Purpose: I need to write a program (or programs) in Python and R that request information from the Nasa Precipitation API and parse the API response for a GeoTiff file. This GeoTiff file can be graphed in R. We will use the data from the GeoTiff file to create an algorithm for glacial change.

My current R program calls a python function for requesting and parsing API information.

Python-related Research:

- How to "GET" request from APIs
  - Module needed: requests
  - Function: requests.get(url, params=query)
    - Query is a list of conditions the API call expects (found on API documentation website)
  - After calling the function, use .json() to format the file as a json file for parsing
    - For the NASA API, the json file is a nested dictionary/list object
- File Parsing Reference
  - I had to read the documentation for NASA's API in-depth to figure out how to parse the file to get the GeoTiff link from the API response
    - The GeoTiff file can be found under: 'items' >> 0 >> 'action' >> 1 >> 'using' >> 1 >> 'url'
- Refresher on opening files in Python
  - I had to perform another GET command before the file was ready to be opened and written into a file on my computer
- Updating the Date in the API Call
  - We need the API to call with the current date → need to find a python module that consistently updates the date
    - Datetime can update the current date using date and can find yesterday's date using timedelta

R-related Research:

- How to "GET" request from API
  - Library needed: httr
  - Function GET(url, query = list())
    - Query is a list of conditions the API call expects (found on API documentation website)
  - After we request the data, we have to parse the API call for the GeoTiff file
- Turning GeoTiff into Raster file
  - Raster files are the file type we need to open and analyze precipitation data in the form of GeoTiff files
  - Documentation on Raster files versus other file types provided by our NASA API: https://arthurhou.pps.eosdis.nasa.gov/Documents/README.GIS.pdf
    - GeoTiff files allow us to access precipitation data for every point in an area

            - This is helpful because we can request data for a glacier and see what the precipitation was at each section of the glacier
                - The precipitation on the glacier affects its movements
    - Use raster("file.tif") to create a raster object
- Downloading GeoTiff files
    - Geotiff files are large and complicated, so they can't be opened like normal files would be opened in R by download.file()\
    - Curl_download effectively downloads GeoTiff files with the right file extension
    - It will save the downloaded file in your current working directory
- Opening GeoTiff files visually
    - Use the plot(data, title_attributes, etc) function
    - Needs the rgdal library and the raster library
        - These libraries support and graph raster objects
    - Precipitation data can be plotted as a DEM (digital elevation model)
        - In this case, "elevation" would be mm of precipitation in the area
    - Result: a heatmap of precipitation in a certain area
    - Important: GeoTiff files from the API are too large for R to handle
        - So use xlim and ylim parameters in plot() to limit the amount of data R is plotting each time it opens the file
- Allowing R to Delete my files to reset them
    - I was having a problem where my GeoTiff files (which I am loading in R) were not resetting and changing to the new data I input to them each time
        - So I had to learn how to clear files that were created in R
            - Use exists(file_name) to figure out if the file exists in the current working directory
            - Delete the file using unlink(file_name) to remove the file from the current working directory
            - From there, we can use curl_download to put new data in our GeoTiff files

Allowing R access to Python Functions:

- Reticulate
    - Library for R that allows R to access functions, classes, etc from .py files in the same directory as the R script
    - Use source_python("file.py") to indicate to R which file to look for a function in
    - The function can be called in R the same way it can be called in the .py file after that

GDAL Package:

- The GDAL package in Python (and rgdal for R) help analyze geographical data in R
    - The package specializes in analyzing API calls for geographical data and interpreting GeoTiff files
- The part of GDAL we want to use in Python is GDALinfo

        - GDALinfo has a special function called -json that makes our API call response a json file which can be parsed as a dictionary more effectively than HTML, Java, or other response types
            - The original API response is not yet in a format that Python / a User can interact with to extract information → it must be converted to JSON
- In R, we use rgdal to graph and interpret GeoTiff files
    - Creates a “heat map” that represents precipitation data for a certain location (latitude and longitude)

Example of Analyzing API/GeoTiff data:

- Landsat 8 is another satellite that is collecting world landmass/coastline geological data, like our NASA API is collecting world precipitation data
- The Landsat 8 data is stored as a multi-layer (multi-band) GeoTiff file
    - These bands correspond to the different colors in the GeoTiff files
        - The band colors represent a certain “histogram bucket”
            - Example: blue is 10-20 mm of precipitation
            - Our data will be in these buckets as well
- In our program, we can analyze our NASA API data by separating the bands and seeing where these bands apply
    - Will tell us how much precipitation is in each area of the glacier
        - Tells us how the glacier moves and changes

What is a GeoTiff File?

- Specific type of Tiff file (store graphics/art in array of pixels)
    - For storing geographical data
- Industry standard adopted by NASA for their satellite API
- Applications: digital elevation models (DEM), precipitation data, storing satellite observation data

Vector Files vs Raster Files

- We have two choices for our API response when requesting data from NASA: the vector data or the raster data
    - Vector data: returns the maximum precipitation value for the area requested, objects represented as curves rather than pixels
    - Raster data: returns a GeoTiff file which maps the precipitation at each point in the raster image (array of points of precipitation measurements)
- We want to use Raster data → increased accuracy on how glaciers might be changing in different places
- However, vector files might be a good starting point for implementation because it might be easier → I’m going to have to discuss this with my project lead

Extra Vector Research

- Shapes in an image are made of mathematically created curves
    - In Raster files (like GeoTiff), images are made of colored pixels

- Application of vector files: .ai files in adobe illustrator
    - Adobe illustrator is a graphic design / digital art tool
    - Never gets pixel-y, which is pretty cool

Starting a New Approach: Using only Python to open and read GeoTiffs

- Using sample code from the National Snow and Ice Data Center on reading and interpreting GeoTiff files
- This approach contains a lot of packages that I'm not sure how to use so I'm doing some research on them
    - Rasterio: python package that opens and interprets GeoTiff files
        - Can visualize geographical data in GeoTiff form
        - Can be converted into a dataset
        - Can also be used to create raster files
            - For more on raster files, see above (vector vs raster section)
        - Dependent on GDAL (package discussed above under the python research section)
    - Affine: a package whose main function is to perform affine transformations on data (preserves shapes without spatial awareness)
        - Basically does transformations of the data (image) its given
            - In our case, it is transforming the raster file we are giving it
                - This results in an image that only contains our shapes and raster bands (see Example of Analyzing Raster Data section for more information on raster bands)
                    - These raster band shapes are what we are looking to analyze
                        - Raster bands contain the areas that a certain level of precipitation fell on
        - Affine is also transforming raster data into data that matplotlib will understand and be capable of graphing
        - Involves matrix algebra, which is beyond my understanding
    - Matplotlib: a python package for creating high-quality graphs in python
        - Once we transform our raster data, matplotlib can graph it
        - We can edit axis labels and other aesthetics through matplotlib as well
        - Matplotlib will also save our graphs if we need to
            - We might be able to create a download option for our website using this feature
                - I will need to talk to the frontend people