A pdf document (-5 point if missing, -2 points if present but in another format like .docx, .pages, .rtf, .txt, .py). The document should contain:

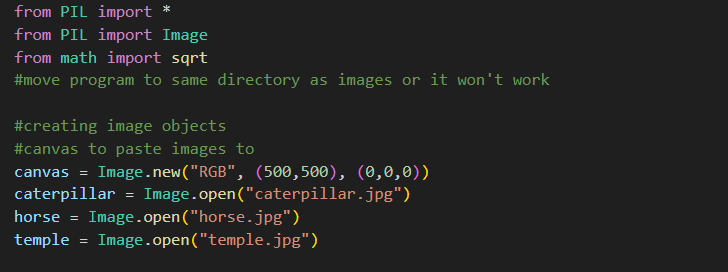
1) Discussion of what you are doing and itemizing the three advanced manipulations you did. (-2 points if missing)

2) the input image(s) (-2 points if missing)

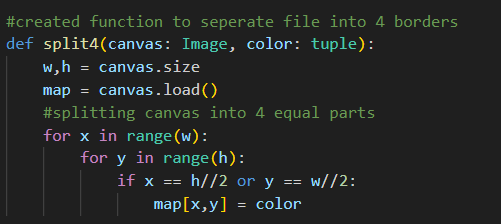
3) snippets of your code with English text explaining what you are doing (-2 points if missing)

4) the output image(s) (-2 points if missing)

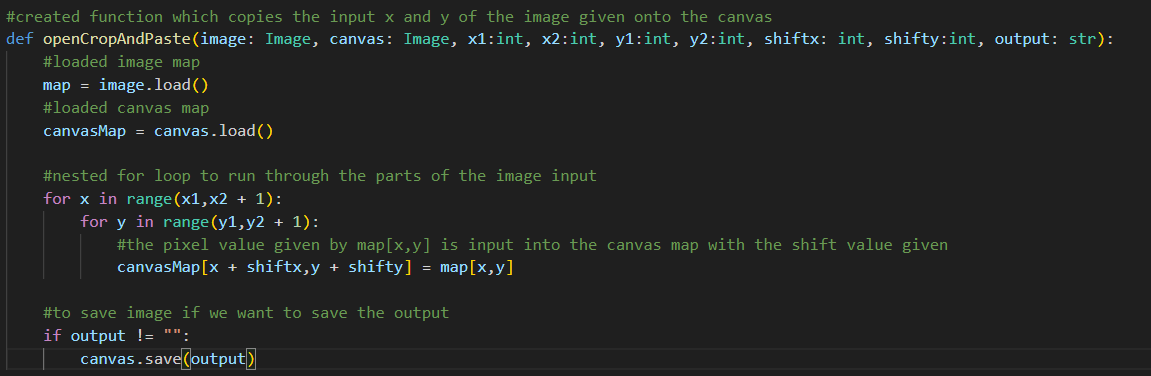
My program uses some basic techniques we previously used along with the more advanced techniques we learned recently, which are mentioned in the assignment rubric. My assignment specifically has image cropping, background manipulation, horizontal and vertical reflection, grey scaling and blurring of images. I used the following images provided in the course content section. The temple, the horse and the caterpillar. My program also produces two outputs. It shows the final image to the user and it saves the image as “output.jpg” in the same directory as the program.   
  
Please move the files into the same directory as the program otherwise the program will not work.



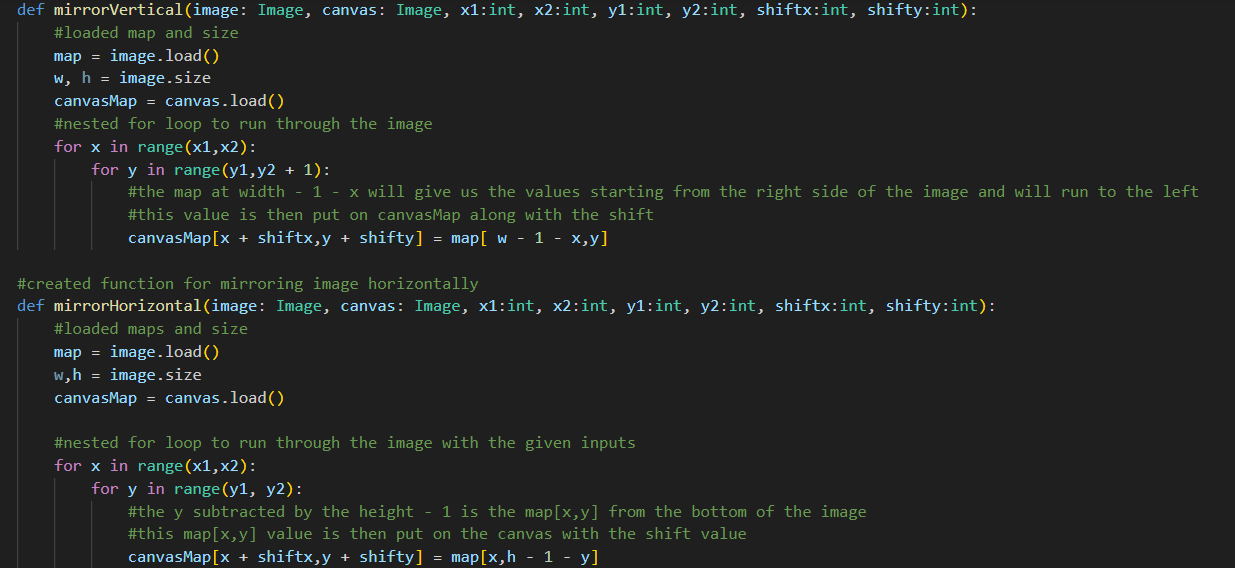
First and foremost, I start importing all the possible modules I use in the program. I create an empty canvas which is my input image so to speak and i open the other images used in this program. My canvas has a size of 500x500 pixels and it is set to black.



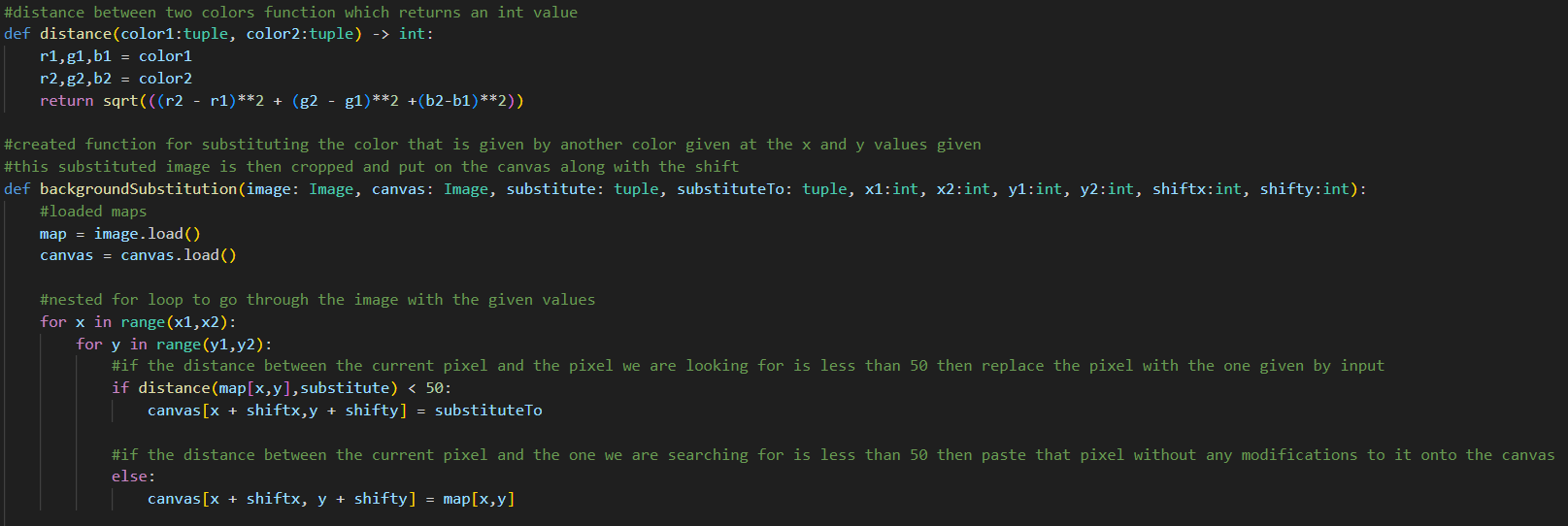
The split function splits the canvas into 4 sections evenly spaced by checking if the x and y pixels are half the height or width. This function just makes it more organized for when i use the functions in the later portion of the program.



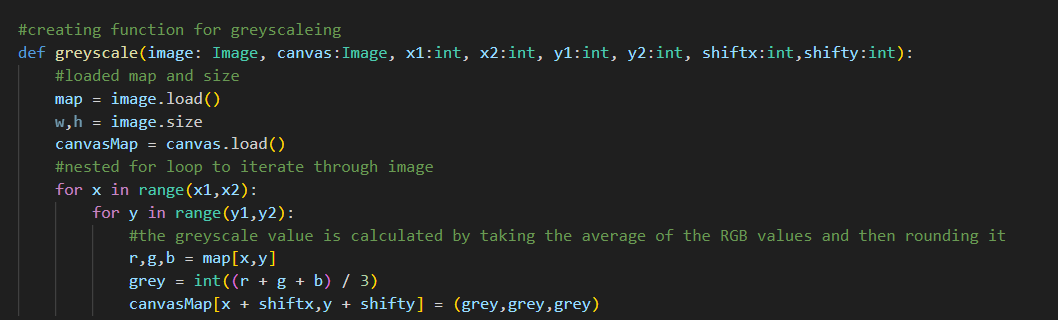
The openCropAndPaste function takes an input image and places it on the canvas given. It crops the portion of the image specified by the user and pastes it into the canvas and shifting the image so that the image is where the user wants it to be. It also has an added functionality of saving the cropped image and the saved file is named what the user wants.



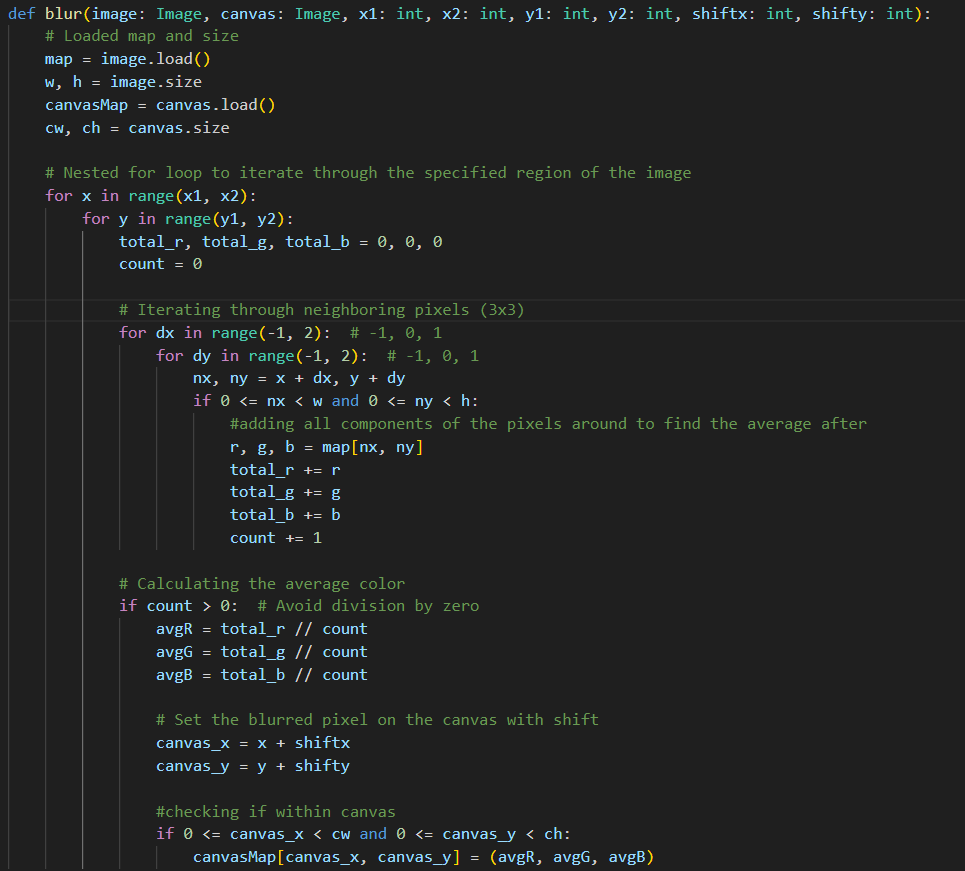
The mirror vertical and horizontal functions mirror the image given onto the canvas. The mirrorVertical function takes the image and flips it vertically by using the width as a reference and subtracting the x value by the width and 1. This pixel is then placed onto the canvas and shifted. The mirrorHorizontal image does the same thing as the mirrorVertical function but flips the image horizontally so it takes the height of the image and subtracts the y value of the pixel by the width -1 and again putting this pixel at the after shift on the canvas.



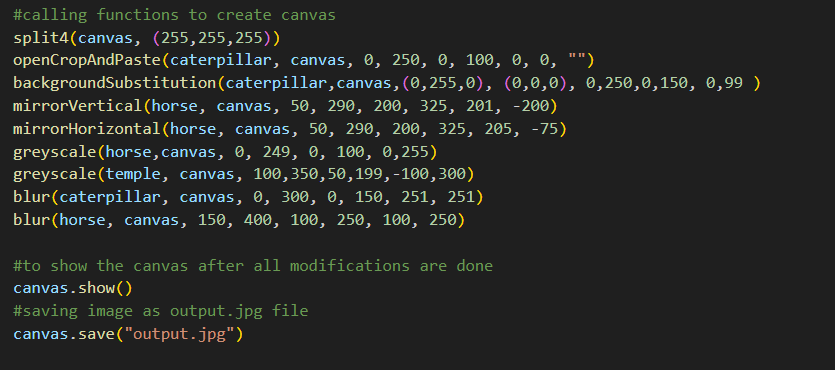
The distance function was taught to us previously, it takes in two tuple values and returns an integer which is the distance between the two colors. Utilizing the distance function, I created the background substitution function which takes in the image, canvas, the pixel we are looking for, the pixel to substitute to, and the coordinate values along with the shift values. The backgroundSubstution image goes through the image given and checks if the distance between the color at that pixel and the color we are looking for is less than 50, if it is then it will change that pixel to the substituted pixel. If the distance is more than 50 then it ignores the pixel. The pixels are then shifted and placed onto the canvas.

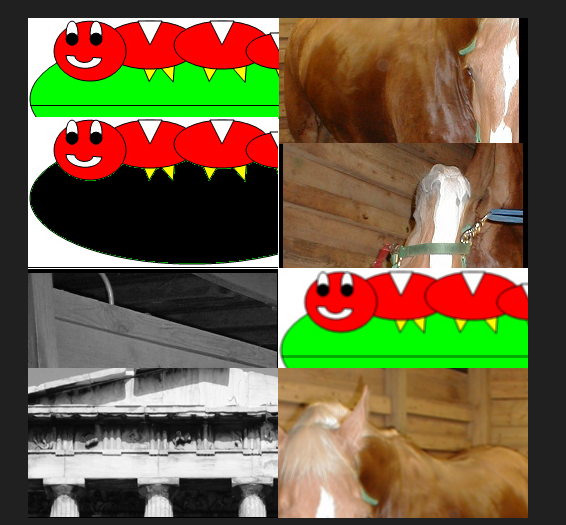


The greyscale function takes an image, canvas and from where to where the user wants to greyscale and put into the canvas along with the shift. The greyscale function works by taking the the average of the red, green and blue of each pixel and setting each pixel to that value. After it does this, the pixel is then placed onto the canvas.



The blur image takes an image, canvas and pixel values. The blue function goes through every pixel and takes the total r,g,b values of the surrounding pixels and divides the total red, green and blue pixels by the number of pixels you counted. We make sure that the count is not zero and the pixel is not at the corner of the image. I kept running into a division by 0 error and so I did some research and came up with this solution. The total average red,blue and green is then taken and put into a pixel that exists within the canvas. Essentially, we take the surrounding pixels of the current pixel[x,y], compact it and place it onto the canvas.

The final portion of my program calls all the functions and displays the final product to the user as well as saving the image in the “output.jpg” file. Final output below.



This is my output image in which you can see that I use image cropping and background substitution in the first quadrant of my file. The second quadrant is vertical and horizontal reflections. The third quadrant is greyscaling images and the final quadrant is blurring of images.