PROJECT 1: SATELLITE IMAGE DATA ANALYSIS USING NUMPY

*#Loading#Loading the libraries we need: numpy, scipy, matplotlib*import numpy as np  
import scipy  
import imageio  
import PIL  
import skimage *#install scikit-image*import matplotlib.pyplot as plt  
import warnings  
warnings.filterwarnings("ignore")  
from skimage import data  
photo\_data = imageio.imread(r'C:\Users\Lenovo PC 16\Documents\sd-3layers.jpg')  
type(photo\_data)  
*#Let us see what is in this image.*plt.figure(figsize=(15,15))  
plt.imshow(photo\_data)  
plt.close('all')  
*#plt.show()*print(photo\_data)  
print(photo\_data.shape)  
print(photo\_data.size)  
print(photo\_data.min())  
print(photo\_data.max())  
print(photo\_data.mean())  
print(photo\_data[150, 250])  
print(photo\_data[150, 250, 1])  
print(photo\_data[1, 1])  
photo\_data = imageio.imread(r'C:\Users\Lenovo PC 16\Documents\sd-3layers.jpg')  
photo\_data[150, 250] = 0  
plt.figure(figsize=(10,10))  
plt.imshow(photo\_data)  
plt.close('all')  
*#plt.show()  
#Changing colors in a Range*photo\_data = imageio.imread(r'C:\Users\Lenovo PC 16\Documents\sd-3layers.jpg')  
photo\_data[200:800, : ,1] = 255  
plt.figure(figsize=(10,10))  
plt.imshow(photo\_data)  
plt.close('all')  
*#plt.show()  
#We will set the value of Red, Green and Blue layer to full intensity*photo\_data = imageio.imread(r'C:\Users\Lenovo PC 16\Documents\sd-3layers.jpg')  
photo\_data[200:800, :] = 255  
plt.figure(figsize=(10,10))  
plt.imshow(photo\_data)  
plt.close('all')  
*#plt.show()  
#We will set the value of Red, Green and Blue layer to least intensity*photo\_data = imageio.imread(r'C:\Users\Lenovo PC 16\Documents\sd-3layers.jpg')  
photo\_data[200:800, :] = 0  
plt.figure(figsize=(10,10))  
plt.imshow(photo\_data)  
plt.close('all')  
*#plt.show()  
#Pick all Pixels with Low Values*photo\_data = imageio.imread(r'C:\Users\Lenovo PC 16\Documents\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  
plt.figure(figsize=(10,10))  
plt.imshow(photo\_data)  
plt.close('all')  
*#plt.show()  
#More Row and Column Operations*rows\_range = np.arange(len(photo\_data))  
print("rows range",rows\_range)  
cols\_range = rows\_range  
print("cols\_range",cols\_range)  
print(type(rows\_range))  
  
*#We are setting the selected rows and columns to the maximum value of 255*photo\_data[rows\_range, cols\_range] = 255  
print(photo\_data)  
plt.figure(figsize=(15,15))  
plt.imshow(photo\_data)  
plt.close('all')  
plt.show()  
*#Masking Images*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)  
import matplotlib.pyplot as plt  
import matplotlib.image as mpimg  
img = mpimg.imread(r'C:\Users\Lenovo PC 16\Documents\sd-3layers.jpg')  
center\_row, center\_col = total\_rows / 2, total\_cols / 2  
print("center\_row = ", center\_row, "AND center\_col = ", center\_col)  
print("X-center row",X - center\_row)  
print("Y-center row",Y - center\_col)  
dist\_from\_center = (X - center\_row)\*\*2 + (Y - center\_col)\*\*2  
print("dist\_from\_center",dist\_from\_center)  
radius = (total\_rows / 2)\*\*2  
print("Radius = ", radius)  
circular\_mask = (dist\_from\_center > radius)  
print("circular\_mask",circular\_mask)  
print("circular\_mask\_New",circular\_mask[1500:1700,2000:2200])  
photo\_data = imageio.imread(r'C:\Users\Lenovo PC 16\Documents\sd-3layers.jpg')  
photo\_data[circular\_mask] = 0  
plt.figure(figsize=(15,15))  
plt.imshow(photo\_data)  
plt.close('all')  
*#plt.show()  
#Further Masking*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(r'C:\Users\Lenovo PC 16\Documents\sd-3layers.jpg')  
photo\_data[half\_upper\_mask] = 255  
plt.figure(figsize=(15,15))  
plt.imshow(photo\_data)  
plt.close('all')  
*#plt.show()  
#detecting high intensity RED Pixels and muting down other areas*photo\_data = imageio.imread(r'C:\Users\Lenovo PC 16\Documents\sd-3layers.jpg')  
red\_mask = photo\_data[:, : ,0] < 150  
photo\_data[red\_mask] = 0  
plt.figure(figsize=(15,15))  
plt.imshow(photo\_data)  
plt.close('all')  
*#plt.show()  
##Detecting Highl-GREEN Pixels*photo\_data = imageio.imread(r'C:\Users\Lenovo PC 16\Documents\sd-3layers.jpg')  
green\_mask = photo\_data[:, : ,1] < 150  
photo\_data[green\_mask] = 0  
plt.figure(figsize=(15,15))  
plt.imshow(photo\_data)  
plt.close('all')  
*#plt.show()  
#Detecting Highly-BLUE Pixels*photo\_data = imageio.imread(r'C:\Users\Lenovo PC 16\Documents\sd-3layers.jpg')  
blue\_mask = photo\_data[:, : ,2] < 150  
photo\_data[blue\_mask] = 0  
plt.figure(figsize=(15,15))  
plt.imshow(photo\_data)  
plt.close('all')  
plt.show()  
*#Composite mask that takes thresholds on all three layers: RED, GREEN, BLUE*photo\_data = imageio.imread(r'C:\Users\Lenovo PC 16\Documents\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=(15,15))  
plt.imshow(photo\_data)  
plt.show()