Introduction

The program, *NPI Reader - 11.ipynb*, is a python-based Jupyter Notebook that given a list of [National Provider Identifiers](https://en.wikipedia.org/wiki/National_Provider_Identifier) (NPIs), extracts each provider’s QPP Participation Status from the [QPP Participation Status web site](https://qpp.cms.gov/participation-lookup) and puts the information into a .csv file.

### Prerequisites

The program was developed on a Windows machine but should run on any OS that supports Jupyter Notebook.

To run the program, you’ll need the following things set up on your machine

1. Jupyter Notebook & Python (best way to get this is to install [Anaconda](https://www.anaconda.com/download/success))
2. The Selenium python library
3. WebDriver (e.g. ChromeDriver)

### Installation Instructions

For Anaconda, go to [Anaconda](https://www.anaconda.com/download/success) and follow the instructions.

For Selenium, once you’ve installed Anaconda, open an Anaconda prompt and run the following command:

pip install selenium

For WebDriver, you can use any WebDriver, but I used ChromeDriver. See the section, Installing ChromeDriver, for detailed instructions.

For *NPI Reader - 11.ipynb* program, download that file as well as the *NPT-4.txt* file which contains 4 NPI numbers. You can put these files anywhere, but a good place to put them is in a folder like the one shown below as it will be handy if you do other projects with Jupyter Notebook.

C:\users\<user name>\projects\qpp-npi

### Using the Program

Here are the steps to run the program.

1. Open Jupyter Notebook
2. From within Jupyter Notebook, navigate to where you stored *NPI Reader - 11.ipynb* and *NPT-4.txt*
3. Open the *NPI Reader - 11.ipynb*
4. Run the program

The program outputs the results into a .csv file in the same directory. The name of the output file will be YYYYMMDDmmss\_<input filename>.csv where the timestamp at the beginning is the time the program was started.

For example, running the program at 8:40am on 5/25/2024 with the default input file resulted in the following output file: *20240525084012 NPI-4 REPORT.csv*

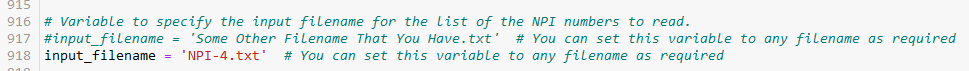
A screenshot of a computer

Description automatically generated

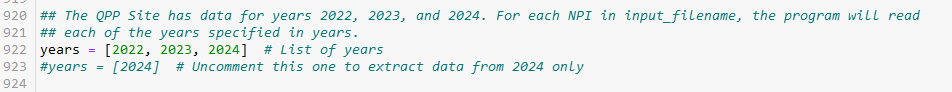
(above) Output of the program using *NPT-4.txt* as the input file and using the default setting for years 2022, 2023, and 2024

On my machine, a 5-year-old Windows machine, it takes about 3 seconds to extract each set of NPI information per year.

The program defaults to using *NPT-4.txt* as the file with the NPI numbers. Change this as needed.



The QPP site supports lookup for years 2022, 2023, and 2024. By default, the program is set to read from all 3 years. Change this as needed.



Occasionally, the QPP website returns a message that a year isn’t supported. This appears to be a glitch in the QPP web site because entering the same NPI by hand on the website will return valid data. To deal with this glitch, the program is set to retry up to 5 times if that error is returned. Change this is needed.

A close up of text

Description automatically generated

If the program has a problem getting the data for an NPI, it will save the generated html file for that NPI to a folder called errors. This is handy for debugging purposes. In the screenshot below, the QPP glitch occurred for NPI = 1720084141, so the program saved the html that it tried to parse to a file called *1720084114.html*.

A screenshot of a computer

Description automatically generated

Over time, this can create a lot of files in the error folder. Comment out the code under the *## When npi\_div* section of the program to stop saving the html files that were problematic.

A close-up of a computer code

Description automatically generated

Finally, know that it took a lot of iterative development to figure out how to read all the data from the generated html. I’ve left quite a few comments in the program to make it easier for someone trying to dig into the details or expand the program.

Suggestions for future work:

* If a provider is associated with multiple practices, the program extracts information for only the first practice that is returned in the html file. Row 8 in the screenshot below is an example.

A screenshot of a computer

Description automatically generated

* If a provider participates in multiple APMs, the program extracts information only for the first APM that is returned in the html file. You’ll see this in column X of the .csv file. In the screenshot example below, the providers are all either in 0 or 1 APM. If a provider had been in 2 APMS, there would have been a 2 in that column.

A screenshot of a document

Description automatically generated

### Installing ChromeDriver

* 1. Get it from the second web site below; choose the win32 platform, download it to Downloads

Look at the

<https://chromedriver.chromium.org/downloads> > <https://googlechromelabs.github.io/chrome-for-testing/#stable>

* 1. Unzip it and you get a .exe file and a license file

A screenshot of a computer

Description automatically generated

* 1. From [this stack overflow post](https://stackoverflow.com/questions/33150351/how-do-i-install-chromedriver-on-windows-10-and-run-selenium-tests-with-chrome), copy chromedriver.exe to C:\Windows

A screenshot of a computer

Description automatically generated

* 1. Verify it works by running the command “chromedriver –version” from a CMD prompt

A screenshot of a computer program

Description automatically generated

(above) Verifying that Chromerdriver is working after putting it in C:\windows

A screenshot of a computer

Description automatically generated

(above) Selecting the Chromedriver for download