Program Explanation

```
1   import pandas as pd
2   import os,rioxarray,warnings,json,datetime
```

- Importing modules which are necessary

```
4   warnings.filterwarnings("ignore")
5   polygons_csv_file_path = "test.csv"
6   indice_folder_path = "D:/JOB/GIS/T43PHR_PROJECT/INDICES"
7   json_file = "D:/JOB/GIS/T43PHR_PROJECT/TEST/JSON/only_ndvi.json"
8   cus_struc = { 'data' : {} }
9   user_indice_list = ('ndvi',"evi2","ndvi","ndwi","npcri","savi",)
10  csv_df = pd.read_csv(polygons_csv_file_path)
11  indice_folders = os.listdir(indice_folder_path)
```

- hile using this program we get some warning from rioxarray module, so to suppress those warning we use warnings.filterwarnings(“ignore”), which will ignore all the warning displayed while running program
- polygons_csv_file_path variable consists the csv file path which have multipolygon boundaries in WKT column and polygon id’s in the PID column as described in the pic below

	WKT	PID
2	MULTIPOLYGON (((866771.477990848 1531373.61415218,866752.691180507 1531379.09180564,866759.622570536 1531396.59059531,866767.843133348 1531390.20242251,866773.120763156 1531382.86493335,866771.477990848 1531373.61415218)))	P_1
3	MULTIPOLYGON (((866781.722681178 1531359.99366804,866749.005949695 1531369.78819105,866752.691180507 1531379.09180564,866771.477990848 1531373.61415218,866786.768902424 1531368.99012879,866781.722681178 1531359.99366804)))	P_2
4	MULTIPOLYGON (((866758.429788906 1531345.14169659,866773.026379233 1531340.03313068,866766.404112627 1531324.83313645,866754.970720232 1531328.45292974,866758.429788906 1531345.14169659)))	P_3
5	MULTIPOLYGON (((866780.789249043 1531312.35196799,866805.605021894 1531301.94156376,866802.725270955 1531291.21435037,866801.019299655 1531291.7414606,866776.76237079 1531303.19691345,866780.789249043 1531312.35196799)))	P_4
6	MULTIPOLYGON (((866741.543102166 1531349.90212734,866758.429788906 1531345.14169659,866754.970720232 1531328.45292974,866744.246326793 1531323.76288992,866734.290838916 1531327.40362751,866741.543102166 1531349.90212734)))	P_5
7	MULTIPOLYGON (((866773.026379233 1531340.03313068,866758.429788906 1531345.14169659,866741.543102166 1531349.90212734,866743.353038861 1531355.51705408,866749.005949695 1531369.78819105,866781.722681178 1531359.99366804,8667	P_6
8	MULTIPOLYGON (((866763.726775791 1531313.76363918,866779.207348314 1531308.75553645,866776.76237079 1531303.19691345,866770.313215275 1531288.88611351,866758.584227046 1531292.50172161,866763.726775791 1531313.76363918)))	P_7
9	MULTIPOLYGON (((866804.549659511 1531336.77636929,866793.32769217 1531340.85801307,866801.948525923 1531360.45745899,866810.736633123 1531357.07678927,866804.549659511 1531336.77636929)))	P_8
10	MULTIPOLYGON (((866793.32769217 1531340.85801307,866776.113571695 1531347.11912842,866781.722681178 1531359.99366804,866799.738040907 1531361.30780419,866801.948525923 1531360.45745899,866793.32769217 1531340.85801307)))	P_9

- indice_folder_path variable consists the directory where the calculated indices are saved.
- json_file variable holds the json file name with the directory.
- cus_struc is dictionary which we are going to create for the data we retrieve from time series.
- User_indices_list is list variable in which indices are added, if you don’t need any indice just remove it from the list but do not remove the (,) at the end of the list.
- We will be reading the polygons_csv_file_path using pandas and assigning it to a dataframe called csv_df

- Now we need to get the folders inside indice directory, for that we will be using os library's listdir() function which will scan all the folders inside a folder and assign the output list to indice_folders variable.

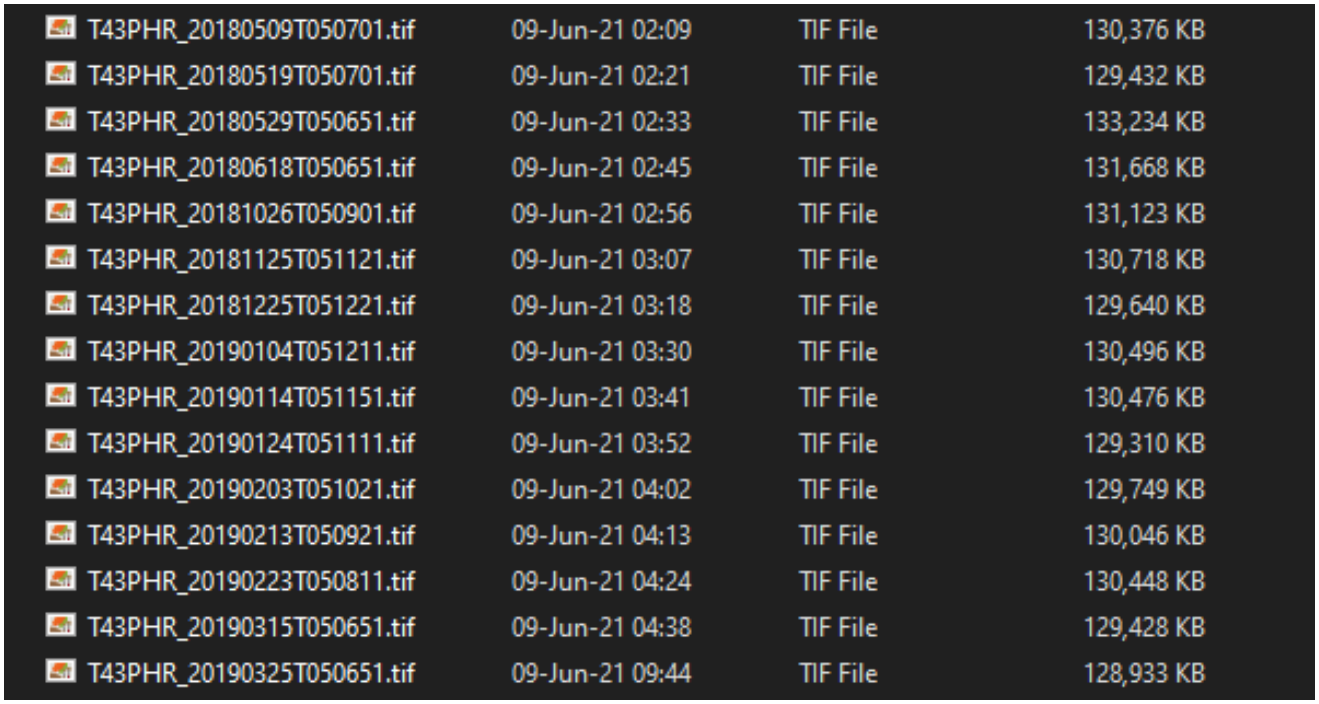
```
12  for indice in indice_folders:
13      if(indice in user_indice_list):
14          print("_________"+indice+"_________",sep="",end="\n")
15          cus_struc['data'][indice] = {}
16          indice_dir = indice_folder_path+"/"+indice
17          files_in_indice_dir = os.listdir(indice_dir)
18          for file in files_in_indice_dir:
19              date = file[7:15]
20              date = str(date[0])+str(date[1])+str(date[2])+str(date[3])+"-"+str(date[4])+str(date[5])+"-"+str(date[6])+str(date[7]
21              start_time = datetime.datetime.now()
22              print(" "+date+" ",sep="",end="")
23              cus_struc['data'][indice][date],r = [],0
```

- We will start looping each indice from indice_folders list for an iteration

- Then check if this indice is in any of indices described in user_indice_list, if true the program gets into the if condition or else it will iterate again and checks again until it reads end of list

- When program gets into if condition we will print which indice is going through loop just to be sure, then we will define the cus_struc dictionary with this indice with value as dictionary.

- After that we will be scanning for indice files inside that indice folder by joining indice_folder_path and indice variables which will act a path to the particular indice in that iteration, then again we will listout files within that folder using os.listdir() and assign the list to files_in_indice_dir variable.



T43PHR_20180509T050701.tif	09-Jun-21 02:09	TIF File	130,376 KB
T43PHR_20180519T050701.tif	09-Jun-21 02:21	TIF File	129,432 KB
T43PHR_20180529T050651.tif	09-Jun-21 02:33	TIF File	133,234 KB
T43PHR_20180618T050651.tif	09-Jun-21 02:45	TIF File	131,668 KB
T43PHR_20181026T050901.tif	09-Jun-21 02:56	TIF File	131,123 KB
T43PHR_20181125T051121.tif	09-Jun-21 03:07	TIF File	130,718 KB
T43PHR_20181225T051221.tif	09-Jun-21 03:18	TIF File	129,640 KB
T43PHR_20190104T051211.tif	09-Jun-21 03:30	TIF File	130,496 KB
T43PHR_20190114T051151.tif	09-Jun-21 03:41	TIF File	130,476 KB
T43PHR_20190124T051111.tif	09-Jun-21 03:52	TIF File	129,310 KB
T43PHR_20190203T051021.tif	09-Jun-21 04:02	TIF File	129,749 KB
T43PHR_20190213T050921.tif	09-Jun-21 04:13	TIF File	130,046 KB
T43PHR_20190223T050811.tif	09-Jun-21 04:24	TIF File	130,448 KB
T43PHR_20190315T050651.tif	09-Jun-21 04:38	TIF File	129,428 KB
T43PHR_20190325T050651.tif	09-Jun-21 09:44	TIF File	128,933 KB

- Again we start inner loop for each file from files_in_indice_dir list, now we will get the filenames

- Each file name consists of the date in it so by using file[7:15] which mean file is the variable having the file name in it, 7:15 means starting cutting the file string from 7th position and up to 15th position so that we will get the date and store it in date variable

- Then again we format the output date into YYYY-MM-DD and store it in same date variable by overlapping.

- start_time variable will have the current time stored in it and by the end of this loop we will be calculating the time taking to for one indice and print that out with other data.

- Now again we will be adding the date variable to the earlier define cus_struc dictionary with value as an array so that we will be storing arrays in the date key of the cus_struc dictionary. And assign r variable a 0 which will be used afterawards.

```
    for polygon in csv_df['WKT']:
        indice_file_path = indice_dir+"/"+file
        polygon_id = csv_df['PID'][r]
        r = r + 1
        poly_geo = polygon[16:len(polygon)-3]
        poly_geo = poly_geo.replace(","," ")
        poly_geo = poly_geo.split()
        gcords,coords = [],[]
        for s in range(0,len(poly_geo),2):
            coords.append([float(poly_geo[s]),float(poly_geo[s+1])])
        gcoords = [coords]
        geometries = [
            {
                'type': 'Polygon',
                'coordinates': gcoords
            }
        ]
        clipped = rioxarray.open_rasterio(indice_file_path,masked=True,).rio.clip(geometries, from_disk=True)
        clip_np = clipped.values
        x=len(clip_np[0])
        y=len(clip_np[0][0])
        str_len,indice_min,indice_max,indice_mean,indice_sum = 0,99999,-99999,0,0
```

- Now we will loop through each and every WKT column's row which contains the boundaries of the polygon and we need to extract all polygons data from each file which is scanner earlier.
- Indice_file_path variable holds the string values of path of the particular file scanned in earlier loop
- Then again we take out the polygon id from the specific column and assign it to polygon_id variable
- We increment r which will be used to get the polygon id from the specific row.
- Now we will scan the multipolygon string which have coordinates in it and cut first 16 characters and last 3 characters which gives us the longitudes and latitudes then assign the string to poly_geo variable
- Then again we will replace the character (,) in the string with a space so that there will be space after each and every longitude and latitude which will be easy to differentiate which will be done in next step.
- Now we will split the string into a list using the predefined function split() which takes a string and return the list of words in it considering the spaces in between the words it will discriminate them as words and save in the same variable poly_geo overlapping with old one cuz we don't need old values of that variable.
- After splitting define 2 empty arrays gcords,cords which will be used afterwards.
- Start a for loop with a range equal to the range of the list created using split.

- In the for loop we will be assigning the first read coordinate and second read coordinate into an array with a set.

- This is how the geometries should be when we are trying to clip the required area from a tif file.

- Then again we will be defining a dictionary with the same structure then store it in geometry variable.

- Using rioxarray module we will be clipping the required area from the indice file path then store the values for every pixel in clipped variable which will be in rioxarray format.

- Using clipped.values we will only get the values of every pixel it read in the area clipped then again assign it into a variable clip_np.

- Rioxarray are nothing but a matrix so by defining the x,y with the limits of rioxarray we should be able to get each and every cell from it.

- Define str_len, indice_min, indice_max, indice_mean, indice_sum as 0, 99999, -99999, 0, 0 respectively which will be used in the loop we are about to create.

```
                for i in range(x):
                    for j in range(y):
                        if str(clip_np[0][i][j]) != "nan":
                            temp_min = str(clip_np[0][i][j])
                            if float(temp_min) <= float(indice_min):
                                indice_min = temp_min
                for i in range(x):
                    for j in range(y):
                        if str(clip_np[0][i][j]) != "nan":
                            temp_max = str(clip_np[0][i][j])
                            if float(temp_max) >= float(indice_max):
                                indice_max = temp_max
                for i in range(x):
                    for j in range(y):
                        if str(clip_np[0][i][j]) != "nan":
                            indice_mean = indice_mean + clip_np[0][i][j]
                            str_len = str_len + 1
                if str_len != 0:
                    indice_mean = indice_mean/str_len
                for i in range(x):
                    for j in range(y):
                        if str(clip_np[0][i][j]) != "nan":
                            indice_sum = indice_sum + clip_np[0][i][j]
                polygon_structure = {"PID":polygon_id,"Geometry":polygon,"min":indice_min,"max":indice_max,"mean":indice_mean,"sum":indice_sum }
                cus_struc['data'][indice][date].append(polygon_structure)
            end_time = datetime.datetime.now()
            print("Calculated Min,Max,Mean,Sum of "+str(r)+" Polygons, Time Taken : "+str(end_time-start_time)+"(HH:MM:SS)",sep="",end="\n")
with open(json_file,'w') as fp:
    json.dump(cus_struc,fp)
```

- There will be 4 for loops which will be used to calculated the min, max, mean and sum of the pixels values that are saved in the rioxarray.

- In the first for loop with I in range of x, then again a nested loop j in range of y now we will be selecting each item from the rioxarray using this, and check if they are “nan” which means nodata which means no pixel, so if its true if will come to into the if else and performs the least number from an array operation. This loop goes on until it found a lowest number of all the numbers scanned. And store the value in temp_min.

- In the second loop we will be taking maximum out of all the values scanned from rioxarray, this is same as the first loop there will be a change in symbols (<=) into (>=) and store the highest number in temp_max.

- In this third loop we will be calculating the sum of all the values from the pixels in rioxarray and store it in indice_mean. Which will be used to calculate the mean in next step.

- Now we should check if str_len is 0 to calculated the mean if its not then we will divide the sum/str_len if not then indice_mean will be the mean.

- In the last loop we will be calculating the sum of all pixels scanned in rioxarray same as the 3rd loop.

- We will be having polygon_id its geometry indice_min and indice_max and indice_mean and indice_sum. Now we will be assigning all of this data into a dictionary variable called polygon_structure, then append this dictionary to the array we created in the nested dictionary of cus_struc variable.

- So for every polygon we will be appending the dictionary, and if the file changes the array will be appended to the newly created array in that file dictionary of cus_struc dictionary variable.

- At the end we will calculated the time take for calculating this whole process for on indice which should be arraound 5-7min depending upon your CPU usage.

- print the data, time taken. At the end write the cust_struc dictionary into a json file.