Assignment 7: CALLING AND WRITING TO ARRAYS

For this assignment, our objective is to learn how to call and write values to arrays. We will be using arrays to write our own image manipulation functions. You will edit three functions: (1) flipping an image, (2) inverting an image's color, and (3) combining two images. You will then create one custom function that does some manipulation of your choosing. Get creative with it!

Deliverables:

1. a PDF document with your figures and your answers to the following questions:
    a. Describe how you chose to combine your images. What conditions did you use to choose which image you got the pixel value from? Did you like the outcome?
    b. Describe your custom function. What were the inputs and outputs? What manipulations did you do to the image?
    c. How could you use arrays in an interactive art setting?
2. Your modified eng103_base7.py file
3. Saved image from each function (flip, invert color, combine, custom)

Calling Values from an Array:

Call a single value: Let's start with an example numpy array that has dimensions (x,y,z). Each dimension is an integer. To call one specific value from this array, we would do the following:

```
array_value = example_array[x,y,z]
```

Here, x,y,and z **must be integers less than the dimension size**. For example, if we have an array with dimensions (20,20,3). In the example above, x and y must be integers 0 to 19, and z must be an integer 0 to 2. The maximum value is one less than the dimension value because python begins its indexing at 0 rather than 1. The array_value would end up being a single value of either float or int, depending on what values were in the example_array.

Call multiple values: What if we want to call multiple values at once? Let's start with an example where we want to call all the z values in the array:

```
array_z_values = example_array[x,y,:]
```

We have replaced the x index with a colon. This allows us to call all the z values for a specific x and y value. The array_z_values will be an array with dimensions (1,1,3).

This can also be done for more than one dimension of the array. For example, the code below would call one specific z value for all the x and y values.

```
array_specific_z_value = example_array[:,:,z]
```

The array_specific_z_value will be an array with dimensions (20,20,1).

Writing to Arrays:

Writing to an array is very similar to calling a value from an array. The way we index the array is the same. Given the dimensions, we call a specific value (or values) and set them equal to a new value (or values).

Writing a specific value: When writing a value to an array, you must make sure that the new value you want to put in the array is the same type as the rest of the array. For example, if your array is filled with integers, the new_value in the example below must also be an integer.

```
example_array[x,y,z] = new_value
```

Writing to multiple values: We can also write to multiple values at once. However, when writing multiple values to an array, you must make sure that the new values are of the same type as the rest of the array and *of the same dimension as the values you want to write to.* For example, in our example_array, if you want to write to all the z values for a given x and y, you must make sure your new_array is of dimension 3 and the same type as the original values.

```
new_array = [a,b,c]
example_array[x,y,:] = new_array
```

STARTING:

1. Open your eng103_base7.py
2. Read through all the comments. The comments contain instructions for how to edit each function and where to put your initial images in the main function,
    a. You will be editing Flip_Image, Invert_Color, and Comine_Images and well as the main function.

EDITING THE CODE:

In this assignment, and all future assignments, the places where you have to edit the code will be denoted in the comments as follows:

```
##### EDIT HERE ######
```

In this assignment, and all future assignments, the places where you have to STOP editing the code will be denoted in the comments as follows:

```
##### STOP EDITING HERE #####
```

This way, you know which parts you can change without losing functionality in the code or adding additional errors.

Reminder: SPACING MATTERS IN PYTHON. You will get errors if the code is not formatted correctly.