CS1110 Summer 2022 — Assignment 5*

Due Wednesday 7/27 at 11:59pm

Read this before you begin

We expect that this assignment will be reasonably hard, so budget time accordingly.

Learning Goals The purpose of this assignment is to help you to solidify your understanding of lists of lists, and how to build and design classes. Relevant material can be found in Canvas: Conditional Statements (Units 8 and 9), Lists (Units 12 and 14), Loops (Unit 13), and Python Classes (Units 16-20).

Submission For this assignment, you will submit three files: `a5pixel.py`, `a5image.py`, and `a5app.py`.

Submission within 24 hours after the deadline (Wednesday) will be accepted with a 10% late penalty. Even if only one file (of several) is submitted late, the assignment as a whole will be marked late. Assignment submission on Canvas closes 24 hours after the deadline and no further submission will be allowed.

Do not put your name in the files that you will submit, but *do* include your NetID. The reason is to help promote fairness in grading. Thank you for heeding our request, even if it sounds a little strange.

Group work and academic integrity You must work either on your own or with one partner. *If you work with a partner, you must first register as a group in Canvas and then submit your work as a group.* For a group, "you" below refers to "your group." You may discuss background issues and general strategies with others and seek help from the course staff, but the work that you submit must be your own. In particular, you may discuss general ideas with others but you may not work out the detailed solutions with others. It is not OK for you to see or hear another student's code and it is *certainly* not OK to copy code from another person or from published/Internet sources. If you feel that you cannot complete the assignment on you own, please seek help from the course staff.

External Libraries For this assignment, there are several Python libraries that can do the operations we are asking you to do pretty much "automatically". Using any external library for this assignment besides `math`, `copy`, `random`, and `typing` is explicitly prohibited and will be considered academic dishonesty.

*Author: Dietrich Geisler; based on assignments by Anne Bracy, Ariel Kellison, Lillian Lee, Steve Marschner, Stephen McDowell, Walker White, and surely the influence of David Gries.

1 Background

Motivation A tool for modifying images is surprisingly common and useful. Operations like color shifting, mirroring, and blurring images are useful and reasonably practical. In this assignment, we will explore designing and building classes that can do such operations, as well as building a simple interface allowing the user to select an operation.

In particular, we will be working with bitmap (`.bmp`) files, which are a standard lossless image format. When an image format is *lossless*, this means that the image contains exactly the RGB (red-green-blue) values written to the image file, thus preventing any loss of information. This comes at the cost of bitmap images being relatively large – a 1920x1080 image alone would take up megabytes of storage! You can read more about bitmap images if you are interested, though you shouldn't need to know details beyond what's provided in this document.

Files The files `a5pixel.py` and `a5image.py` contain the code you are to work with. Each contain a single class with a collection of methods – your primary task is to implement these methods. Each class and method includes a docstring, though feel free to add to the provided documentation if you would like.

The files `a5helper.py` and `a5test.py` contain helper functions and functions for testing your class, respectively. The former, in particular, contains functions for reading and writing images using the `Image` class. Note that these functions will not work until you implement the initialization methods for both `Pixel` and `Image` per the method specifications.

As usual, we include a copy of `testcase.py` to aid with testing.

Finally, we have provided four image files for testing purposes: `simple.bmp`, `bmp_24.bmp`, `python_logo.bmp`, and `dragon.bmp`. You may of course convert any image to a bitmap image (there are handy online converters) if you want to experiment!

Classes For this assignment, you will be working with the `Pixel` and `Image` classes. A `Pixel` object represents a single pixel, and has four attributes: `r`, `g`, `b`, and `a`. `rgb` represent the red, green, and blue values of that pixel, respectively, while `a` represents the opacity of the pixel.

An `Image` object represents a collection of `Pixel` that form an image. Specifically, an image consists of a grid of pixels and methods that can manipulate this grid. `Image` methods generally modify the image in–place (to avoid costly data copying).

2 Your Task

This assignment is divided into 3 parts:

1. Implementing the `Pixel` class
2. Implementing the `Image` class
3. Implementing a command–line interface for reading images from a file, modifying them, and writing the result to another file.

Unlike previous assignments, we have written tests in `a5test.py` for you – feel free to add tests if you worry these tests are insufficient or you would like to verify an edge case. Your code should pass all the tests we provide.

2.1 The Pixel Class

Your first goal should be to implement the methods outlined in the `Pixel` class of `a5pixel.py`. The specifications of each object method are given in this file – be sure to read the class description and method specifications carefully!

2.2 The Image Class

Implementing this class will likely be the bulk of this assignment. Your task is to implement each method outlined in the `Image` class of `a5image.py`. Again, the specifications of each method are given in the associated file.

Since these specifications are designed to be fairly terse and readable, they can be difficult to understand the technical details of what to implement. We recommend reading the provided test cases for behavior examples if you're not sure how a method should work.

Additionally, the `WRITE_IMAGES` flag at the top of the file (`True` by default) causes images to be generated when running the test suite. These images should be visually inspected to help ensure that the respective image method does what you expect. To help understand these images, the approximate expected look of each (after implementing the associated method) is described in the table below:

Image name (`.bmp`)	Description (relative to the base `dragon.bmp` file)
`dragon_add_pixel`	Blue-ish Green
`dragon_red_shift`	Reddish
`dragon_darker`	Generally darker, but same general colors
`dragon_monochrome`	Black–and–white
`dragon_mirrored_horizontal`	Flipped over the x–axis
`dragon_mirrored_vertical`	Flipped over the x–axis
`dragon_noisy`	Grainy with, well, a lot of noise
`dragon_blur`	Blurry / lower–definition

2.3 User Interface

For this last piece of the assignment, you should implement a simple command-line interface in `a5app.py` to read, modify, and write images. This code will be similar to what you wrote in assignment 4, so feel free to reuse ideas or code from your codebase there.

In particular, you will be graded on your application performing the following steps:

1. Read in an image by filename
2. Perform one manipulation of that image (run one of the methods defined in the `Image` class)
3. Write the resulting image to another filename

As a reminder, `read_image` and `write_image` in `a5helper.py` will be very important for this application. Note that, for full points, you should be able to accept reasonable input without crashing. As with assignment 4, no need to test this application with unit tests, just make sure it works before submitting.

As always, let us know if you have any questions, and good luck!