CptS 111 Introduction to Algorithmic Problem Solving (http://piazza.com/wsu/fall2016/cpts111/home)

Washington State University (https://wsu.edu)

Gina Sprint (http://eecs.wsu.edu/~gsprint/)

PA6 Strings (100 pts)

Due Wednesday, November 16 at Midnight.

Learner Objectives

At the conclusion of this programming assignment, participants should be able to:

- Iterate through characters in a string
- Perform string indexing
- Concatenate strings

Prerequisites

Before starting this programming assignment, participants should be able to:

- Apply loops
- Read and write from/to files

Acknowledgments

Content used in this assignment is based upon information in the following sources:

- Adam Carter's (http://www2.humboldt.edu/computerscience/carter.html) CptS 111 labs
- Stanford's Image Editor Assignment (http://nifty.stanford.edu/2012/guerin-image-editor/image_editor.html)

Overview and Requirements

Write a program to modify images (image_modifier.py). The image format we are going to use is PPM (http://netpbm.sourceforge.net/doc/ppm.html). Don't worry if you are not familiar with PPM images. This document will tell you everything you need to know!

PPM Image Format

The PPM image format is encoded in human-readable plain text. A PPM image has two main parts:

1. Header
2. Body

PPM Header

The header is always the top three uncommented lines in the file:

```
P3
4 4
255
```

The first line specifies the image encoding that is contained within the file. We will always use the "P3" specification. The second line specifies the number of pixel columns and rows present in the image. In this example, we have a 4 pixel by 4 pixel image. The final number indicates the maximum value for each red, green, and blue (RGB) element in the picture. We will always use a max value of 255.

PPM Body

Below the header is the body of the PPM file. Each pixel has a red, green, and blue value. For example, the body content of our 4x4 image might be:

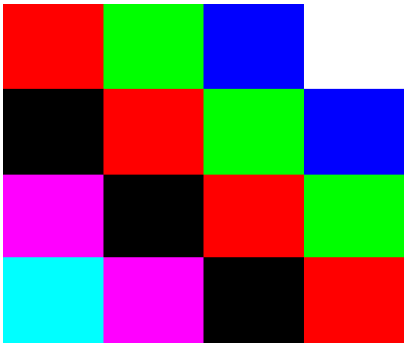
```
255 0   0       0   255 0       0   0   255     255 255 255

0   0   0       255 0   0       0   255 0       0   0   255

255 0   255     0   0   0       255 0   0       0   255 0

0   255 255     255 0   255     0   0   0       255 0   0
```

With these values, the pixel in the first column and first row has a RGB value of (255, 0, 0), which equates to red. The pixel in the last column and the first row has a RGB value of (255, 255, 255), which equates to white. A blown-up image containing these values would look like:



Note: You can explore RGB color values at this website (http://www.colorpicker.com/).

Example PPM Image

Download this PPM image of the NY sky line: ny.ppm (https://raw.githubusercontent.com/gsprint23/cpts111/master/progassignments/files/ny.ppm). You can open the image with software such as Ifranview (http://www.irfanview.com/) to view it as an image. You can also open it as a text file (in Spyder IDE or a text editor) to view it as plain text (useful for debugging your code!).

(Image from https://pixabay.com/en/new-york-skyline-manhattan-hudson-540807/ (https://pixabay.com/en/new-york-skyline-manhattan-hudson-540807/))

Program Details

Write a program to modify a PPM image in two ways:

1. Negate the colors
2. Apply a high contrast
3. Apply a gray scale
4. Remove a primary color (red, green, blue)

To do this, the flow of your program should be as follows:

1. Prompt the user to enter the input PPM file name, the output PPM file name, and one of the following commands to determine what modification to apply the input PPM image:
   A. "negate"
   B. "high contrast"
   C. "gray scale"
   D. "remove `<color>`" where `<color>` is "red", "green", or "blue"
2. Apply the modification. Your solution must contain the functions below to perform the image modification. These functions will help you get started! You are encouraged to define more than the ones listed as you wish.
   A. `process_header(infile, outfile)`: Writes the first three lines of the input image to the output image (no need to modify these lines)
   B. `process_body(infile, outfile, modification)`: Walks through each line in the body of the input image, modifies the RGB values according to `modification`, and writes out the modified output image. For the `modification`, `process_body()` calls one of the following functions for each RGB triple in the line:
      a. `negate()`: Accepts a *single* integer RGB value and the output file object. To negate the value, subtract 255 and take the absolute value of the result. Writes the result to the output file.
      b. `high_contrast()`: Accepts a *single* integer RGB value and the output file object. To apply high contrast, if the value is greater than 127, set it to 255, otherwise set it to 0. Writes the result to the output file.
      c. `gray_scale()`: Accepts *three* integer RGB values and the output file object. Inspect all three red, green, and blue values for each pixel. For each RGB triple, convert the values to the triplet's average. Writes the result to the output file.
   C. `remove_color()`: Accepts *three* integer RGB values, a string representing the color to remove, and the output file object. For example, suppose the color to remove is green. This function sets all green values to 0 (green is the 2nd value in an RGB triple). Writes the result to the output file.
3. Write the modified image to the user-specified output file.

Negate line example: `1 2 3 200 100 150` negated is `254 253 252 55 155 105`

High contrast line example: `1 2 3 200 100 150` in high contrast is `0 0 0 255 255 255`

Gray scale line example: `1 2 3 200 100 150` as gray scale is `2 2 2 150 150 150`

Remove green line example: `1 2 3 200 100 150` without green is `1 0 3 200 0 150`

Note: Other than `strip()`, **do not use any other string methods** in your code.

Note: I recommend you **add code to prompt the user last**. You can debug faster if you hard code the file names and the command, rather than typing them in each time as you implement your code to perform the image modifications.

Example Runs

Here is an example run of the program:

```
Please enter the input file name: brelsford.ppm
Please enter the output file name: brelsford_negate.ppm
Modification commands include:
"negate"
"high contrast"
"gray scale"
"remove red"
"remove green"
"remove blue"
Please enter the modification to perform: negate
Image modification "negate" complete. Closing files.
```

Here are the image files to compare with:

Original

(ny.ppm) (https://raw.githubusercontent.com/gsprint23/cpts111/master/progassignments/files/ny.ppm):

Negate

(ny_negate.ppm)
(https://raw.githubusercontent.com/gsprint23/cpts111/master/progassignments/files/ny_negate.ppm):

High Contrast

(ny_high_contrast.ppm)
(https://raw.githubusercontent.com/gsprint23/cpts111/master/progassignments/files/ny_high_contrast.ppm):

Gray Scale

(ny_gray_scale.ppm)
(https://raw.githubusercontent.com/gsprint23/cpts111/master/progassignments/files/ny_gray_scale.ppm):

Remove Red

(ny_remove_red.ppm)
(https://raw.githubusercontent.com/gsprint23/cpts111/master/progassignments/files/ny_remove_red.ppm):

Remove Green

(ny_remove_green.ppm)
(https://raw.githubusercontent.com/gsprint23/cpts111/master/progassignments/files/ny_remove_green.ppm):

Remove Blue

(ny_remove_blue.ppm)
(https://raw.githubusercontent.com/gsprint23/cpts111/master/progassignments/files/ny_remove_blue.ppm):

Bonus (5 pts)

Validate the input file specified by the user is valid, i.e. the input file adheres to the following requirements:

1. The file exists in the current working directory
2. The file has a .ppm extension

If the user specifies an invalid input file, tell the user why the file is invalid, and *re-prompt until a valid file is specified by the user.*

Hint: To check if a file exists, import the `os` module and call the predicate function `os.path.isfile(<string file_name>)`(https://docs.python.org/3/library/os.path.html#os.path.isfile):

In [8]:
```
import os.path

print(os.path.isfile("ny.ppm"))
print(os.path.isfile("file_that_doesnt_exist.ppm"))
```

```
True
False
```

Submitting Assignments

1. Use the Blackboard tool https://learn.wsu.edu (https://learn.wsu.edu) to submit your assignment to your TA. You will submit your code to the corresponding programming assignment under the "Content" tab. You must upload your solutions as `<your last name>_pa6.zip` by the due date and time.
2. Your .zip file should contain your .py file and your .ppm files you used to test your program (originals, negatives, and high contrasts).

Grading Guidelines

This assignment is worth 100 points + 5 points bonus. Your assignment will be evaluated based on a successful compilation and adherence to the program requirements. We will grade according to the following criteria:

- 10 pts for prompting and receiving input from the user
- 10 pts for correct `process_header()`
- 35 pts for correct `process_body()` (and other helper functions you may choose to define)
- 10 pts for correct `negate()`
- 10 pts for correct `high_contrast()`
- 10 pts for correct `gray_scale()`
- 10 pts for correct `remove_color()`
- 5 pts for adherence to proper programming style and comments established for the class