The number of results you want to see with your request, the max is 100 and if not set it will default to 40. Now that we understand all the parameters let's make a request by defining our endpoint, and then a dictionary of our parameters. Where the key of the dictionary is the parameter name, and the value is the value we want to set for that parameter. Once we've defined these two components we can make our request and parse the response using BeautifulSoup.

- The following piece of code loops for the given SIC code to extract the list of companies and their CIK Codes. We define the parameters for the search and then request the URL from the website. After getting the URL, parse the response to save in the required form so that it can be used later to reach further in the search, to obtain CIK page of companies and so on.

```
for i in range(0,len(x)):
    # define our parameters dictionary
    #Paramters: action, SIC code, Count[range(0,100) where 100 displays 100 companies per page]
    param_dict = {'action':'getcompany',
                  'myowner':'exclude',
                  'SIC': (x)[i],
                  'count': '100'}
    y=x[i]
    # request the url, and then parse the response.
    response = requests.get(url = endpoint, params = param_dict)
    soup = BeautifulSoup(response.content, "html.parser")

```

- In many cases, search resulted in multiple pages containing the list of all the companies in a particular SIC. To extract all the companies, the code had to be written to be able to extract the whole list, like from 1-20 in one page, 21-40 in the next page, and so on. The following code parses the next pages until the list ends. We leverage the html parser output to find the link that takes us to the additional results. This process is easy; we merely find the link tag that has a *rel* attribute set to *next.*

```
    start = int(100)
    first_page= [ base_url + tr.td.a['href'] for tr in soup.find_all('tr')[1:] ]

   # company_links = company_links + first_page
    for i in first_page:
            if i not in company_links:
                company_links.append(i)

    while soup.find_all('input',{'type':'button'})!= []:

#       next_page_link = base_url +'/cgi-bin/browse-edgar?action=getcompany&amp;SIC=' + y + '
        str1= '/cgi-bin/browse-edgar?action=getcompany&amp;SIC='
        str2= '&amp;owner=include&amp;match=&amp;start='
        str3='&amp;count=100&amp;hidefilings=0'

        next_page_link = ''.join(map(str, (base_url,str1, y, str2, start,str3)))
#         print(next_page_link)
        start= start+100

        # request the next page
        response = requests.get(url = next_page_link)
        soup = BeautifulSoup(response.content, 'html.parser')
        more_companies_list=[ base_url + tr.td.a['href'] for tr in soup.find_all('tr')[1:] ]
        company_links= company_links + (more_companies_list)
        for i in more_companies_list:
            if i not in company_links:
                company_links.append(i)
```

- After setting the parameters for the format and time-period of filings as required by the user, the following block of code will give the company-wise page URL but with the parameters of format and time-period incorporated. Request the URL and then parse the response. Then it looks for the documents of the company for which “interactive data” is present. That is the data that we need, it contains the filings.

```
filing_page = requests.get(url = company_link.replace('&hidefilings=0','').replace('&count=40',''), params = param_dict)
soup = BeautifulSoup(filing_page.content, 'html.parser')
print(filing_page.url)

x= soup.find_all('a', {'id' : 'interactiveDataBtn'})
document_links= [ base_url + a['href'] for a in x ]
print('Below are links to your specified form type')

```

- Finally, the following code extracts the final URL of the excel file and downloads the filing into the current working directory. In many cases, one company had multiple filings for a specific time period, so we had to name each file uniquely, but recognizable

under the name of the company. This code downloads and saves these files as "<company name and CIK> 1,2,3,….so on".

```
131         download_file=[base_url + a['href'] for a in soup.find_all('a',{'class':"xbrlviewer"})[:2] ]
132         final_download_file= download_file[1]
133         print(final_download_file)
134 
135         outfile_name = company_name + " " + str(a)+ ".xls"
136         outfile_name = outfile_name.replace(',','').replace('#','').replace(':','').replace('(see all company filings)','')
137         print(outfile_name)
138         urllib.request.urlretrieve(final_download_file, outfile_name)
139         a = a+ 1
```

This code is available for download from the following google drive link:
https://drive.google.com/drive/folders/1h51pCcXQH4tVxP7Se9bewj2PE7DV933Y?usp=sharing
Python installer for 64-bit system and the 32-bit systems are also provided.

Contacts of the programmers:
pragunnanda1610@gmail.com
upreti.ayushman@gmail.com