PROJECT 1: SATELLITE IMAGE DATA ANALYSIS USING NUMPY

Reference : <https://drive.google.com/drive/folders/1Xe3HftLxL1T6HsEBUfjq_zXANjTnr6Cz?usp=sharing>

import numpy as np  
import scipy  
import imageio  
import matplotlib.pyplot as plt  
import skimage  
import warnings  
warnings.filterwarnings("ignore")  
  
  
from skimage import data  
photo\_data = imageio.imread('data/sd-3layers.jpg')  
type(photo\_data)  
  
  
  
plt.figure(figsize=(10,10))  
plt.imshow(photo\_data)  
plt.show()  
  
  
print(photo\_data.shape)  
  
  
  
photo\_data.size  
photo\_data.min(), photo\_data.max()  
  
photo\_data.mean()  
  
photo\_data[150, 250]  
  
photo\_data[150, 250, 1]  
  
photo\_data[1, 1]

photo\_data = imageio.imread('data/sd-3layers.jpg')  
photo\_data[150, 250] = 0   
plt.figure(figsize=(10, 10))  
plt.imshow(photo\_data)  
plt.show()  
  
  
photo\_data = imageio.imread('data/sd-3layers.jpg')  
photo\_data[200:800, :, 1] = 255  
plt.figure(figsize=(10, 10))  
plt.imshow(photo\_data)  
plt.show()  
  
photo\_data = imageio.imread('data/sd-3layers.jpg')  
photo\_data[200:800, :] = 255  
plt.figure(figsize=(10,10))  
plt.imshow(photo\_data)  
plt.show()  
  
  
  
  
photo\_data = imageio.imread('data/sd-3layers.jpg')  
photo\_data[200:800, :] = 0  
plt.figure(figsize=(10, 10))  
plt.imshow(photo\_data)  
plt.show()  
  
  
photo\_data = imageio.imread('data/sd-3layers.jpg')  
print("Shape of photo\_data:", photo\_data.shape)  
low\_value\_filter = photo\_data < 100  
print("Shape of low\_value\_filter:", low\_value\_filter.shape)  
  
  
  
plt.figure(figsize=(10,10))  
plt.imshow(photo\_data)  
photo\_data[low\_value\_filter] = 0 #set low values to 0  
plt.figure(figsize=(10, 10))  
plt.imshow(photo\_data)  
plt.show()  
  
  
  
rows\_range = np.arange(len(photo\_data)) #create a range array  
print(rows\_range)  
cols\_range = rows\_range  
print(cols\_range)  
print(type(rows\_range))  
  
  
  
photo\_data[rows\_range, cols\_range] = 255  
print(photo\_data)  
plt.figure(figsize=(10, 10))  
plt.imshow(photo\_data)  
plt.show()  
  
  
total\_rows, total\_cols, total\_layers = photo\_data.shape  
print("photo\_data = ", photo\_data.shape)  
X, Y = np.ogrid[:total\_rows, :total\_cols]  
print("X = ", X.shape, " and Y =", Y.shape)  
from IPython.display import Image  
Image("Images/figure.png")  
center\_row, center\_col = total\_rows / 2, total\_cols / 2  
print("center\_row = ", center\_row, "AND center\_col = ", center\_col)  
  
dist\_from\_center = (X - center\_row) \*\*2 + (Y - center\_col) \*\*2  
  
radius = (total\_rows / 2)\*\*2  
  
circular\_mask = (dist\_from\_center > radius)  
  
print(circular\_mask[1500:1700, 2000:2200])  
photo\_data = imageio.imread('data/sd-3layers.jpg')  
photo\_data[circular\_mask] = 0  
plt.figure(figsize=(10,10))  
plt.imshow(photo\_data)  
  
X, Y = np.ogrid[:total\_rows, :total\_cols]  
half\_upper = X < center\_row   
half\_upper\_mask = np.logical\_and(half\_upper, circular\_mask)  
photo\_data = imageio.imread('data/sd-3layers.jpg')  
photo\_data[half\_upper\_mask] = 255  
plt.figure(figsize=(10, 10))  
plt.imshow(photo\_data)  
plt.show()  
  
  
  
photo\_data = imageio.imread('data/sd-3layers.jpg')  
red\_mask = photo\_data[:, : ,0] < 150  
photo\_data[red\_mask] = 0  
plt.figure(figsize=(10,10))  
plt.imshow(photo\_data)  
plt.show()  
  
photo\_data = imageio.imread('data/sd-3layers.jpg')  
green\_mask = photo\_data[:, :, 1] < 150  
photo\_data[green\_mask] = 0  
plt.figure(figsize=(10,10))  
plt.imshow(photo\_data)  
plt.show()  
  
  
photo\_data = imageio.imread('data/sd-3layers.jpg')  
blue\_mask = photo\_data[:, :, 2] < 150  
photo\_data[blue\_mask] = 0  
plt.figure(figsize=(10,10))  
plt.imshow(photo\_data)  
plt.show()  
  
  
photo\_data = imageio.imread('data/sd-3layers.jpg')  
red\_mask = photo\_data[:, : ,0] < 150  
green\_mask = photo\_data[:, : ,1] > 100  
blue\_mask = photo\_data[:, : ,2] < 100  
final\_mask = np.logical\_and(red\_mask, green\_mask, blue\_mask)  
photo\_data[final\_mask] = 0  
plt.figure(figsize=(10,10))  
plt.imshow(photo\_data)  
plt.show()