**What’s the goal of this?**

Being able to run scripts in python is incredibly valuable. You can pull data from all sorts of exciting places and do it far quicker than you would be able to through the user interface. You can also often get more data.

You don’t need to be able to write scripts in python, but it is very helpful to be able to run them. In this example document we’re going to run a script to pull down data from search console.

**How do we run python?**

If you’re on a mac, you can move straight onto downloading the script. This isn’t the case for windows. By default Windows doesn’t have Python so we have to install a program, [Anaconda](https://www.continuum.io/downloads).

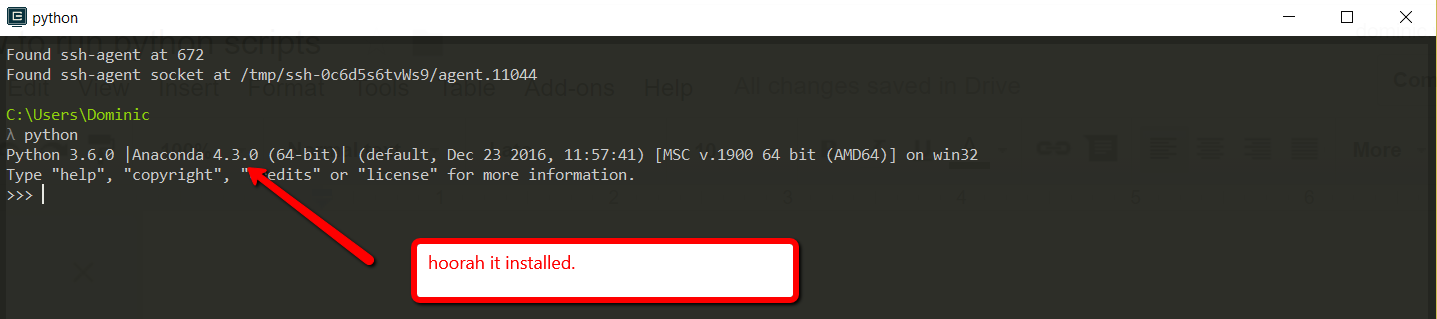
Head here and download it.

Then you’re going to need a good shell. I would recommend [Cmder](http://cmder.net/).

Go ahead and install that.

Check all those steps worked

Open up the command line and type in python and hit enter.



Exit that by typing in exit().

**Download our example script**

For this example we’ll be using the search console script, written by one of our consultants, [Stephan](https://www.distilled.net/about/people/stephan-solomonidis/).

You can download it from his personal Github [here](https://github.com/stephan765/Google-Search-Console-bulk-query/blob/master/search_console_query.py). I’m not going to include a full tutorial on git in this, so if you’re unsure how to download a git repository just go ahead and copy it. (Although it’s definitely worth learning!)

**Running our example script**

Once we’ve downloaded it, we should navigate in Cmder to the folder where we just downloaded it.

The command line functions like the windows file explorer or Finder that you normally use, only everything happens through text. You don’t get a mouse or a GUI.

Some command line basics

To move around it you’ll need some command line basics, these two super important commands:

* cd [path]
* ls -g

The first navigates you to the path given. (or if you use .. takes you step backwards.)

The second lists all the files and folders in the directory you’re currently in.

That’s all you need, but to make yourself faster there are two other things that are useful to know:

Hitting tab, will cause the shell to try and complete the path you’re typing.

Suppose you’re in a folder with two files:

* Moz\_1990\_ranking\_data.txt
* Moz\_180\_rankings.txt
* 180\_rankings.txt

Suppose you type:

* 180 and hit tab: it will autocomplete to *180\_rankings.txt*
* Moz and hit tab: it will autocomplete to *Moz\_*

Secondly, hitting the up key, goes through all the commands you’ve used.

Congratz, now you’re ready to run the script. We just need to get permission for the api.

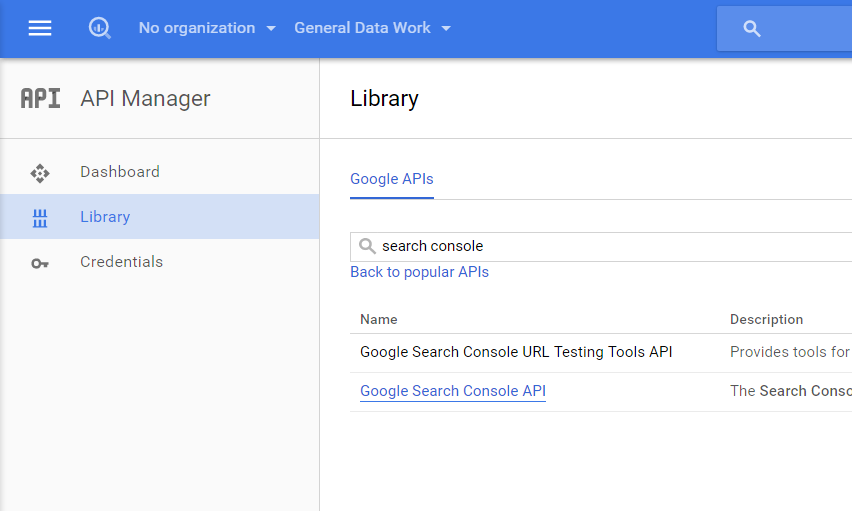
Turning on the API

In the same way you have to login to see search console data, you need permission to use the API. We also need to check its turned on.

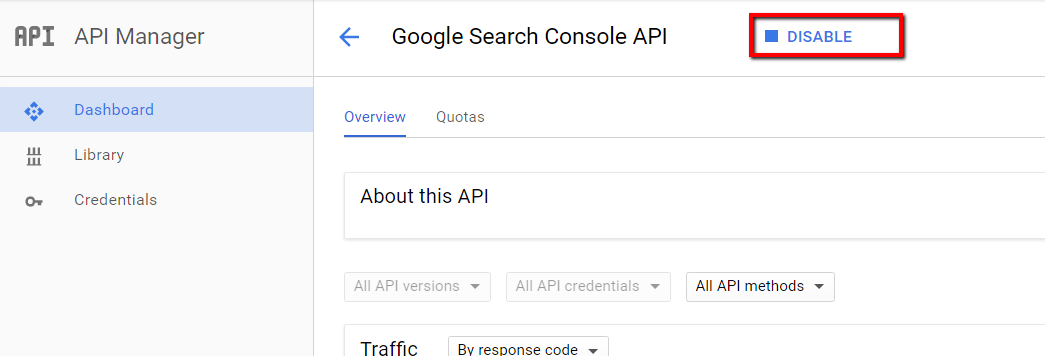
All the Google APIs live in the same place (Google Analytics) is there too. You can find them all at:

* https://console.cloud.google.com/apis/

You'll need to sign in (making sure to use the account with access to your search console). Then you can search for the search console API.



Then once it’s selected, if it says enable here, you’ll need to enable it.

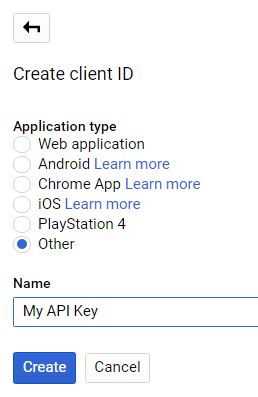


Once that's done we need to download an api key. A single API key gives you access to all of the Google services. Different APIs have different types of keys, sometimes it will just be a text string, other times a file.

Once we’ve enabled the API we select the credentials tab and then create credentials. The three main kinds of API key, a basic text key, user OAuth credentials and service account keys.

The first is quick and simple, the second is more secure and intended for users who will authenticate with a login, the third for automated data pulling.

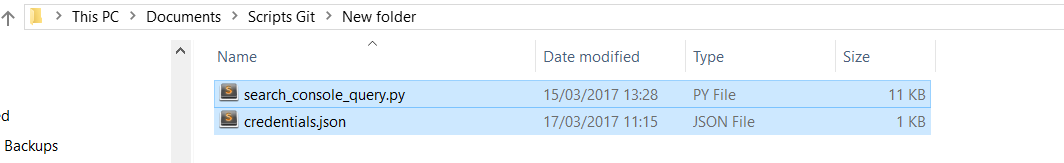
There are some subtleties around permissions with these that we don't really want to delve into here. The script is set-up to use the second so we’ll use that. Go ahead create an OAuth Client ID:



Ignore the pop-up and download the file from the credentials screen:

https://lh6.googleusercontent.com/Wyf-ieyWbamJvSz5njGFutId30vPxpYe6sJ4wytj1KexzJnJaeBg5oOfSradiV_T1yyIavRhupJEphtMWgWaTGK-baua-a-PwlOsTxGEPye7yQqzkd_steUjlvYUe1oaPxqALGtG

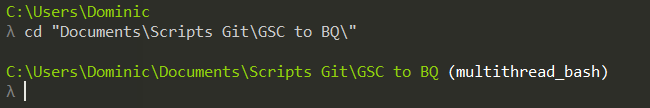
And then move it to the same folder as your script. For ease of use we’ll also rename it “credentials.json” which is what the script is looking for (a script will tell you what it’s expecting this file to be called when you run it or will have this in the documentation, assuming it’s well written of course):



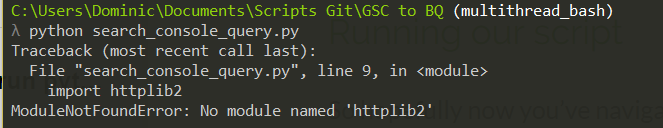
Running our script

And we’re ready to go!

Hopefully now you’ve navigated to the folder with the script in.



So now we try and run the script:



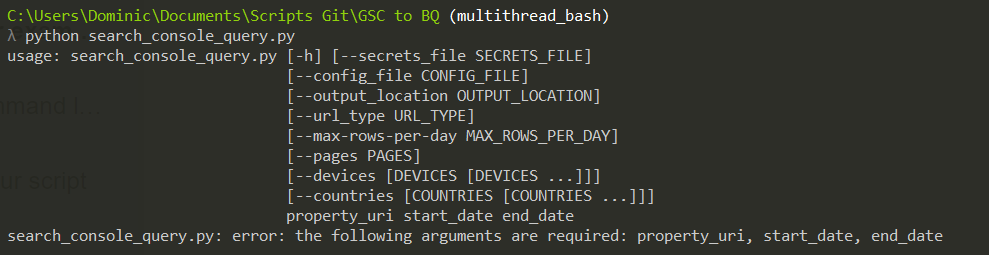
We get a module missing error. Normally you can solve this by running --

* pip install missing\_module -- or in our case
* pip install httplib2

And because we’ll get several of these errors we need to install a couple modules.

* pip install httplib2 oauth2client
* pip install --user --upgrade google-api-python-client

Then we can try again and now we’ll get the script help, which will tell us how to run it. Any well documented script should return something like this:



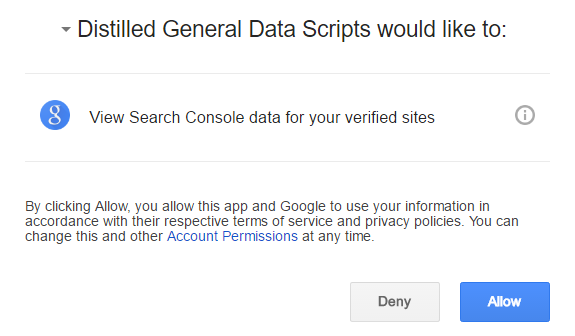
First we pay attention to the last line. Which arguments are required?

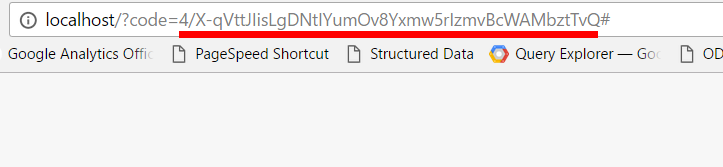
* property\_uri
* start\_date
* end\_date

Our script needs to have these 3 first (remember to change the URL to a property you have access to!) :

python search\_console\_query.py https://www.distilled.net/ 2017-02-05 2017-02-06

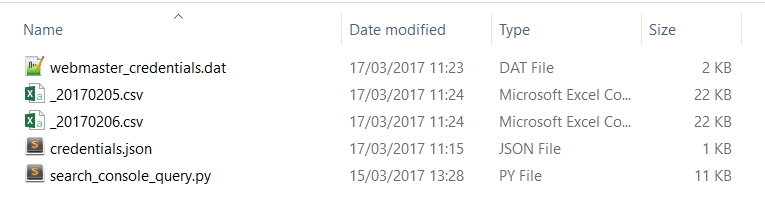
Your browser will open up, you’ll need to login and authenticate the script (because it’s the first time we’re running the script):



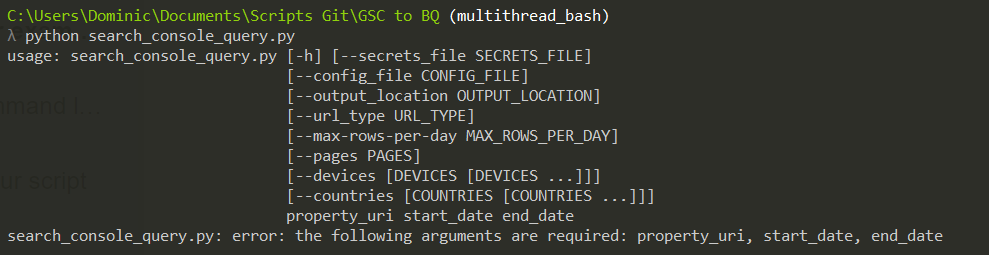
Then once the code will appear in the URL of the page, everything from the = up to the hash, which you’ll need to copy and paste back into the script:

Check your folder and it should now look something like this:

The permission we gave the script is now saved in webmaster\_credentials.dat and each of our search console data sits in those CSV files!



If we look back at our script options:



We can see some of the other options we can use.

* Pages takes a file of pages to individually query (example file)
  + By default pulls for the entire property.
* Devices takes a space separated list
  + By default it queries mobile, desktop and tablet.
* Countries
  + By default does worldwide. Takes a space separated [list of country codes](https://en.wikipedia.org/wiki/ISO_3166-1_alpha-3).
* By default it will pull 100 rows of data per day. The API allows a limit of up to 5000.

So for example:

#get top queries for the property

python search\_console\_query.py https://www.distilled.net/ 2017-02-05 2017-02-06

#get top queries for multiple pages and aggregate together

python search\_console\_query.py https://www.distilled.net/ 2017-02-05 2017-02-06 --pages file\_of\_pages

#get top queries for the property from desktop and mobile

python search\_console\_query.py https://www.distilled.net/ 2017-02-05 2017-02-06 --devices desktop mobile

#get the top queries for the property from the US & the UK

python search\_console\_query.py https://www.distilled.net/ 2017-02-05 2017-02-06 --countries USA GBR