MN GEOSPATIAL CKAN API ¶

This API is a metadata retrieval method.

In [ ]:

```
import requests
import json
import zipfile
import pprint

###########---MODIFY---#######
search_query = 'waterways'
result_num = 1
resource_num = 1
##############################

big_url = 'https://gisdata.mn.gov/api/3/action/package_search?q=' + search_que
ry
#call API to search packages with search query
response = requests.get(big_url, verify = False)
#turn result into JSON
json_response = json.loads(response.content)
#dig down through dictionary layers
result_options = json_response['result']['results']
chosen_result = result_options[result_num]
resources_under_result= chosen_result['resources'][resource_num]
chosen_resource = resources_under_result['url']
print(chosen_resource)




x=input('Type "next" to continue to download and extraction.' )
if x == 'next':
    pass
else:
    print("Sorry, please restart script")
#now send HTTP request to the spit-out url and save to file
URL_request = requests.get(chosen_resource)
with open('filename4.zip', 'wb') as f:
    f.write(URL_request.content)
    f.close()
#unzip the file into the same directory
with zipfile.ZipFile("filename4.zip","r") as zip_ref:
    zip_ref.extractall()
print('Download and extraction complete. Check notebook folder')
```

```
C:\Program Files\ArcGIS\Pro\bin\Python\envs\arcgispro-py3\lib\site-packages\u
rllib3\connectionpool.py:1004: InsecureRequestWarning: Unverified HTTPS reque
st is being made to host 'gisdata.mn.gov'. Adding certificate verification is
strongly advised. See: https://urllib3.readthedocs.io/en/latest/advanced-usag
e.html#ssl-warnings
  InsecureRequestWarning,

https://resources.gisdata.mn.gov/pub/gdrs/data/pub/us_mn_state_dot/trans_rail
_lines/fgdb_trans_rail_lines.zip
```

GOOGLE PLACES API

The Google Places API makes use of the Google Map Cloud to find locations/information based on an input. It requires a billing account and API key. The requests library's 'get' is still used to send the completed API url string to the server. Formatting of certain parameters in html url format is required, and additional params are optional, before the get. The response comes back as html formatting, and the local id can be clicked to connect to the web.

In [29]:
```
'''
#required params:
    key: the API key
    input: the search terms
    input type: either textquery or phonenumber (in intl' format)
#opt params:
    language
    fields
    locationbias (see docs for more),
#if you omit "fields", only place_ID will returned
#fields param:
    BASIC:
        business_status
        formatted_address
        geometry
        icon
        name
        photos
        place_id
        plus_code
        types
    CONTACT:
        open_now
    ATMOS:
        price_level
        rating
        user_ratings_total
#full url examples:
#.../json?input=Museum%20of%20Contemporary%20Art%20Australia&inputtype=textque
ry&fields=photos,formatted_address,name,rating,opening_hours,geometry&key=YOUR
_API_KEY
#.../json?input=%2B61293744000&inputtype=phonenumber&fields=place_id&key=YOUR_
API_KEY
'''
import requests
import json
API_KEY = "AIzaSyBjn4F98m1QtOZC0VzPfbQpOXicJPOzEN8"
#APIKey aquired from Places API
BASE_URL = "https://maps.googleapis.com/maps/api/place/findplacefromtext/jso
n?"
#this base url is for a place search request
#output can be json or xml. suggest use JSON

search = 'Potato%20Museum'
#the terms to search for. put in %20 for space
inp_type = 'textquery'
#input type
fields = 'place_id,formatted_address'
#desired fields

Full_URL = BASE_URL+'input='+search+'&inputtype='+inp_type+'&fields='+fields+
'&key='+API_KEY
#build full search URL
print(Full_URL)
#print the complete URL
response = requests.get(Full_URL)
```

```
# i can probably use the payload method to specify how to build the URL like i
n NDAWN
#use that requests boi
for c in response.iter_lines():
    print(c)
#ptint the retireved content

#it found a potato museum in Blackfoot, ID!
```

```
https://maps.googleapis.com/maps/api/place/findplacefromtext/json?input=Potat
o%20Museum&inputtype=textquery&fields=place_id,formatted_address&key=AIzaSyBj
n4F98m1QtOZC0VzPfbQpOXicJPOzEN8
b'{'
b'   "candidates" : ['
b'      {'
b'         "formatted_address" : "130 NW Main St, Blackfoot, ID 83221, United
States",'
b'         "place_id" : "ChIJPTg-hZ0aVVMRiEnRXIW3Lt0"'
b'      }'
b'   ],'
b'   "status" : "OK"'
b'}'
```

NDAWN API

NDAWN or North Dakota Agricultural Weather Network, is a set of 155 recording stations across ND, MN, and MT.

This website provides an array of statistics at hourly, daily, monthly, and yearly summaries: max/min/avg air temperature, windspeed and direction, rainfall, soil temp, humidity, etc

Data retrieval from this website utilizes the requests python library and html inspect skills. This particular script is set up to retrieve a csv for every station, variable, and year specified in a list. The 'payload' is a dictionary of parameters to be sent in the request. A 'get' function sends this information to a specified url, which is set to the monthly data page.

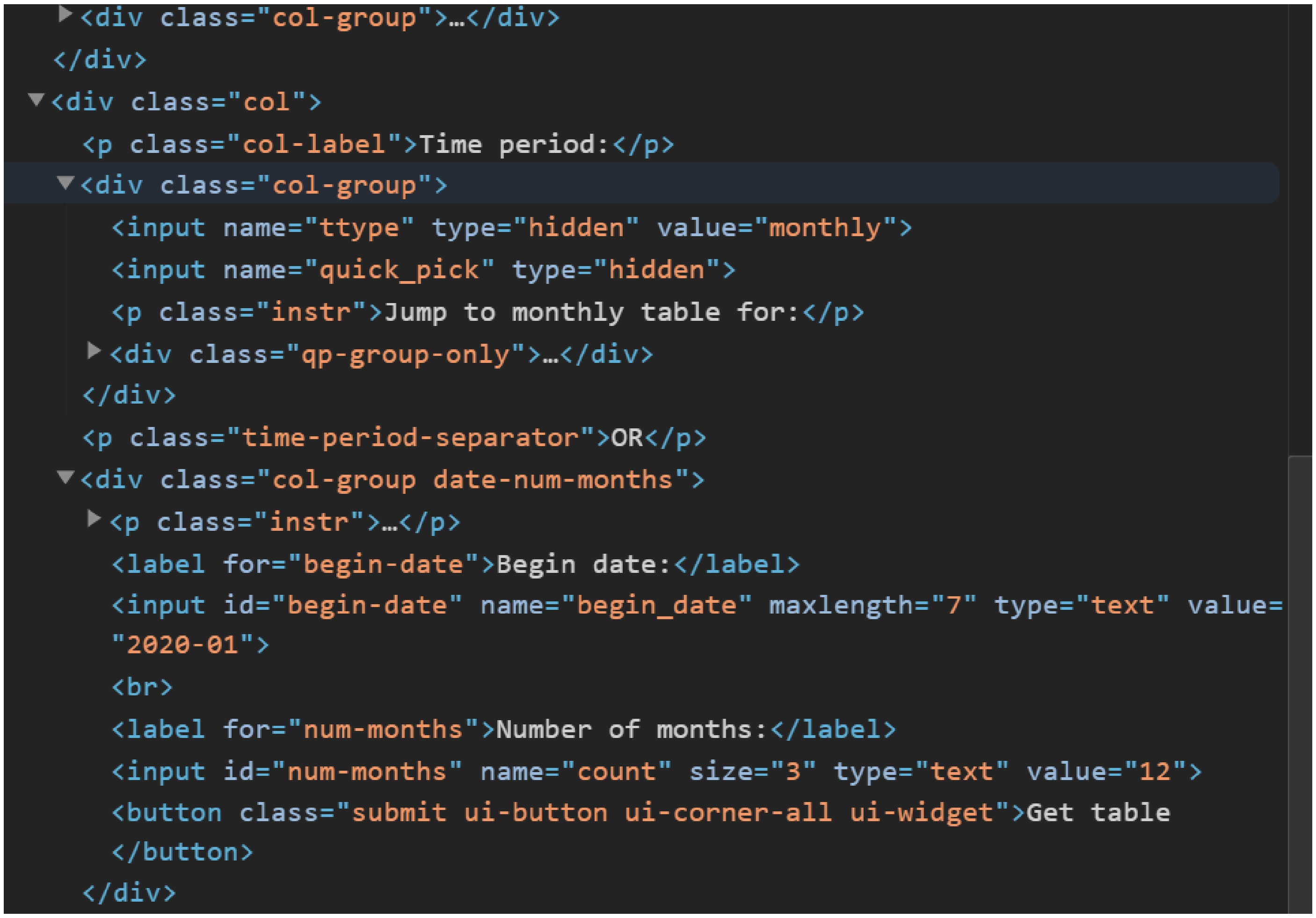
File names are saved as a string of the measured variable, station, and year. Each file is saved to the local drive as a csv.

In [22]:
```
#NDAWN MONTHLY API

'''
# the payload refers to the key: value pairs found in the URL string.
#....station=5&variable=mdmxws&...etc
# you can find what the important variables are by using the inspector on the
 page after data selection
#or looking at an example URL and recording
Required keys:
    station: use any number here. numbers reference station locations
    variable:
        mdmxt Air Temp Max
        mdmnt Air Temp Min
        mdavt Air Temp Avg.
        mdbst....
    year: explanatory/not needed
    ttype: monthly, daily, etc.
    begin_date: YYYY-MM
    quickpick: blank
    count: number of months requested
'''
import requests

#put the desired adjustable parameters here!
stations = [3,56,72]
variables = ['mdmxt', 'mdmnt']
years = [2019, 2020, 2021]

#say we want to download a data set for each station/variable/year combinatio
n. Use a nested loop
for station in stations:
    for variable in variables:
        for year in years:
            payload = {'station': station,
                  'variable':variable,
                  'year': year,
                  'ttype':'monthly',
                  'quickpick':'',
                   #begin date needs a DD if weekly is selected
                  'begin_date': str(year)+'-01',
                  'count':12}
            #the requests.get command finds the file location and accepts the
 parameter payload
            http_location = requests.get('https://ndawn.ndsu.nodak.edu/table.c
sv', params = payload, stream = True)

            # ok, so apparently I had the 'table.csv' set to get-table.html, s
ince that was what was in the URL.
            #apparently that was not right
            #if you click 'export CSV file' on the get-table page, it tries to
save as a csv called 'table'
            #the inspector on the export button refers to a /table.csv before
 the parameters
            #ie, .edu/table.csv?station=....

            #this line writes the content of the response object to the file n
```

```
ame/type designated
            filename = 'MonthlyWindMax' + '_station_' + str(station) + str(var
iable) + str(year) + '.csv'
            print(filename)
            #open, write, and close the file
            with open(filename, 'wb') as f:
                f.write(http_location.content)
                f.close()

    #helpful but way more complicated source for really nice one:
    #https://gettecr.github.io/noaa-api.html
```

```
MonthlyWindMax_station_3mdmxt2019.csv
MonthlyWindMax_station_3mdmxt2020.csv
MonthlyWindMax_station_3mdmxt2021.csv
MonthlyWindMax_station_3mdmnt2019.csv
MonthlyWindMax_station_3mdmnt2020.csv
MonthlyWindMax_station_3mdmnt2021.csv
MonthlyWindMax_station_56mdmxt2019.csv
MonthlyWindMax_station_56mdmxt2020.csv
MonthlyWindMax_station_56mdmxt2021.csv
MonthlyWindMax_station_56mdmnt2019.csv
MonthlyWindMax_station_56mdmnt2020.csv
MonthlyWindMax_station_56mdmnt2021.csv
MonthlyWindMax_station_72mdmxt2019.csv
MonthlyWindMax_station_72mdmxt2020.csv
MonthlyWindMax_station_72mdmxt2021.csv
MonthlyWindMax_station_72mdmnt2019.csv
MonthlyWindMax_station_72mdmnt2020.csv
MonthlyWindMax_station_72mdmnt2021.csv
```

an image of the html inspect screen on Brave browser. begin_date and count represent two of the params included in the get request



```
  ▶<div class="col-group">…</div>
  </div>
 ▼<div class="col">
    <p class="col-label">Time period:</p>
  ▼<div class="col-group">
      <input name="ttype" type="hidden" value="monthly">
      <input name="quick_pick" type="hidden">
      <p class="instr">Jump to monthly table for:</p>
    ▶<div class="qp-group-only">…</div>
    </div>
    <p class="time-period-separator">OR</p>
  ▼<div class="col-group date-num-months">
    ▶<p class="instr">…</p>
      <label for="begin-date">Begin date:</label>
      <input id="begin-date" name="begin_date" maxlength="7" type="text" value=
      "2020-01">
      <br>
      <label for="num-months">Number of months:</label>
      <input id="num-months" name="count" size="3" type="text" value="12">
      <button class="submit ui-button ui-corner-all ui-widget">Get table
      </button>
    </div>
```