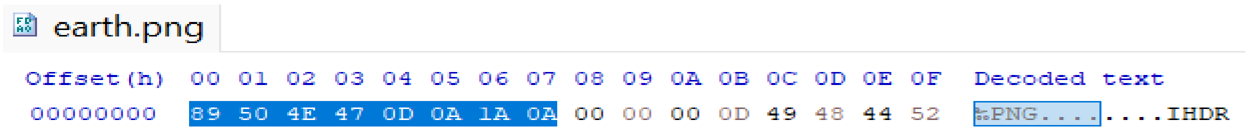
# Report 3

This week we worked on handling image files using the command line and python. This involved using programs that allow you to see hashed values of files. We also utilized the command line in Windows to create the hash values. Then we used python to create the hash values and compared them to verify if they were identical. The last task was to create hidden messages and pixel values to then be located using python.

The first part was to use HxD File Analysis. The first step was to download an image from pixabay.com. The picture I choose was an image of Earth. I saved the image as a .png file.

This is the picture that I used

The next step was to open Hxd. I looked at the signature to verify that this .png file has the correct corresponding signature for a PNG file. The signature is highlighted below.

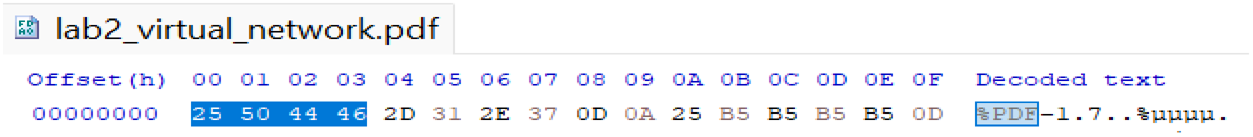


earth.png

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text
00000000   89 50 4E 47 0D 0A 1A 0A 00 00 00 0D 49 48 44 52  ‰PNG........IHDR
```

This signature corresponds to a PNG file. Each type of file has a unique signature. This can be used to distinguish what type of file this is.

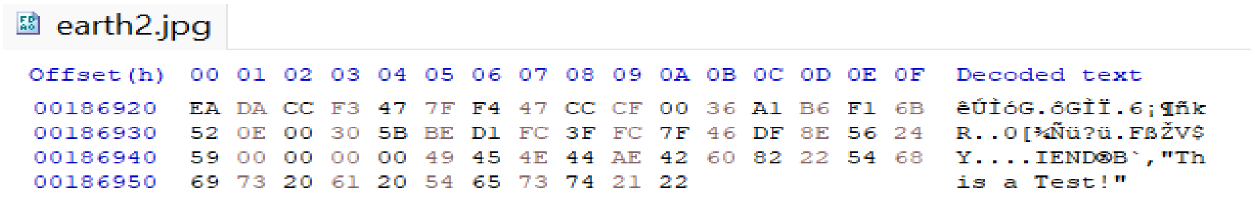
Here is the PDF signature of last week's lab slides



lab2_virtual_network.pdf

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text
00000000   25 50 44 46 2D 31 2E 37 0D 0A 25 B5 B5 B5 B5 0D  %PDF-1.7..%µµµµ.
```

The next step was to add a secret message to the end of the file. I created files called earth.jpg and earth2.jpg file. And I added the message **"This is a Test"** to the end of the file of the earth2.jpg file.

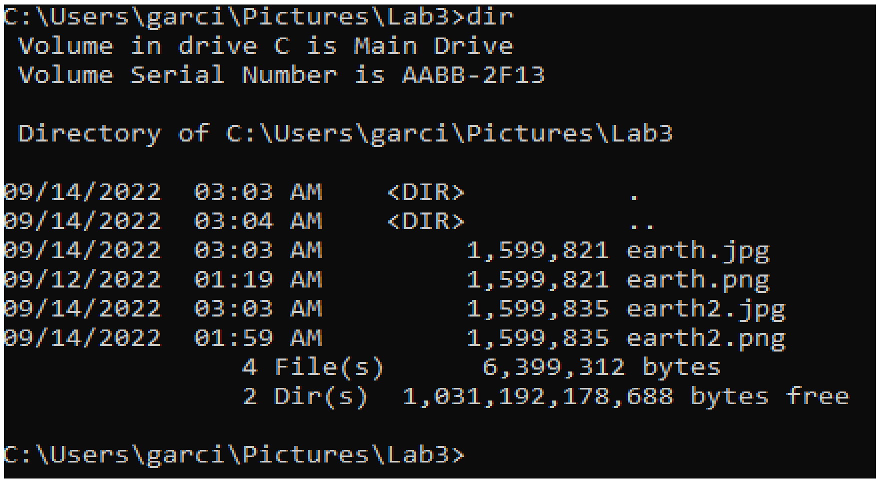


earth2.jpg

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text
00186920   EA DA CC F3 47 7F F4 47 CC CF 00 36 A1 B6 F1 6B  êÚÌóG.ôGÌÏ.6¡¶ñk
00186930   52 0E 00 30 5B BE D1 FC 3F FC 7F 46 DF 8E 56 24  R..0[¾Ñü?ü.FßŽV$
00186940   59 00 00 00 00 49 45 4E 44 AE 42 60 82 22 54 68  Y....IEND®B`‚"Th
00186950   69 73 20 61 20 54 65 73 74 21 22                 is a Test!"
```

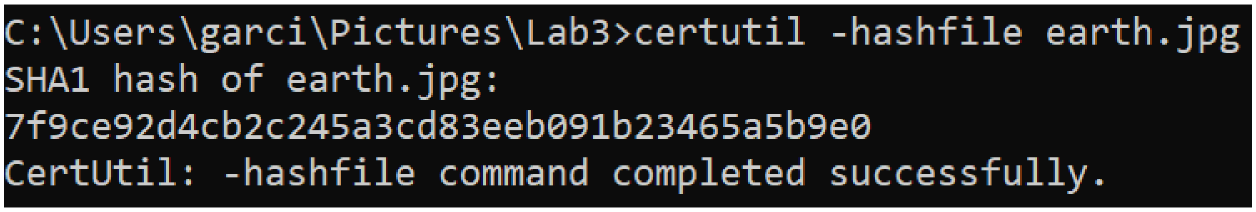
The file sizes appear to be nearly identical in size. Th earth2.jpg file has only a bit more bytes than the earth.png file. It is interesting because the side of the image by side looks identical. There were no visible changes in pixel quality and color that I can see.

The next part of the lab involved hashing using the command prompt. The first step was to open a command prompt and run it as an administrator. The next step was to change the working directory to where the images are saved. The command I used to access the location of my saved pictures was **cd C:\Users\garci\Pictures\lab3**



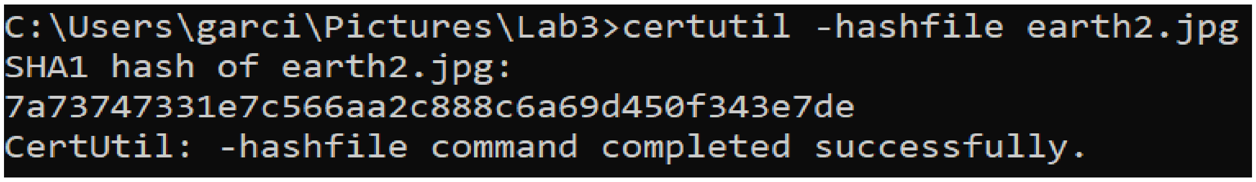
```
C:\Users\garci\Pictures\Lab3>dir
 Volume in drive C is Main Drive
 Volume Serial Number is AABB-2F13

 Directory of C:\Users\garci\Pictures\Lab3

09/14/2022  03:03 AM    <DIR>          .
09/14/2022  03:04 AM    <DIR>          ..
09/14/2022  03:03 AM         1,599,821 earth.jpg
09/12/2022  01:19 AM         1,599,821 earth.png
09/14/2022  03:03 AM         1,599,835 earth2.jpg
09/14/2022  01:59 AM         1,599,835 earth2.png
               4 File(s)      6,399,312 bytes
               2 Dir(s)  1,031,192,178,688 bytes free

C:\Users\garci\Pictures\Lab3>
```

I used the command **certutil -hashfile earth.jpg** for the first image



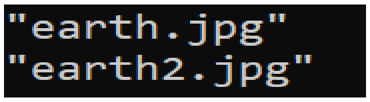
```
C:\Users\garci\Pictures\Lab3>certutil -hashfile earth.jpg
SHA1 hash of earth.jpg:
7f9ce92d4cb2c245a3cd83eeb091b23465a5b9e0
CertUtil: -hashfile command completed successfully.
```

And the **certutil -hashfile earth2.jpg** for the first second image



```
C:\Users\garci\Pictures\Lab3>certutil -hashfile earth2.jpg
SHA1 hash of earth2.jpg:
7a73747331e7c566aa2c888c6a69d450f343e7de
CertUtil: -hashfile command completed successfully.
```

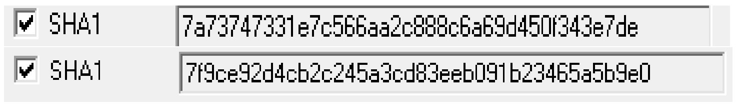
The **FORFILES /M *.jpg /C “cmd /c echo @FILE”** command displays both files that are JPG.



```
"earth.jpg"
"earth2.jpg"
```

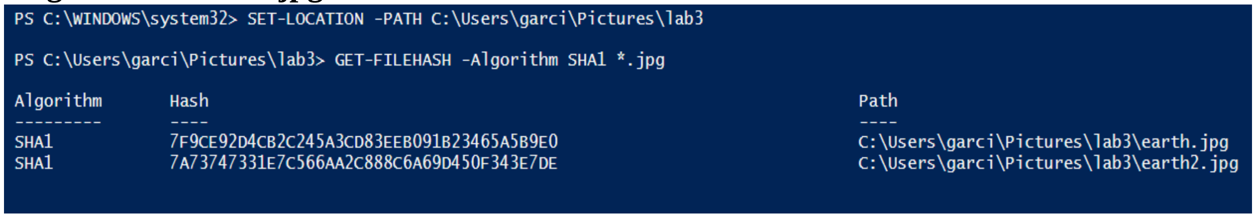
The  **FORFILES /M *.jpg /C “cmd /c certutil -hashfile @FILE”** command hashes both values at once by looking for files for a specific mask. They are the same hash values as above.

Using the HashCalc Software they resulted in the same hash values as in the command line for both earth.jpg files.



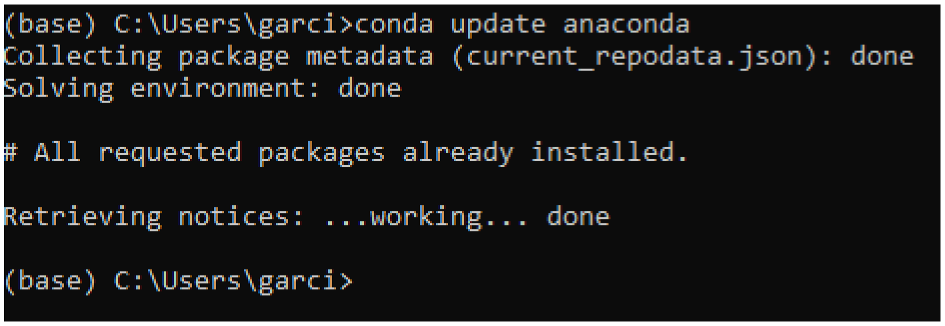
☑ SHA1 7a73747331e7c566aa2c888c6a69d450f343e7de
☑ SHA1 7f9ce92d4cb2c245a3cd83eeb091b23465a5b9e0

The next step was the hash files using PowerShell. I used the command **SET-LOCATION -PATH** to set the path to **C:\Users\garci\Pictures\lab3** to hash the files using **GET-FILEHASH -Algorithm SHA1 *.jpg** command.



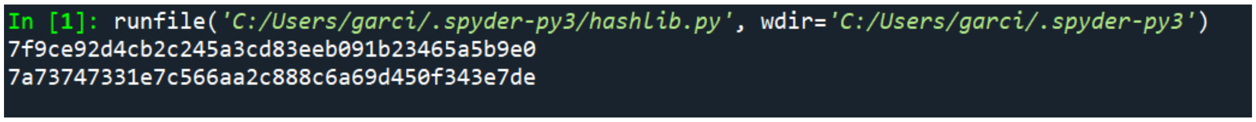
```
PS C:\WINDOWS\system32> SET-LOCATION -PATH C:\Users\garci\Pictures\lab3

PS C:\Users\garci\Pictures\lab3> GET-FILEHASH -Algorithm SHA1 *.jpg

Algorithm       Hash                                                                   Path
---------       ----                                                                   ----
SHA1            7F9CE92D4CB2C245A3CD83EEB091B23465A5B9E0                               C:\Users\garci\Pictures\lab3\earth.jpg
SHA1            7A73747331E7C566AA2C888C6A69D450F343E7DE                               C:\Users\garci\Pictures\lab3\earth2.jpg
```

The next part of the lab involved using python to hash the values. Before I opened Spyder (anaconda3), had to update it first so I used the **conda update anaconda** command to do this. All the latest packages were already installed on my computer.



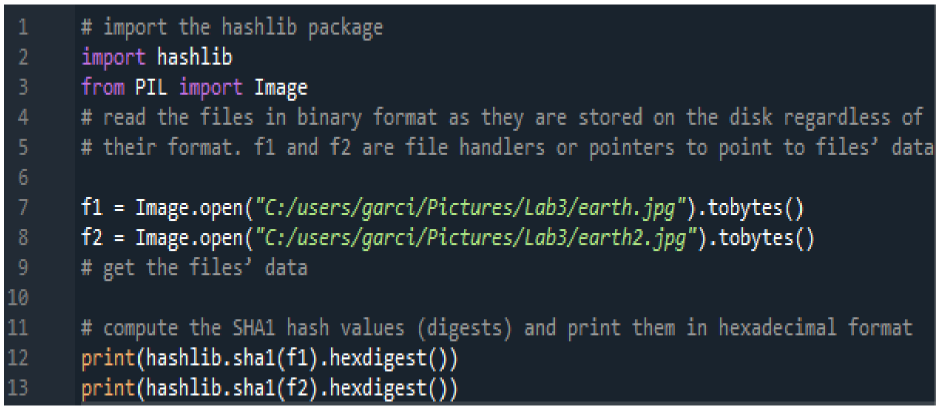
```
(base) C:\Users\garci>conda update anaconda
Collecting package metadata (current_repodata.json): done
Solving environment: done

# All requested packages already installed.

Retrieving notices: ...working... done

(base) C:\Users\garci>
```

The next step was to hash the two JPG files and print out the hash values. I created a file called lab3_part3.py to hash both files using the **import hashlib** package. The earth.jpg and earth2.jpg contained the same hash values as the ones I did in the command line and using HashCalc.



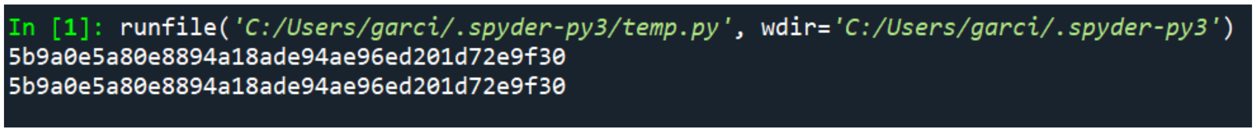
```
In [1]: runfile('C:/Users/garci/.spyder-py3/hashlib.py', wdir='C:/Users/garci/.spyder-py3')
7f9ce92d4cb2c245a3cd83eeb091b23465a5b9e0
7a73747331e7c566aa2c888c6a69d450f343e7de
```

The next part was to convert the image content to bytes. So, I added **PIL import Image** to my python code in order to do this.



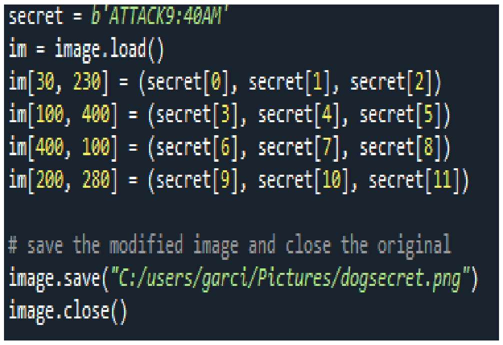
```python
# import the hashlib package
import hashlib
from PIL import Image
# read the files in binary format as they are stored on the disk regardless of
# their format. f1 and f2 are file handlers or pointers to point to files' data

f1 = Image.open("C:/users/garci/Pictures/Lab3/earth.jpg").tobytes()
f2 = Image.open("C:/users/garci/Pictures/Lab3/earth2.jpg").tobytes()
# get the files' data

# compute the SHA1 hash values (digests) and print them in hexadecimal format
print(hashlib.sha1(f1).hexdigest())
print(hashlib.sha1(f2).hexdigest())
```

As you can see the problem when comparing bytes is that you are only comparing the data. Anything outside of that data will not be considered. So, we must compare the entire file to get the information we need. This is why it prints the same hash when we know there are the same files.



```
In [1]: runfile('C:/Users/garci/.spyder-py3/temp.py', wdir='C:/Users/garci/.spyder-py3')
5b9a0e5a80e8894a18ade94ae96ed201d72e9f30
5b9a0e5a80e8894a18ade94ae96ed201d72e9f30
```

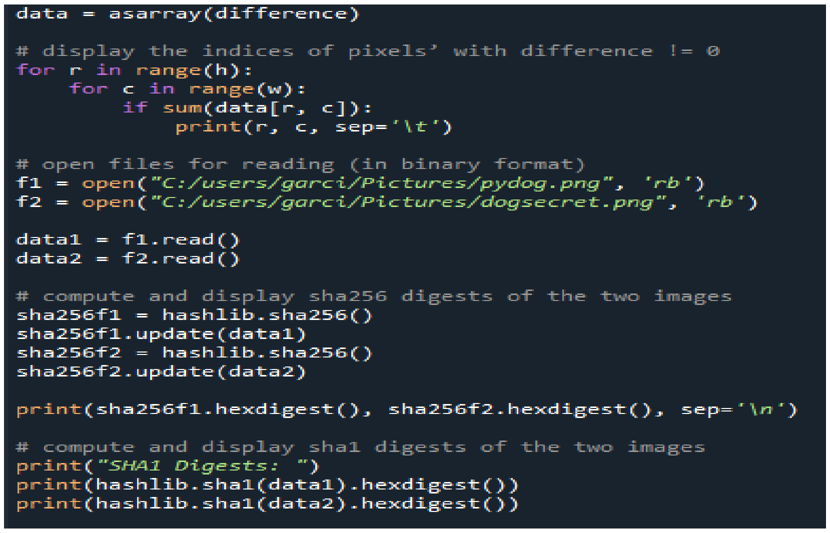
The last part of the lab involves hiding secrets in an image.  I created a new python file called lab3_part6t. I created an image called dog2.png. Using the previous code, I added the from **numpy import asarray** package. I added **w,h = image.size** to get the height and the width of the image.



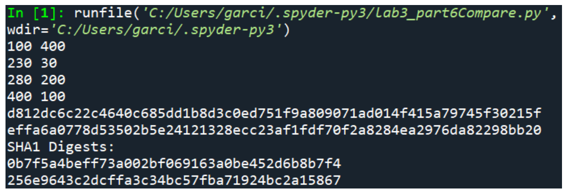
```
secret = b'ATTACK9:40AM'
im = image.load()
im[30, 230] = (secret[0], secret[1], secret[2])
im[100, 400] = (secret[3], secret[4], secret[5])
im[400, 100] = (secret[6], secret[7], secret[8])
im[200, 280] = (secret[9], secret[10], secret[11])

# save the modified image and close the original
image.save("C:/users/garci/Pictures/dogsecret.png")
image.close()
```

I Opened the image using **image = Image.open("C:/users/garci/Pictures/dog2.png")** and I saved the original image using **image.save("C:/users/garci/Pictures/pydog.png")** I also created a secret message, loaded the image to so that the pixel values can be modified, and then save the image using **image.save("C:/users/garci/Pictures/dogsecret.png")**

I opened both files and did not see any noticeable changes to both images. In order to find the differences, you need to convert the difference image into a numpy array to be able to perform the required mathematical operations. The code **difference = ImageChops.difference(dog, dogsecret)** is used to find the difference between the 2 dog images. The for loop will display the indices of pixels that aren't a difference of 0. Then I compute the files using sha256 and print them out.



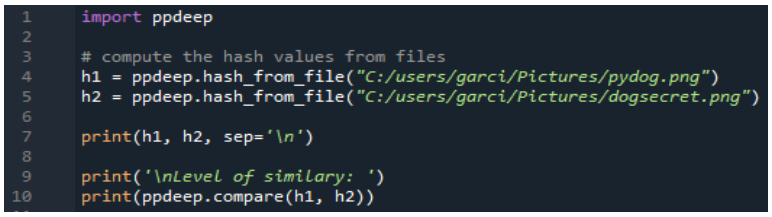
```
data = asarray(difference)

# display the indices of pixels' with difference != 0
for r in range(h):
    for c in range(w):
        if sum(data[r, c]):
            print(r, c, sep='\t')

# open files for reading (in binary format)
f1 = open("C:/users/garci/Pictures/pydog.png", 'rb')
f2 = open("C:/users/garci/Pictures/dogsecret.png", 'rb')

data1 = f1.read()
data2 = f2.read()

# compute and display sha256 digests of the two images
sha256f1 = hashlib.sha256()
sha256f1.update(data1)
sha256f2 = hashlib.sha256()
sha256f2.update(data2)

print(sha256f1.hexdigest(), sha256f2.hexdigest(), sep='\n')

# compute and display sha1 digests of the two images
print("SHA1 Digests: ")
print(hashlib.sha1(data1).hexdigest())
print(hashlib.sha1(data2).hexdigest())
```



```
In [1]: runfile('C:/Users/garci/.spyder-py3/Lab3_part6Compare.py',
wdir='C:/Users/garci/.spyder-py3')
100 400
230 30
280 200
400 100
d812dc6c22c4640c685dd1b8d3c0ed751f9a809071ad014f415a79745f30215f
effa6a0778d53502b5e24121328ecc23af1fdf70f2a8284ea2976da82298bb20
SHA1 Digests:
0b7f5a4beff73a002bf069163a0be452d6b8b7f4
256e9643c2dcffa3c34bc57fba71924bc2a15867
```

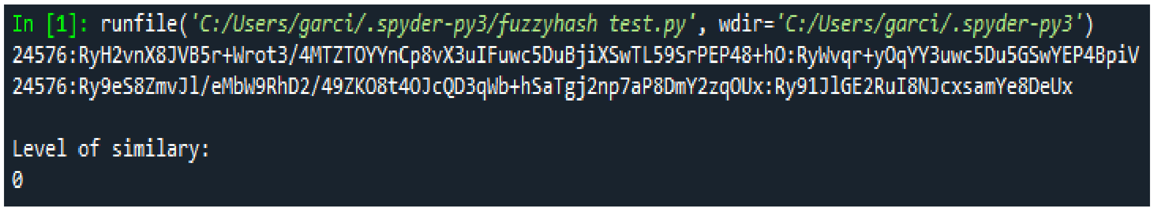
These are hash values for data 1 and 2. Also, the modified pixel values are displayed here.

All that was left is to compute the pydog.png I created earlier and the modified dogsecret.png using the **import ppdeep** package.



```
import ppdeep

# compute the hash values from files
h1 = ppdeep.hash_from_file("C:/users/garci/Pictures/pydog.png")
h2 = ppdeep.hash_from_file("C:/users/garci/Pictures/dogsecret.png")

print(h1, h2, sep='\n')

print('\nLevel of similary: ')
print(ppdeep.compare(h1, h2))
```

In my case, the modified pictures have zero levels of similarity since their hash values did not match.



```
In [1]: runfile('C:/Users/garci/.spyder-py3/fuzzyhash test.py', wdir='C:/Users/garci/.spyder-py3')
24576:RyH2vnX8JVB5r+Wrot3/4MTZTOYYnCp8vX3uIFuwc5DuBjiXSwTL59SrPEP48+hO:RyWvqr+yOqYY3uwc5Du5GSwYEP4BpiV
24576:Ry9eS8ZmvJl/eMbW9RhD2/49ZKO8t4OJcQD3qWb+hSaTgj2np7aP8DmY2zqOUx:Ry91JlGE2RuI8NJcxsamYe8DeUx

Level of similary:
0
```