BYU IDAHO CSE 110 | Programming Building Blocks

07 Prove: Assignment Milestone

Regarding Milestones and Assignments: Just as with the previous lessons, the Prove assignments for Lessons 07 and 08 will be working toward completing the same assignment. At the end of Lesson 07, you will complete part of the overall program, and then finish it in Lesson 08.

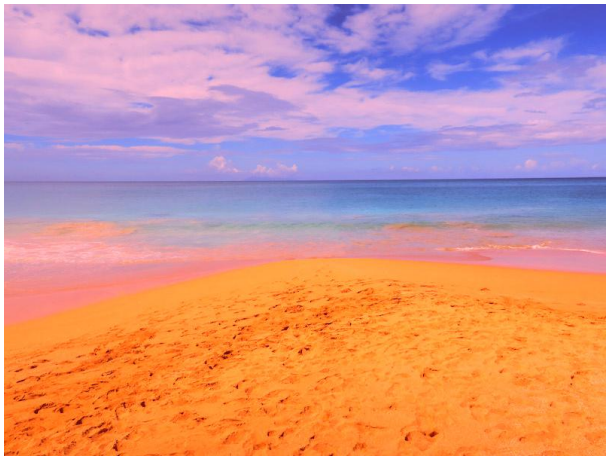
Read over the full requirements first. Then, at the bottom of this page, you'll see which features are required for the milestone at the end of Lesson 07.

Overview

Recall from the preparation material that digital images can be stored as a collection of pixel values. One of the things that computers are really good at is applying the same process over and over again. When it comes to the pixels of an image, if there were a process or rule or series of steps that you defined for one pixel, you could then have the computer "loop through" each pixel in the image and apply it to each one. For example, you could have the program loop through each pixel and add a certain amount to the red value of every pixel. This would add an overall red color to the image in general.

For example, consider the following image:

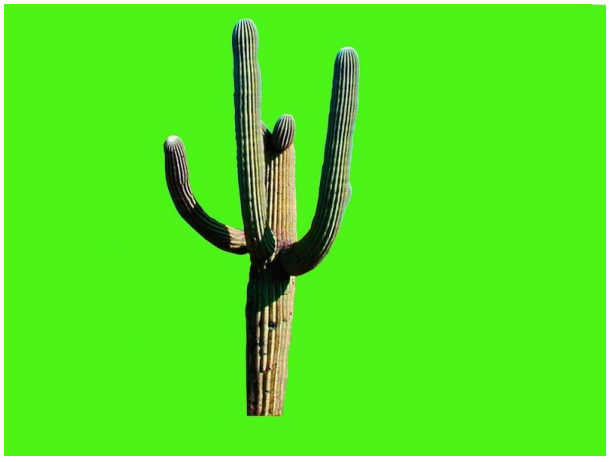
If the red value of each pixel was increased, it would look as follows:



Project Description

For this project, you will use this idea of looping, or iterating, through each pixel in an image to produce a green screen effect.

For a green screen effect, you will start with an image that has someone or something in front of a green background such as the following:



Then, you take a second image that has a scene or background in it (for example, a beach or forest) such as the following:

The goal is to take the item from the first picture and insert it over the top of the scene in the second picture. As you might expect, in high-end movie or image editing software there are sophisticated rules around identifying and blending the edges, etc., but the basic idea is that you want the computer to go through all the pixels from the first image. If the pixel is not green, you want to keep it, but if it is green, you want to use the corresponding pixel value from the other image.

This could produce an image such as the following:

Assignment

In order to complete this assignment, you need to successfully work through a number of different problems, including:

01. Downloading and saving image files

02. Installing and using a Python library
03. Opening an image file
04. Iterating through the pixels of an image
05. Using **`if`** statements to make decisions about how to handle each pixel
06. Producing a new image file and saving it to your computer

> **Note from Instructor:**
>
> To complete this assignment, you may need to push through some technical challenges along the way, especially related to installing libraries. Working through these is important for a few reasons:
>
> 1. It enables us to do something cool for this assignment!
>
> 2. You'll be prepared to install other libraries in the future, which you'll make use of for other courses and in other projects outside of school.
>
> 3. Learning a little bit more about how your computer is configured, where files are stored, etc., will be valuable to you, as you start to perform more complex tasks down the road—for this class, or for other things you'll do.
>
> With this in mind, if you run into trouble, don't be discouraged, but make sure to reach out for help. It will be worth it for you in the end!

For the milestone deliverable due at the end of Lesson 07, you need to have the library working and use it to open and display an image and retrieve and update some pixel values (no loops yet).

For the final deliverable at the end of Lesson 08, you need to complete the other parts of the project for the full green screen effect.

I. DOWNLOADING AND SAVING IMAGE FILES

A set of sample images for you to use for this project can be downloaded here:

» cse110_images.zip

Please download that zip file, extract the images from it and copy them to a folder on your computer where you want to work with them, and where you will create your Python file.

II. INSTALLING AND USING A PYTHON LIBRARY

Please refer back to the instructions in the preparation material about installing the Pillow library.

To verify that you have installed the library correctly. Start a new python program and include the following code:

```
from PIL import Image
print("The library is loaded correctly")
```

Run the program like you would any other Python program. If it runs, and displays, "The library is loaded correctly" without producing any error messages, you'll know that everything is installed and working correctly.

If you see error messages, you'll have to keep working to get it installed. Sometimes this takes a little bit of work to get exactly right, so please try it early and ask for help if you run into problems.

III. OPENING AN IMAGE FILE

Once you have installed the image library, you can use it to open an image. If everything is set up and running correctly, the Python code is as simple as:

```
# This line imports or includes the library we will need
from PIL import Image

# This line opens the image and loads it into a variable called "image_original"
image_original = Image.open("beach.jpg")

# This line attempts to open a new window to display the image.
image_original.show()
```

There is a potential problem with the code above, and that is, it expects the image file, "beach.jpg" to be in the same place where you are running your Python command. This may already be the case for you, but it's possible that you are running Python from one location but your pictures are stored in another.

One of the easiest ways to make it so that VS Code is running in the folder or directory you expect is to have it "open the folder" where you want it to run. To do this, in VS Code, select File menu and choose "Open Folder" (Windows) or "Open..." (macOS) and then browse to and open the *folder* that will contain both your Python program and your image files. Then, when you click the green play button to run your program, Python will be running in that folder by default.

If you aren't sure how to open a folder in VS Code rather than a file, please watch the following video to learn more about it:

» Video: Folders and Files (4 mins)

Before continuing, verify that you can open and display an image using the code snippet above.

IV. WORKING WITH PIXEL VALUES AND IMAGES

In order to access the pixels of the image, you'll need to use a few library functions.

If you have loaded an image into a variable **`image_original`**, you can use that variable to perform various operations that related to the image. For example, you can get the size like this:

```
width, height = image_original.size
```

If you want to access the pixels of the image, you can use the image variable to get access to another, internal variable that holds all of the information about the pixels. You can get the pixels of the image and store them into a variable like this:

```
pixels_original = image_original.load()
```

Once you have the pixels stored in a variable, you can access the red, green, and blue (RGB) values of an individual pixel at any spot in the image, for example, the coordinates (100, 200), using the square bracket notation **`[100, 200]`** like this:

```
r, g, b = pixels_original[100, 200]
```

Keep in mind that 100 and 200 could be replaced with any numeric values, include variables that hold numeric values, as long as you don't choose a number larger than the image. Also, please be aware that while in mathematics, the origin (0, 0) is located in the bottom left. In computer graphics, it is located in the top left. (The reason for this is not too important, but it has to do with the history of computers and CRT monitors.)

If you display the r, g, b values that were obtained from the last code, you can learn the color of that particular pixel.

Similarly, you can set new red, green, and blue values for the pixel at some x, y location in the picture like this:

```
# Don't forget to use parentheses around your (r, g, b)
pixels_original[100, 200] = (r, g, b)
```

Keep in mind that the computer always works with the red, green, and blue values in that order. If you mix up the variable names

and put `r` last, the computer will not realize this, and you'll have a bug in your code.

Take a minute to play around with this. Retrieve the pixel values from a few different locations. Then, set the value at a few locations to something really out of place, such as (255, 0, 255) which would be a mix of strong red and blue (a type of purple). Then, display the picture and see if you can zoom in at that spot and see the new pixel.

When you are ready to save an image out to a new file, you can call the save function like this:

```
image_original.save("the_file_goes_here.jpg")
```

> **Frequently Asked Questions:**
>
> One variable for images one for pixels? What's going on here?
>
> This library is making use of some techniques that you will learn if you continue on to CSE 210, Programming with Classes, but in short, the image variable is used to access information about the image in general such as its size, or to perform operations like saving it. The pixel variable helps us access the value of individual pixels. It maintains a link to the pixels in the image, so that if you use the image variable later to save or display an updated image, you'll see your changes.

Once you have reached this point, **you have completed the steps of the milestone deliverable**! The remaining steps are needed to complete project.

V. ITERATING THROUGH THE PIXELS

Equipped with the functions above, you can combine them to get the width and the height of the image, then, loop through all the pixels (using two for loops, that are nested, one within the other) to load the RGB values for each pixel, do something to those values, and finally set new values.

VI. USING IF-STATEMENTS TO MAKE DECISIONS

When you have the RGB values for a pixel stored in variables, you can use **`if`** statements to do anything you'd like with them. For example, you could say, "if g is greater than something and r and b are less than something else, do the following."

VII. PRODUCING AND SAVING A NEW IMAGE

If you want to produce a new image (for example, a new image that will be the combined version of your green-screen and background images, you call the **`Image.new()`**function as follows:

```
# Create a new image in RGB format that is the same size as image_original
image_new = Image.new("RGB", image_original.size)
pixels_new = image_new.load()
```

You can set its pixels values as shown above (using the new variable **`pixels_new`** to hold the values of the pixels for the new image):

```
pixels_new[x, y] = (r, g, b)
```

You can show the image on the screen as shown above (keep in mind that **`pixels_new`** is still connected to **`image_new`**. So when you set the pixels in the **`pixels_new`** variable, you then "show" or "save" the image, using the **`image_new`** variable.):

```
image_new.show()
```

And you can save the image as shown above:

```
image_new.save("the_new_image.jpg")
```

Video Demos

Whew! That is a lot of new information to process on top of the concept of loops that you are mastering this week. To help see some of these library functions and tools in action, please watch the following video demo that shows how to use these tools to load, manipulate, and display images in Python.

- » Python Image Demo Part 1: Saving images and opening them in Python (6 mins)
- » Python Image Demo Part 2: Getting and setting pixel values (8 mins)
- » Python Image Demo Part 3: Using loops to update pixels (7 mins)
- » Python Image Demo Part 4: More tools and strategies (6 mins)

Milestone Requirements

At the end of Lesson 07, to help make sure you are on track to finish the assignment, you need to complete the following:

01. Download the images for this assignment, unzip them, and save them to a location on your computer.
02. Install the image library and import it in a Python program without errors.
03. Load an image in Python and have it shown on the screen.
04. Retrieve and print out the numbers corresponding to the pixel RGB values for 5 different pixels.
05. Set a new color value for 5 different pixels.
06. Show the version of the picture on the screen and observe the changed pixels.

Final Requirements

01. Load a green image and a background image.

02. Create a new image that is the same size as the other images.

03. Iterate through all the pixels of the green image. Check to see if the green value is greater than a certain number and that the red and blue values are less than a certain number. If so, get the red, green, and blue values from the pixel at that location in the *background* image, if it's not a bright green color, get the red, green, and blue values from the pixel at that location in the green, foreground image.

04. Set the pixel in your new image to have the appropriate red, green, and blue value, as described.

05. Save the new image to a file.

06. Try a few different images.

Make it your own

See if you can find some ways to make this project your own. Perhaps you could try blending images by averaging values. Perhaps you could play with different thresholds or filters. Ask the user for the location of the image files, and use those. See what you can come up with and have fun!

Submission

When the Milestone is complete, answer the questions in the associated quiz in I-learn.

When the final project is complete, upload your code and an image or two that you created to I-learn.