CISC-121 Winter 2013

Weeks 8 and 9 Lab Assignment: Image Manipulation

This is the assignment for the labs in Weeks 8 and 9 (March 4 to 15). **This assignment will be graded and will be worth 5% of your final grade. See below for details of how to submit your solution.**

Deadline:

Please submit your solution by 8:00 AM Monday March 18. I will give you a 24-hour grace period that ends at 8:00 AM Tuesday March 19. Assignments submitted during the grace period will be penalized 10% for lateness. Assignments will not be accepted after the end of the grace period.

Introduction:

The importance of image manipulation software can hardly be overstated. In this assignment you will develop a Python program that allows the user to perform some simple operations on images. This assignment requires the **pypng** library. If you were not able to install this library on your own computer, you should use the computers in the lab.

Starting Point:

You are provided with a stub of a program. The provided program contains a function called get_image(fn), which takes as its parameter the name of a file (fn) which is expected to contain an image in PNG format, and returns an integer-based representation of the image.

The PNG image format represents each pixel of the image by three integers – a red value, a green value, and a blue value, each in the range [0 ... 255]. A pixel with representation [0,0,0] is black, while[255,255,255] represents a pixel which displays as white. A pure red pixel is represented by [255,0,0], etc. Every combination of red, green and blue values gives a different colour.

We represent each row of pixels in an image as a list of integers, three integers for each pixel. The rows are then combined as elements in a nested list that represents the entire image.

As an example, consider this Python fragment

```
small_image = [  [255,0,0,0,255,0],          # list representing first row
                 [0,0,255,0,0,0]              # list representing second row
```

```
]
```

This represents an image with two rows of pixels, with two pixels in each row. The first row consists of a red pixel and a green pixel, and the second row consists of a blue pixel and a black pixel. Note that there is no separation between the values that represent consecutive pixels in a row.

We can access individual pixels using slices. For example

```
apixel = small_image[0][0:3]
print apixel
```

produces output

```
[255,0,0]
```

and

```
small_image[1][3:6] = [90,20,120]
```

assigns a blended colour to the bottom right pixel.

Accessing each pixel in an image in sequence is straightforward using nested for loops and specifying a **step value** of 3 for the loop that works across each row.

For example, suppose that my_image is a nested list that represents a PNG image as explained above. Then

```
rows = len(my_image)
cols = len(my_image[0])
for row in range(rows):
    for col in range(0,cols,3):
        pixel = my_image[row][col:col+3]
        pixel[0], pixel[2] = pixel[2], pixel[0]
        my_image[row][col:col+3] = pixel
```

exchanges the red and blue values in each pixel. (Yes, this can be done with fewer lines. This example is meant to be illustrative only.)

Converting the integer-based representation to an image file is handled by the **save_image(list,file)** function. Look at the provided program to see how this is used.

The provided program uses the **listdir()** function, imported from the **os** library, to get a listing of the files in the current directory. It eliminates all but the PNG files, and uses the **choicebox** function from **easygui** to let the user select an image to work on. If you decide not to use easygui, you need to make sure that the user cannot accidentally choose a non-existent image file.

Requirements:

The program as provided allows the user to load an image, convert it to its negative, and save it to a file.

PART A (Simple Operations)

1. Study this program and make sure you understand how it works.

2. Add a function that produces a left-to-right reflection of the image.

3. Add a function that produces a top-to-bottom reflection of the image.

4. Add a function that rotates the image 90 degrees to the left. That is, the top row must become the left-most column of the resulting image.

5. Add your functions to the provided operation menu in the program. If you are not using easygui, provide equivalent functionality.

PART B (Advanced Operations)

6. Add a function that converts the image to grey-scale. A grey pixel is produced by assigning equal values to the red, green and blue integers that represent the pixel. Choosing the shade of grey for a coloured pixel can be done by taking the average of the original red, green and blue values for the pixel.

7. Add a function that blends one image left-to-right into another: in each row, the first pixel is purely from the first image, the final pixel is purely from the second image, and each pixel in between is a combination of the two in a smoothly graduated fashion.

   As an example of this, suppose the images have 10 pixels in each row. We can treat the colours independently, so consider just the red values for now. In the first pixel of the blended row, the red value should be
   (9*red from first image + 0*red from second image)/9
   The second pixel's red value should be
   (8*red from first image + 1*red from second image)/9
   The third pixel's red value should be
   (7*red from first image + 2*red from second image)/9
   ...
   The ninth pixel's red value should be
   (1*red from first image + 8*red from second image)/9
   and the final pixel's red value should be
   (0*red from first image + 9*red from second image)/9
   Applying the same technique to the green and blue values produces the blended row.

8. Add a function that replaces the background in the image. This requires the use of another image that duplicates the original image with the foreground elements removed, and a third image that becomes the new background. The algorithm is straightforward:
   1. Allow the user to specify the original image, the image that contains the duplicate of the old background, and the image to be used as the new background.
   2. Create the composite image by comparing each pixel of the original image to the corresponding pixel in the "background-only" copy. If the pixels from the two images are different, treat the pixel as part of the foreground and copy the pixel from the original image to the composite image. However, if the pixels from the two images are the same, treat the pixel as part of the background and copy the pixel from the new background image to the composite image. Note that exact matches are very unlikely, so you need to establish a "close" measure for colours.

9. Add your new methods to the user menu in your program.

Sample Images:

I have provided a few images that you may use for developing and testing your functions.

Feel free to use your own images if you wish. You may find the free program **irfanview** useful – it allows conversion between different formats, and easy resizing of images.

Dealing with Errors:

Your program and the functions you write should be robust. You can assume that any file that ends in ".png" does contain a valid PNG image. Whenever you are implementing a function that deals with two or more images, you should make sure the images have the same size. If you are allowing the user to enter a file name, you must make sure the file exists (note that the **listdir()** and **choicebox()** approach takes care of this).

Marking Scheme:

This assignment will be marked out of 100:

1. PART A : 40
2. PART B : 30
3. Error Handling : 15
4. Style : 15

Documentation:

1. At the beginning of your program, identify yourself, and your working-partner (if any)
2. Identify the purpose of each function
3. Add comments where ever you feel they will help the reader understand your algorithm.

Uploading Your Solution:

Upload your solution to Moodle in the usual way.

Academic Integrity:

You must observe the Queen's Academic Integrity Policy in this assignment. All of the work you submit must be your own work (or the work of you and your partner, if you work in a pair). You may **not** copy from **any source –** the Internet, a book, or any other source. Even if you cite your source, this is not acceptable: you should be learning how to write your own code. The TA's and I will give advice but we will not write your code for you. You may discuss the assignment with other students but you must not look at or exchange code or even pseudo-code with each other (except your working-partner, if any). If we discover two solutions that are too similar to be a coincidence, you will be charged with a departure from academic integrity. Queen's has a zero-tolerance policy in such cases.