Coding Notes
Author: Hude Hude

This document illustrates how I managed to complete each task, including how I found code references, what problems I encountered, and how I handled issues. I had no experience dealing with TIF data in Python before, so I studied several libraries and packages that could be helpful in handling this format of data.

Several initial experiments were made:

from PIL import Image; import numpy. The NumPy Array list that extracts each pixel value and stores it into an array did not work because of a MemoryError. Even controlling via Image.MAX_IMAGE_PIXELS = None, a similar memory error occurred.
import rioxarray; import matplotlib.pyplot as plt. I tried these, hoping to first visualize the GeoTIFF image. It failed due to the same reason.
Use import fiona; import rasterio; and from rasterio.mask import mask. It successfully read the data and stored it in a nested N-dimensional array. In our case, it is 2D since the band value = 1.

Task 1:

Shapefiles were downloaded from GitHub:
https://github.com/gamamo/AmazonBasinLimits

Methods of masking are consulted from Stack Exchange and Google results in general. This part is mainly from here:
https://stackoverflow.com/questions/41462999/how-do-i-use-rasterio-python-to-mask-a-raster-using-a-shapefile-to-set-the-rast
https://rasterio.readthedocs.io/en/stable/topics/masking-by-shapefile.html

Here is a record of failure: (not quite sure why it doesn't work)

```
# bounding_box = box(*amazon_biome.total_bounds)
# clipped_geometry = gpd.GeoDataFrame(geometry=[bounding_box], crs=amazon_biome.crs)
#
# # Clip the satellite image to the Amazon biome boundary
# # Extract the geometry of the Amazon biome
# amazon_biome_geometry = amazon_biome.geometry.iloc[0]
#
# # Use the mask function to clip the satellite image based on the Amazon biome geometry
# clipped_image, transform = mask(dataset, [mapping(amazon_biome_geometry)], crop=True)
#
# # Get metadata from the original image and update with the new transform
```

```
# # This metadata will be used when saving the clipped image
# meta = dataset.meta.copy()
# meta.update({
#     'driver': 'GTiff',  # The driver for GeoTIFF format
#     'height': clipped_image.shape[1],  # Height of the clipped image
#     'width': clipped_image.shape[2],   # Width of the clipped image
#     'transform': transform  # Affine transform for mapping pixel coordinates to geographic
coordinates
# })
#
# # Write the clipped image to a new file
# output_path = "coverage_brazil/brasil_coverage_1985_clipped.tif"
# with rasterio.open(output_path, 'w', **meta) as dest:
#     dest.write(clipped_image)
```

As a result, this gives me a BLUE SQUARE rather than the Amazon Biome shape image.

Reasonable step is checking if the shapefiles are representing the real shape of Amazon Biome. So, I created a program, namely Check_Shapefile, to visualize the shape described by the shapefile.

Google results suggest use these two packages:
https://stackoverflow.com/questions/30447790/displaying-a-shapefile

```
import geopandas as gpd
import matplotlib.pyplot as plt
```

Example:

```
import geopandas as gpd
shape=gpd.read_file('shapefile')
shape.plot()
```

As a result, Check_shapefiles confirmed that the shapefiles obtained are valid.

To check if I obtained the right subsample, I created Check_clip.py that prints the value of random points and plot a subset of the entire image. By trying different windows, I confirmed that the amazon_coverage_1985.tif is correctly collected.

****

Task2:

When reading pixel values from one dataset and compare it with another dataset, memory error occurs again

Error Message: Unable to allocate 22.9 GiB for an array with shape (158459, 155239) and data type bool
Solution: I tried to use "chunk" via iterating through several windows of a image.
Here, I referred to this webpage: https://rasterio.readthedocs.io/en/stable/topics/windowed-rw.html

I also used several built-in methods that, ideally, provide me a shortcut to assign and compare pixel values. This one (Lable_Forest.py) failed as it gave an output with pixel value of zero for all points.

A Case test using Check_Legacy:

```
# This returns unexpected result:
# Coordinates: (55785, 28536), Legacy Pixel Value: 0
# Coordinates: (55785, 28536), 1986 Pixel Value: 0
# Coordinates: (55785, 28536), 1985 Pixel Value: 0
# I should expect pixel value for both year to be 3, and legacy pixel value = 1
```

Revised version using dask to let the PC determine proper size of chunk to work with.
It works successfully. Result form Check_Legacy:

```
C:\RA_Projects\RA_CodingTest\.venv\Scripts\python.exe C:
\RA_Projects\RA_CodingTest\Check_Legacy.py
Pixel at (61222, 5039): Classified as Forest - False
Pixel at (44265, 37567): Classified as Forest - True
Pixel at (51345, 65140): Classified as Forest - True
Pixel at (105101, 15473): Classified as Forest - False
Pixel at (33951, 22039): Classified as Forest - True
Number of Bands (Legacy): 1
Coordinates: (55785, 28536), Legacy Pixel Value: 1
Coordinates: (55785, 28536), 1986 Pixel Value: 3
Coordinates: (55785, 28536), 1985 Pixel Value: 3
```

Here, Coordinates: (55785, 28536), 1986 Pixel Value: 3 and Coordinates: (55785, 28536), 1985 Pixel Value: 3 suggest that this pixel should be assigned as Forest. Equivalently, in boolean value, it should correspond with value of 1. Hence, the result, Coordinates: (55785, 28536), Legacy Pixel Value: 1, is consistent with expectation.

Another way to check legacy forest data is to use Check_clip again. Check_clip provide means to check data pints as well as provide a visual representation.

****

Task3:

Result from Get_LegacyArea.py:

Total Number of Pixels: 11892635340
Total Legacy Forest Area: 404506286.19 hectares.

Method for counting number of occurrence for certain elements in the numpy array:
https://stackoverflow.com/questions/28663856/how-do-i-count-the-occurrence-of-a-certain-item-in-an-ndarray

Problem encountered initially:

```
# Here is a method that leads to an unreasonable result. I don't know why it goes wrong.
# forest_pixels = (legacy_coverage == 1).sum()
# # Calculate the total area covered by legacy forest in square meters
# total_forest_area_m2 = forest_pixels * (30.0 * 30.0)  # Use floating-point numbers
# # Convert the area to hectares
# total_forest_area_hectares = total_forest_area_m2 * 0.0001
# print(f"Total Legacy Forest Pixels: {forest_pixels}")
# print(f"Total Legacy Forest Area: {total_forest_area_hectares:.2f} hectares")
# # This gives the following result, which is considered to be wrong as it is too small:
# # Total Legacy Forest Pixels: 199546995
# # Total Legacy Forest Area: 17959229.55 hectares
```

I know this result is not trustworthy by checking a statistics done by other authorities.

One credible reference is found here:
https://www.maaproject.org/2022/amazon-tipping-point/
Quote: "We found that the original Amazon forest covered over 647 million hectares (647,607,020 ha). This is equivalent to 1.6 billion acres."

Result obtained via total_forest_pixels2 is about 404 million hectares, which is reasonable.

****

Task 4:

Problem 1: chunk dimension mismatch:

```
 # Create a new array with three values: 0 for non-legacy, 1 for remained legacy, and 2 for deforested
    data_1987 = da.select(
        [~is_legacy, is_legacy & ~is_deforest_1987, is_legacy & is_deforest_1987],
        [0, 1, 2],
        default=0  # Default value for pixels not covered by any condition
    )
```

ValueError: Chunks and shape must be the same length/dimension.
Got chunks=(), shape=(105405, 112828)

Solution:

Adopting the following built-in functions from dask.

```
# Create a new array with the same shape and three values: 0 for non-legacy, 1 for remained legacy, and 2 for deforested
    data_1987 = da.full_like(image_legacy, fill_value=0, dtype=np.uint8)
    data_1987 = da.where(is_legacy & ~is_deforest_1987, 1, data_1987)
    data_1987 = da.where(is_legacy & is_deforest_1987, 2, data_1987)
```

Fixed Chunk Error by creating a copy of data with same shape and dimension to start with.

Get_Deforest_1987.py:

```
C:\RA_Projects\RA_CodingTest\.venv\Scripts\python.exe C:
\RA_Projects\RA_CodingTest\Get_Deforest_1987.py
Total Deforest Area in 1987: 1911754.26 hectares
Total Legacy Forest Area: 404506286.19 hectares
Deforest Rate: 0.00
```

Here exhibits rounding bias. The ratio is very small, which is approximately 0.004726. It is not equal to 0.

Improvement in percision and achieving update in calculation with more than one year:
Get_Deforest.py:

```
C:\RA_Projects\RA_CodingTest\.venv\Scripts\python.exe C:
\RA_Projects\RA_CodingTest\Get_Deforest.py
Total Deforest Area in 1987: 1911754.26 hectares
Deforest Rate in 1987: 0.4726%
Total Deforest Area in 1988: 3641055.30 hectares
Deforest Rate in 1988: 0.9001%
```

Final Verion:
Get_Deforest_Nested: A loop that takes care of all the years.

Problem Remained:
Can't write out the output data. My guess is that the problem has something to do with the use of chunk nested in a loop.
Originally, the method of chunk allows me to write out the output data without a dataset that is updating through each year.
I tried to fix it, but so far there is no good solution.

Here is a list of calculation of deforest rate from Get_Deforest_Nested.py:
Total Deforest Area in 1987: 1911754.26 hectares
Deforest Rate in 1987: 0.4726%
Total Deforest Area in 1988: 3641055.30 hectares
Deforest Rate in 1988: 0.9001%
Total Deforest Area in 1989: 5053891.41 hectares
Deforest Rate in 1989: 1.2494%
Total Deforest Area in 1990: 6229313.46 hectares
Deforest Rate in 1990: 1.5400%
Total Deforest Area in 1991: 7460172.54 hectares
Deforest Rate in 1991: 1.8443%
Total Deforest Area in 1992: 9145510.20 hectares
Deforest Rate in 1992: 2.2609%
Total Deforest Area in 1993: 10772832.15 hectares
Deforest Rate in 1993: 2.6632%
Total Deforest Area in 1994: 12597069.42 hectares
Deforest Rate in 1994: 3.1142%

****

Task 5:

Since I am unable to get an output dataset from Get_Deforest_Nested.py, I am unable to work on this task.

Here is an idea on how to do task 5:
1. Read both legacy_coverage.tif and the resulting data from Get_Deforest_Nested.py.
2. Downsample the data by some factor
3. Assign colors based on the pixel value for plot
4. Plot use methods similar to what I used in Check_clip.py