

**Carleton University**  
**Department of Systems and Computer Engineering**  
**SYSC 1005 - Introduction to Software Development - Fall 2018**

**Summative Exercises - Photo Editor, Version 2**

**Exercise 1**

**Background:** This is a question from the Fall 2017 final exam. If you can do this question without assistance from a TA or one of your colleagues, you should have no difficulties with the image processing questions in this term's exam.

**Exam Question:**

A filter function named `scatter` is passed an `Image` object. This function returns a new image that looks like a copy of the original image in which the pixels have been randomly scattered. To do this, it first creates a new image that is a copy of the original image. For each pixel in the new image, `scatter` randomly selects a pixel in the original image. The colour of the random pixel in the original image provides the new colour of the pixel in the new image.

To randomly select a pixel, `scatter` randomly selects a column and row that are within 10 of the  $(x, y)$  coordinates of the current pixel in the new image. In other words, the column of the randomly-selected pixel will be in the range  $x - 10$  to  $x + 10$ , inclusive, and the row of the randomly-selected pixel will be in the range  $y - 10$  to  $y + 10$ , inclusive. If the column or row are out of bounds, the function will randomly select another column and row. It will repeat this until the column and row are both in-bounds.

Example: assume `scatter` is currently visiting the pixel at  $(7, 3)$  in the new image. The column and row coordinates of the randomly-selected pixel must be within 10 of  $(7, 3)$ . Suppose the function generates random numbers 2 and -5. The coordinates of the random pixel are therefore  $(7 + 2, 3 - 5) = (9, -2)$ , which is out of bounds. As such, the function generates another pair of random numbers, say -2 and 1. The coordinates of the random pixel are  $(7 - 2, 3 + 1) = (5, 4)$ , which is in-bounds. The colour of the pixel at  $(7, 3)$  in the new image is then replaced with the colour of the pixel at  $(5, 4)$  in the original image.

Here is an incomplete definition of `scatter`. Parts of several expressions and statements have been replaced by ruled lines. Your task is to complete the function by writing the missing pieces of code.

```

from Cimpl import *
import random

def scatter(image):
    """ (Cimpl.Image) -> Cimpl.Image

    Return a new image that looks like a copy of an image in which the
    pixels have been randomly scattered.

    >>> original = load_image(choose_file())
    >>> scattered = scatter(original)
    >>> show(scattered)
    """

    # Create an image that is a copy of the original.

    new_image = _____

    # Visit all the pixels in new_image.

    for _____ in _____:

        # Generate the row and column coordinates of a random pixel
        # in the original image. Repeat this step if either coordinate
        # is out of bounds.

        row_and_column_are_in_bounds = _____
        while not row_and_column_are_in_bounds:

            # Generate two random numbers between -10 and 10, inclusive.

            rand1 = _____

            rand2 = _____

            # Calculate the column and row coordinates of a
            # randomly-selected pixel in image.

            random_column = _____

            random_row = _____

```

```

    # Determine if the random coordinates are in bounds.

    if _____
        _____
        _____:

        row_and_column_are_in_bounds = _____

    # Get the color of the randomly-selected pixel.

    _____ = get_color(image, _____)

    # Use that color to replace the color of the pixel we're visiting.

    _____(new_image, _____)

# Return the scattered image.

_____

```

**Exercise:** Download `scatter.py` from cuLearn and open the file in Wing 101. This file contains the incomplete function definition shown above. Replace the ruled lines with the missing code. Use the Python shell to test your `scatter` function.

## Exercise 2

Open the `filters` module you developed in Labs 5 and 6. Copy/paste your `scatter` function you developed in Exercise 1 into your `filters` module. This module should now contain 9 filters: `grayscale`, `weighted_grayscale`, `extreme_contrast`, `sepia_tint` and `posterize` (from Lab 5), `detect_edges`, `detect_edges_better` and `blur` (from Lab 6), and `scatter`.

Complete any filters that you didn't finish during Labs 5 and 6.

## Exercise 3

If you finished Lab 7, coding and testing the solution to this exercise should take less than 15-20 minutes.

Make a copy of file `lab_7_ui.py`, which you developed during Lab 7. Rename the copy to `photo_editor.py`.

Open `photo_editor.py` in Wing 101. Modify the script so that users can run some of the functions in your `filters` module. The user interface must look like this:

L)oad image  
B)lur E)dge detect P)osterize S)catter T)int sepia  
W)eighted grayscale X)treme contrast  
Q)uit

Notes:

- This user interface will not call your `grayscale` and `detect_edges` filters.
- The `W` command will call your `weighted_grayscale` filter.
- The `E` command will call your `detect_edges_better` filter. For this command, you'll need to allow the user to enter a threshold value (you did something very similar to this in Lab 7, for the solarizing filter).

In all other respects, this photo editor should operate in the same way as the one you developed for Lab 7. Specifically:

- **the image manipulations must be cumulative.** For example, if you load an image and type the letter `P`, a posterized image is created and displayed. If you then type the letter `W`, the program creates and displays a grayscale version of the posterized image, not a grayscale version of the image that was originally loaded.
- the program should display the error message **"No image loaded"** if you attempt to use one of the filters (the `W`, `X`, `T`, `E`, `P`, `B` and `S` commands) before you've loaded an image into the editor. (For this part, you should be able to reuse the code you wrote for Exercise 7 in Lab 7.)
- the program should display the error message **"No such command"** if an invalid command is typed. Note that if an invalid command is entered before an image has been loaded, only **"No such command"** should be displayed; that is, **"No image loaded"** should not be displayed at the same time. (For this part, you should be able to reuse most of the code you wrote for Exercise 8 in Lab 7.)

## Wrap-up

1. Make sure that each function definition in `photo_editor.py` and `filters.py` has a docstring containing:
  - a type contract;
  - a brief description of what the function does. (Remember, this description does not explain the function's implementation details/algorithm.)
  - at least one example of how to call the function from the Python shell.
2. Make sure that each function follows the widely-accepted conventions for good Python programming style that were introduced throughout the course. For example:
  - function names and variable names should be descriptive (your functions shouldn't contain a sequence of variables named `a1`, `a2`, `a3`, `a4`, etc.);

- use one space to separate operands and operators and use one space to separate each argument in an argument list; e.g., write:

```
new_colour = create_color(r // 2, g // 4, b)
```

instead of:

```
new_colour=create_color(r//2,g//4,b)
```

## Submitting Your Photo Editor

Log in to cuLearn and submit `filters.py` and `photo_editor.py`. To do this:

1. Click the Submit Photo Editor link. A page containing instructions and your submission status will be displayed. After you've read the instructions, click the **Add submission** button. A page containing a **File submissions** box will appear. Drag `filters.py` and `photo_editor.py` to the **File submissions** box. **Do not submit files with different names. Do not submit another type of file (e.g., a ZIP file, a RAR file, a .txt file, etc.)**
2. After the icons for the files appear in the box, click the **Save changes** button. At this point, the submission status is "Draft (not submitted)". If you're ready to finish submitting the files, jump to Step 2.2. If you aren't ready to do this; for example, you want to do some more work on the editor and submit revised files later, you can leave the files with "draft" submission status. When you're ready to submit the final version of your editor, you can update your "draft" file submission by following the instructions in Step 2.1, then finish the submission process by following the instructions in Step 2.2.
  - 2.1. You can replace or delete a previously submitted file by clicking the **Edit my submission** button. The page containing the **File submissions** box will appear.
    - 2.1.1. To overwrite a file you previously submitted with a file having the same name, drag another copy of the file to the **File submissions** box, then click the **Overwrite** button when you are told the file exists ("There is already a file called..."). After the icon for the file reappears in the box, click the **Save changes** button.
    - 2.1.2. To delete a file you previously submitted, click its icon. A dialogue box will appear. Click the **Delete** button., then click the **OK** button when you are asked, "Are you sure you want to delete this file?" After the icon for the file disappears, you can drag another file to the **File submissions** box. When you're done, click the **Save changes** button.
  - 2.2. Once you're sure that you don't want to make any changes, click the **Submit assignment** button. A **Submit assignment** page will be displayed containing the message, "Are you sure you want to submit your work for grading? You will not be able to make any more changes." Click the **Continue** button to confirm that you are ready to submit your lab work. This will change the submission status to "Submitted for grading".