ISS-LIS_download files_based on user input of time & region of interest_netcdffile filtered_v2.0

Earthdata API is used to access and query NASA data, for details: https://pypi.org/project/earthdata/

In [16]:
```
import earthdata
from earthdata import Auth, Store, DataCollections, DataGranules
```

To access the cell below one should have to create a .netrc file in home directory, please follow the given steps below:-

BY CMD IT, COULD BE DONE AS FOLLOWS: (1) cd ~ or cd $HOME, (2) touch .netrc, (3) echo "machine urs.earthdata.nasa.gov login password " >> .netrc (Note that here is your user name and is your password used for Earthdata search Login, enter both without the brackets), (4) chmod 0600 .netrc (so only you can access it)

OR

BY MANUAL PROCESS, IT COULD BE DONE AS FOLLOWS: (1) Go to dir (find your dir: On Windows, this might look like 'C:\Users\\', on a Mac '/opt/'), (2) Open notepad, and paste the content in the brackets "machine urs.earthdata.nasa.gov login password " (Note that here is your user name and is your password used for Earthdata search Login, enter both without the brackets), (3) Save the notepad as '.netric'

Finally once the netric file is successfully created in your dir, you would be able to RUN the next cell and it would show the output as "You're now authenticated with NASA Earthdata Login, True"

In [17]:
```
### AUTHENTICATION STEP

auth = Auth()
auth.login(strategy="netrc")
# are we authenticated?
print(auth.authenticated)
```

```
You're now authenticated with NASA Earthdata Login
True
```

**Note here we need NASA authentication to access the files directly from GHRC server*

We can search for collections using a pythonic API client for CMR

###Locate DAAC (in oue case it is GHRC-DAAC to access LIS files)### #Query = DataCollections().daac("GHRCDAAC") ###Find collections in the mentioned DAAC### #print(f'Collections found: {Query.hits()}') #collections = Query.fields(['ShortName']).get(10) ###Printing collection 6 of our interest which has ISS science data### #collections[6]

Please enter the date from 2017-03-01 onwards till date

Input start time of interest

In [18]:
```
start_time = input('Enter start time in format: YYYY-MM-DD \n')
```

```
Enter start time in format: YYYY-MM-DD
2022-11-12
```

Input end time of interest

In [19]:
```
end_time = input('Enter end time in format: YYYY-MM-DD \n')
```

```
Enter end time in format: YYYY-MM-DD
2022-11-13
```

LET now find Non-Quality Controlled Lightning Imaging Sensor (LIS) on International Space Station (ISS) Science Data V2,
LIS granules for given dates and access their metadata using earthdata API get() method

In [20]:
```
%%time
### We build our query, note as ISS_LIS data comes 1file per orbit so we can temporally query the data###
### direct spatial query not possible in case of ISS_LIS data###
### The short name for collection was found from cell[2]###

from pprint import pprint
Query = DataGranules().short_name('isslis_v2_nqc').temporal(start_time,end_time)
```

```
###We get all metadata records###
granules = Query.get()

###Please uncomment to display granules
#print(granules)
```

```
CPU times: total: 62.5 ms
Wall time: 22.1 s
```

In [21]:
```
%%time
### Access bounding rectangle coordinates from each granules (as required to indirectly query ISS_LIS dat
### Note here these coordinates are ones when stellite is changing its orbit or doing some*
### *maintainance/instrumentation health check

space = []

for i in range(len(granules)):
    space_iss = granules[i]['umm']['SpatialExtent']['HorizontalSpatialDomain']['Geometry']['BoundingRecta

    ### to remove square brackets from start and end of each string
    space_iss_1 = str(space_iss)[1:-1]

    # change datatype from string to dict
    space_iss_1 = eval(space_iss_1)
    space.append(space_iss_1)

###Please uncomment to display bounding rectangle coordinates
#space
```

```
CPU times: total: 15.6 ms
Wall time: 8.98 ms
```

In [22]:
```
%%time
###Change the datatype from dict to dataframe

import pandas as pd
spatial = pd.DataFrame(space)

###Please uncomment to display table of bounding rectangle coordinates
#spatial
```

```
CPU times: total: 15.6 ms
Wall time: 998 µs
```

Now lets try to extract data URLS from the metadata of each datasets of our interest

In [23]:
```
%%time
data_links = [granule.data_links() for granule in granules]

###Please uncomment to display links
#print(data_links)
```

```
CPU times: total: 15.6 ms
Wall time: 6.02 ms
```

OOPS!! we are not able to access the GHRC DAAC server to directly download the files using get() method from earthdata library so lets take a long cut (atleast for time being)

'https' url for GHRC-DAAC server are sorted (from 's3' ie AWS links) and than the bounding rectangle coordinates dataframe is merged with 'https' sorted dataframe

In [24]:
```
%%time
###covert list to dataframe .. a long cut again :)
import pandas as pd
url = pd.DataFrame(data_links, columns = ['https' , 'sf3'])

###converted dataframe back to list sorting hhtps urls
url_https = url['https']#.values.tolist()
#print(inprem)
type(url_https)

### merge the bounding rectangle coordinates dataframe is merged with 'https' dorted dataframe
result_1 = pd.concat([spatial, url_https], axis=1, join="outer")

result_1.head()
```

```
CPU times: total: 15.6 ms
Wall time: 2.99 ms
```

Out[24]:

	WestBoundingCoordinate	EastBoundingCoordinate	NorthBoundingCoordinate	SouthBoundingCoordinate	
0	-156.066	178.875	51.773	-51.814	https://data.ghrc.e
1	-156.066	178.875	51.773	-51.814	https://data.ghrc.e
2	-175.257	152.025	51.758	-51.731	https://data.ghrc.e
3	-175.257	152.025	51.758	-51.731	https://data.ghrc.e
4	157.392	128.146	51.752	-51.813	https://data.ghrc.e

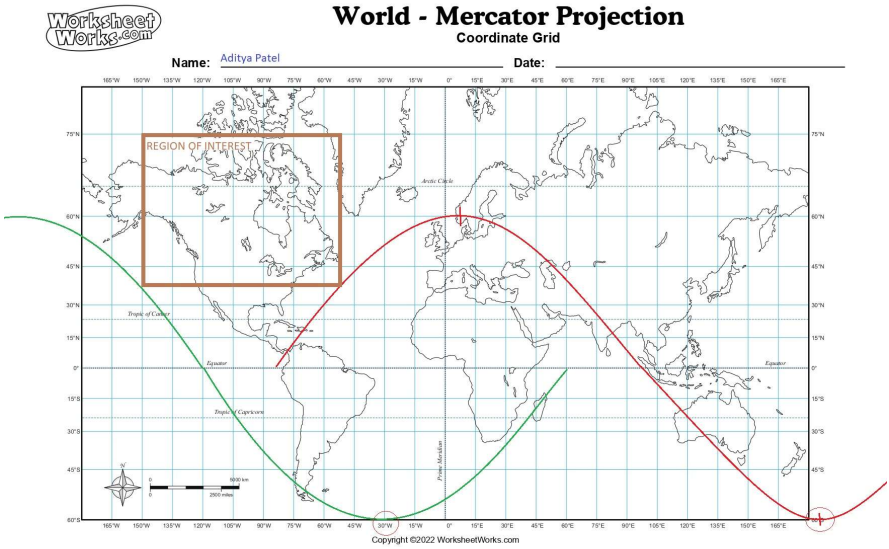
In [25]:

```
%%time
### An indirect method to sptially query ISS_LIS data wrt region of interest
### Finally we filter the iss_lis data based on the condition that whenever the bounding coordinates are

import pandas as pd
result = result_1[(result_1['WestBoundingCoordinate'] > 0) & (result_1['EastBoundingCoordinate'] > 0)]

### sorted coordinates
result
```

```
CPU times: total: 15.6 ms
Wall time: 7.98 ms
```

Out[25]:

	WestBoundingCoordinate	EastBoundingCoordinate	NorthBoundingCoordinate	SouthBoundingCoordinate	
4	157.392	128.146	51.752	-51.813	https://data.ghrc
5	157.392	128.146	51.752	-51.813	https://data.ghrc
6	133.512	105.785	51.782	-51.823	https://data.ghrc
7	133.512	105.785	51.782	-51.823	https://data.ghrc
8	111.355	83.314	51.766	-51.806	https://data.ghrc
9	111.355	83.314	51.766	-51.806	https://data.ghrc
10	88.687	60.448	51.779	-51.799	https://data.ghrc
11	88.687	60.448	51.779	-51.799	https://data.ghrc
12	65.821	36.481	51.780	-51.791	https://data.ghrc
13	65.821	36.481	51.780	-51.791	https://data.ghrc
14	41.855	12.627	51.758	-51.784	https://data.ghrc
15	41.855	12.627	51.758	-51.784	https://data.ghrc



NOTE: THE ISS_LIS SATELLITE BOUNDING RECTANGLES ARE DEFINED AT THE LOWER PEAK SINUSOIDAL WAVE (REF.ABOVE FIGURE), THUS, TO COVER REGION OF INTEREST WE CAN CONSIDER THE ORBITS FROM LONGITUDINAL COORDINATES FROM 30 DEGREE WEST TO 180 DEGREE EAST

In [26]:
```
%%time
### Now let's filter list URLs with extension '.nc'
### NOTE that here we sort netCDF4 files from list of HDF and NETCDF4 files
### Though generally HDF file size are smaller than netCDF4 files,netCDF4 files are easy to read and proc
### Using List comprehension method (filtering List of strings based on the substring list)

import re

### convert dataframe to back to list dtype
download_https = result["https"].values.tolist()

def Filter(download_https, substr):
    return [str for str in download_https if
              any(sub in str for sub in substr)]

substr = ['.nc']

download_https_nc = Filter(download_https, substr)
```

```
CPU times: total: 0 ns
Wall time: 0 ns
```

In [27]:
```
%%time
download_https_nc[:10]
```

```
CPU times: total: 0 ns
Wall time: 0 ns
```

Out[27]:
```
['https://data.ghrc.earthdata.nasa.gov/ghrcw-protected/isslis_v2_nqc__2/202211/ISS_LIS_SC_V2.2_20221112_0
14312_NQC.nc',
 'https://data.ghrc.earthdata.nasa.gov/ghrcw-protected/isslis_v2_nqc__2/202211/ISS_LIS_SC_V2.2_20221112_0
31602_NQC.nc',
 'https://data.ghrc.earthdata.nasa.gov/ghrcw-protected/isslis_v2_nqc__2/202211/ISS_LIS_SC_V2.2_20221112_0
44853_NQC.nc',
 'https://data.ghrc.earthdata.nasa.gov/ghrcw-protected/isslis_v2_nqc__2/202211/ISS_LIS_SC_V2.2_20221112_0
62144_NQC.nc',
 'https://data.ghrc.earthdata.nasa.gov/ghrcw-protected/isslis_v2_nqc__2/202211/ISS_LIS_SC_V2.2_20221112_0
75434_NQC.nc',
 'https://data.ghrc.earthdata.nasa.gov/ghrcw-protected/isslis_v2_nqc__2/202211/ISS_LIS_SC_V2.2_20221112_0
92725_NQC.nc']
```

In [28]:
```
%%time
download_https_nc[-10:]
```

```
CPU times: total: 0 ns
Wall time: 0 ns
```

Out[28]:
```
['https://data.ghrc.earthdata.nasa.gov/ghrcw-protected/isslis_v2_nqc__2/202211/ISS_LIS_SC_V2.2_20221112_014312_NQC.nc',
 'https://data.ghrc.earthdata.nasa.gov/ghrcw-protected/isslis_v2_nqc__2/202211/ISS_LIS_SC_V2.2_20221112_031602_NQC.nc',
 'https://data.ghrc.earthdata.nasa.gov/ghrcw-protected/isslis_v2_nqc__2/202211/ISS_LIS_SC_V2.2_20221112_044853_NQC.nc',
 'https://data.ghrc.earthdata.nasa.gov/ghrcw-protected/isslis_v2_nqc__2/202211/ISS_LIS_SC_V2.2_20221112_062144_NQC.nc',
 'https://data.ghrc.earthdata.nasa.gov/ghrcw-protected/isslis_v2_nqc__2/202211/ISS_LIS_SC_V2.2_20221112_075434_NQC.nc',
 'https://data.ghrc.earthdata.nasa.gov/ghrcw-protected/isslis_v2_nqc__2/202211/ISS_LIS_SC_V2.2_20221112_092725_NQC.nc']
```

Finally now, https urls for GHRC-DAAC server are opened using loop and temporarily sorted files are downloaded ;)

In [29]:
```
##Works https url opens and data is downloaded to downloads as default location
import time
import webbrowser


#### Please uncomment to download queried data from server
for i in download_https_nc:
    reponse = webbrowser.open(i)
    time.sleep(2)
```

THE CODE ENDs HERE

.

*Management of huge sized data files from NASA is a big challenge and so in this code not only helps to get connected with NASA server using API to get ISS_LIS data files but also further enables to query the data wrt time , bounding rectangle coordinates according to ones research interest and netCDF4 file type**

ALL THE FILTERING EFFORTS ARE JUST DONE TO OPTIMALLY USE AVAILABLE SPACE IN COMPUTER*

AS SEEN BELOW, a significant reduction in number of datefiles to be downloaded can be clearly observed ~1/5th wrt the initial total datafiles

In [15]:
```
### Total number of data files after inputing time period of interest
len(granules)
```

Out[15]:
```
968
```

In [16]:
```
### Number of data files would be reduced due to orbital sorting (orbits passing over region of interest:
len(result)
```

Out[16]:
```
394
```

In [17]:
```
### Finally further reduction in the number of datafiles is achieved by sorting netcdf files
len(download_https_nc)
```

Out[17]:
```
197
```

the code ends here

Issues yet to be addressed

ISSUE: 01_NOT WORKING: code not excecuting

Preference:

High priority

Details: ISS files are available in HDF4 and NETCDF4 formats

In the code we have sorted netCDF4 files though generally HDF file size are smaller than netCDF4 files,as netCDF4 files were found

to be easy to read and process in python (if HDF4 file reading and processing technique known this would save lot of disk space)

Related efforts&resources: HDF4 files exploration using rioxarray, example:https://www.earthdatascience.org/courses/use-data-open-source-python/hierarchical-data-formats-hdf/open-MODIS-hdf4-files-python/

import rioxarray as rxr

iss_lis = rxr.open_rasterio('/C/Users/Aditya/ISS_LIS_SC_V2.1_20180

ISSUE: 02_Save all LIS files in a given folder (How to download it to a specific path?)

Preference:

High priority

Details: The webbrowser function used in the code to download queried files actually opens the links on webbrowser and then saves all files in the default download folder location on PC. If we want to analyses each set of lis downloaded data than will have to create a new folder to download lis files there.

ISSUE: 03_NOT WORKING: General method to download data by earthdata API but no data is downloaded from s3 links for ISS_LIS

Preference:

Medium priority

Details: Not able to access the files from local DAAC server directly using get() method

(By default the AWS links gets accessed and no file is downloaded)

Related efforts&resources:

(a quick try; please check open the below links of a same given dataset and check if you are able to download the data using s3 link: https://data.ghrc.earthdata.nasa.gov/ghrcw-protected/isslis_v2_nqc__2/202105/ISS_LIS_SC_V2.1_20210501_223343_NQC.nc s3://ghrcw-protected/isslis_v2_nqc__2/202105/ISS_LIS_SC_V2.1_20210501_223343_NQC.nc)

```
#data_links = [granule.data_links(access="direct") for granule in granules]
##or if the data is an on-prem dataset
#data_links = [granule.data_links(access="onprem") for granule in granules]
##The Store class allows to get the granules from on-prem locations with get()
```

NOTE: Some datasets require users to accept a Licence Agreement before accessing them

```
#store = Store(auth)
#files = store.get(granules, local_path="./data/")
```

ISSUE: 04

Preference:

Low priority

It can be seen in the whole code that the datatype is changed from dict to list to pandas dataFrame and this can help to shorten the length of the overall code

All new ideas to excecute above code more efficiently are always welcomed and Thank you very much for contributing your precious time here! :)