

## Incorporating Data Representation into your Classroom

1)
2)
3)



The Mars alphabet

Consonants

V	Λ			>	<
p	t			ʈ	ɕ
U	ʌ			ɔ	ɕ
b	d			dʒ	g
ʋ	Δ	ʒ		ʒ	ʒ
f	θ	s		f	x
ɸ	ʌ	ʒ		ɸ	ɕ
v	ð	z		ʒ	Y
ʊ	ʌ			ʌ	ɕ
m	n			ɲ	ɲ
W	M	ʒ		E	X
w	l			j	h

Vowels

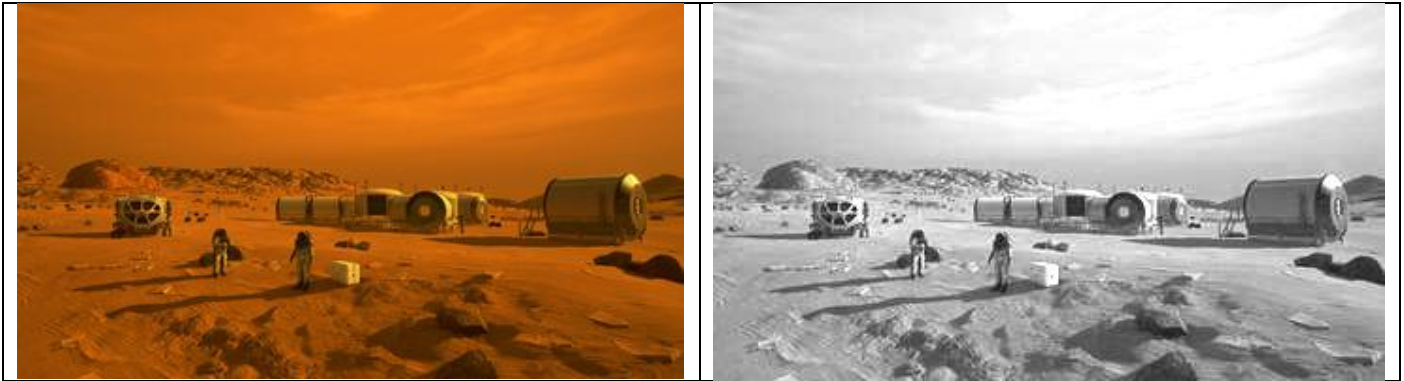
q	P	d	b
i	I	e	ɛ
q	ɸ	ɔ	ɕ
u	ʊ	o	ɔ
ʌ	ʌ	J	L
æ	a	ɔ	ɕ
ɕ	ɕ	ɕ	ɕ
y	Y	ø	œ
ɸP	ɸP	bP	bP
ai	au	eu	ɔi

Numerals and punctuation

.	+	+	+	+	+	+	+	+	+
0	1	2	3	4	5	6	7	8	9

• , ʌ ! > < > <

**Problem 2:** Upon return to earth, you discover that your pictures taken with your phone are a bit too Red. We would like to convert these pictures to grayscale.



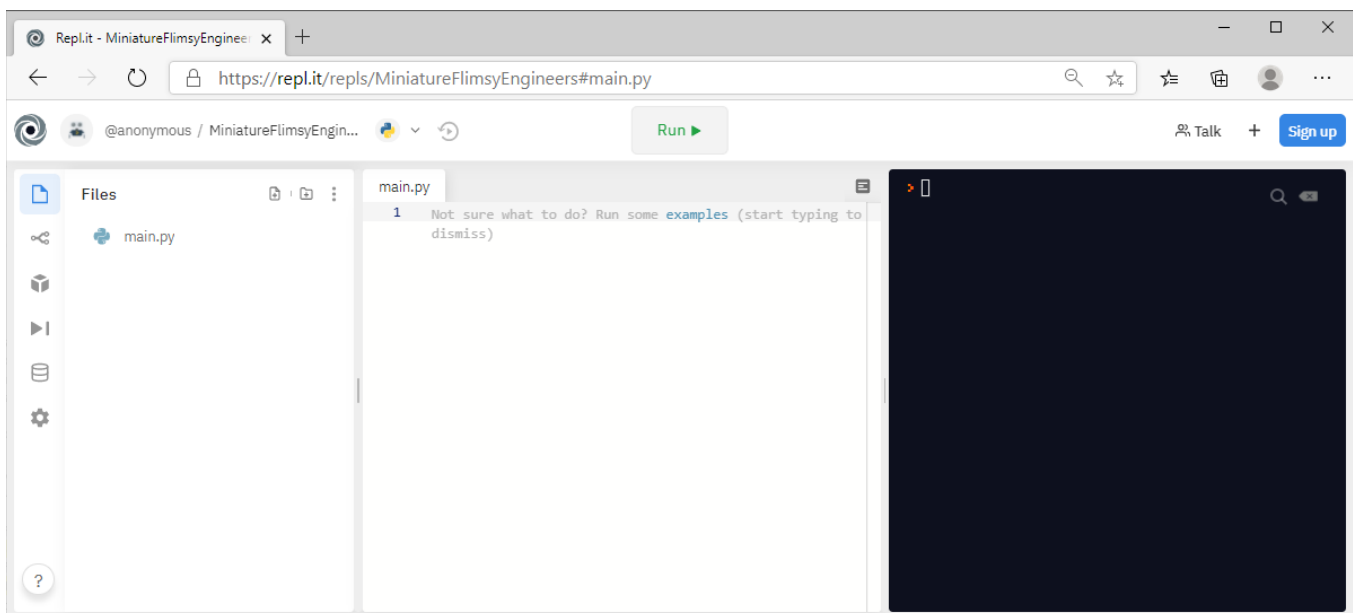
We can do this using a Python program that reads each pixel of the original image and changes the pixel's RGB color value and then writes it back to a new image.

There are different methods for accomplishing grayscale images. One approach is to sum all the colors (Red, Green, and Blue) and then divide them by 3.

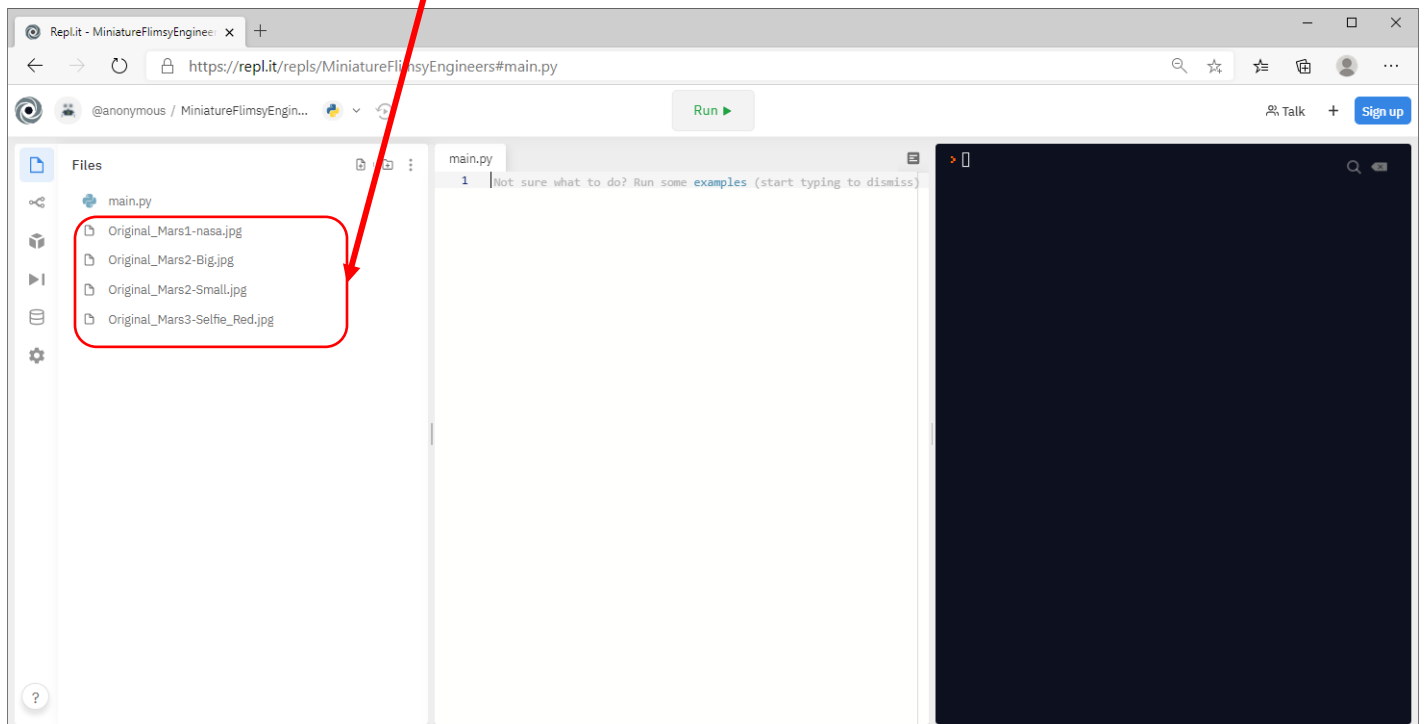
**You can use the following steps to process your images:**

**Step 1:**

1. Point your browser to <https://repl.it>
2. Click on the "Start coding" button.
3. From the list of languages select "Python".
4. Click the "Create repl" button.



**Step 2:** Copy (drag and drop) the following 4 images in the “Files” window of the repl.it site.



In the following steps, we will use these 4 “Original” images to experiment with our filter program.

**Step 3:** Copy the following code and paste it into the editor window. It is typically named "main.py". The code below includes two functions. The first function called "printTime()" will get the local computer time and returns it. The second function is a multi-use filter function. It can take a source image file, apply a filter to it and then create a new destination image file. Currently, the filters include "RED", "GREEN", and "BLUE". In other words, the function can filter the image by extracting and preserving one of those bands of colors. The function also includes a place holder for creating a forth filter called "GRAYSCALE". (This is your job to implement. See more in step 5)

**NOTE:** when copying the code make sure that the indentation of the lines is preserved.

```
from PIL import Image, ImageDraw, ImageFont
from resizeimage import resizeimage
import time

def printTime():
    t = time.localtime()
    current_time = time.strftime("%M:%S", t);
    return current_time

## Convert image by applying a filter
#
def imageFilter(filter, sourceFile, destFile):

    print('Begin applying ', filter, ' filter to image: ', sourceFile)
    print('Start Processing at: ', printTime());

    mySourceImage = Image.open(sourceFile);          # Open the source image file
    myDestImage = Image.new('RGB',mySourceImage.size); # create a new image object for the destination

    width = mySourceImage.size[0];
    height = mySourceImage.size[1];

    # Go through the image line by line and extract each pixel, and
    # manipulate its RGB depending on the filter parameter.

    for x in range(width):
        for y in range(height):
            curPix = mySourceImage.getpixel( (x,y) )

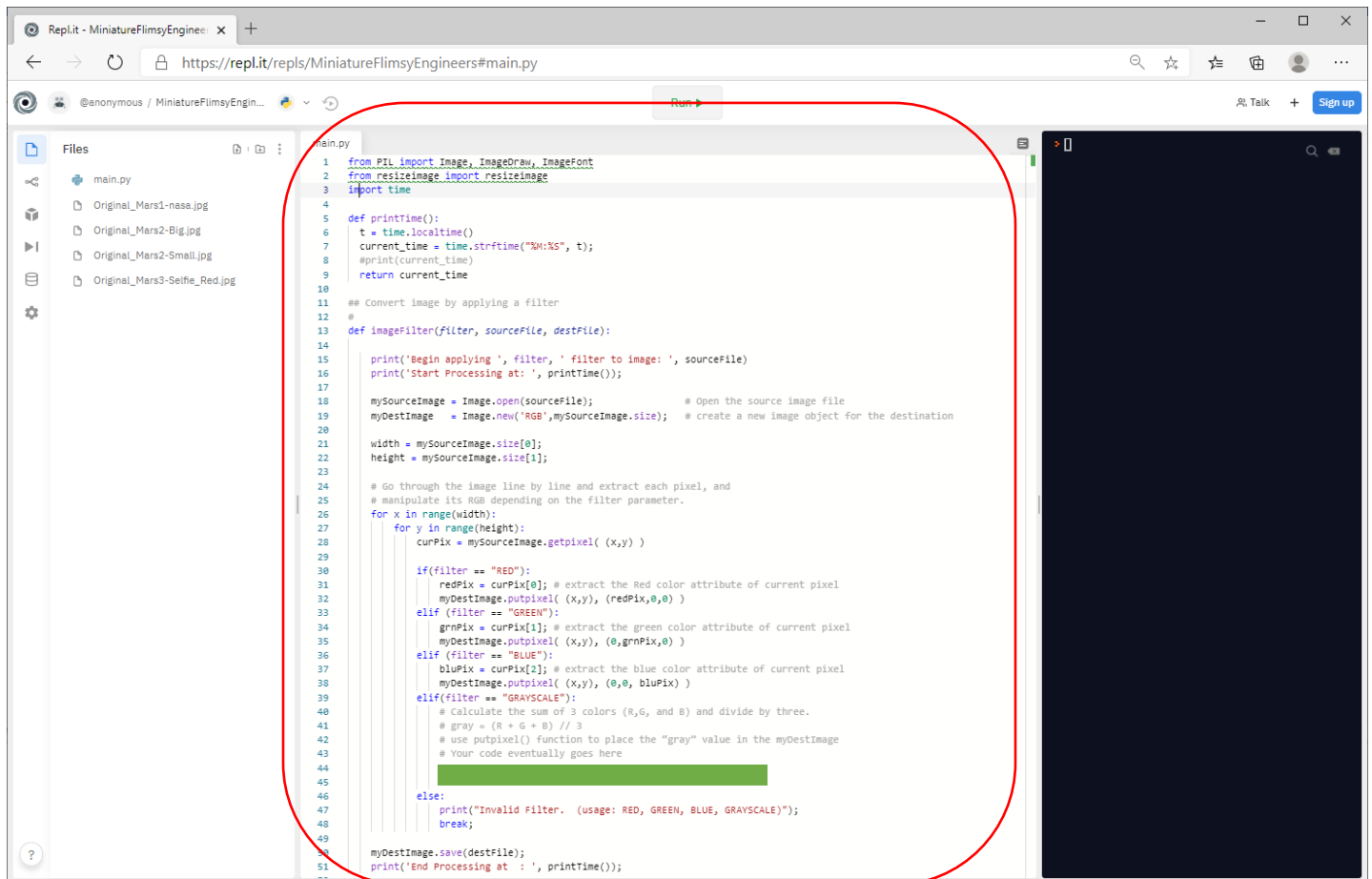
            if(filter == "RED"):
                redPix = curPix[0]; # extract the Red color attribute of current pixel
                myDestImage.putpixel( (x,y), (redPix,0,0) )
            elif (filter == "GREEN"):
                grnPix = curPix[1]; # extract the green color attribute of current pixel
                myDestImage.putpixel( (x,y), (0,grnPix,0) )
            elif (filter == "BLUE"):
                bluPix = curPix[2]; # extract the blue color attribute of current pixel
                myDestImage.putpixel( (x,y), (0,0, bluPix) )
            elif(filter == "GRAYSCALE1"):

                # Calculate the sum of 3 colors (R,G, and B) and divide by three.
                # gray = (R + G + B) // 3
                # use putpixel() function to place the "gray" value in the myDestImage
                # Your code eventually goes here (See step 5)

            else:
                print("Invalid Filter. (usage: RED, GREEN, BLUE, GRAYSCALE)");
                break;

    myDestImage.save(destFile);
    print('End Processing at : ', printTime());
```

Now your screen should look like the image below:



```
1 from PIL import Image, ImageDraw, ImageFont
2 from resizeimage import resizeimage
3 import time
4
5 def printTime():
6     t = time.localtime()
7     current_time = time.strftime("%M:%S", t)
8     print(current_time)
9     return current_time
10
11 ## Convert image by applying a filter
12 #
13 def imageFilter(filter, sourceFile, destFile):
14
15     print('Begin applying ', filter, ' filter to image: ', sourceFile)
16     print('Start Processing at: ', printTime());
17
18     mySourceImage = Image.open(sourceFile);           # Open the source image file
19     myDestImage = Image.new('RGB',mySourceImage.size); # create a new image object for the destination
20
21     width = mySourceImage.size[0];
22     height = mySourceImage.size[1];
23
24     # Go through the image line by line and extract each pixel, and
25     # manipulate its RGB depending on the filter parameter.
26     for x in range(width):
27         for y in range(height):
28             curPix = mySourceImage.getpixel( (x,y) )
29
30             if(filter == "RED"):
31                 redPix = curPix[0]; # extract the Red color attribute of current pixel
32                 myDestImage.putpixel( (x,y), (redPix,0,0) )
33             elif (filter == "GREEN"):
34                 grnPix = curPix[1]; # extract the green color attribute of current pixel
35                 myDestImage.putpixel( (x,y), (0,grnPix,0) )
36             elif (filter == "BLUE"):
37                 bluPix = curPix[2]; # extract the blue color attribute of current pixel
38                 myDestImage.putpixel( (x,y), (0,0,bluPix) )
39             elif(filter == "GRAYSCALE"):
40                 # Calculate the sum of 3 colors (R,G, and B) and divide by three.
41                 # gray = (R + G + B) // 3
42                 # use putpixel() function to place the "gray" value in the myDestImage
43                 # Your code eventually goes here
44
45             else:
46                 print("Invalid Filter. (usage: RED, GREEN, BLUE, GRAYSCALE)");
47                 break;
48
49     myDestImage.save(destFile);
50     print('End Processing at: ', printTime());
```

**Step 4:** Now add the following lines, to the bottom of the program. These lines call the imageFilter() function above, sending it 3 parameters. First is the type of filter we would like to apply, second is the source file and finally the destination file that the function must produce.

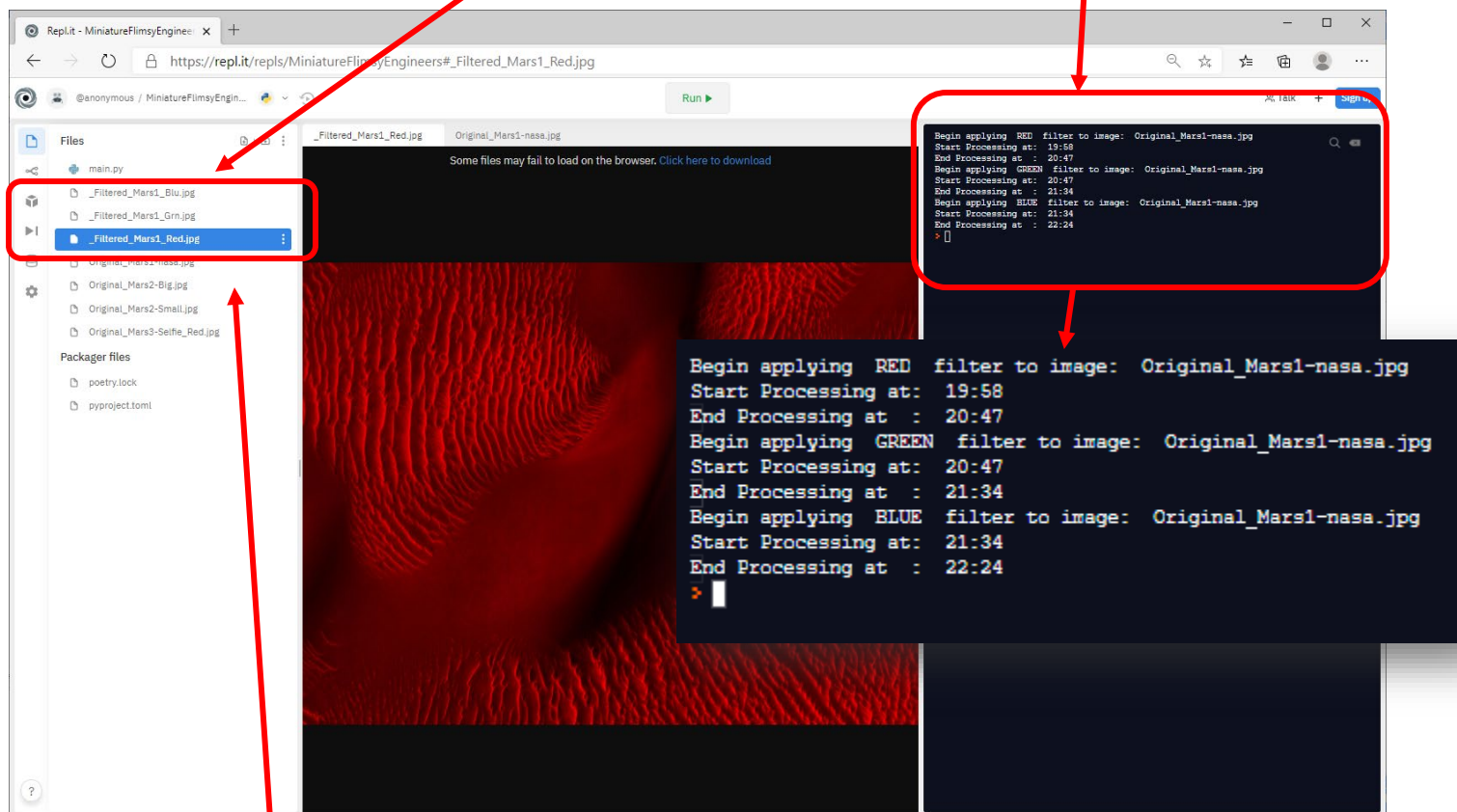
```
##----- MAIN -----
def main():

    #Apply Filter
    imageFilter("RED", "Original_Mars1-nasa.jpg", "_Filtered_Mars1_Red.jpg" );
    imageFilter("GREEN", "Original_Mars1-nasa.jpg", "_Filtered_Mars1_Grn.jpg" );
    imageFilter("BLUE", "Original_Mars1-nasa.jpg", "_Filtered_Mars1_Blu.jpg" );

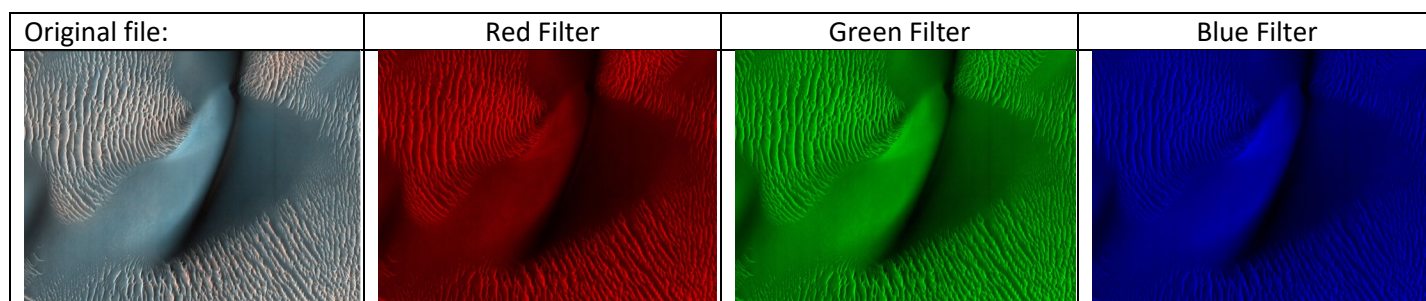
    #imageFilter("GRAYSCALE", "Original_Mars2-Small.jpg", "_Filtered_Mars2_Small_Gry1.jpg" );
    #imageFilter("GRAYSCALE", "Original_Mars2-Big.jpg", "_Filtered_Mars2_Big_Gry1.jpg" );
    #imageFilter("GRAYSCALE", "Original_Mars3-Selfie_Red.jpg", "_Filtered_Mars3_Selfie_Gry1.jpg" );

    # Call the main function
    main();
```

Then Click the "RUN" button and view the results in the execution window (right side of your screen). Also see if there are any new images created. You should see **3 new images**. (Please note that this process may take a while depending on how busy the "Repl.it" server is and the size of your source file. In my case it took **about a minute** to process each filter. See image to the right.)



Click each file on the "**files**" window to see the result.





**Step 5:** Now it is your turn. We now want to create our own GRAYSCALE filter. Refer to the code in step 3 and note the **green** highlighted area, **this is where you want to add your code**.

The steps are as follows:

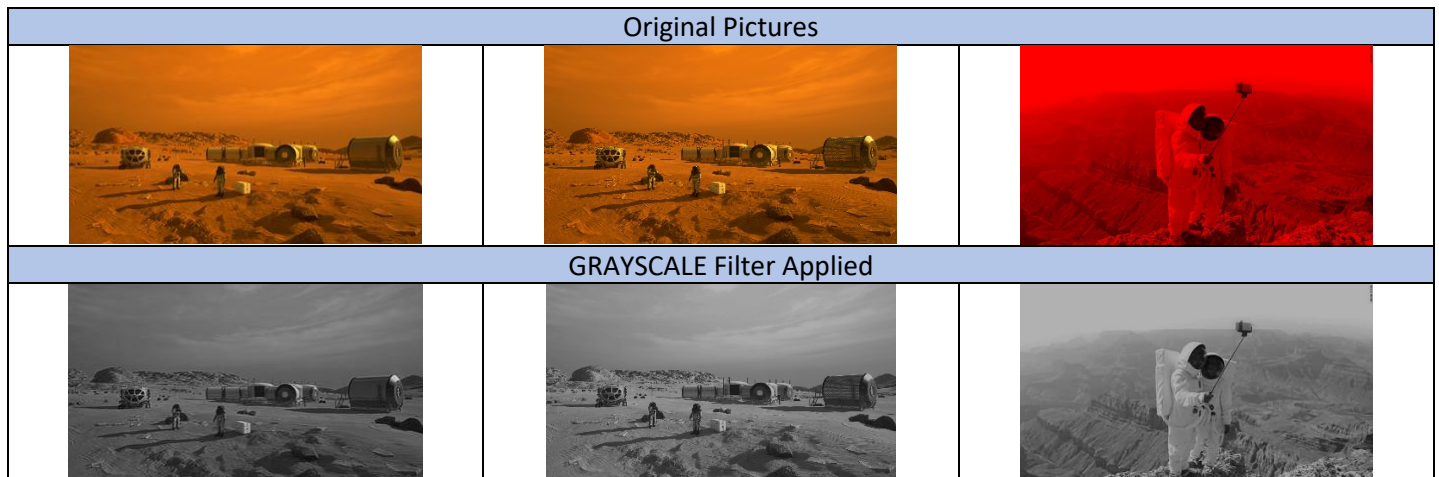
- 1) Create a variable called gryPix and set it to the sum of current pixel's (Red, Green, Blue), divide the result by 3. (use double // instead of /). We need to do this because we need the result of the division to be an integer value. Read about int() function or about Floor Division using this link [https://www.tutorialspoint.com/python/python\\_basic\\_operators.htm](https://www.tutorialspoint.com/python/python_basic_operators.htm)
- 2) Now we want to replace all 3 color bands of the "destination image's" current pixel with a gray value that you calculated above. In other words instead of having (R, G, B) values, we will use (gryPix,gryPix,gryPix)

**That is it, 2 lines should do the job!**

**Step 6:** Once you have modified the code for the GRAYSCALE filter, **uncomment** the first commented imageFilter() line that uses the GRAYSCALE filter and **RUN** the program to test your code. If your code works, you will get a new image called **Filtered\_Mars2\_Small\_Gry1.jpg** on the "Files" window.

```
def main():  
  
    #Apply Filter  
    imageFilter("RED", "Original_Mars1-nasa.jpg", "_Filtered_Mars1_Red.jpg" );  
    imageFilter("GREEN", "Original_Mars1-nasa.jpg", "_Filtered_Mars1_Grn.jpg" );  
    imageFilter("BLUE", "Original_Mars1-nasa.jpg", "_Filtered_Mars1_Blu.jpg" );  
    #imageFilter("GRAYSCALE", "Original_Mars2-Small.jpg", "_Filtered_Mars2_Small_Gry1.jpg" );  
    #imageFilter("GRAYSCALE", "Original_Mars2-Big.jpg", "_Filtered_Mars2_Big_Gry1.jpg" );  
    #imageFilter("GRAYSCALE", "Original_Mars3-Selfie_Red.jpg", "_Filtered_Mars3_Selfie_Gry1.jpg" );  
  
# Call the main function  
main();
```

If your code works, then UNCOMMENT the other two imageFilter() function calls, and note the additional files created.



Once again, note that it takes a while to process some of these images:

```
Begin applying RED filter to image: Original_Mars1-nasa.jpg
Start Processing at: 01:39
End Processing at : 02:27
Begin applying GREEN filter to image: Original_Mars1-nasa.jpg
Start Processing at: 02:27
End Processing at : 03:16
Begin applying BLUE filter to image: Original_Mars1-nasa.jpg
Start Processing at: 03:16
End Processing at : 04:07
Begin applying GRAYSCALE filter to image: Original_Mars2-Small.jpg
Start Processing at: 04:07
End Processing at : 04:07
Begin applying GRAYSCALE filter to image: Original_Mars2-Big.jpg
Start Processing at: 04:07
End Processing at : 04:23
Begin applying GRAYSCALE filter to image: Original_Mars3-Selfie_Red.jpg
Start Processing at: 04:23
End Processing at : 04:31
> 
```

## What to Submit?

For problem 1:

- Three arguments to advocate for using binary vs. ternary number system for representing data.

For problem 2:

- Create a word document, copy and paste the code in the document, followed by copies of the images produced.
- For you hackers, review the last page of this document, try the four options provided for GRAYSCALE filter to determine which may represent an optimization. Feel free to import your own personal images and try the filters.

If you have difficulty completing this exiting project, contact your session facilitator.



## Hacker's Corner (Optional)

### Refining our GRAYSCALE algorithm

Some of you who are familiar with color schemes and may already have some ideas about how to get better grayscale images. If not, try the following options and process the images after every option.

Ideas taken from: # <https://stackoverflow.com/questions/9839013/how-to-convert-an-image-to-gray-scale>

Option 1	<pre># get the sum of 3 colors level = (R + G + B) * 0.70 # Not bad  Rpix = int((curPix[0]+curPix[1]+curPix[2]) * 0.70); Gpix = int((curPix[0]+curPix[1]+curPix[2]) * 0.70); Bpix = int((curPix[0]+curPix[1]+curPix[2]) * 0.70); myDestImage.putpixel( (x,y), (Rpix,Gpix,Bpix) );</pre>
Option 2	<pre># (max(R, G, B) + min(R, G, B)) / 2 # not bad  Rpix = int(max(curPix[0],curPix[1],curPix[2]) - min(curPix[0],curPix[1],curPix[2]) / 2); Gpix = int(max(curPix[0],curPix[1],curPix[2]) - min(curPix[0],curPix[1],curPix[2]) / 2); Bpix = int(max(curPix[0],curPix[1],curPix[2]) - min(curPix[0],curPix[1],curPix[2]) / 2); myDestImage.putpixel( (x,y), (Rpix,Gpix,Bpix) );</pre>
Option 3	<pre># get the <b>Min</b> of the 3 colors and then set all colors to it level = max (R , G , B) # Too Dark, but may work for some images. Try changing the 1.0 to 2.0  Rpix = int(min(curPix[0],curPix[1],curPix[2]) * 1.0); Gpix = int(min(curPix[0],curPix[1],curPix[2]) * 1.0); Bpix = int(min(curPix[0],curPix[1],curPix[2]) * 1.0); myDestImage.putpixel( (x,y), (Rpix,Gpix,Bpix) );</pre>
Option 4	<pre># get the <b>Max</b> of the 3 colors and adjust it by some percentage. # then set all colors to it level = max (R , G , B) # Better  Rpix = int(max(curPix[0],curPix[1],curPix[2]) * 0.70); Gpix = int(max(curPix[0],curPix[1],curPix[2]) * 0.70); Bpix = int(max(curPix[0],curPix[1],curPix[2]) * 0.70); myDestImage.putpixel( (x,y), (Rpix,Gpix,Bpix) );</pre>