Visualizing and Analyzing GOES-R Data Using Google Colab

Diego Souza
diego.souza@inpe.br
Engineer - INPE (Brazil)
National Institute for Space Research

Marcial Garbanzo
marcial.garbanzo@ucr.ac.cr
Professor - UCR
University of Costa Rica

Presentation Outline

- Satellite Data Access and Processing
- Introduction to Google Colab
- GOES-R Data Processing Examples
- Additional Resources

Satellite Data Access and Processing

GRB

HRIT/EMWIN

HRD

GEONETCast

INTERNET / CLOUD SERVICES

amazon web services

Environmental Data Commons

Google Cloud Platform

unidata IDD

LONG-TERM ARCHIVE

NOAA CLASS

EUMETSAT Data Centre

WEB PAGES

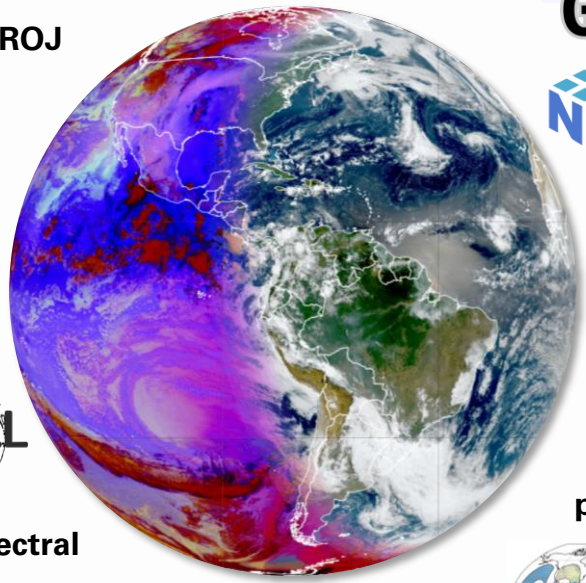
ANACONDA

python

netcdf4

GDAL

pyPROJ



MetPy

NumPy

Folium

PyOrbital

PyTROLL

Siphon

pyHDF

PySpectral

matplotlib

Cartopy

pyGRIB

SciPy

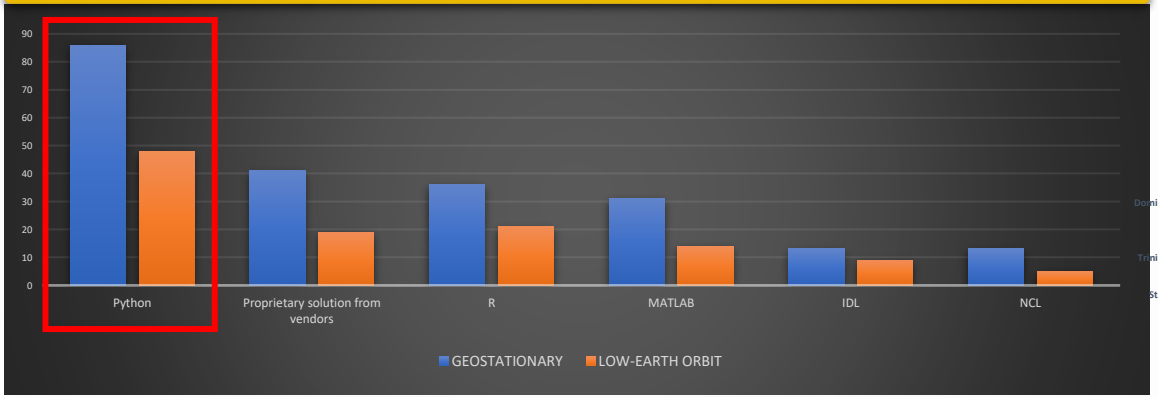
... and much more

The Python Programming Language

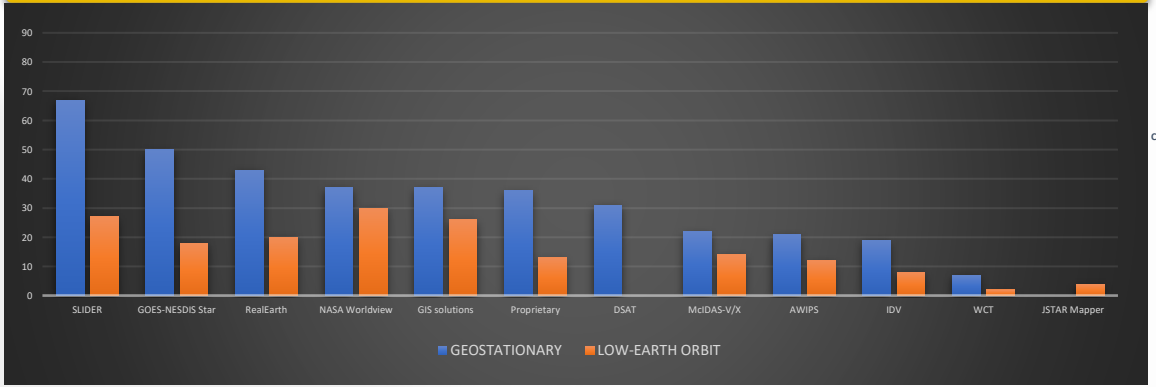
- Freely Available
- Easy to Get Started
- Gradual Learning Curve
- (Very) Active Community
- Many (Many!) Libraries / Modules
- Multiple Input and Output Formats
- Works on Different Operating Systems
- Integration with Other Software / Applications

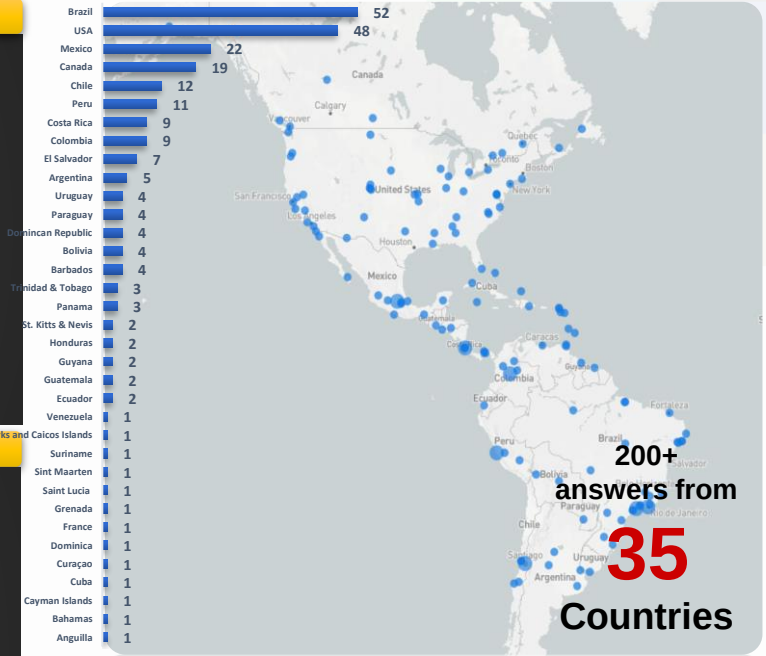
Use of Python by the Satellite Community

9. Data processing - Please indicate the tools you currently use for GEO and LEO satellites



10. Data visualization - Please indicate the tools you currently use for GEO and LEO satellites



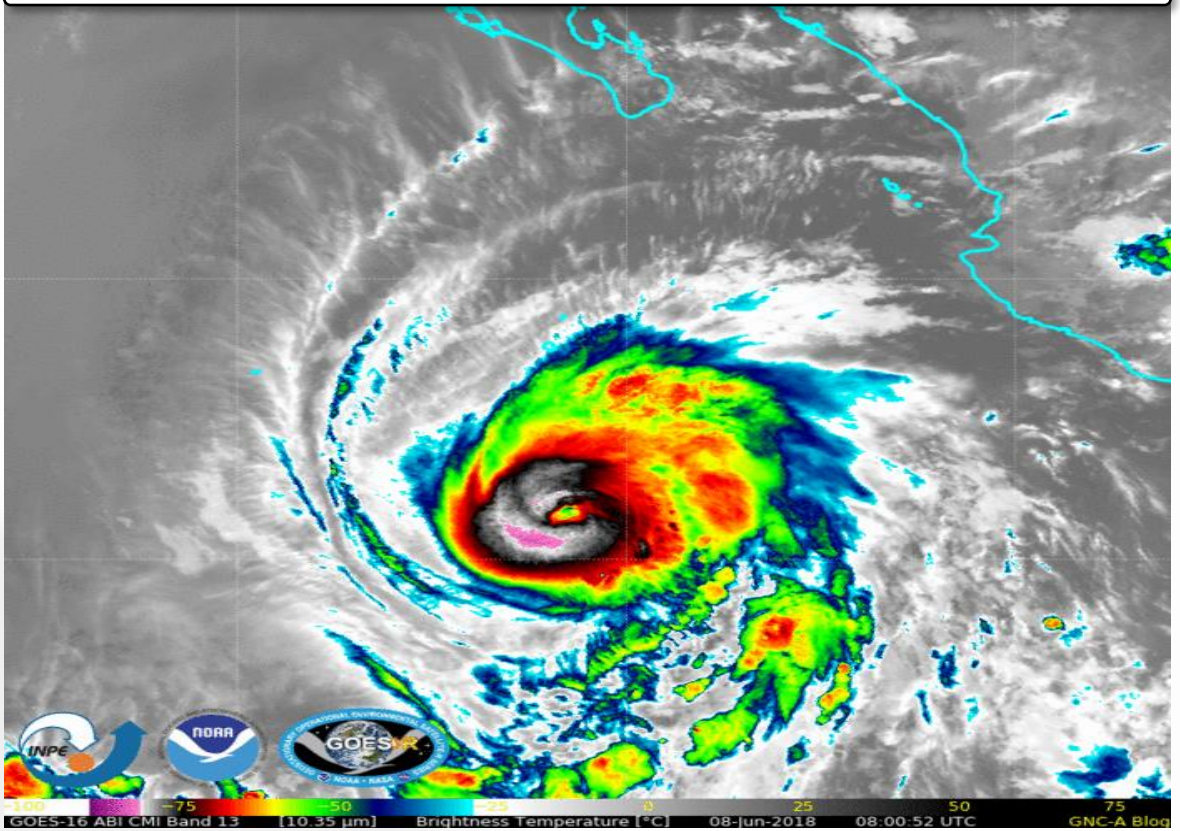


200+ answers from 35 Countries

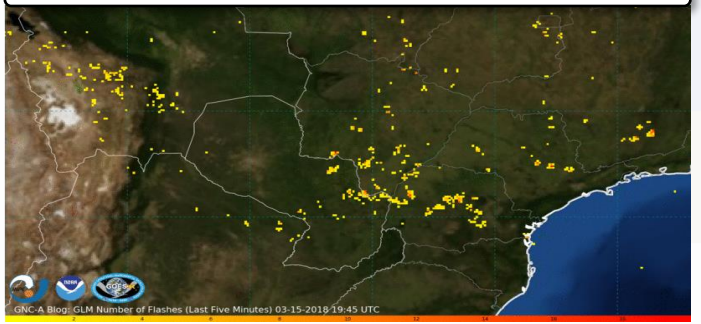
The results of the WMO Regional Survey can be seen at the following link: Link

What Can You Do With Python? Some Examples

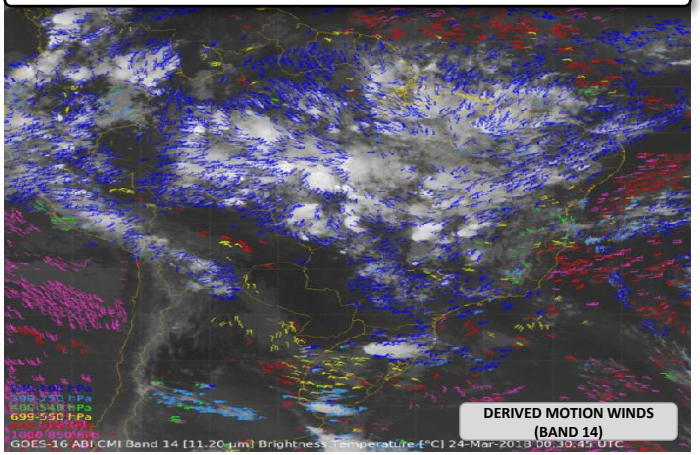
GOES-R ABI BANDS



GOES-R GLM (GEOSTATIONARY LIGHTNING MAPPER)

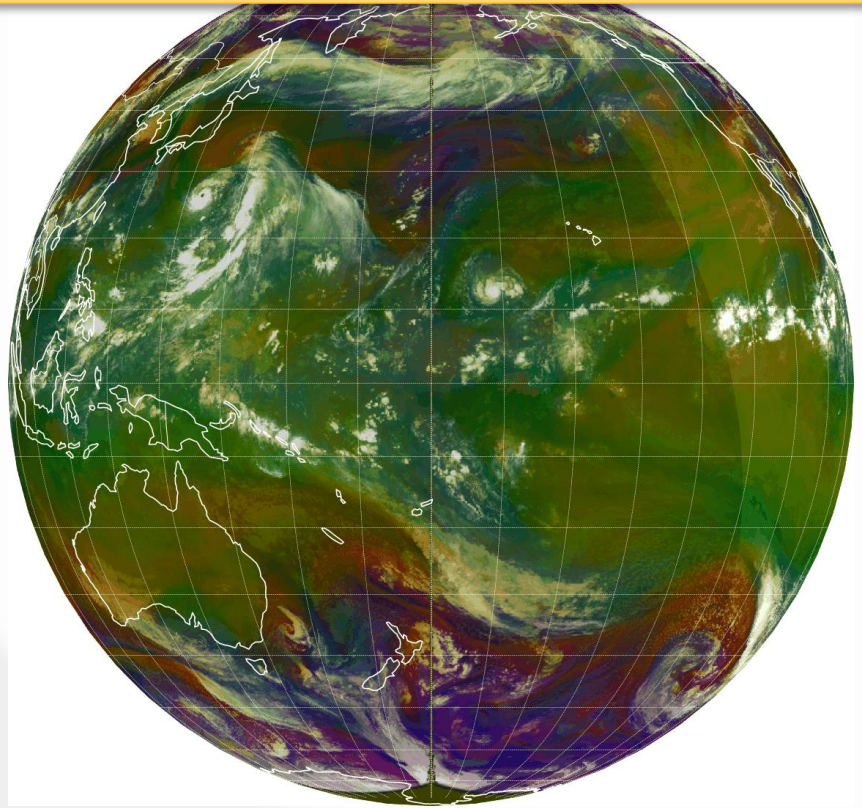


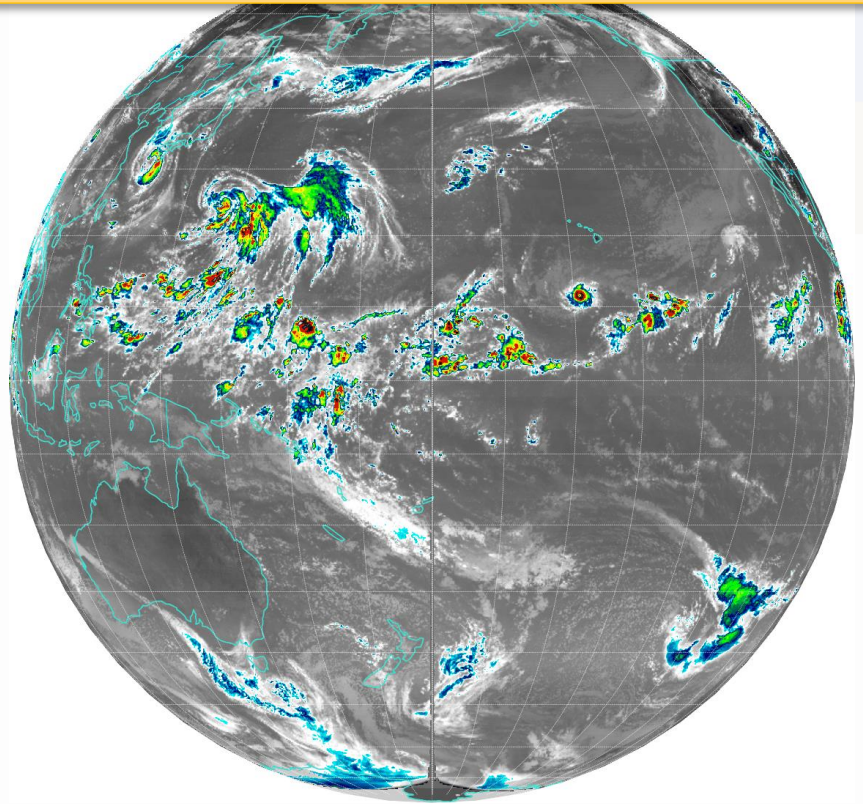
GOES-R DERIVED PRODUCTS



What Can You Do With Python? Some Examples

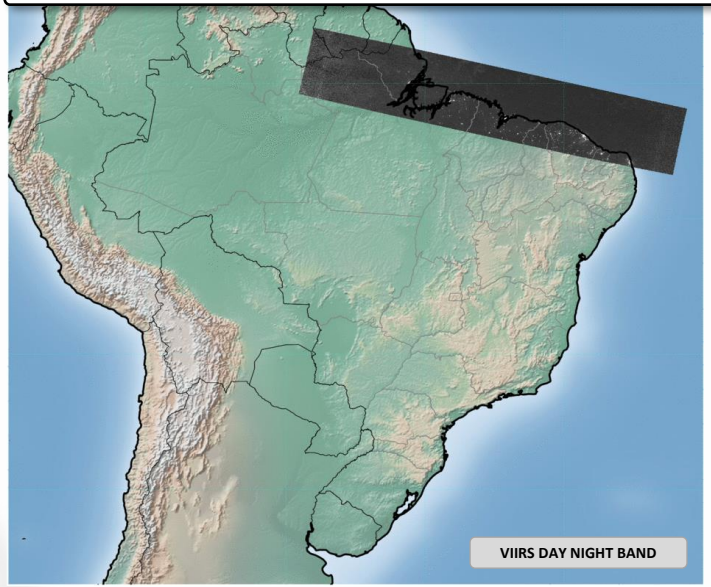
GOES-East / GOES-West (NOAA) , Meteosat 0° / Meteosat 45.5°E (EUMETSAT) and Himawari-8 (JMA)

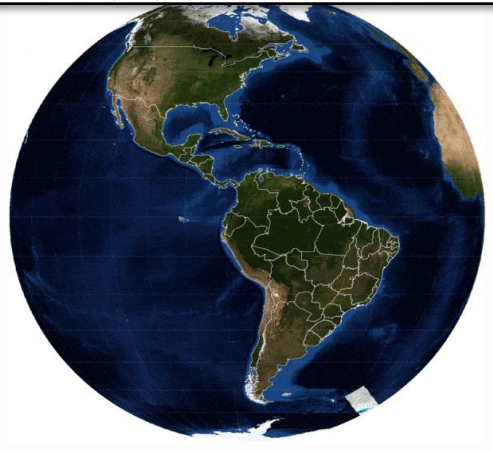


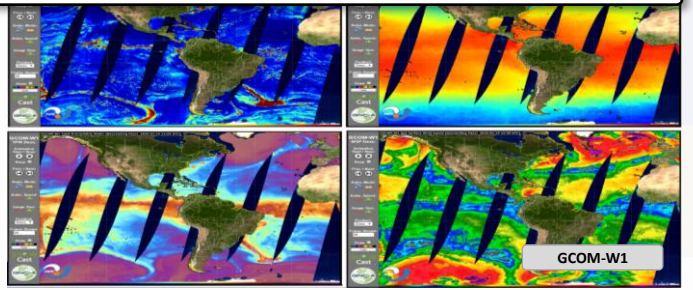


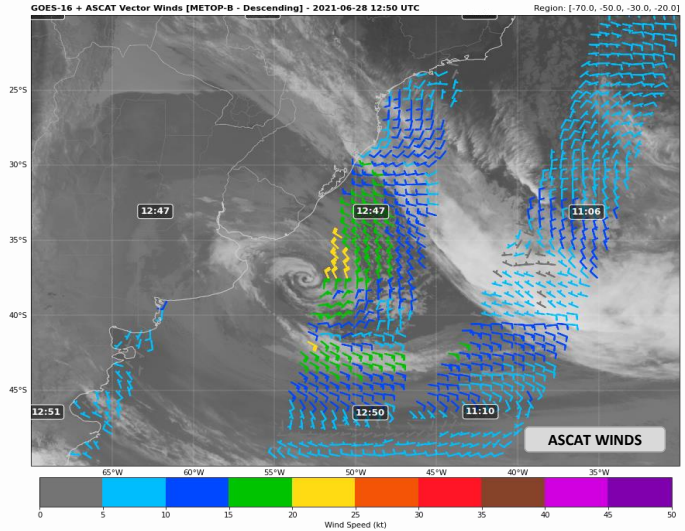
What Can You Do With Python? Some Examples

LOW-EARTH ORBIT (LEO) SATELLITES - IMAGERY AND PRODUCTS



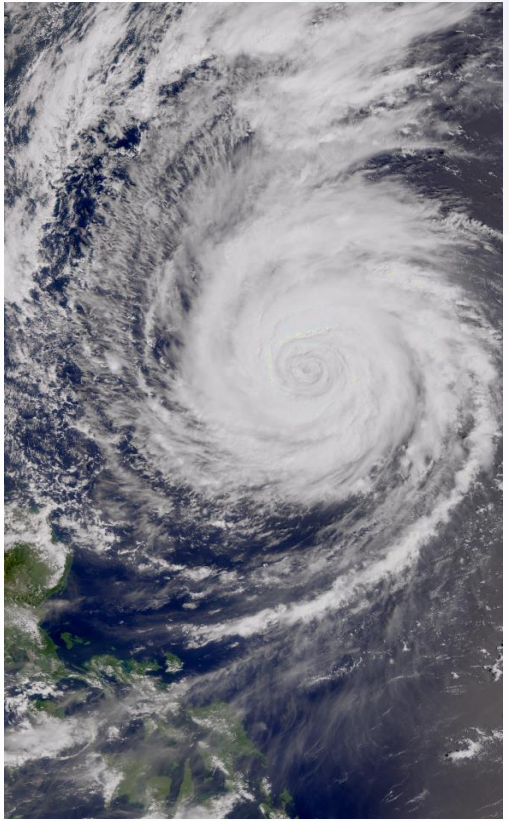






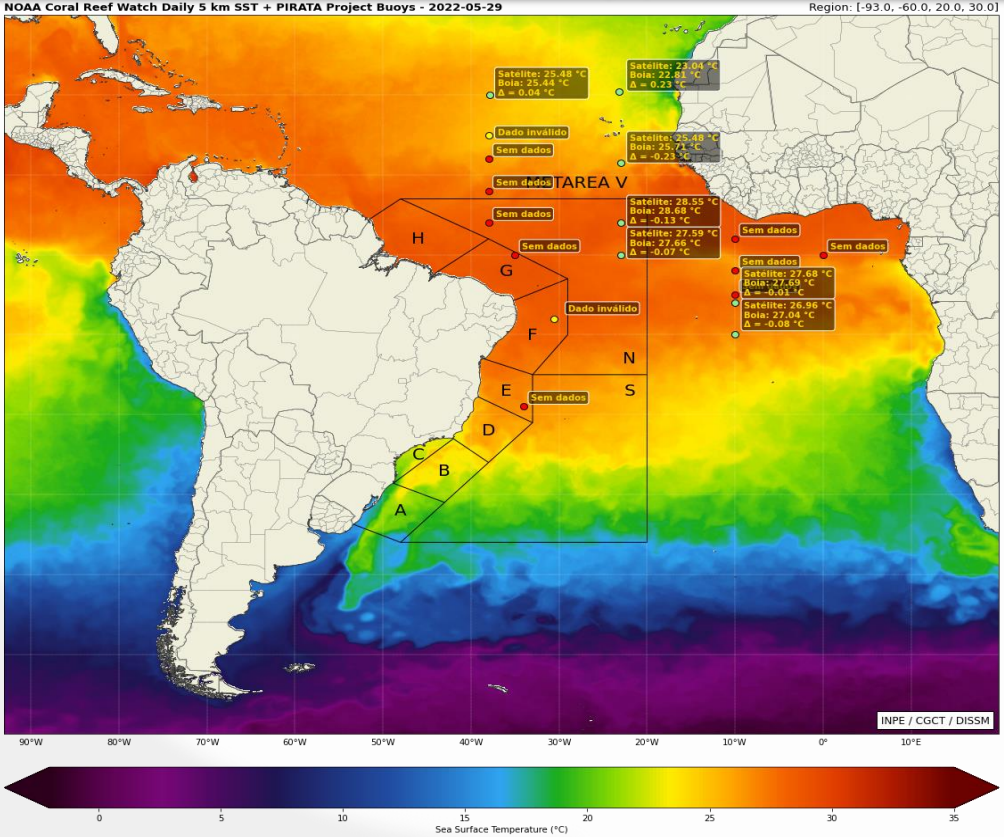
What Can You Do With Python? Some Examples

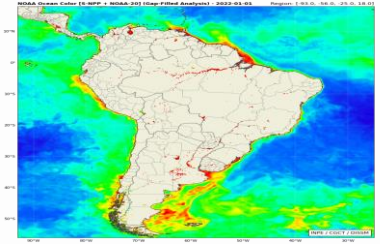
HIGH RESOLUTION IMAGERY (IN THE EXAMPLES BELOW, SENTINEL-2 AND SENTINEL-3 IMAGERY)

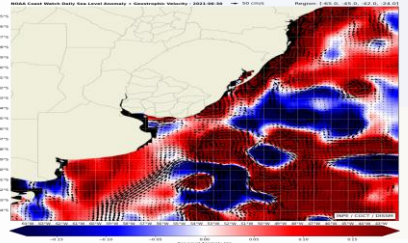


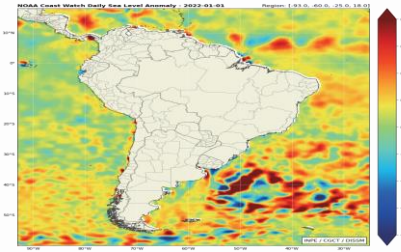
What Can You Do With Python? Some Examples

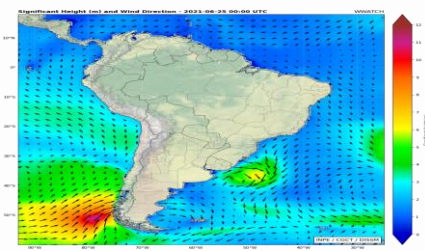
OCEAN PRODUCTS (SST, SST-ANOMALIES, ALTIMETRY, CHLOROPHYLL, GEOSTROPHIC CURRENTS, ETC)

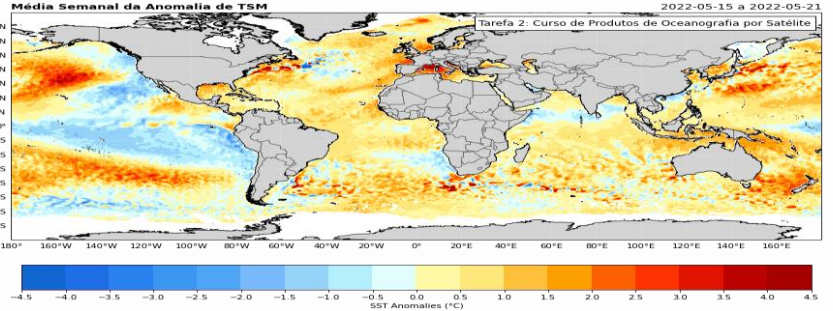






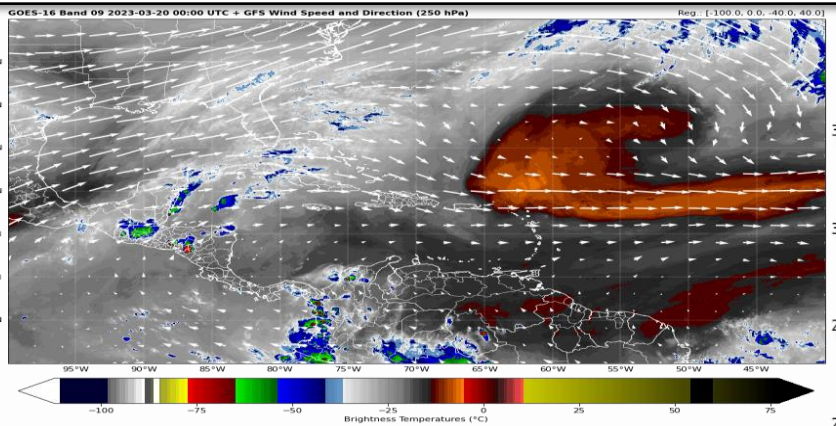


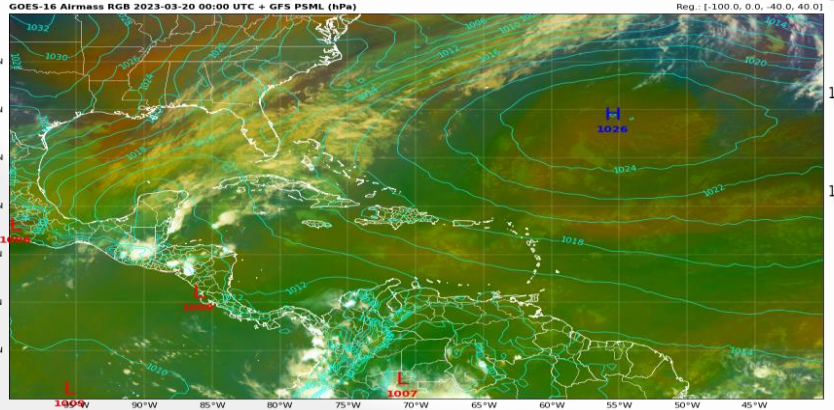


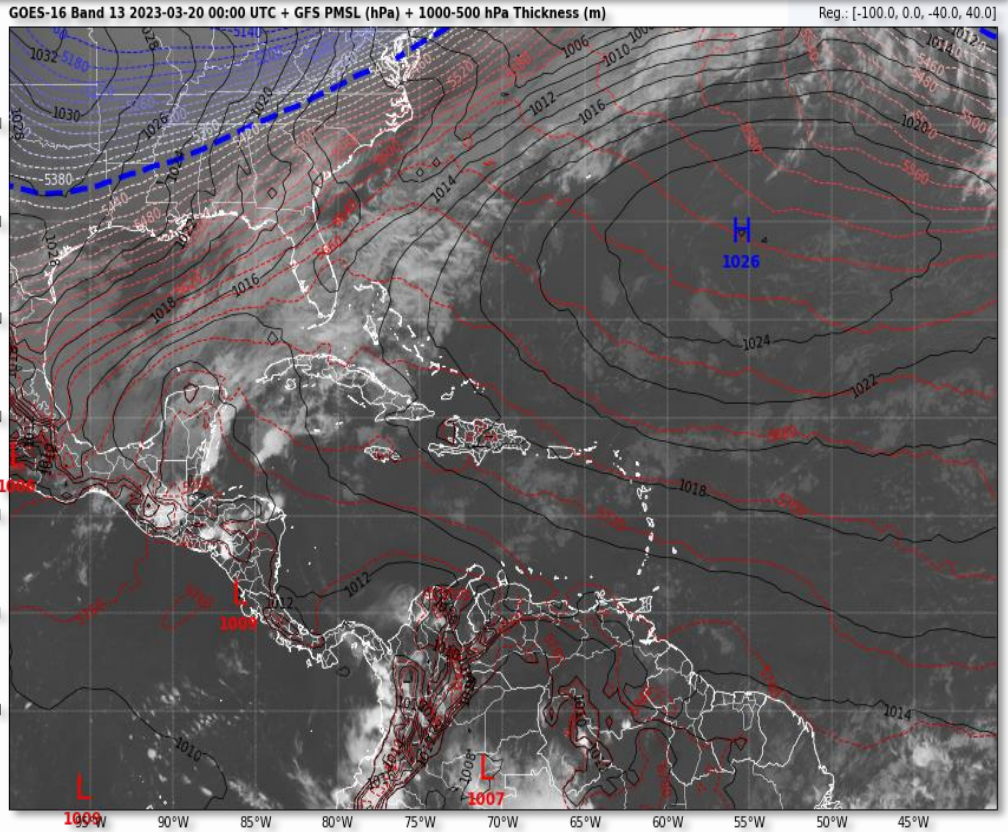


What Can You Do With Python? Some Examples

NWP OUTPUT + SATELLITE IMAGERY

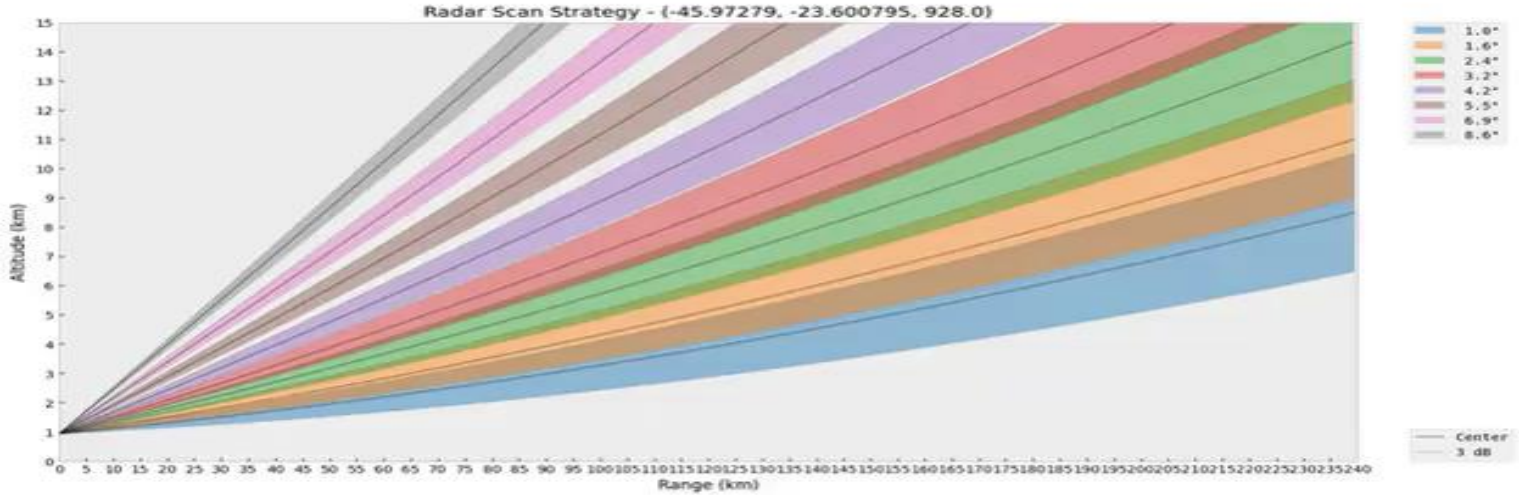






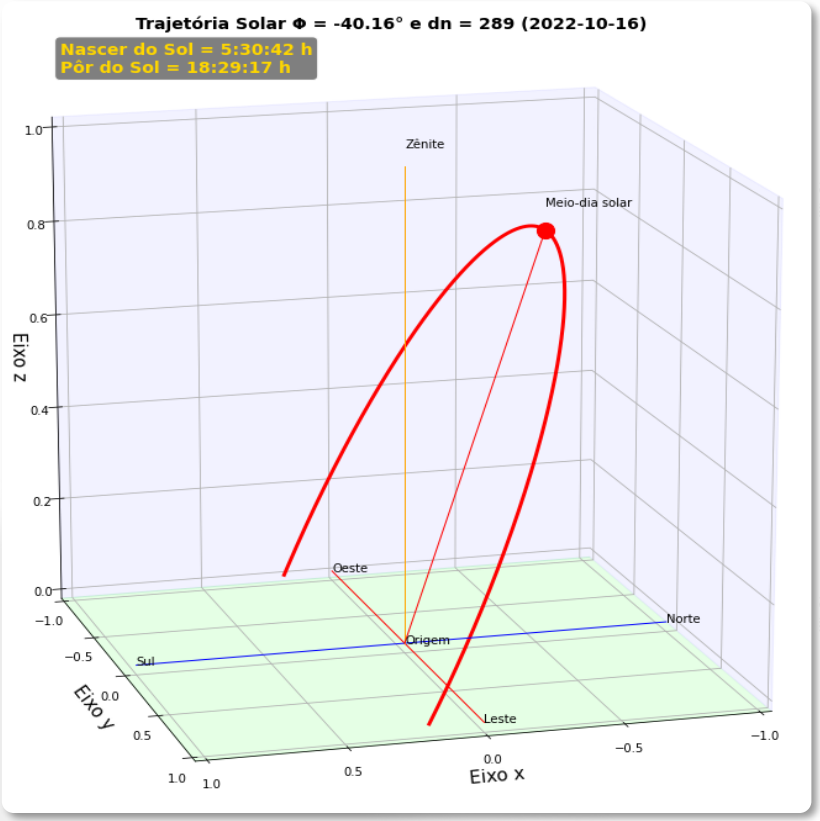
What Can You Do With Python? Some Examples

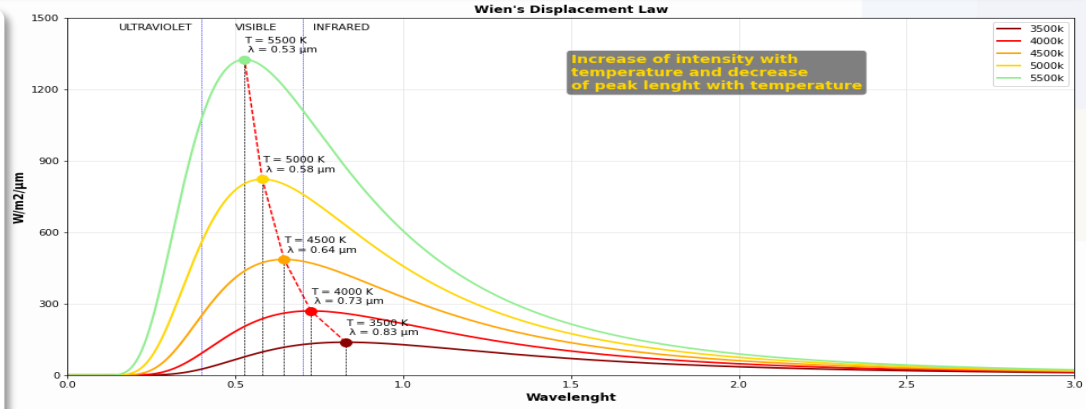
WEATHER RADAR - THIS CONETENT IS BEING PREPARED FOR AN UPCOMING TRAINING ORGANIZED IN BRAZIL

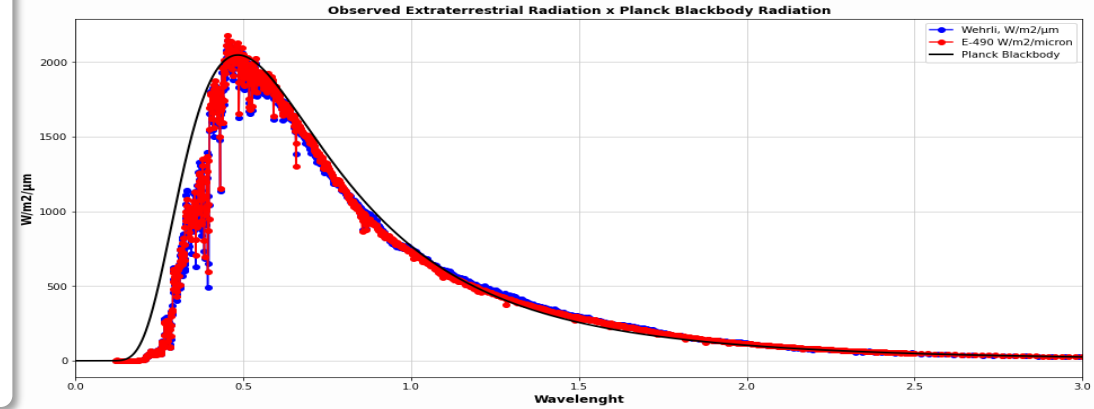


What Can You Do With Python? Some Examples

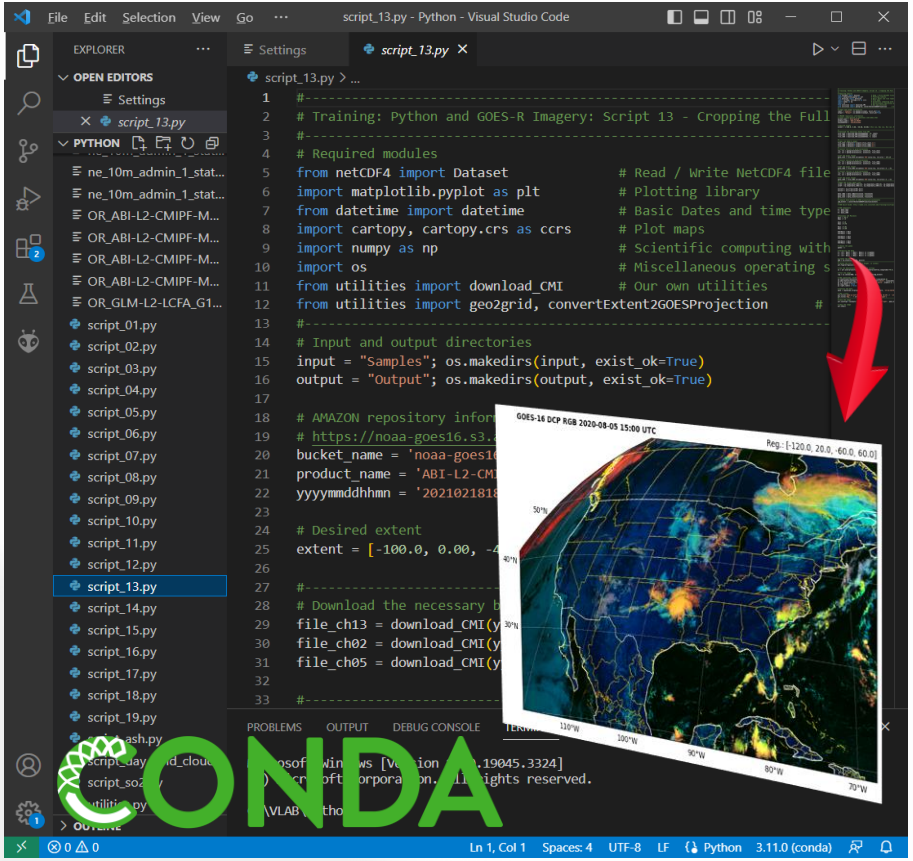
GENERAL GRAPHS - IN THIS EXAMPLE, SOME PLOTS CREATED FOR AN UPCOMING TRAINING ON RADIATION PRODUCTS

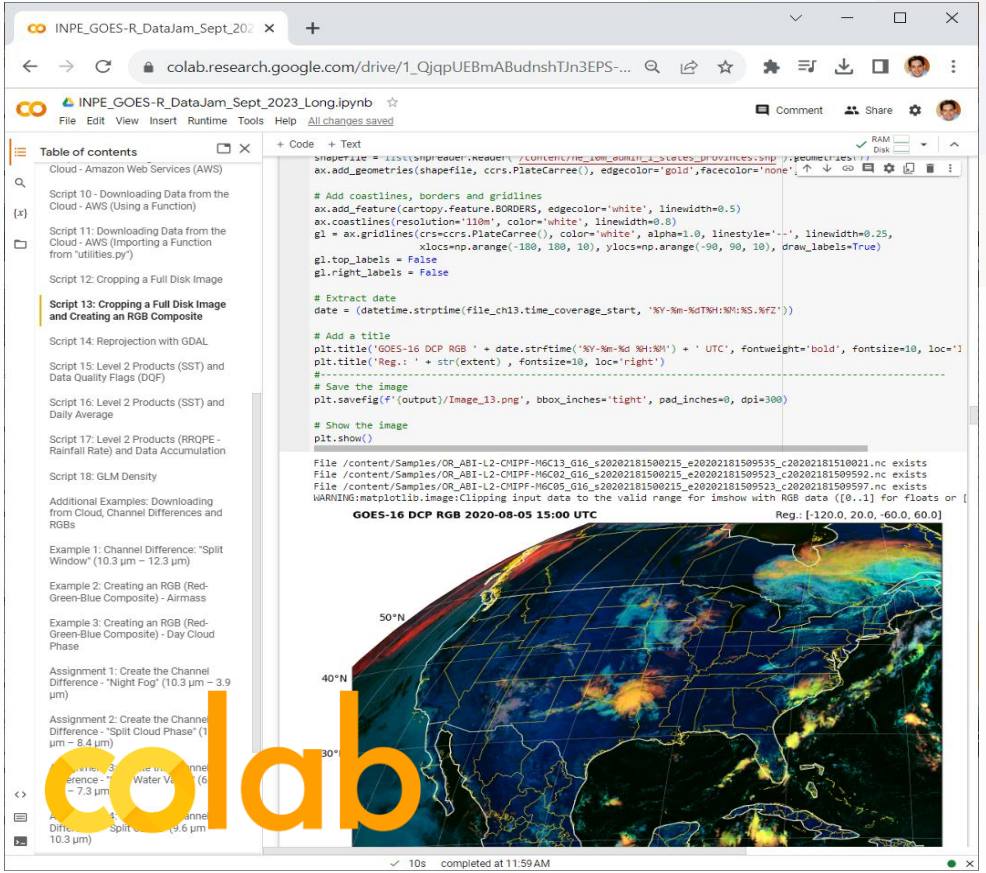






Running Python Scripts Locally or in the Cloud



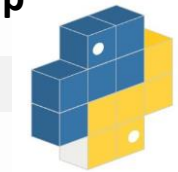


Running Python Scripts Locally

This is what you need in your computer or server: Conda (or pip), and editor to write your scripts (or IDE) and some sample imagery

- Python
- Package Manager
- Virtual Environment Manager

pip



- Text Editor / IDE (Integrated Development Environment)

VS Code

PyCharm

and many others!

- Sample Data

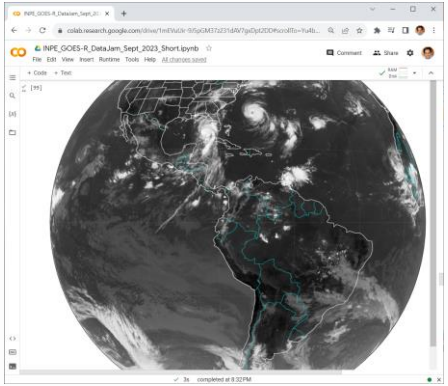
A variety of data access mechanisms available

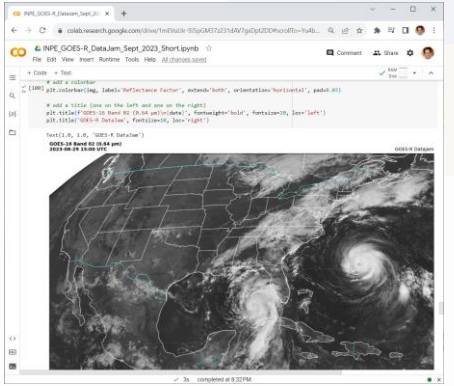
We will not see it in this video but there's a link in the end showing the steps needed

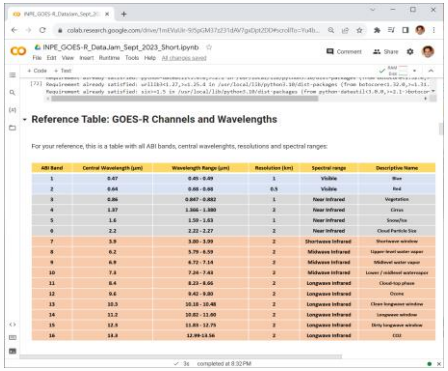
Running Python Scripts in the Cloud

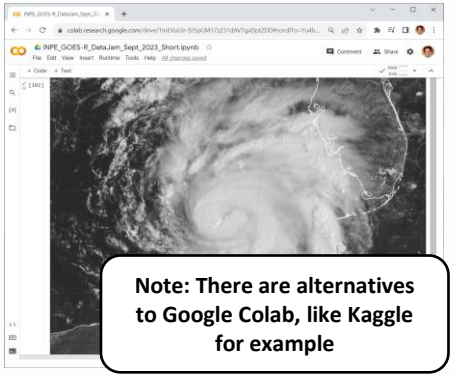
You only need a Colab "Notebook"! We have prepared some examples.

https://colab.research.google.com/drive/1mEVuUir-9J5pGM37z231dAV7gxDpt2DD?usp=sharing





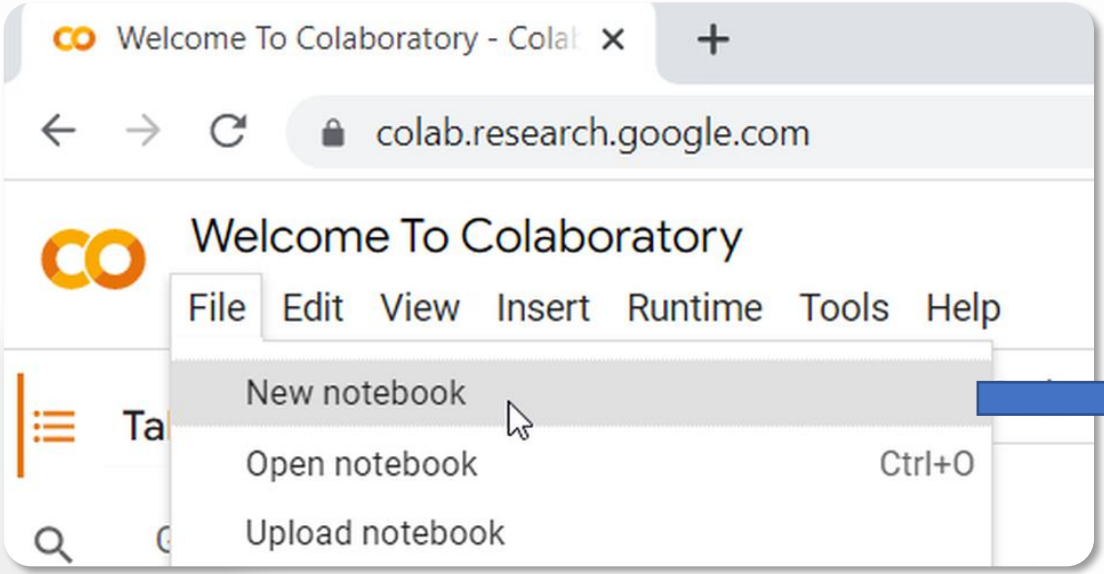


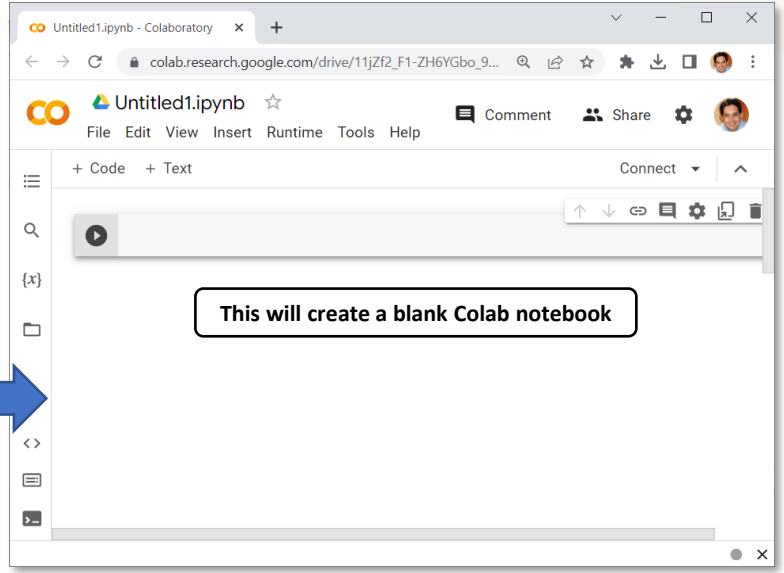


Note: There are alternatives to Google Colab, like Kaggle for example

GOOGLE COLAB: BASIC CONCEPTS

Getting started with Google COLAB: https://colab.research.google.com/

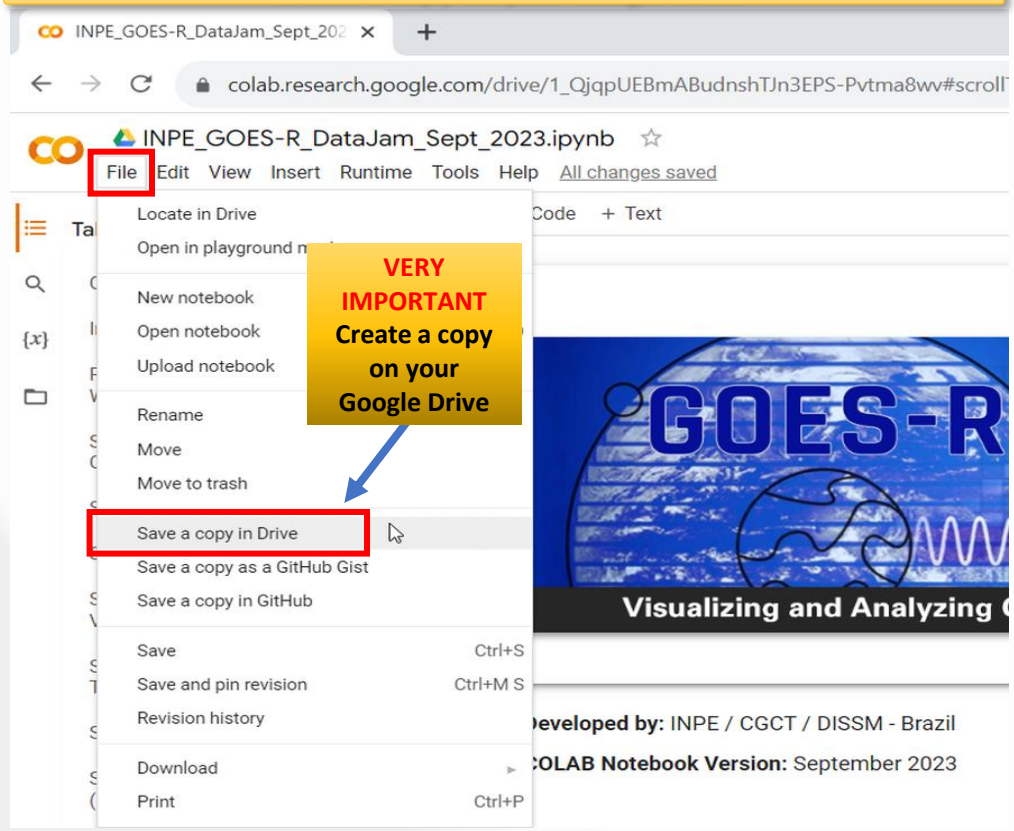




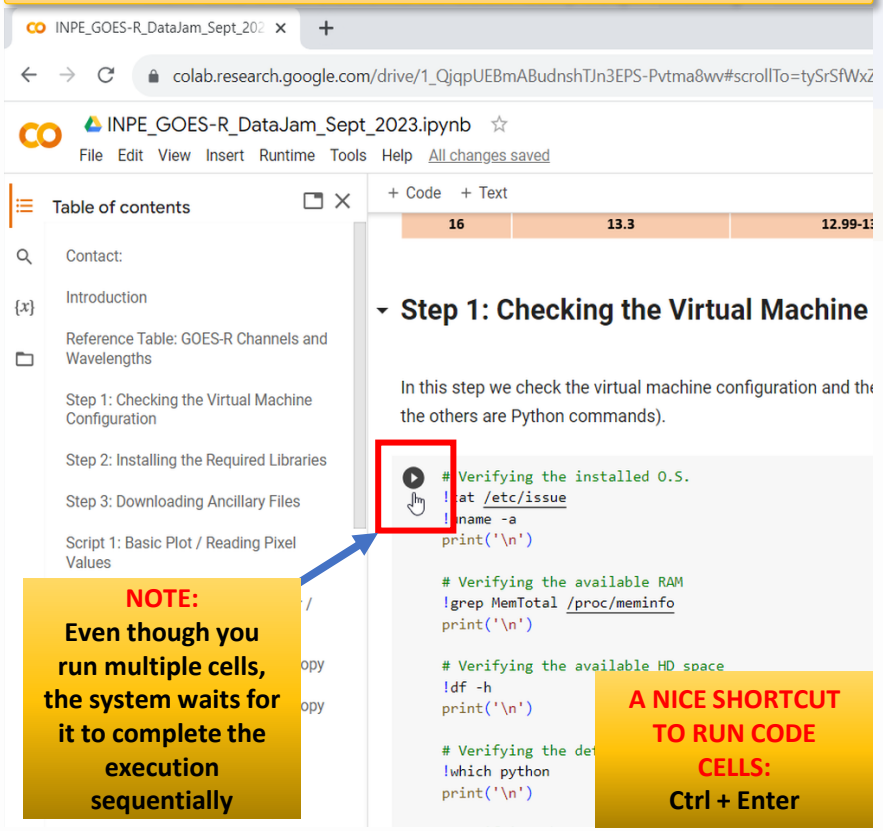
… let's make a quick demonstration

Creating a Copy and Running Code Cells

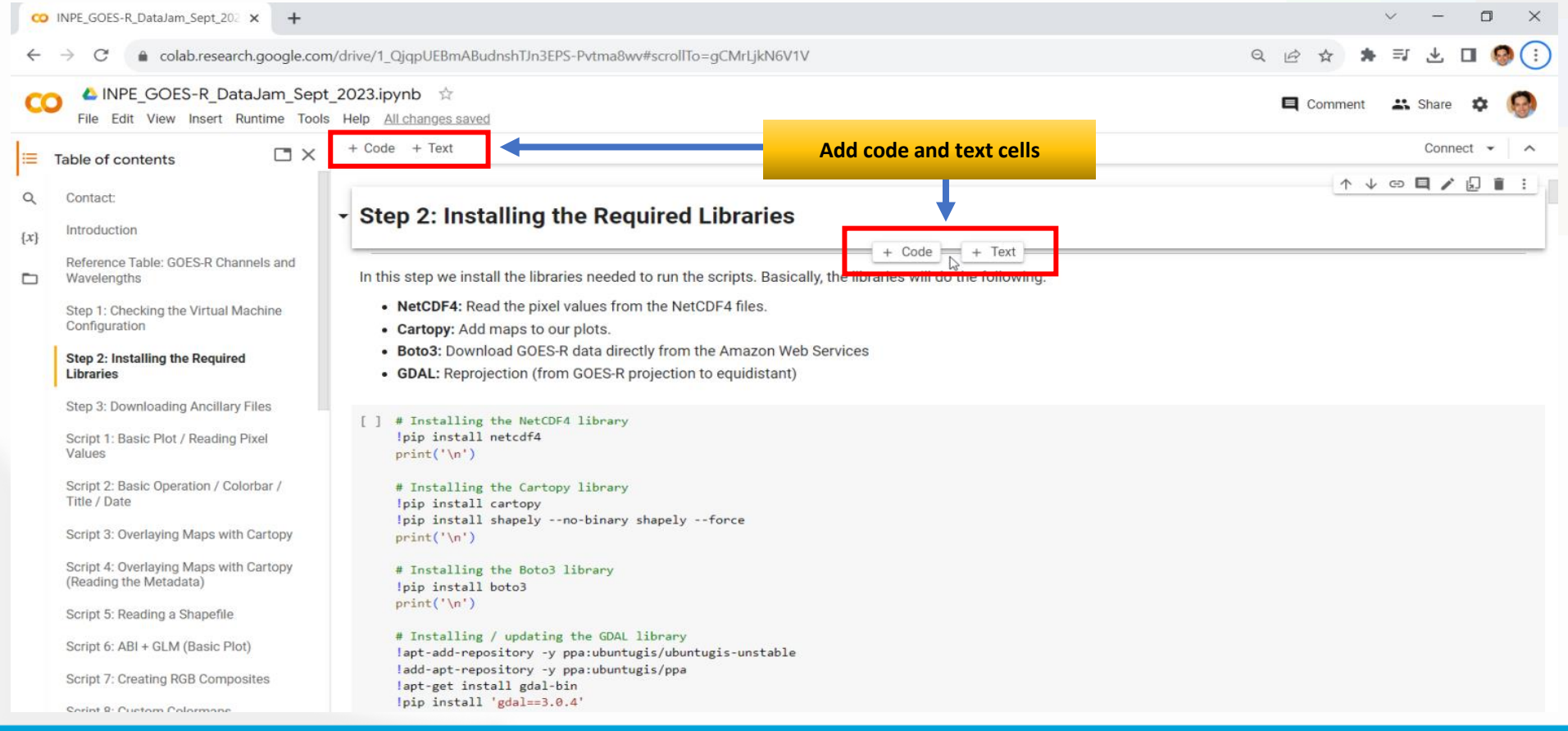
Create a copy on your Google Drive



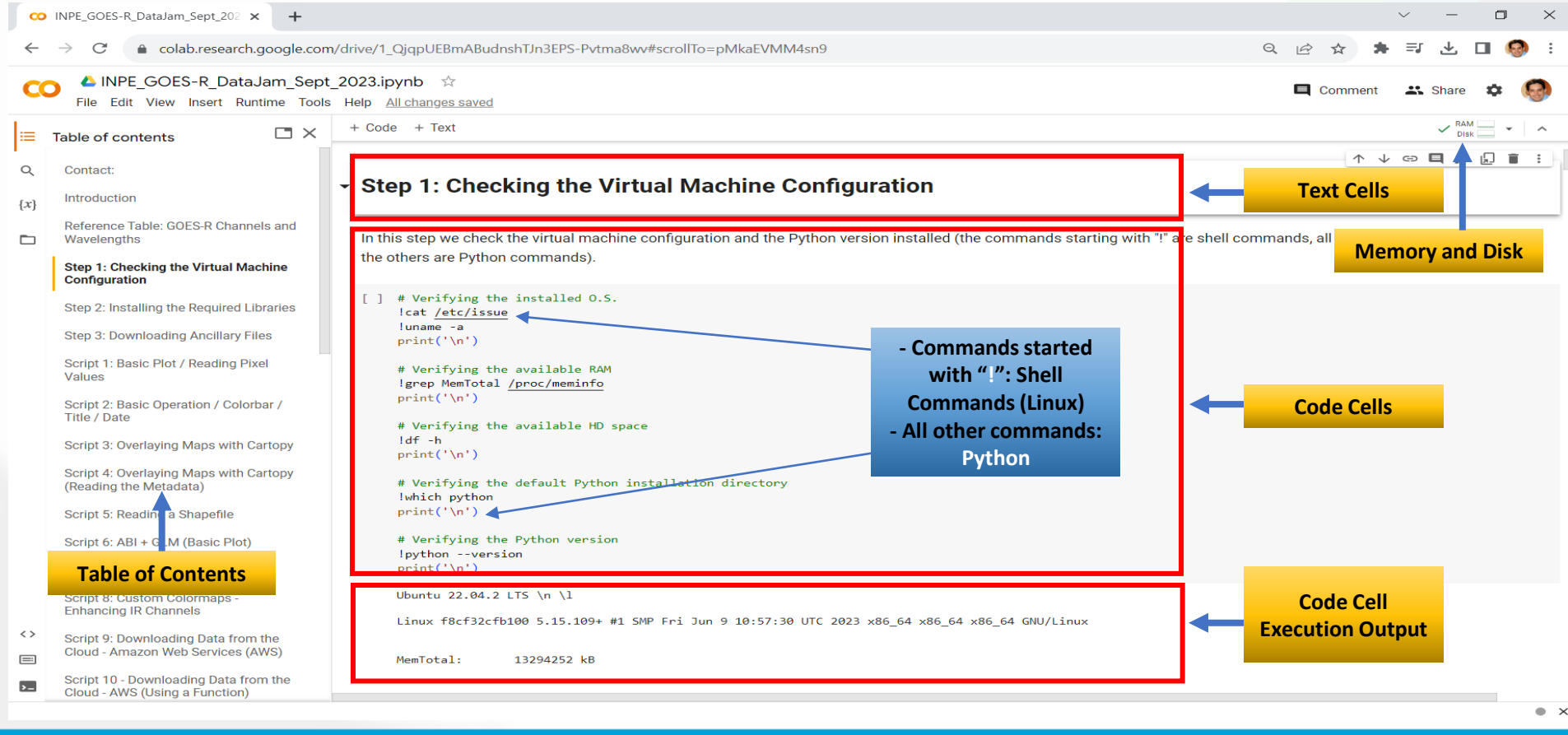
Run the instructions in each cell, sequentially



Adding Code and Text Cells

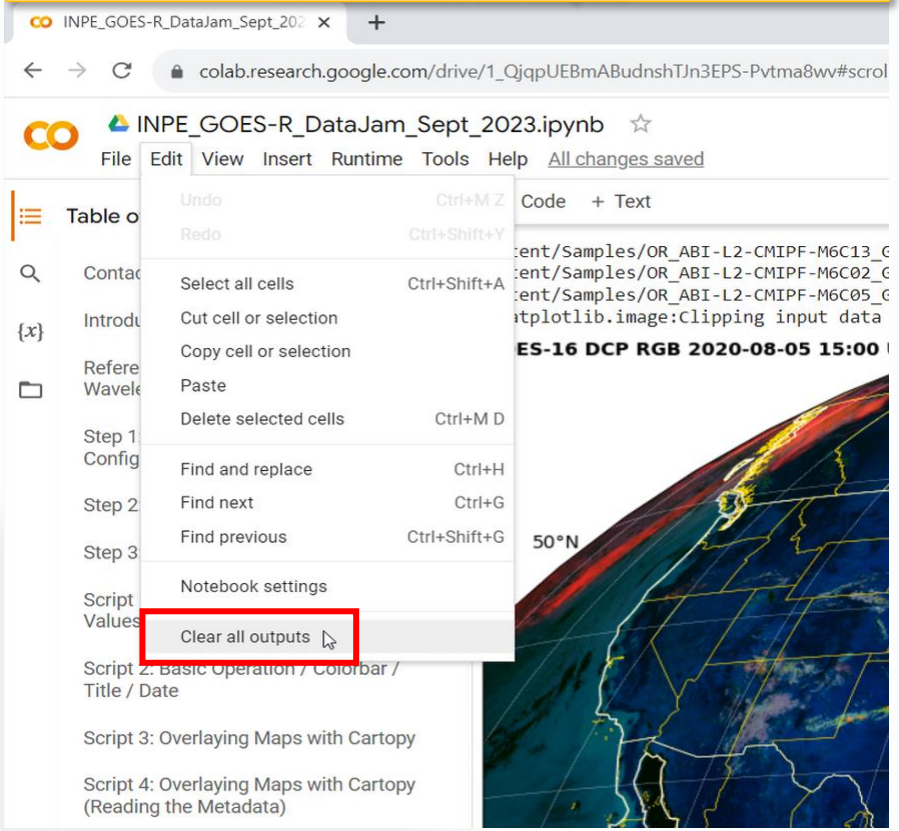


Linux Shell and Python Commands

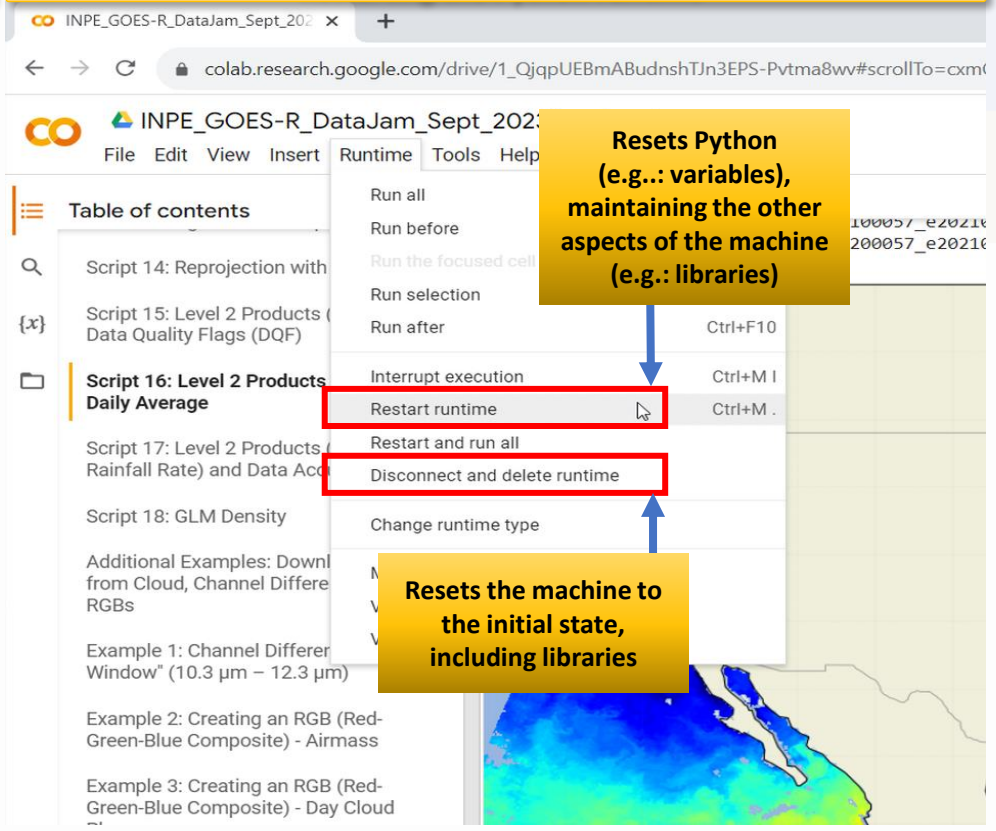


Clearing Outputs and Restarting the Runtime

Clears all code cell outputs



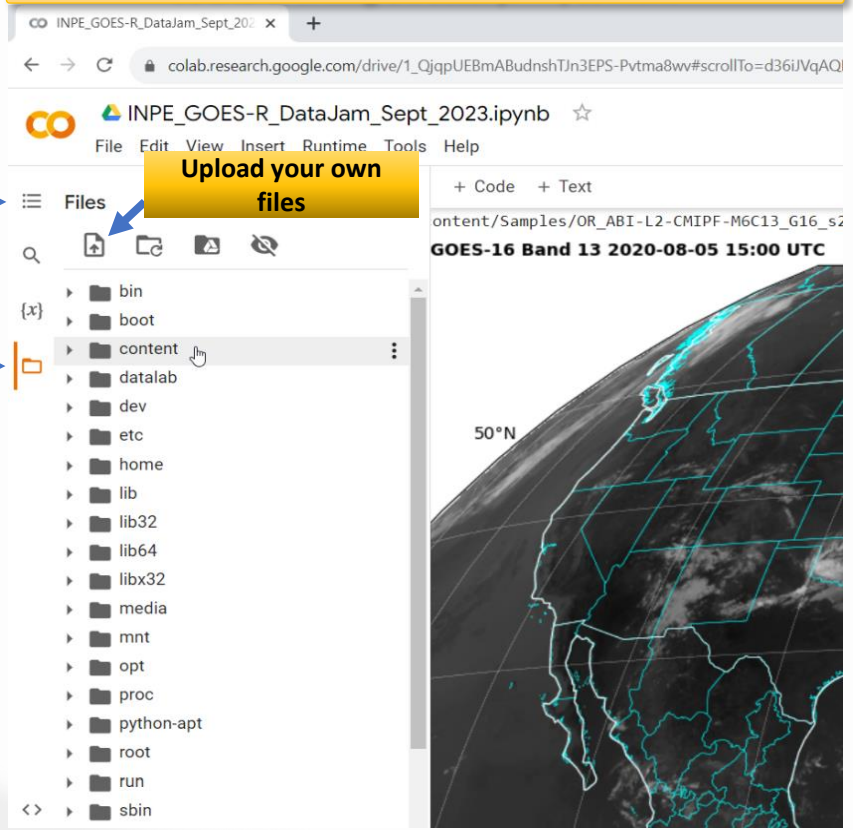
Restarting the Python runtime

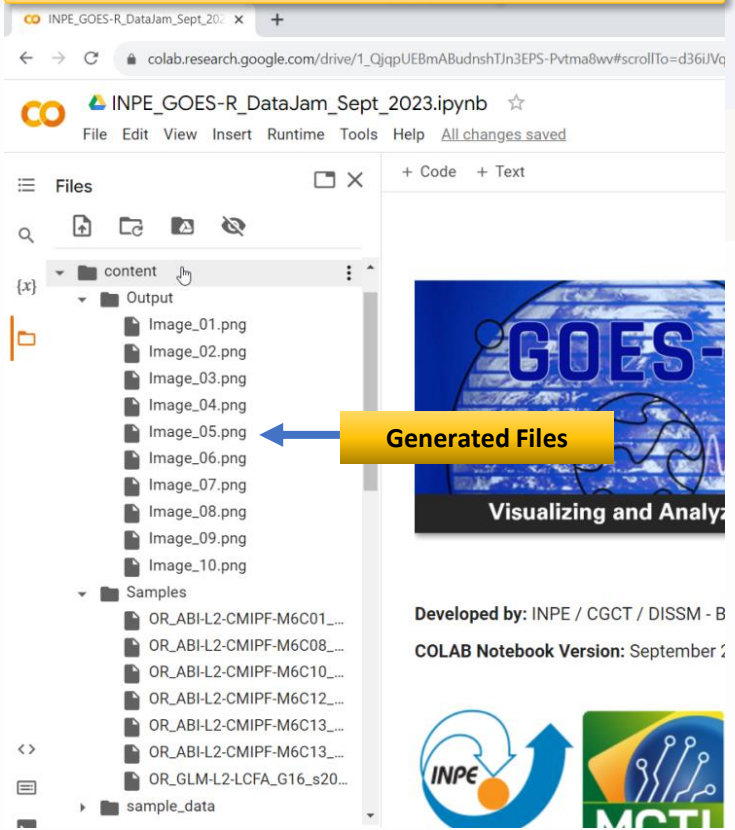


Viewing Table of Contents and Directories

COLAB Working Directory: "/content"

"/content" Directory





Installing Libraries and Downloading Samples

Let's explore the short version of our Colab Notebooks

https://colab.research.google.com/drive/1mEVuUir-9J5pGM37z231dAV7gxDpt2DD?usp=sharing

```
# installing the NetCDF4 library
!pip install netcdf4
print('\n')

# installing the Cartopy library
!pip install cartopy
!pip install shapely --no-binary shapely --force
print('\n')

# Create the 'samples' directory
!mkdir -p samples # where we will store the GOES-R samples
print('\n')

# download some GOES-16 samples from Amazon Web Services (AWS) with wget
# note: the links were taken from the following page: https://home.chpc.utah.edu/~u0553130/Brian_Blaylock/cgi-bin/goes16_download.cgi

# Band 13 (10.3µ) - August 29, 15:00 UTC (Full Disk)
!wget -c https://noaa-goes16.s3.amazonaws.com/ABI-L2-CMIPF/2023/241/15/OR_ABI-L2-CMIPF-M6C13_G16_s20232411500206_e20232411509526_c20232411509585.nc -P /content/samples/

# Band 02 (0.64µ) - August 29, 15:00 UTC (CONUS)
!wget -c https://noaa-goes16.s3.amazonaws.com/ABI-L2-CMIPC/2023/241/15/OR_ABI-L2-CMIPC-M6C02_G16_s20232411501172_e20232411503545_c20232411504026.nc -P /content/samples/

# Band 02 (0.64µ) - August 29, 15:00 UTC (MESOSCALE 1)
!wget -c https://noaa-goes16.s3.amazonaws.com/ABI-L2-CMIPM/2023/241/15/OR_ABI-L2-CMIPM1-M6C02_G16_s20232411500279_e20232411500336_c20232411500395.nc -P /content/samples/
```

Install the NetCDF4 library

Install the Cartopy library

Create a folder called "samples" (to store our samples)

Download GOES-R samples from the Amazon Web Services (AWS)

GOES-R Channels and Wavelengths

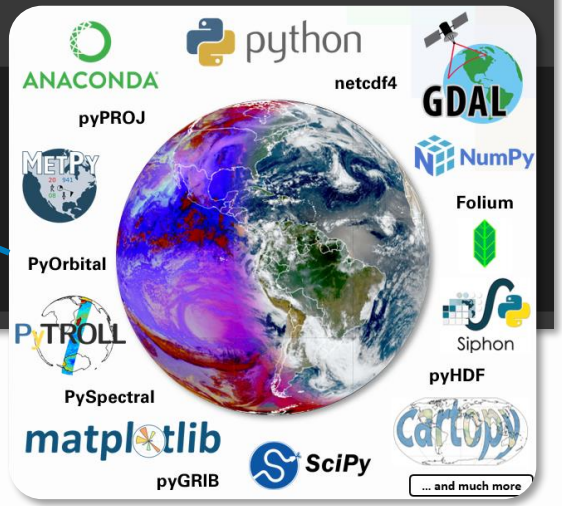
For your reference, this is a table with all ABI bands, central wavelenghts, resolutions and spectral ranges:

ABI Band	Central Wavelength (µm)	Wavelength Range (µm)	Resolution (km)	Spectral range	Descriptive Name
1	0.47	0.45 - 0.49	1	Visible	Blue
2	0.64	0.68 - 0.68	0.5	Visible	Red
3	0.86	0.847 - 0.882	1	Near Infrared	Vegetation
4	1.37	1.366 - 1.380	2	Near Infrared	Cirrus
5	1.6	1.59 - 1.63	1	Near Infrared	Snow/Ice
6	2.2	2.22 - 2.27	2	Near Infrared	Cloud Particle Size
7	3.9	3.80 - 3.99	2	Shortwave Infrared	Shortwave window
8	6.2	5.79 - 6.59	2	Midwave Infrared	Upper-level water vapor
9	6.9	6.72 - 7.14	2	Midwave Infrared	Midlevel water vapor
10	7.3	7.24 - 7.43	2	Midwave Infrared	Lower / midlevel watervapor
11	8.4	8.23 - 8.66	2	Longwave Infrared	Cloud-top phase
12	9.6	9.42 - 9.80	2	Longwave Infrared	Ozone
13	10.3	10.18 - 10.48	2	Longwave Infrared	Clean longwave window
14	11.2	10.82 - 11.60	2	Longwave Infrared	Longwave window
15	12.3	11.83 - 12.75	2	Longwave Infrared	Dirty longwave window
16	13.3	12.99-13.56	2	Longwave Infrared	CO2

Importing the Necessary Python Libraries

In this step we will import all the libraries we will need for this demonstration: "**netCDF4**" to open files and read the pixel values, "**matplotlib**" to generate images, "**datetime**" to read the aqcuisition time and date, "**cartopy**" to add maps and "**numpy**" to make some calculations when needed.

```
[ ]  # required modules
     from netCDF4 import Dataset          # read / write NetCDF4 files
     import matplotlib.pyplot as plt      # plotting library
     from datetime import datetime        # basic dates and time types
     import cartopy, cartopy.crs as ccrs # plot maps
     import numpy as np                   # scientific computing with Python
```



Opening a G16 File and Checking the Datasets

GOES-R File Naming Convention

OR_ABI-L2-CMIPF-M6C13_G16_s20232411500206_e20232411509526_c20232411509585.nc

<Operational System Real-Time Data>_<sensor>-<level>-<short product name>-M<operation mode><channel>-G<GOES>-s<start time of scan:year-julianday-hh:mm:ssUTC>/_e<end time of scan:year-julianday-hh:mm:ssUTC>/_c< data creation time ><year-julianday-hh:mm:ssUTC>/.nc

First, we will create an object called **"file"** (you may call it anyway you want) and open a GOES-16 ABI image with the NetCDF4 **"Dataset"** command:

```
# open the GOES-R NetCDF file
file = Dataset("samples/OR_ABI-L2-CMIPF-M6C13_G16_s20232411500206_e20232411509526_c20232411509585.nc")
```

We have several datasets inside a NetCDF file (with very useful information!). With the **"variables.keys()"** command, we can list all the available datasets.

```
# listing the key NetCDF variables
file.variables.keys()
```

https://www.goes-r.gov/products/baseline-cloud-moisture-imagery.html

```
dict_keys(['CMI', 'DQF', 't', 'y', 'x', 'time_bounds', 'goes_imager_projection', 'y_image', 'y_image_bounds', 'x_image', 'x_image_bounds', 'nominal_satellite_subpoint_lat',
'nominal_satellite_subpoint_lon', 'nominal_satellite_height', 'geospatial_lat_lon_extent', 'band_wavelength', 'band_id', 'total_number_of_points', 'valid_pixel_count',
'outlier_pixel_count', 'min_brightness_temperature', 'max_brightness_temperature', 'mean_brightness_temperature', 'std_dev_brightness_temperature', 'planck_fk1', 'planck_fk2',
'planck_bc1', 'planck_bc2', 'algorithm_dynamic_input_data_container', 'percent_uncorrectable_GRB_errors', 'percent_uncorrectable_L0_errors',
'earth_sun_distance_anomaly_in_AU', 'processing_parm_version_container', 'algorithm_product_version_container', 'esun', 'kappa0',
'focal_plane_temperature_threshold_exceeded_count', 'maximum_focal_plane_temperature', 'focal_plane_temperature_threshold_increasing',
'focal_plane_temperature_threshold_decreasing', 'channel_integration_time', 'channel_gain_field'])
```

Verifying the Characteristics of the CMI Dataset

Let's check the characteristics of the **"CMI"** dataset (or "Cloud and Moisture Imagery", as NOAA calls it). According to the product documentation, this is the dataset where we may retrieve the **"reflectance factor"** (for visible bands) and the **"brightness temperatures"** for IR bands:

https://www.goes-r.gov/products/baseline-cloud-moisture-imagery.html

```
# reading a specific variable
file.variables['CMI']
```

```
<class 'netCDF4._netCDF4.Variable'>
int16 CMI(y, x)
    _FillValue: -1
    long_name: ABI L2+ Cloud and Moisture Imagery brightness temperature
    standard_name: toa_brightness_temperature
    _Unsigned: true
    sensor_band_bit_depth: 12
    valid_range: [    0 4095]
    scale_factor: 0.06145332
    add_offset: 89.62
    units: K
    resolution: y: 0.000056 rad x: 0.000056 rad
    coordinates: band_id band_wavelength t y x
    grid_mapping: goes_imager_projection
    cell_methods: t: point area: point
    ancillary_variables: DQF
unlimited dimensions:
current shape = (5424, 5424)
filling on
```

This particular band (channel 13 - 10.3 um) has 5424 rows x 5424 columns and the unit is Kelvin

Please note that for this particular channel (Band 13 - 10.3 µm - 2 km - Full Disk) we have **5424 rows x 5424 columns**. For a 1 km Full Disk band we would have twice as much and for a 0.5 km Full Disk band we would have 4 times as much (plotting a 0.5 km band requires good computational power). Please also note that the unit for this particular band is Kelvin.

Considering that, let's read the CMI dataset and create a preliminary plot.

Reading and Plotting the CMI Dataset

First, we will read the CMI dataset pixel values, convert the pixels to °C and store all values from this 2D array (5424 x 542
variable (again, you may call it anyway you want). We will use this variable to plot an image, but remember, you have a
2D array, so you can use it anyway you want (perform any calculation, make graphs, e

My variable name

CMI Dataset

The whole 2D array

```
# Get the pixel values
data = file.variables['CMI'][:] - 273.15
```

Convert from K to °C

Now, we will set the image size (in inches) and plot the "**data**" variable with a grayscale colormap, called "
temperature values and black for higher BT values) using the "imshow" command from matplotlib.

We will pass some parameters to the "**imshow**" command:

- "**data**": the variable we want to plot.
- "**vmin**" and "**vmax**": the minimum and maximum values of the colormap. You may adjust this values
- "**cmap**": the colormap to be used. Please access the following link for a list of default colormaps fro
  https://matplotlib.org/stable/tutorials/colors/colormaps.html

```
# choose the plot size (width x height, in inches)
plt.figure(figsize=(20,20))

# plot the image
plt.imshow(data, vmin=-80, vmax=40, cmap='Greys')
```

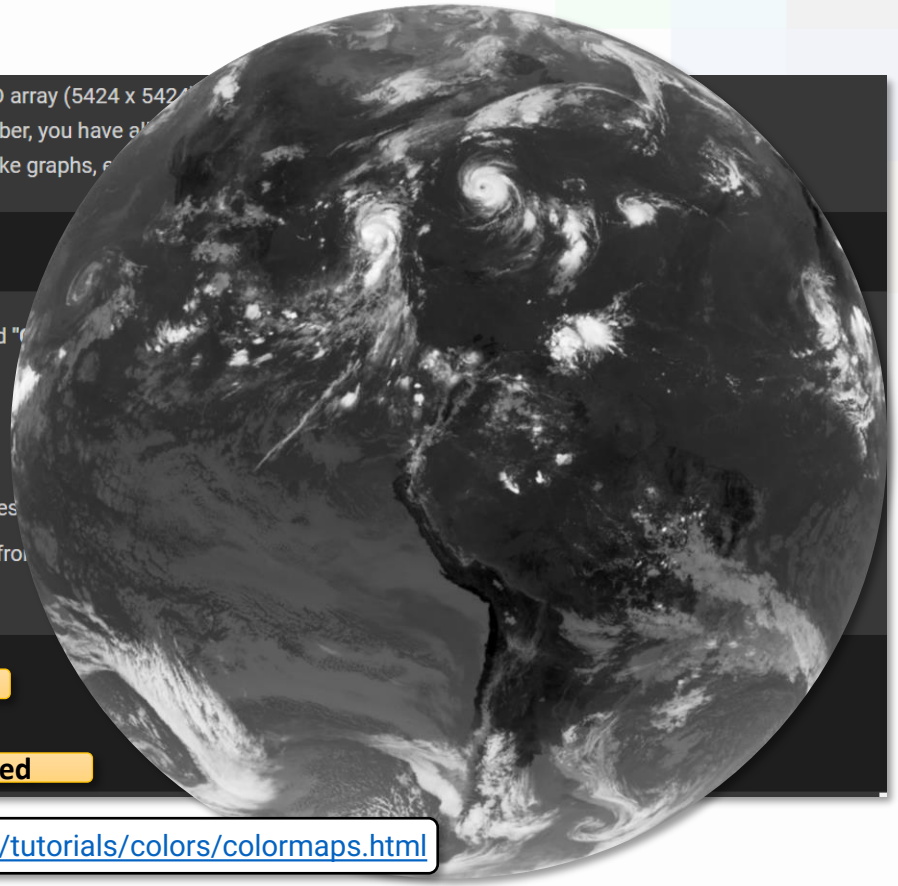
Plot size (width x height)

Colormap to be used

Minimum value of the colormap

Maximum value of the colormap

https://matplotlib.org/stable/tutorials/colors/colormaps.html



Reading the Acquisition Time (Start of the Scan)

1 - Reading the time / date from file

```
file.getncattr('time_coverage_start')
```

```
'2023-08-29T15:00:20.6Z'
```

2 - Informing the "datetime" library how the time / date is formatted

```
# read the time/date from the NetCDF file metadata as a string
date_string = file.getncattr('time_coverage_start')

# how this time/date is formatted (using the timedate convention)
date_format = '%Y-%m-%dT%H:%M:%S.%fZ'

# create the datetime object
date_obj = datetime.strptime(date_string, date_format)
print(date_obj)
```

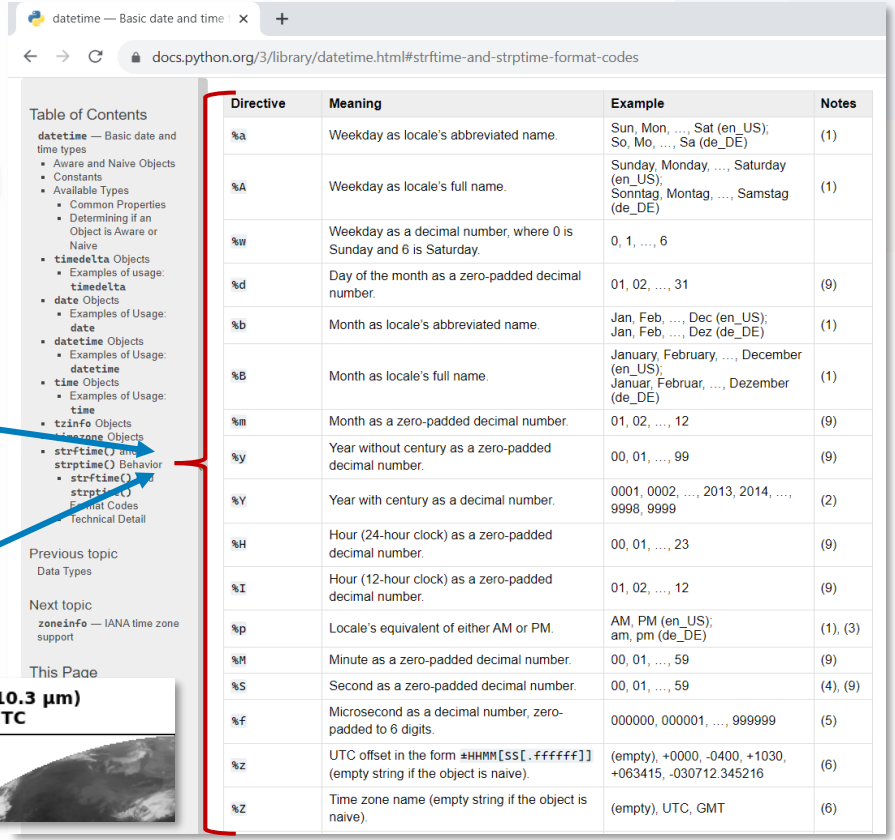
```
2023-08-29 15:00:20.600000
```

3 - How do I want to format and show it

```
date = date_obj.strftime('%Y-%m-%d %H:%M UTC')
print(date)
```

```
2023-08-29 15:00 UTC
```

https://docs.python.org/3/library/datetime.html#strftime-and-strptime-format-codes



datetime — Basic date and time

docs.python.org/3/library/datetime.html#strftime-and-strptime-format-codes

Table of Contents

- datetime — Basic date and time types
  - Aware and Naive Objects
  - Constants
  - Available Types
    - Common Properties
    - Determining if an Object Is Aware or Naive
  - timedelta Objects
    - Examples of usage: timedelta
  - date Objects
    - Examples of Usage: date
  - datetime Objects
    - Examples of Usage: datetime
  - time Objects
    - Examples of Usage: time
  - tzinfo Objects
  - timezone Objects
  - strftime() and strptime() Behavior
    - strftime() and strptime() Format Codes
    - Technical Detail

Previous topic: Data Types

Next topic: zoneinfo — IANA time zone support

This Page

Directive	Meaning	Example	Notes
%a	Weekday as locale's abbreviated name.	Sun, Mon, …, Sat (en_US); So, Mo, …, Sa (de_DE)	(1)
%A	Weekday as locale's full name.	Sunday, Monday, …, Saturday (en_US); Sonntag, Montag, …, Samstag (de_DE)	(1)
%w	Weekday as a decimal number, where 0 is Sunday and 6 is Saturday.	0, 1, …, 6	
%d	Day of the month as a zero-padded decimal number.	01, 02, …, 31	(9)
%b	Month as locale's abbreviated name.	Jan, Feb, …, Dec (en_US); Jan, Feb, …, Dez (de_DE)	(1)
%B	Month as locale's full name.	January, February, …, December (en_US); Januar, Februar, …, Dezember (de_DE)	(1)
%m	Month as a zero-padded decimal number.	01, 02, …, 12	(9)
%y	Year without century as a zero-padded decimal number.	00, 01, …, 99	(9)
%Y	Year with century as a decimal number.	0001, 0002, …, 2013, 2014, …, 9998, 9999	(2)
%H	Hour (24-hour clock) as a zero-padded decimal number.	00, 01, …, 23	(9)
%I	Hour (12-hour clock) as a zero-padded decimal number.	01, 02, …, 12	(9)
%p	Locale's equivalent of either AM or PM.	AM, PM (en_US); am, pm (de_DE)	(1), (3)
%M	Minute as a zero-padded decimal number.	00, 01, …, 59	(9)
%S	Second as a zero-padded decimal number.	00, 01, …, 59	(4), (9)
%f	Microsecond as a decimal number, zero-padded to 6 digits.	000000, 000001, …, 999999	(5)
%z	UTC offset in the form ±HHMM[SS[.ffffff]] (empty string if the object is naive).	(empty), +0000, -0400, +1030, +063415, -030712.345216	(6)
%Z	Time zone name (empty string if the object is naive).	(empty), UTC, GMT	(6)

Adding Titles and a Colorbar

How do I want to format the time / date

```
date = date_obj.strftime('%Y-%m-%d %H:%M UTC')
print(date)
```

```
2023-08-29 15:00 UTC
```

Now that we already know how to read and format the aqcuisition time and date, let's create a plot adding the datetime string as a title. We will also add a colorbar with the "**colorbar**" command:

```
# choose the plot size (width x height, in inch
plt.figure(figsize=(10,10))

# plot the image
plt.imshow(data, vmin=-80, vmax=40, cmap='Greys')

# add a colorbar
plt.colorbar(label='Brightness Temperatures (°C)', extend='both', orientation='horizontal', pad=0.05, shrink=0.75)

# add a title (one on the left and one on the right)
plt.title(f'GOES-16 Band 13 (10.3 µm)\n{date}', fontweight='bold', fontsize=10, loc='left')
plt.title('GOES-R DataJam', fontsize=10, loc='right')
```

Title of the colorbar

Extend the colorbar on which side?

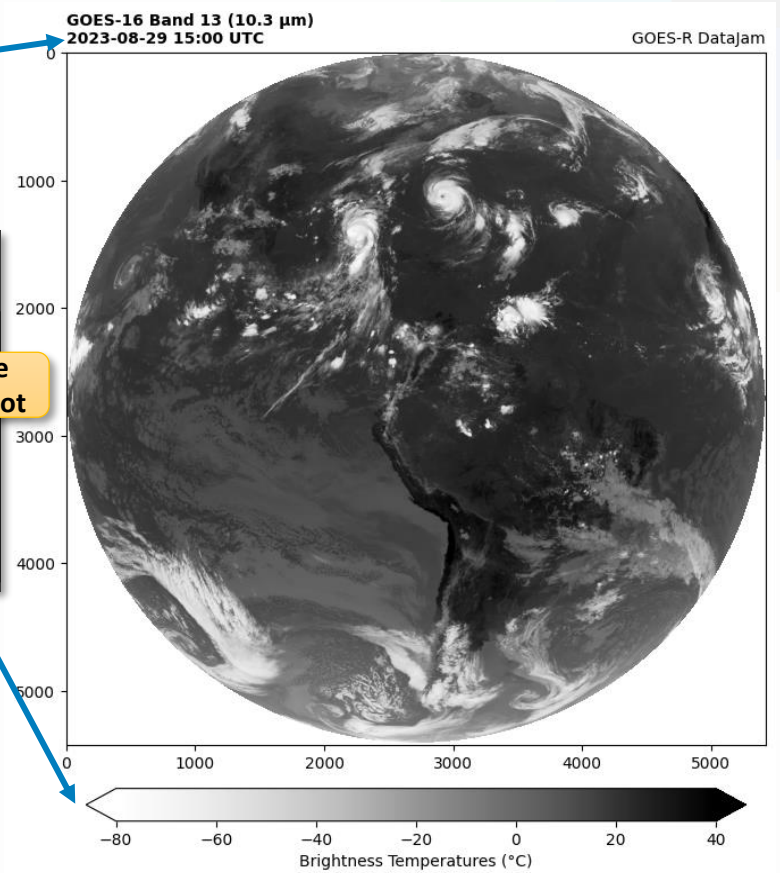
Orientation of the colorbar

Relative distance (%) to the main plot

Size multiplier

Add title on the left

Add another title on the right



Adding Maps With Cartopy

```
# read the satellite longitude from the metadata
longitude = file.variables['goes_imager_projection'].longitude_of_projection_origin

# read the satellite altitude from the metadata
height = file.variables['goes_imager_projection'].perspective_point_height

# use the Geostationary projection in cartopy
ax = plt.axes(projection=ccrs.Geostationary(central_longitude=longitude, satellite_height=height))

# add coastlines, borders and gridlines
ax.add_feature(cartopy.feature.BORDERS, edgecolor='black', linewidth=0.5)
ax.coastlines(resolution='110m', color='black', linewidth=0.8)
ax.gridlines(color='black', alpha=0.5, linestyle='--', linewidth=0.5)
```

```
<cartopy.mpl.gridliner.Gridliner at 0x7879b537e170>
```

Read the satellite central longitude from the metadata

Read the satellite height from the metadata

Tell Cartopy the projection of our data - “geostationary” in this case, passing the central longitude and satellite height as parameters

https://scitools.org.uk/cartopy/docs/v0.15/crs/projections.html

Add country borders with Cartopy

Add coastlines with Cartopy

Add gridlines with Cartopy

Resulting plot

… now let’s see how to plot this map with our GOES-R ABI image

Adding Maps With Cartopy

We also need to inform Cartopy the "extent" of our data

We also need to inform the "**extent**" of our data. According to the product documentation (https://www.goes-r.gov/products/docs/PUG-L2+-vol5.pdf), the extent can be calculated by **multiplying half of the full disk by the pixel size in radians by the satellite height**. Again, all this information is provided in the metadata. Calculating this by reading the metadata is very useful: as we will see, we can use the same scripts to plot the **Full Disk, CONUS and Mesoscale sectors**:

Paragraph 4.2 - ABI Fixed Grid: https://www.goes-r.gov/products/docs/PUG-L2+-vol5.pdf

```
# calculate the extent of the GOES-R full disk in decimals (2712*0.000056*35786023.0)
xmin = file.variables['x'][:].min() * height
xmax = file.variables['x'][:].max() * height
ymin = file.variables['y'][:].min() * height
ymax = file.variables['y'][:].max() * height

# list with the extent
img_extent = (xmin, xmax, ymin, ymax)
print(img_extent)
```

```
(-5433892.69232443, 5433892.69232443, -5433892.69232443, 5433892.69232443)
```

According to the GOES-R documentation, the extent is:
Number of pixels (half of full disk) x pixel size (radians) x satellite height

This first multiplication is already available in the metadata, called as 'x' and 'y' datasets

This is what Cartopy needs

```
# plot the image
img = ax.imshow(data, vmin=-80, vmax=40, origin='upper', extent=img_extent, cmap='Greys')
```

Adding Maps With Cartopy

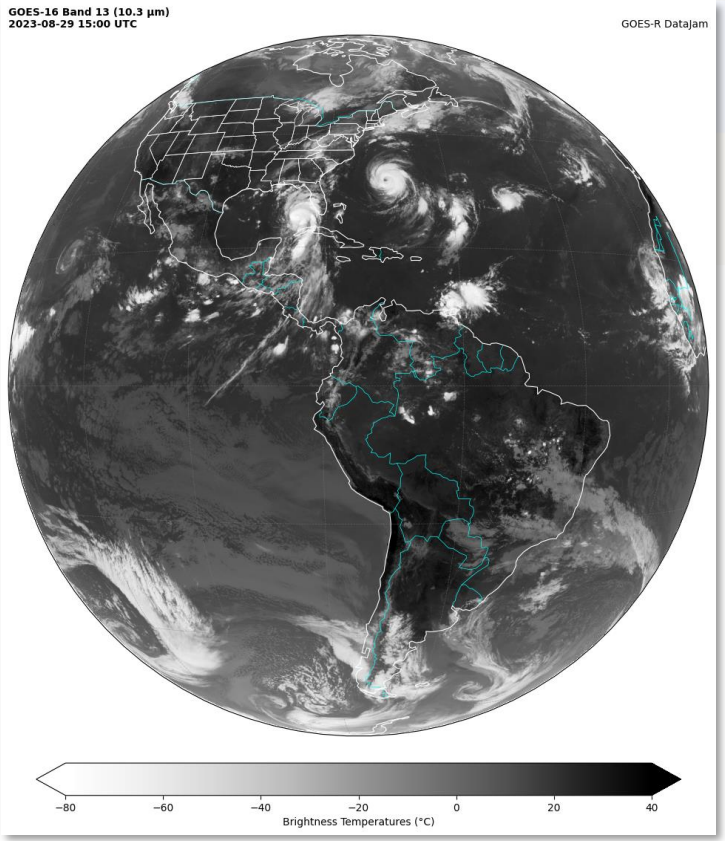
All the instructions needed to create the image on the right

```
# open the GOES-R NetCDF file
file = Dataset("samples/OR_ABI-L2-CMIPF-M6C13_G16_s20232411500206_e20232411509526_c20232411509585.nc")

# get the pixel values
data = file.variables['CMI'][:] - 273.15
```

DATA READING AND MANIPULATION

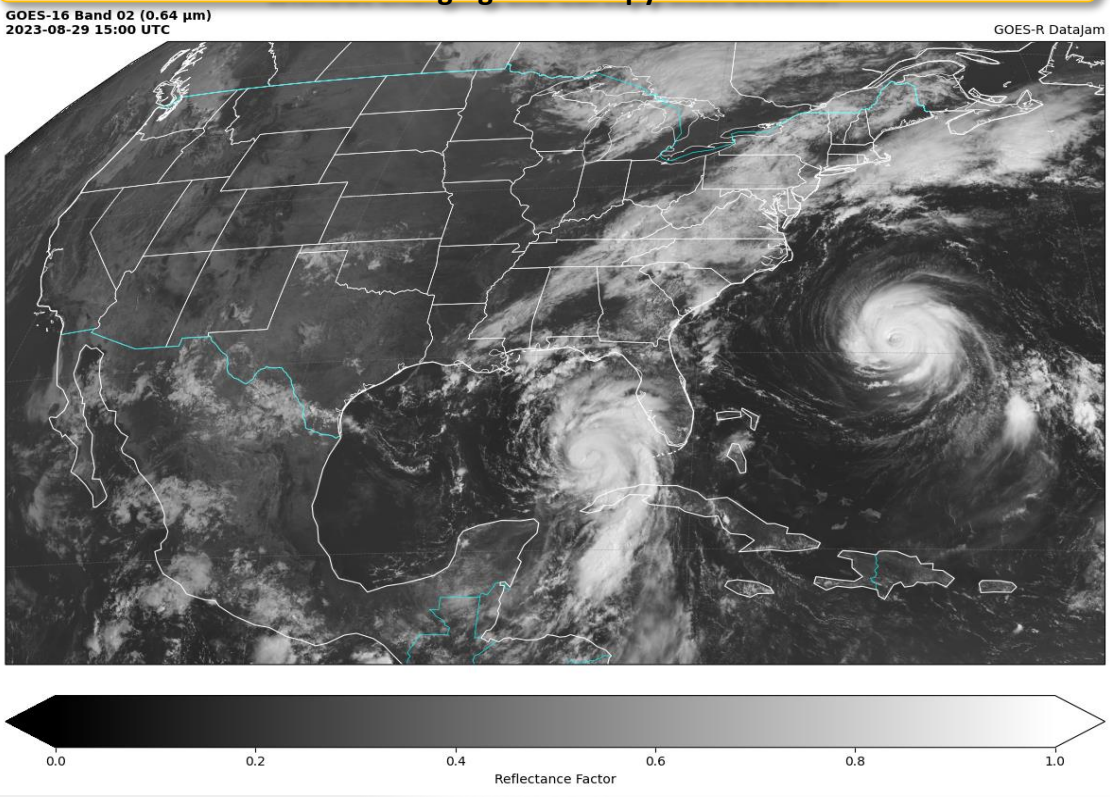
```
# choose the plot size (width x height, in inches)
plt.figure(figsize=(15,15))

# use the Geostationary projection in cartopy
ax = plt.axes(projection=ccrs.Geostationary(central_longitude=longitude, satellite_height=height))

# calculate the extent of the GOES-R full disk in decimals (2712*0.000056*35786023.0)
xmin = file.variables['x'][:].min() * height
xmax = file.variables['x'][:].max() * height
ymin = file.variables['y'][:].min() * height
ymax = file.variables['y'][:].max() * height

# list with the extent
img_extent = (xmin, xmax, ymin, ymax)

# add coastlines, borders and gridlines
ax.add_feature(cartopy.feature.BORDERS, edgecolor='cyan', linewidth=0.5)
ax.add_feature(cartopy.feature.STATES, edgecolor='white', linewidth=0.5)
ax.coastlines(resolution='110m', color='white', linewidth=0.8)
ax.gridlines(color='gray', alpha=0.5, linestyle='--', linewidth=0.5)

# plot the image
img = ax.imshow(data, vmin=-80, vmax=40, origin='upper', extent=img_extent, cmap='Greys')

# add a colorbar
plt.colorbar(img, label='Brightness Temperatures (°C)', extend='both', orientation='horizontal', pad=0.03, shrink=0.75)

# add a title (one on the left and one on the right)
plt.title(f'GOES-16 Band 13 (10.3 µm)\n{date}', fontweight='bold', fontsize=10, loc='left')
plt.title('GOES-R DataJam', fontsize=10, loc='right')
```

PLOT CONFIGURATION



Plotting CONUS and MESOSCALES

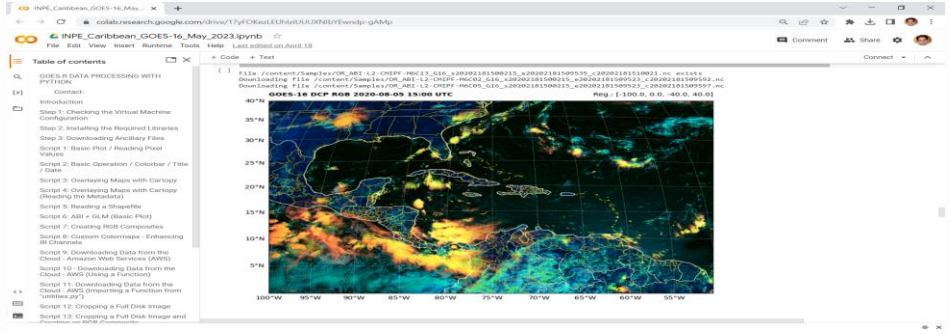
We are reading parameters from the metadata, so we can plot other sectors without changing the Cartopy instructions!



For Your Future Reference

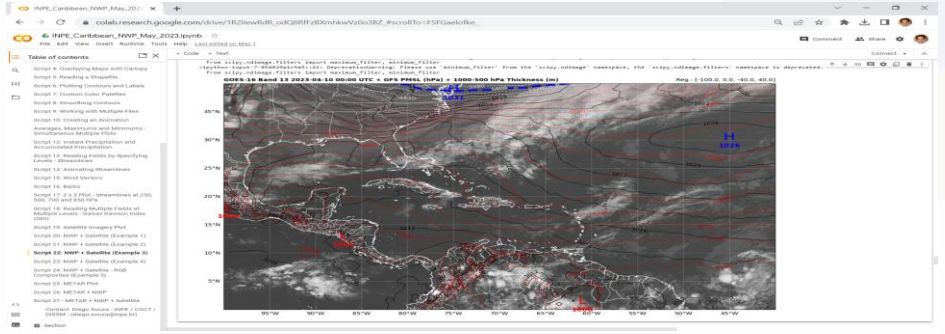
GOES-R DATA AND PRODUCTS - LONGER VERSION WITH ABI, GLM, RGBs AND MUCH MORE!

https://colab.research.google.com/drive/1_QjqpUEBmABudnshTJn3EPS-Pvtma8wv?usp=sharing



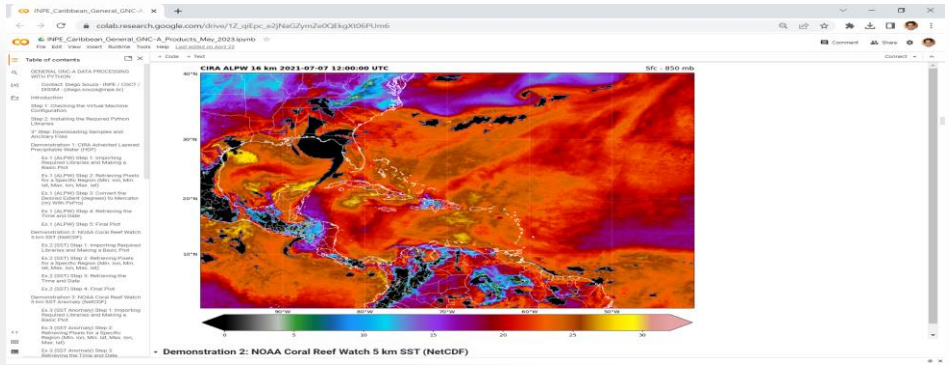
NWP (VARIOUS PLOT TYPES, OVERLAY WITH SATELLITES, METAR, ETC)

https://colab.research.google.com/drive/17yFOKezLEUhlziUUUXNlbYEwndp-gAMp?usp=sharing



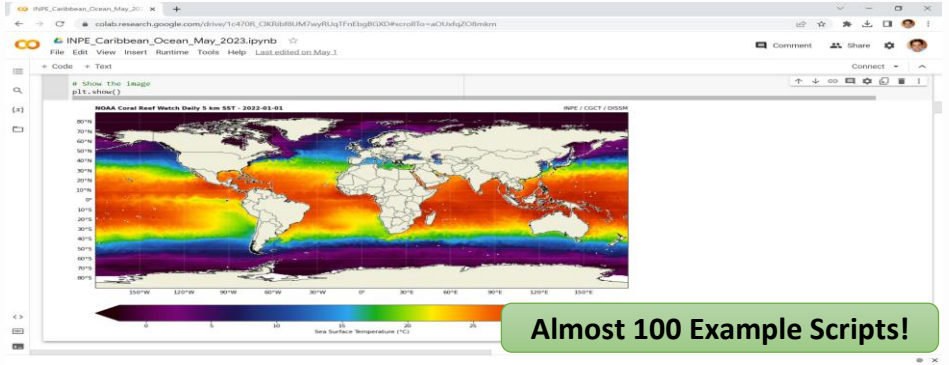
MISCELLANEOUS PRODUCTS (TPW, VEGETATION, HOTSPOTS, ETC)

https://colab.research.google.com/drive/1Z_qiEpc_e2jNaGZymZe0QEkgXt06PUm6?usp=sharing



OCEANOGRAPHIC PRODUCTS

https://colab.research.google.com/drive/1c470R_ClKRibf8UM7wyRUqTFnEbgBGXD?usp=sharing



Almost 100 Example Scripts!

For Your Future Reference

https://goesrdatajam.sched.com/event/1RI91/visualizing-and-analyzing-goes-r-data-using-google-colab

Procedure to process GOES-R data locally:
https://geonetcast.wordpress.com/2023/03/19/getting-started-with-python-and-satellite-imagery/

Main Slides From Recording PDF

Slides used for this presentation

Extra Introduction To Python PDF

Extra GOES R Data Processing With Python PDF

Extra NWP Data Processing With Python PDF

Extra Introduction To Visual Studio Code PDF

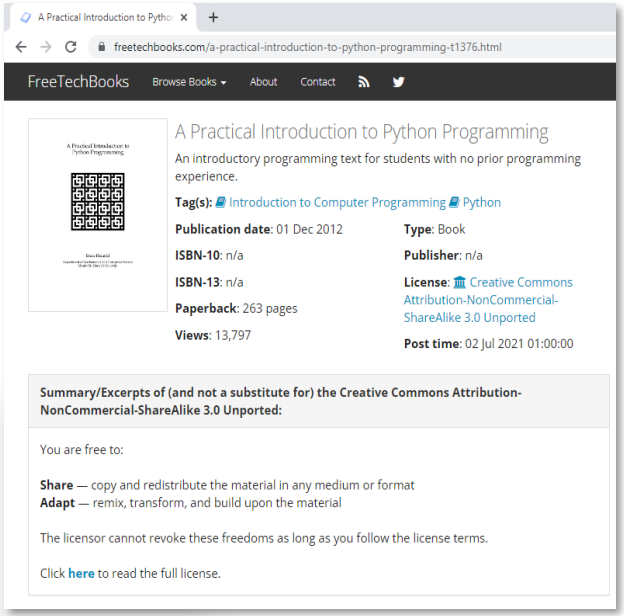
Extra resources:
- Longer introduction to Python
- Slide deck explaining each instruction from the long version of the notebook.
- Slide deck explaining how to process NWP data
- Slide deck explaining how to use Visual Studio Code, a very nice IDE (Integrated Development Environment) to process data locally.

Monday September 18, 2023 12:15pm - 12:30pm EDT
Virtual
Trainer Material, Coding Examples, ABI, GLM, Google Colab, Google

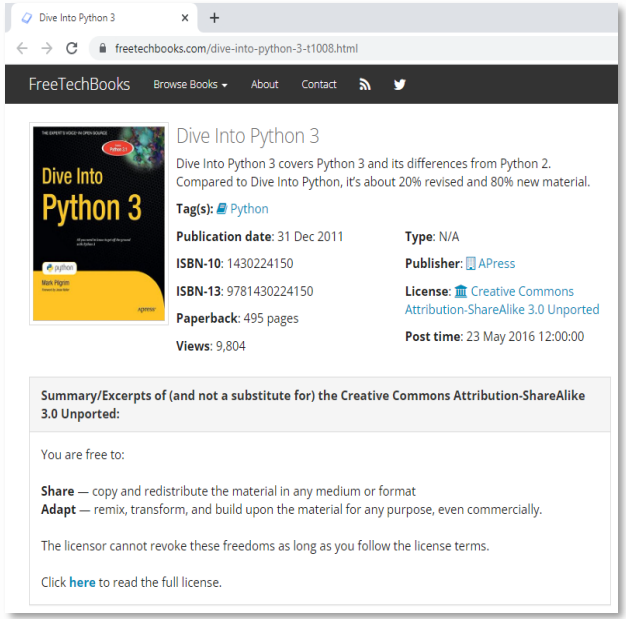
Some Useful Extra Resources

Practical introduction to Python from scratch: There are many materials available for free

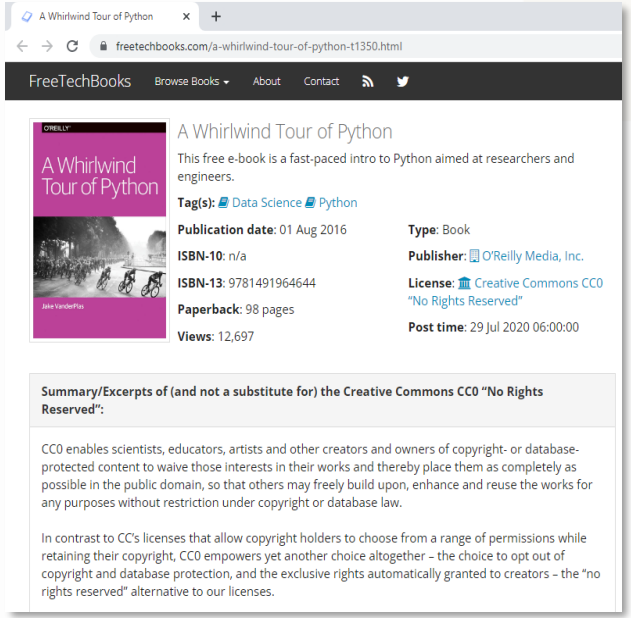
E-books distributed free of charge by the authors (CC licenses): https://www.freetechbooks.com/python-f6.html



https://www.freetechbooks.com/a-practical-introduction-to-python-programming-t1376.html



http://getpython3.com/diveintopython3/



https://jakevdp.github.io/WhirlwindTourOfPython/

...and much more

Visualizing and Analyzing GOES-R Data Using Google Colab

Diego Souza
diego.souza@inpe.br
Engineer - INPE (Brazil)
National Institute for Space Research

Marcial Garbanzo
marcial.garbanzo@ucr.ac.cr
Professor - UCR
University of Costa Rica