MSDS 501 - Homework 3

Requirement - [0.2 pt]

- **Please do not hardcode any variables including** *`file_name, r_ratio, g_ratio, b_ratio, pixel, red, green,`* and *`blue`***.**
- **Please do not add any additional libraries or packages.**
- Make sure that everything passes when you run `$pytest`.

Data Overview

In digital imaging, grayscale images are represented as 2-dimensional arrays, where each pixel is a single number representing a color that varies from white to black.
Color images, on the other hand, are stored as 3-dimensional arrays. Each pixel is represented by a triplet of integers corresponding to the red, green, and blue (RGB) channels — each ranging from 0 to 255. **In this homework, we will create and manipulate digital color images from numbers.**

`file_name` includes

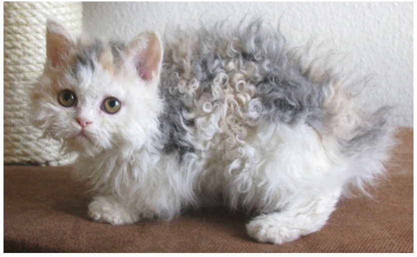
- First line: width of the image
- Second line: height of the image
- Remaining lines: pixel data, where each pixel is a triplet `[r, g, b]`
    - *r,g,* and *b* values are between 0 and 255.

For example, the following input indicates that the image can be represented as [[[1,2,3],[4,5,6]],[[7,8,9],[10,11,12]]].

```
2
2
1,2,3
4,5,6
7,8,9
10,11,12
```

Question

1. Create a function called *`create_image_array()`* which takes *`file_name`* as an input variable and returns a list with the given *width* and *height*. [0.3 pt]
   `create_image(create_image_array(file_name))` returns the following image.



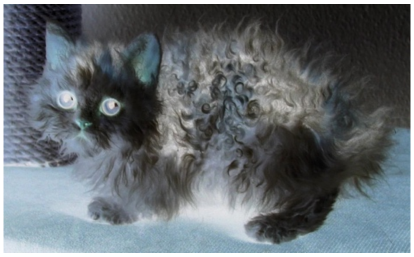
2. Create *`xray_filter()`* that takes a list and returns a new list. This new list includes updated r,g,b values that `r_value = 255 – r_value`, `g_value = 255 – g_value`, and `b_value = 255 – b_value`. [0.5pt]
   Ex.

```
numbers = [[[1,2,3],[5,6,7],[9,10,11]],
           [[11,12,13],[15,16,17],[19,20,21]]]

xray_filter(numbers) returns

[[[254, 253, 252], [250, 249, 248], [246, 245, 244]],
 [[244, 243, 242], [240, 239, 238], [236, 235, 234]]]
```

`create_image(xray_filter(create_image_array(file_name)))` returns the following image.



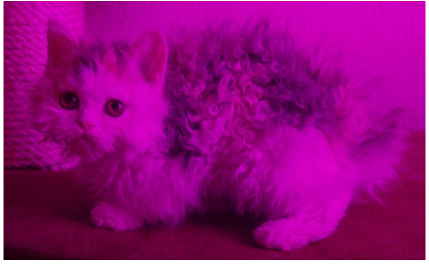
3. Create a function called *adjust_r_g_b*() that takes the image array and three float values that are multiplied to r,g, and b values accordingly. The resulting value should be rounded to the nearest integer. [0.5pt]
   Ex.

```
numbers = [[[1,2,3],[5,6,7],[9,10,11]],
           [[11,12,13],[15,16,17],[19,20,21]]]

adjust_r_g_b(numbers, 1, 0.9, 0.8) returns

[[[1, 2, 2], [5, 5, 6], [9, 9, 9]],
 [[11, 11, 10], [15, 14, 14], [19, 18, 17]]]
```

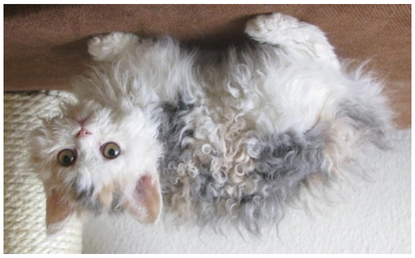
   create_image(adjust_r_g_b(create_image_array(file_name), r_ratio, g_ratio, b_ratio)) returns the following image.



4. Create a function called *upside_down()* that takes a list and reverses the list to flip the image. [0.5pt]
   Ex.

```
numbers = [[[1,2,3],[5,6,7],[9,10,11]],
           [[11,12,13],[15,16,17],[19,20,21]]]
upside_down(numbers) returns
[[[11, 12, 13], [15, 16, 17], [19, 20, 21]],
 [[1, 2, 3], [5, 6, 7], [9, 10, 11]]]
```

   create_image(upside_down(create_image_array(file_name))) returns the following image.



5. Create a *function called* *`vertical_flip()`* that takes a list and returns a list where values in each row are vertically flipped. (i.e., reverses the pixel order in each row)[0.5pt] Ex.

```
numbers = [[[1,2,3],[5,6,7],[9,10,11]],
           [[11,12,13],[15,16,17],[19,20,21]]]

vertical_flip(numbers) returns

[[[9, 10, 11], [5, 6, 7], [1, 2, 3]],
 [[19, 20, 21], [15, 16, 17], [11, 12, 13]]]
```

create_image(vertical_flip(create_image_array(file_name))) returns the following image.

6. Create a function called *`create_border()`* which adds a border around the image with given red, green, blue and pixel values [0.5 pt]. Input parameters are given as **arbitrary keyword arguments** including *`numbers, red, green, blue`* and *`pixel`*.
   a. *`numbers`* : A list of pixel values of the input image created by create_image_array().
   b. *`red, green, blue`* : r, g, b values for the color of the border.
   c. *`Pixel`* : the number of pixels indicating how many pixels of [*`red, green, blue`*] value should be added at the beginning and end of each row. In addition, the returned list should have the *`pixel`* number of rows only consists with the given *`red, green`*, and *`blue`* at the beginning and end of *`numbers`*. - In summary, 1) add pixel pixels at the beginning and end of each row. 2) Add pixel rows to the top and bottom, each filled with [red, green, blue].

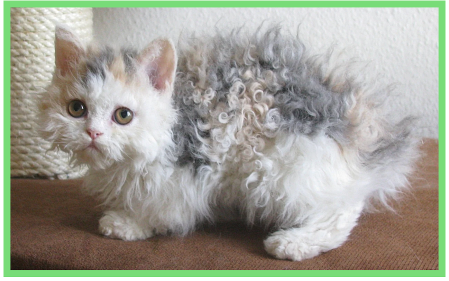
```
numbers = [[[1,2,3],[5,6,7],[9,10,11]],[[11,12,13],[15,16,17],
[19,20,21]]]
r = 0
g = 0
b = 0
pixel = 2


create_border(numbers=numbers, red=r, green=g, blue=b, pixel=pixel)
```

returns

```
[[[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0]],
 [[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0]],
 [[0, 0, 0], [0, 0, 0], [1, 2, 3], [5, 6, 7], [9, 10, 11], [0, 0, 0], [0, 0, 0]],
 [[0, 0, 0], [0, 0, 0], [11, 12, 13], [15, 16, 17], [19, 20, 21],
  [0, 0, 0], [0, 0, 0]],
 [[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0]],
 [[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0]]]
```

create_image(create_border(numbers = create_image_array(file_name), red=r, green=g, blue=b, pixel=pixel)) returns the following image.



Submit the hw3.py file (**ONLY**) on Canvas - the name of your file should be **hw3.py** .