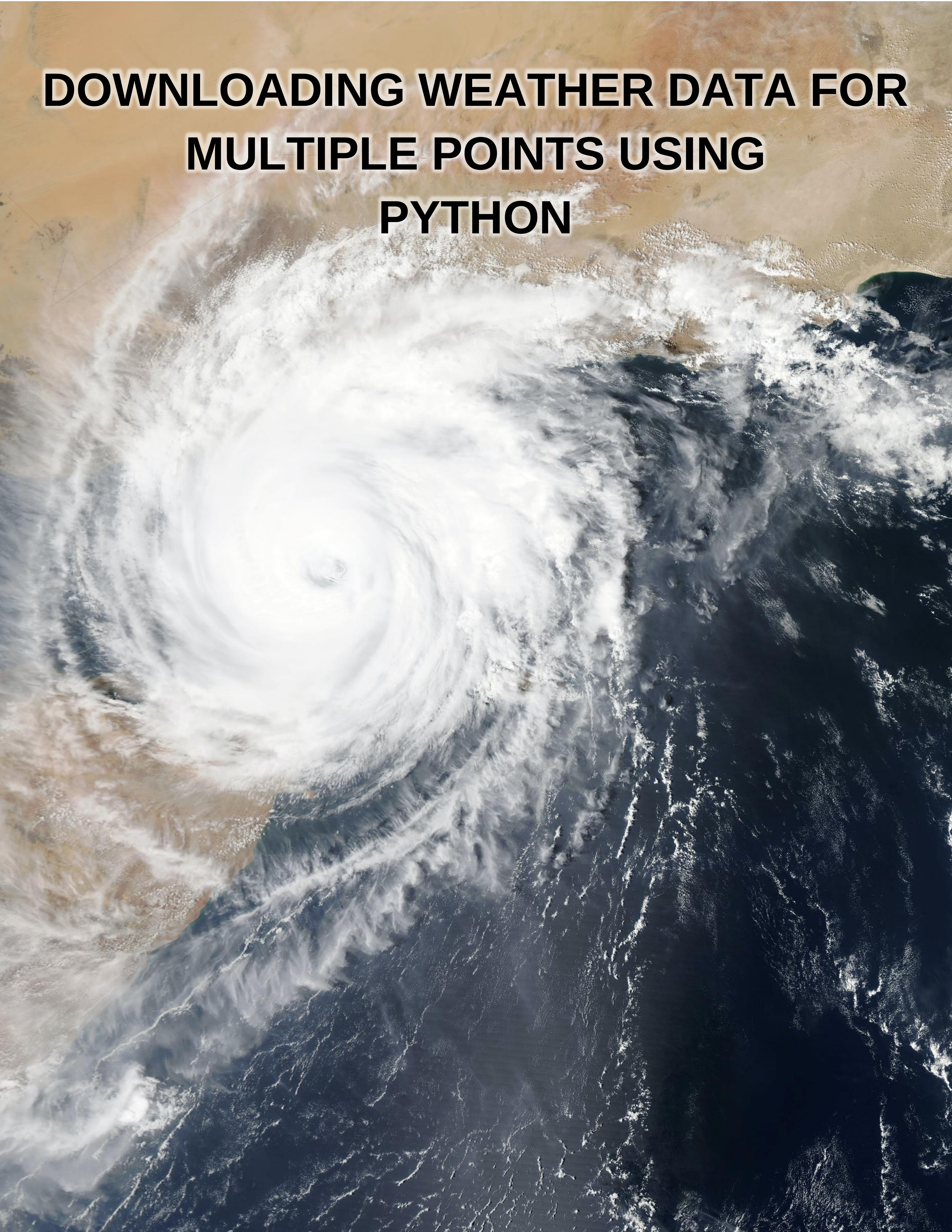
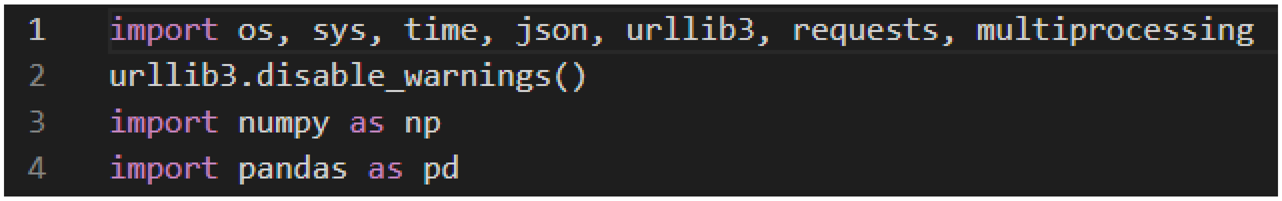
DOWNLOADING WEATHER DATA FOR MULTIPLE POINTS USING PYTHON



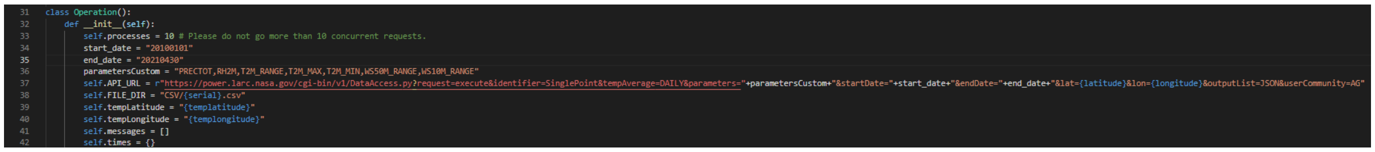
1. Program Explanation

1.1 Weather Data Code Explanation



```python
import os, sys, time, json, urllib3, requests, multiprocessing
urllib3.disable_warnings()
import numpy as np
import pandas as pd
```

- We will be importing libraries os, sys, time, json, urlibr3, requests, multiprocessing, numpy and pandas

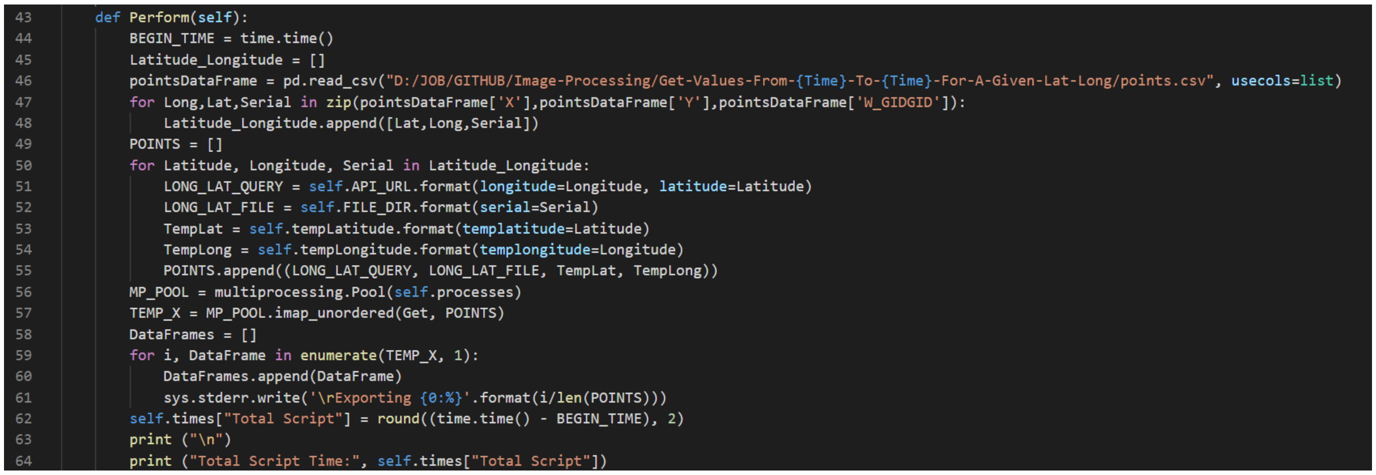


```python
class Operation():
    def __init__(self):
        self.processes = 10 # Please do not go more than 10 concurrent requests.
        start_date = "20100101"
        end_date = "20210430"
        parametersCustom = "PRECTOT,RH2M,T2M_RANGE,T2M_MAX,T2M_MIN,WS50M_RANGE,WS10M_RANGE"
        self.API_URL = r"https://power.larc.nasa.gov/cgi-bin/v1/DataAccess.py?request=execute&identifier=SinglePoint&tempAverage=DAILY&parameters="+parametersCustom+"&startDate="+start_date+"&endDate="+end_date+"&lat={latitude}&lon={longitude}&outputList=JSON&userCommunity=AG"
        self.FILE_DIR = "CSV/{serial}.csv"
        self.tempLatitude = "{templatitude}"
        self.tempLongitude = "{templongitude}"
        self.messages = []
        self.times = {}
```

- Create a class Operation() with constructor.
- In constructor method we will be defining variables process, start_date, end_date, parametersCustom, API_URL, FILE_DIR, tempLatitude, tempLongitude, messages and times
- The process variable defines how many times the code should access the API_URL maximum should be only 10, if exceeded it will give an error saying too many requests from API_URL.
- The start_date and end_date parameters are for the user to define the date range of the data (s)he wants. Should be same format as specified.

- The parametersCustom variable defines how many parameters of data should the code download which should be defined by the user. Should be the same format defined in the website.

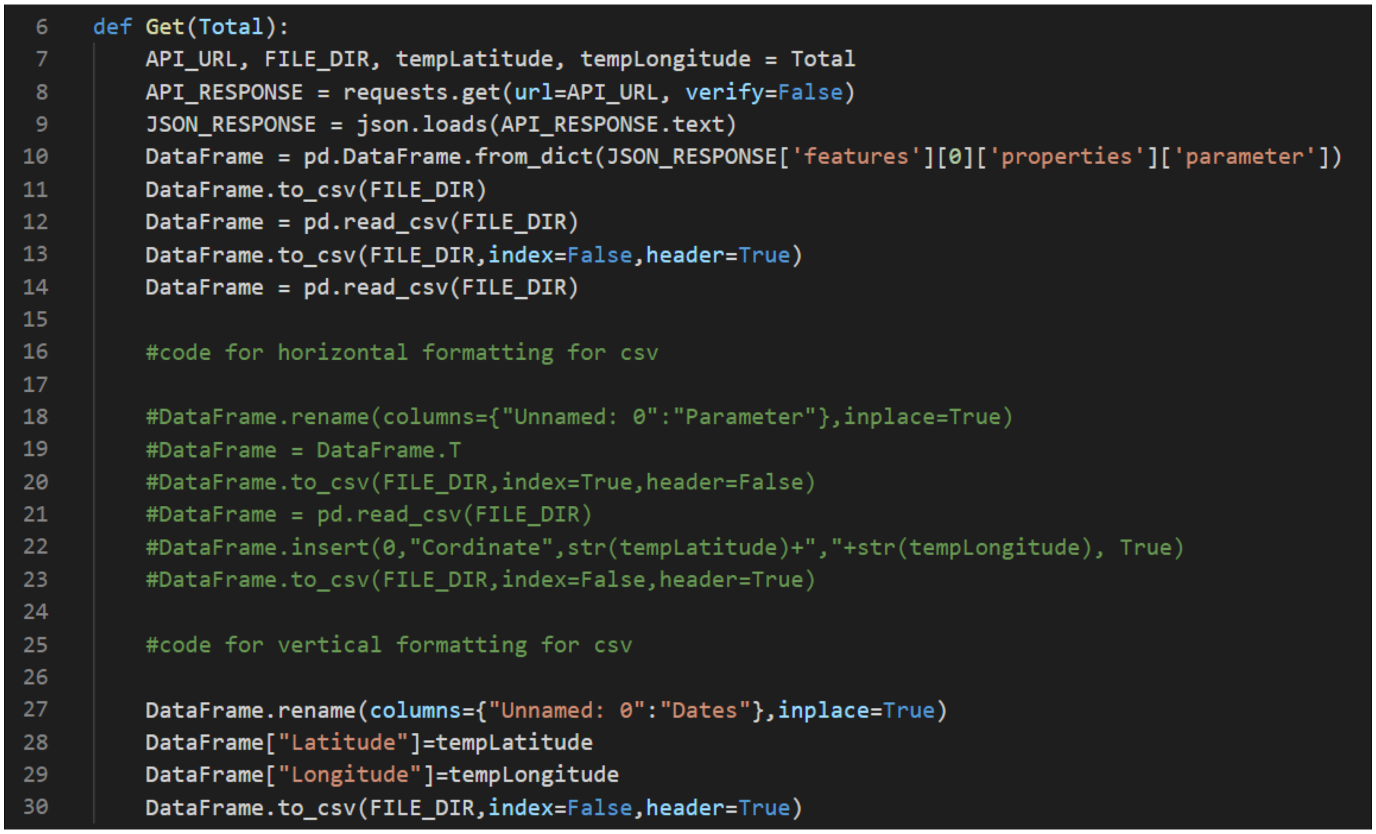
- API_URL variable have the url to the api to get data with the parameters defined by user added within the url passed to that variable.
- FILE_DIR variable holds the directory where the downloaded csv files should be.
- tempLatitude and temp Longitude have the latitude and longitude(points specified in the csv file)
- Messages and times are used for Perform() method for error message storing.



```
    def Perform(self):
        BEGIN_TIME = time.time()
        Latitude_Longitude = []
        pointsDataFrame = pd.read_csv("D:/JOB/GITHUB/Image-Processing/Get-Values-From-{Time}-To-{Time}-For-A-Given-Lat-Long/points.csv", usecols=list)
        for Long,Lat,Serial in zip(pointsDataFrame['X'],pointsDataFrame['Y'],pointsDataFrame['W_GIDGID']):
            Latitude_Longitude.append([Lat,Long,Serial])
        POINTS = []
        for Latitude, Longitude, Serial in Latitude_Longitude:
            LONG_LAT_QUERY = self.API_URL.format(longitude=Longitude, latitude=Latitude)
            LONG_LAT_FILE = self.FILE_DIR.format(serial=Serial)
            TempLat = self.tempLatitude.format(templatitude=Latitude)
            TempLong = self.tempLongitude.format(templongitude=Longitude)
            POINTS.append((LONG_LAT_QUERY, LONG_LAT_FILE, TempLat, TempLong))
        MP_POOL = multiprocessing.Pool(self.processes)
        TEMP_X = MP_POOL.imap_unordered(Get, POINTS)
        DataFrames = []
        for i, DataFrame in enumerate(TEMP_X, 1):
            DataFrames.append(DataFrame)
            sys.stderr.write('\rExporting {0:%}'.format(i/len(POINTS)))
        self.times["Total Script"] = round((time.time() - BEGIN_TIME), 2)
        print ("\n")
        print ("Total Script Time:", self.times["Total Script"])
```

- Now define a method Perform() in the same class, the BEGIN_TIME variable records the current time while the performing the operation and it will be used to determine the time taken for the download.
- Latitude_Longitude variable is an array which stores the latitude and longitudes given in the csv input file.
- pointsDataFrame variable holds the data of a csv file which will be given by the user.

- For more clarification we will be looping that data frame and take out the only data we were gonna need, which will be the latitude, longitude and the W_ID.
- Those 3 variables will be appended to the array we created earlier, every time it loops it will add one row from the dataframe until the end of rows.
- Now we will use the data from the Latitude_longitude array and extract the 3 variables (latitude, longitude, W_ID) and start the loop for each latitude, longitude and W_ID respectively.
- We will be passing the latitude and longitude to the variable we created earilier in the constructor API_URL.
- And we will be passing the W_ID as the serial number of the csv files which are being created.
- Same for the variables Templat and Templong aslo formatted with latitude and longitude respectively.
- All this data will be appended to the array called points, each row will be appended while the loop continues.
- MP_POOL variable will be set to the processes that were define in the constructor earlier.
- TEMP_X will have the out data by passing the GET method and the points array in it.
- After the data we will be calculating the live percentage which should be printed on the screen will downloading the data using for loop through the dataframe of the csv file we read earilier.

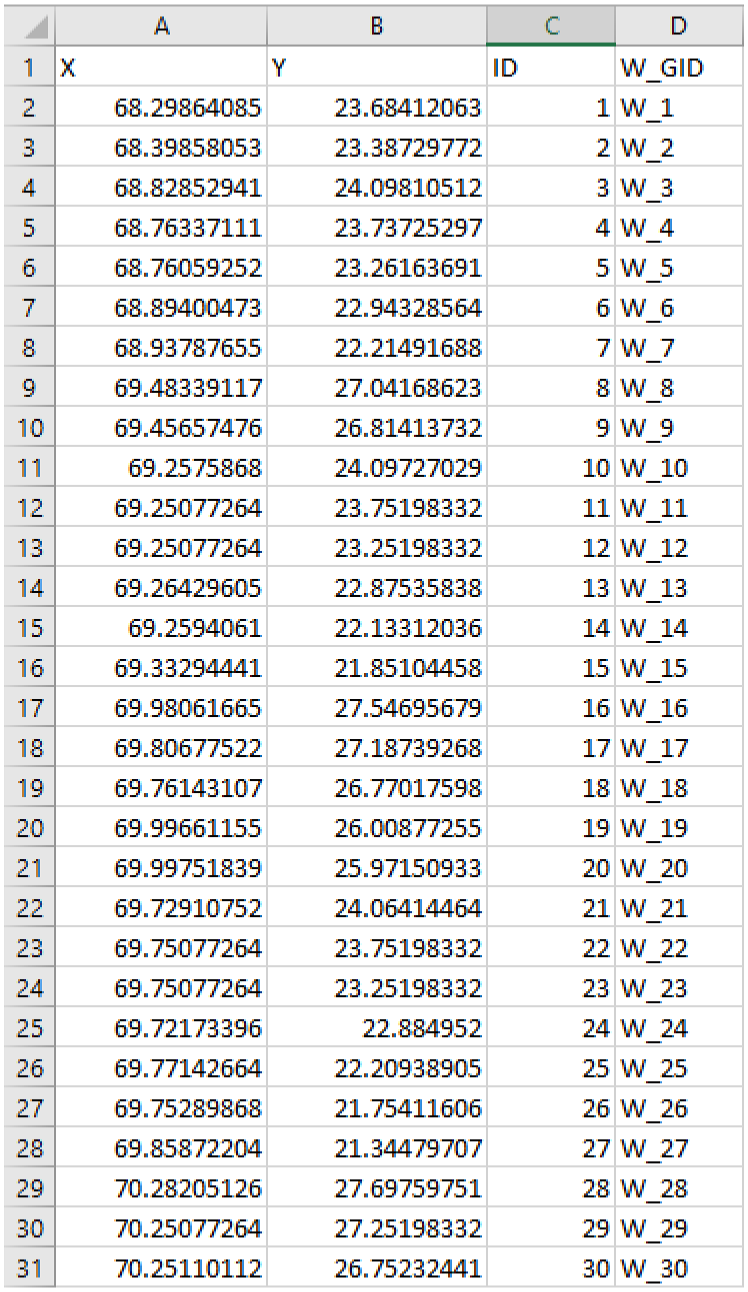


```python
def Get(Total):
    API_URL, FILE_DIR, tempLatitude, tempLongitude = Total
    API_RESPONSE = requests.get(url=API_URL, verify=False)
    JSON_RESPONSE = json.loads(API_RESPONSE.text)
    DataFrame = pd.DataFrame.from_dict(JSON_RESPONSE['features'][0]['properties']['parameter'])
    DataFrame.to_csv(FILE_DIR)
    DataFrame = pd.read_csv(FILE_DIR)
    DataFrame.to_csv(FILE_DIR,index=False,header=True)
    DataFrame = pd.read_csv(FILE_DIR)

    #code for horizontal formatting for csv

    #DataFrame.rename(columns={"Unnamed: 0":"Parameter"},inplace=True)
    #DataFrame = DataFrame.T
    #DataFrame.to_csv(FILE_DIR,index=True,header=False)
    #DataFrame = pd.read_csv(FILE_DIR)
    #DataFrame.insert(0,"Cordinate",str(tempLatitude)+","+str(tempLongitude), True)
    #DataFrame.to_csv(FILE_DIR,index=False,header=True)

    #code for vertical formatting for csv

    DataFrame.rename(columns={"Unnamed: 0":"Dates"},inplace=True)
    DataFrame["Latitude"]=tempLatitude
    DataFrame["Longitude"]=tempLongitude
    DataFrame.to_csv(FILE_DIR,index=False,header=True)
```

- Outside of the class Operation() we will be defining a method called Get(), which has been called in the method Perfrom().
- This method consists of variable API_URL, FILE_DIR, tempLatitude, tempLongitude which will be defined to a variable total which consists of 4 types of variable in it which makes it equal to be define it with another 4 variables.
- API_RESPOSE gets the data the we need by passing the API_URL variable into the requests using the requests library.
- JSON_RESPOSE variable will possess the data that have been retrieved by the API_RESPONSE variable and loads it in plain text which will be easy to be read or manipulated.
- DataFrame variable stores the data from the dictionary of JSON data which have been retrieved and stored as plain text
- After that the DataFrame will be will save to a csv file and read again into the same variable
- Then the DataFrame variable will be added a header and removed the index and saved to csv again

- Again the csv will be read into the same variable now the columns will be renamed using a function called rename from pandas library.
- And 2 columns will be added Latitude and Longitude with latitude and longitude points then again saved to the cv file
- This process will repeat for every row read from the given csv file.

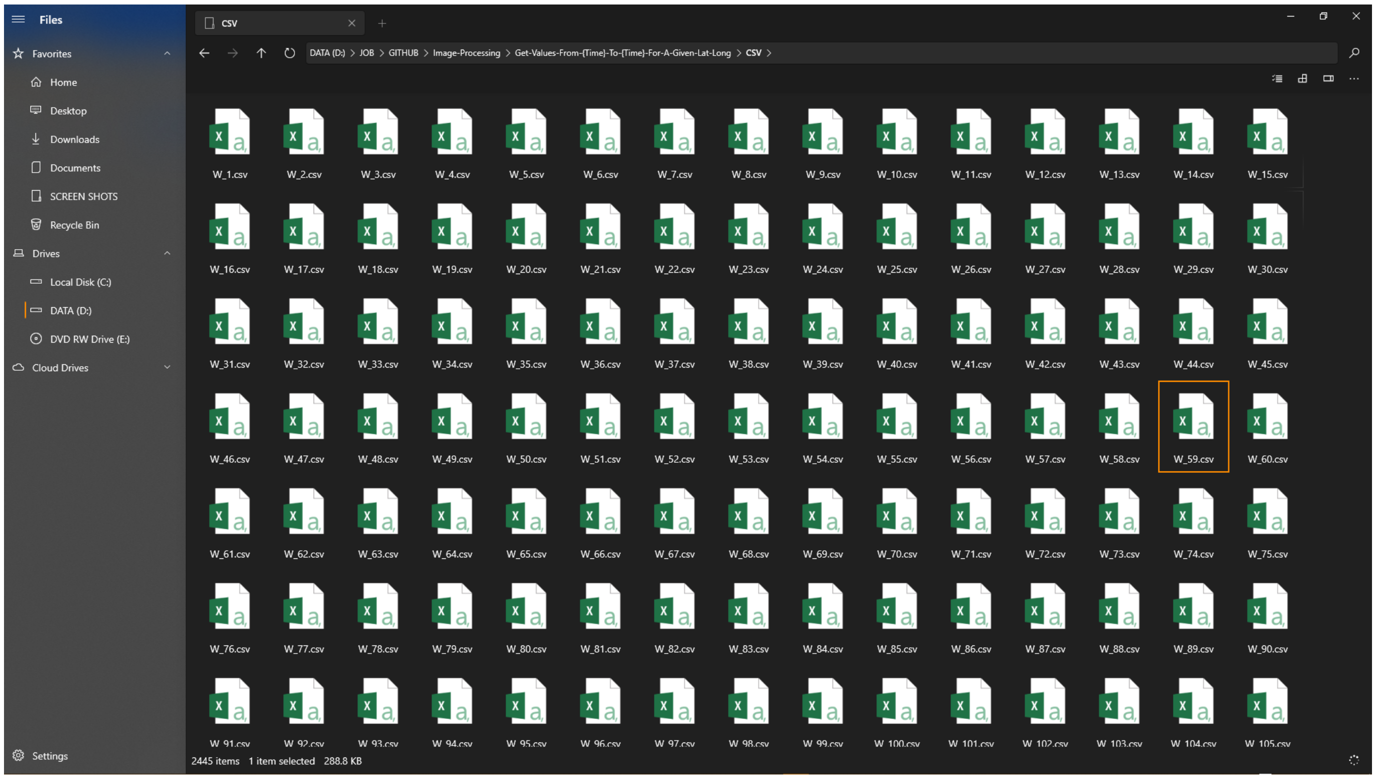
1.2 Input Data Format



	A	B	C	D
1	X	Y	ID	W_GID
2	68.29864085	23.68412063	1	W_1
3	68.39858053	23.38729772	2	W_2
4	68.82852941	24.09810512	3	W_3
5	68.76337111	23.73725297	4	W_4
6	68.76059252	23.26163691	5	W_5
7	68.89400473	22.94328564	6	W_6
8	68.93787655	22.21491688	7	W_7
9	69.48339117	27.04168623	8	W_8
10	69.45657476	26.81413732	9	W_9
11	69.2575868	24.09727029	10	W_10
12	69.25077264	23.75198332	11	W_11
13	69.25077264	23.25198332	12	W_12
14	69.26429605	22.87535838	13	W_13
15	69.2594061	22.13312036	14	W_14
16	69.33294441	21.85104458	15	W_15
17	69.98061665	27.54695679	16	W_16
18	69.80677522	27.18739268	17	W_17
19	69.76143107	26.77017598	18	W_18
20	69.99661155	26.00877255	19	W_19
21	69.99751839	25.97150933	20	W_20
22	69.72910752	24.06414464	21	W_21
23	69.75077264	23.75198332	22	W_22
24	69.75077264	23.25198332	23	W_23
25	69.72173396	22.884952	24	W_24
26	69.77142664	22.20938905	25	W_25
27	69.75289868	21.75411606	26	W_26
28	69.85872204	21.34479707	27	W_27
29	70.28205126	27.69759751	28	W_28
30	70.25077264	27.25198332	29	W_29
31	70.25110112	26.75232441	30	W_30

- The csv file which is going to be included in the code should at least contain 3 columns in which 2 columns are Longitude and Latitude respectively and the other column should be the W_ID.
- Failed to provide the columns as is should result error as output.

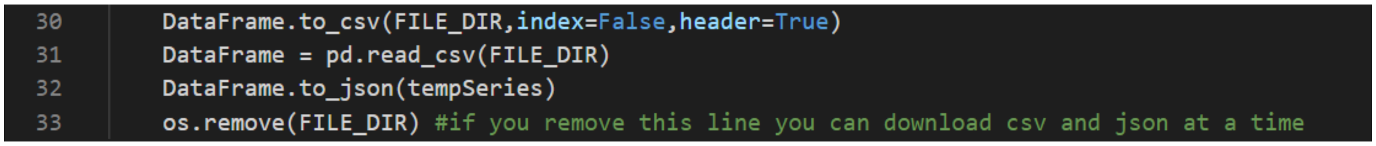
1.3 Output Data


Files
Favorites
Home
Desktop
Downloads
Documents
SCREEN SHOTS
Recycle Bin
Drives
Local Disk (C:)
DATA (D:)
DVD RW Drive (E:)
Cloud Drives
Settings
csv
DATA (D:) > JOB > GITHUB > Image-Processing > Get-Values-From-{Time}-To-{Time}-For-A-Given-Lat-Long > CSV >
W_1.csv W_2.csv W_3.csv W_4.csv W_5.csv W_6.csv W_7.csv W_8.csv W_9.csv W_10.csv W_11.csv W_12.csv W_13.csv W_14.csv W_15.csv
W_16.csv W_17.csv W_18.csv W_19.csv W_20.csv W_21.csv W_22.csv W_23.csv W_24.csv W_25.csv W_26.csv W_27.csv W_28.csv W_29.csv W_30.csv
W_31.csv W_32.csv W_33.csv W_34.csv W_35.csv W_36.csv W_37.csv W_38.csv W_39.csv W_40.csv W_41.csv W_42.csv W_43.csv W_44.csv W_45.csv
W_46.csv W_47.csv W_48.csv W_49.csv W_50.csv W_51.csv W_52.csv W_53.csv W_54.csv W_55.csv W_56.csv W_57.csv W_58.csv W_59.csv W_60.csv
W_61.csv W_62.csv W_63.csv W_64.csv W_65.csv W_66.csv W_67.csv W_68.csv W_69.csv W_70.csv W_71.csv W_72.csv W_73.csv W_74.csv W_75.csv
W_76.csv W_77.csv W_78.csv W_79.csv W_80.csv W_81.csv W_82.csv W_83.csv W_84.csv W_85.csv W_86.csv W_87.csv W_88.csv W_89.csv W_90.csv
W_91.csv W_92.csv W_93.csv W_94.csv W_95.csv W_96.csv W_97.csv W_98.csv W_99.csv W_100.csv W_101.csv W_102.csv W_103.csv W_104.csv W_105.csv

2445 items 1 item selected 288.8 KB

- For every point one csv file will be created with the parameters and date range specified by the user.

1.3 For JSON as output



```
DataFrame.to_csv(FILE_DIR,index=False,header=True)
DataFrame = pd.read_csv(FILE_DIR)
DataFrame.to_json(tempSeries)
os.remove(FILE_DIR) #if you remove this line you can download csv and json at a time
```

- A small 3 lines of code will be added which reads the output csv and converts into JSON and removes the csv this way without writing a whole code for JSON output these 3 lines does the trick.