Planet image processing batch codes.

1. **Set up directory structure.**
   a. Make a folder, I suggest using site name and year i.e. EasternShore_2023.
      i. *from this point forward this is always going to be your top working directory folder*
   b. Inside this folder make another folder called 0_Zipped (important to use this exact name)
   c. Place your downloaded zipped image files in this folder.

Final directory structure

Folder name	File type	File names (examples)
EasternShore_2023	Top directory, all other folders will reside here.	
0_Zipped	Your downloaded, zipped files	EasternShore_2June2023_psscene_analytic_sr_udm2.zip
1_Images	Unzipped files, in named folders	EasternShore_2June2023\PSScene\20230602_153140_36_2483_3B_AnalyticMS_SR_harmonized_clip.tif
2a_Mosaic	Mosaic tiles, same day, same sensor	20230602_2483_mosaic.tif
2b_Composite_images	Images composited with DEM raster	20230602_2483_composite.tif
3_Classified_composite	Composited images classified	20230602_2483_classified.tif
4_LAI	Retrieved LAI	20230602_2483_LAI.tif
5a_SAV_presence	Presence/absence raster, 1 = SAV, 0 =all others	20230602_2483_SAVpresence.tif
5b_pixels_imaged	Identified valid imaged pixels, 1 = valid pixel	20230602_2483_reclass_imaged.tif

2. **Unzipping files.**
   a. Code needed unzippmyfiles.py
   b. Python packages needed: tkinter, zippfile, os
   c. This code will ask you to select your working directory, that is the folder with the site name and year.
   d. It will look for a folder named 0_Zipped, it will identify all zipped files, unzip them and place them in a new folder called 1_Images.
      i. Each unzipped file set will be in its own folder inside 1_Images that retains the site name, date and year.
      ii. Depending on how you name your Planet orders you might want to change this.

3. **Mosaic tiles from same day, same sensor**
    a. Code needed PlanetProcessing1_Mosaic_V1.py
    b. Python packages needed: tkinter, zippfile, os, shutil, arcpy
    c. This code will ask for your working directory, that is the folder with the site name and year.
    d. It searches 1_Images for valid .tif files by looking for files with names ending "harmonized_clip.tif", you will need to change this if your files have different names.
    e. It makes a list of all individual days and sensor IDs.
    f. It makes a folder called "1b_MosaicTxtFiles" and places in here text files containing the names of all .tif files that have the same day and sensor number.
    g. It makes a folder called 2a_Mosaic
    h. It iterates through these text files and mosaics tiles together, placing them in folder 2a_Mosaic and renaming the new files as YYYYMMDD_sensorID_moasic.tif
        i. At the start of this code there is a variable named "numberofbands" you will need to set this o 4 or 8 depending on what data you downloaded.
    i. Due to the need to have all files retain the same naming convention, at the moment, if there is only 1 tile, it will copy this tile to 2a_Mosaic with the new name YYYYMMDD_sensorID_moasic.tif

4. **Clipping and generating composite with DEM**
    a. Code needed PlanetProcessing2_Clip_and Composite_V1.py
    b. Python packages needed: tkinter, zippfile, os, shutil, arcpy
    c. This code will ask for your working directory, that is the folder with the site name and year.
    d. Code will ask if you want to clip your images. You may want to clip your images smaller than the clip that Planet did. If so, you will need to make a shapefile polygon of your clip extent and indicate this file when asked by the code.
    e. If you type no to clipping then the code will skip straight to compositing with DEM.
    f. Code makes a folder called 2b_Composite_images that it places the new files in.
    g. Composite files have name YYYYMMDD_sensorID_composite.tif
    h. Code also makes a TEMP folder to place temp files in, it will be deleted after the code runs.
    i. Code will check to see if this file has already been created, if so it will skip to the next file. This means that as you download and add more images to 1_Images you can run this code and it will only process those files that haven't been created yet.
    j. As part of the code it will take non-valid pixels and replace values with nan, this needs to be done as other those pixels will get classified. Non-valid pixels are those outside of the clip extent.

5. **Classifying and generating frequency of SAV presence raster's.**
    a. Code needed PlanetProcessing3_Classify_and_generateFreq_V1.py
    b. There is another version of the code that works if you have one set of ROIs for each image. PlanetProcessing3_Classify_and_generateFreq_V1_indivROIs.py
    c. Again you will indicate which working directory you want.
    d. Three new folders are made. 3a_Classified_composite, 5a_SAV_presence and 5b_pixels_imaged
    e. You will be asked for the location of the DEM file and your ROI shapefile
    f. The code uses the ROIs and train a classification it then makes a .ecd file
    g. The .ecd file is used to classify the entire image.
    h. The code uses the DEM to mask out pixels where the depth is positive or deeper than 2 m. You can change that if you want. Output is saved as YYYYMMDD_sensorID_classified.tif
    i. Next a SAVpresence raster is made, where SAV = 1 and all other pixels are = 0. This is saved in 5s_SAV_presence folder. This is a binary raster.
    j. Next a pixels imaged raster is made, if a pixel has a valid value then it is set to 1 and all others = 0. This provides a binary raster where any pixel that was imaged is set to 1.
        i. This is used to calculate % times pixel is classified as SAV.
    k. Once all files have been classified the code will generate a raster called EasternShore2023_SAVpresence.tif which is calculated as total number of times a pixels was classified as SAV in all images / number if times a pixel was imaged.
        i. The code will always overwrite this file with a new one each time you run the code. When you add new images to your folders and run this code it will recalculate the % SAV raster.