WEATHER DATA

# **Introduction:**

This project is used for storing the weather data retrieved, into a GeoJSON or CSV file format. In order to retrieve weather data we will be using an API from NASA website: <https://power.larc.nasa.gov/> . Through this API we can retrieve Moisture, Temperature, Wind, Sky, Long wave, Earth skin temperature, Pressure, and many more sub attributes, but for our project there is a need of 6 attributes which are:

PRECTOT – Precipitation

RH2M – Relative humidity at 2 meters

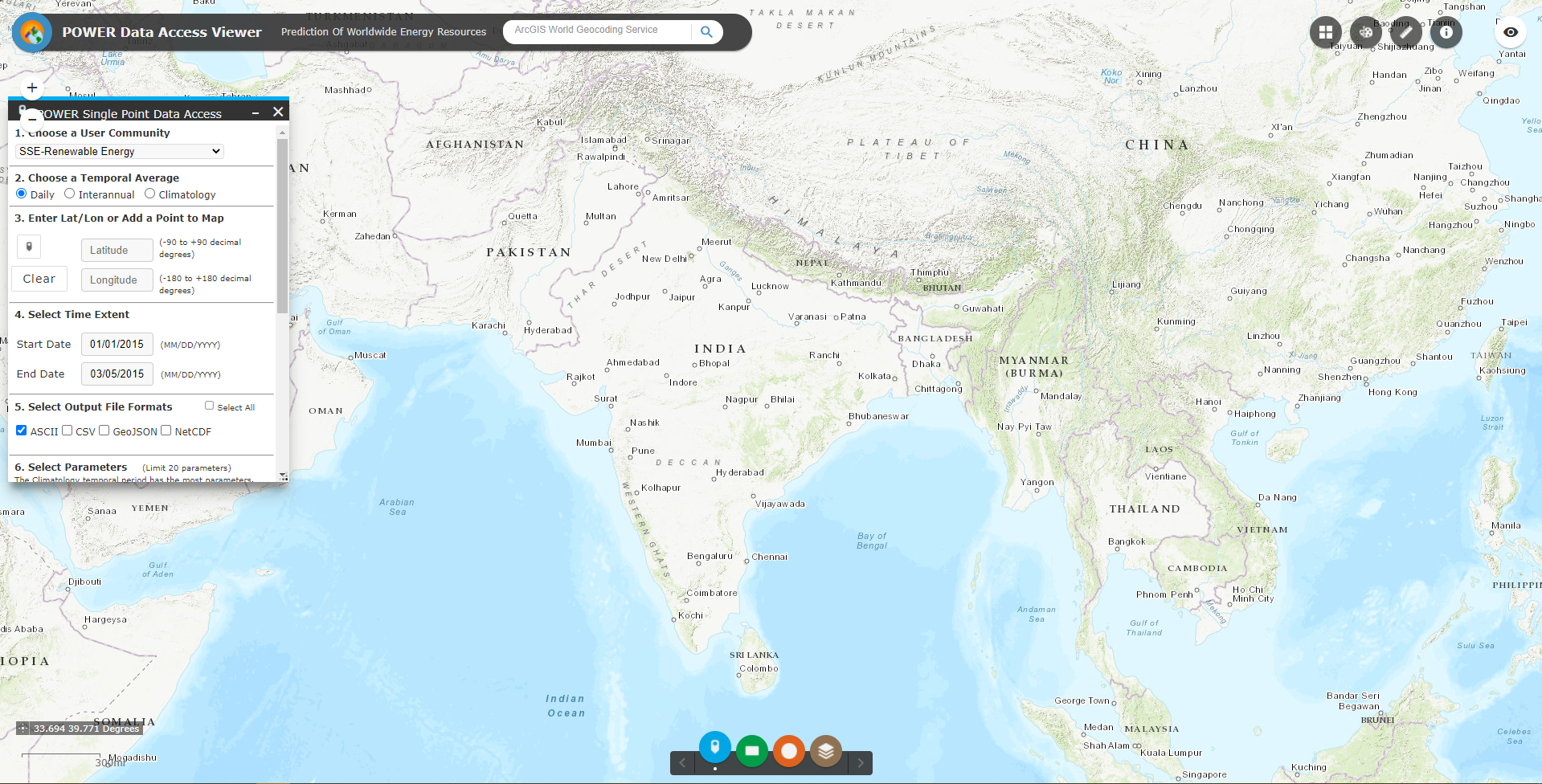
T2M\_MAX – Maximum temperature at 2 meters

T2M\_MIN – Minimum temperature at 2 meters

T2M\_RANGE – Temperature range at 2 meters

WS10M\_RANGE – Wind speed at 10 meter range

WS50M\_RANGE – Wind speed at 50 meter range



# **Program:**

We will be given a CSV file with coordinates (longitude and latitude) in it, the output of this program should create CSV or GeoJSON files which include the date, longitude and latitude with weather data.

1. Importing necessary modules.

import os, sys, time, json, urllib3, requests, multiprocessing

urllib3.disable\_warnings()

import numpy as np

import pandas as pd

2. Define a class and a method naming Operation() and Get() respectively

Def Get(Total):

Class Operation():

If \_\_name\_\_ == ‘\_\_main\_\_’:

3. The Operation() class will be having a constructor and a method

def \_\_init\_\_(self):

def Perform(self):

4. Constructor have all the variables that needed to be retrieved like, number of processes, start date, end date, parameters, api url, file directory, latitude and longitude.

self.processes – Number of parallel processors needed (should be < 10)

start\_date – from this date the data will be retrieved

end\_date - upto this date the data will be retrieved

parametersCustom – defining the parameters which we should retrieve

self.API\_URL – this attribute contains the url from which the data will be retrieved using the parametersCustom attribute from the start\_date to end\_date.

self.FILE\_DIR – contains the directory where the CSV or GeoJSON file should be saved at.

self.tempLatitude – contains the latitude of the present point which was imported from the given csv.

self.tempLongitude - contains the longitude of the present point which was imported from the given csv.

self.messages – contains any error message occurred while running the program

def \_\_init\_\_(self):

self.processes = 10

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 = {}

5. Coming to Perform() method, this method executes the program with the defined attributes in the constructor. Begin time will be assigned to a variable and a list name Latitude\_Longitude will be derived with no length, after that the CSV given will be read into a dataframe object using a predefined method in pandas module called read\_csv(). The Imported CSV file contains 3 columns, X for latitude, Y for longitude and W\_GIDGID as the coordinated id. Using that dataframe we will be retrieving the data and assigning them in a list created before.

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 = []

6. In this ‘for’ loop we will be assigning list values row by row into the variables. The format() function in each line will replaces the present value to the variable created in the constructor. After that a points() list will be collecting the assigned variable and using multiprocessing.Pool() method these variable will be sent to constructor. After sending data to constructor, using MP\_POOL.imap\_unordered() method we will be sending POINTS data to Get() method we defined earlier

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 = []

7. This for loop will be executed after sending POINTS data to Get() method. For each point read from CSV this whole process repeats, this for loop represents the percentage of work done it will out the progress bar using the equation

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"])

8. Now coming to Get() method, this methods gets the POINTS data from the MP\_POOL.imap\_unordered() method define in the Perform() Class. Using the data we get we will be creating a CSV file and storing every data we get for each point in the CSV in a GeoJSON or CSV format

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)

DataFrame.rename(columns={"Unnamed: 0":"Dates"},inplace=True)

DataFrame["Latitude"]=tempLatitude

DataFrame["Longitude"]=tempLongitude

DataFrame.to\_csv(FILE\_DIR,index=False,header=True)