Homework 0 Questions

Document Instructions

- 5 questions **[4 + 4 + 8 + 6 + 4 + 3 + 3 = 32 points]**.
- Fill all your answers within the answer boxes, and **please do NOT remove the answer box outlines**.
- Questions are highlighted in the **orange boxes**, bonus questions are highlighted in **blue boxes**, answers should be recorded in the **green boxes**.
- Include code, images, and equations where appropriate.
- To identify all places where your responses are expected, search for 'TODO'.
- The answer box sizes have been set by the staff beforehand and will truncate your text if it goes beyond the limit. Please make sure your responses fit in the appropriate spaces. **Extra pages are not permitted unless otherwise specified.**
- Make sure your submission has the right number of pages to validate page alignment sanity (check the footer).
- Please make this document anonymous.

Gradescope Instructions

- When you are finished, compile this document to a PDF and submit it directly to Gradescope.
- The pages will be automatically assigned to the right questions on Gradescope *assuming you do not add any unnecessary pages*. **Inconsistently assigned pages will lead to a deduction of 2 points per misaligned page (capped at a maximum 6 point deduction).**

Q1: [4 points]

For each of the following, complete the task and check the box to mark it as done.

TODO: Check off all items

- ☑ This is an example of a checked box
- ☐ Read the GitHub tutorial here.
- ☐ Create a GitHub account, if you don't have one.
- ☐ Join the Gradescope course.
- ☐ Join the course Ed.
- ☐ Set up the python environment and virtual environment.
- ☐ Set up an editing environment (VSCode), get it to use your python virtual environment, and know how to debug within it by setting breakpoints.
- ☐ Read the Python tutorial.

Q2: [4 points]

Please find and read the course collaboration policy on the course website and mark whether each of the following scenarios violates the policy.

Note: To fill in boxes, replace '\square' with '\blacksquare' for your answer.

(a) Another cs1430 student looking at your code to help you debug, after you have spent time trying to tackle the bug or have come to TA office hours/Ed.

TODO: Check the right option
- ☐ Acceptable
- ☐ Violation

(b) Using the result images from another student's code for your write up because your code is broken.

TODO: Check the right option
- ☐ Acceptable
- ☐ Violation

(c) Googling third party sites to clarify concepts for written and code assignments, with proper citation.

TODO: Check the right option
- ☐ Acceptable
- ☐ Violation

(d) A student who has previously taken the course and is not currently a TA sharing code with you to help you get through a bug.

TODO: Check the right option
- ☐ Acceptable
- ☐ Violation

Q3: **[8 points]** Computer vision is all around us, sometimes in surprising ways. Answer the following questions so we can get to know you better :)

(a) **[1 + 1 points]**

If you could have any computer vision related superpower, with no limitations, what would it be? How would you use it? **[2-3 sentences]**

TODO: Your answer for (a) here

(b) **[6 points]** All students enter the course with different backgrounds in socially responsible computing.

Please first review our ethics primer that introduces basic concepts to everyone.
Please list at least three values that are at stake if you use your superpower as intended (or if your archenemy misused your superpower). **[5-6 sentences]**

TODO: Your answer for (b) here

Q4: **[6 points]** Here is an image: grizzlypeakg.png (in images folder)

(a) **[2 points]** Below is some code that sets pixels that have a value of 50 or less to 0. This removes some of the lower-intensity haze around the bright lights. However, the code only works on single-channel grayscale images.

How could we convert the above code to handle color images? Might we use another for loop?

```
from skimage import io

A = io.imread('grizzlypeakg.png')
height, width = A.shape

# TODO: introduce a for loop that allows this
# code to work on RGB images
for i in range(height):
    for j in range(width):
        if A[i,j] <= 50 :
            A[i,j] = 0
```

(b) **[4 points]** Imagine we wanted to process 1000 images in the same way, but we weren't sure if our program was fast enough in execution time to cope with that many images.

(i) **[2 points]** We could time code execution for a single image (we learned about time execution in the Python tutorial), but it's dangerous to assume that the time taken for one image $\times 1000$ will equal 1000 image computations. This is because a single short tasks on multitasking computers often take variable time.

Instead, compute the time on a smaller number, say, 10 images, and compute the average time for a single image.

Note: When measuring the time, please ignore the file loading. To do so, either write your code in a way such that you make copies of the loaded

image inside the loop or in a way such that you load the image in a loop, but only time the modification.

TODO: Your answer for (b) (i) here

(ii) **[1 point]**

You might find the code would be too slow to handle 1000 images. Why is it slow? Let's speed it up. Using what we learned in our Python tutorial, what single line would replace the logic in the innermost for loop?

```
    # TODO: Your line here
```

TODO: Your answer for (b) (ii) here

(iii) **[1 point]**

Time the execution of your faster code for a single image. To do this, average out from running over multiple images (say 10) to get a reliable estimate. How much faster is the new version, as a multiplicative factor? (eg., 2×, 5×.)

TODO: Your answer for (b) (iii) here

Q5: [3 + 1 points] We wish to darken an image by editing the values in its matrix. But, when trying to visualize the result, we see some 'errors'.

Image: gigi.jpg (in images folder)

Find out what's incorrect with the approach given below. Then, fix the code such that it maintains the same intended operation, and include the resultant image.

```
from skimage import io
import matplotlib.pyplot as plt
import numpy as np

# TODO: fix the issue that's causing the unwanted artefacts
# ('errors') in the output image
I = io.imread('gigi.jpg').astype(np.float32)
I = I - 50
plt.imshow(I)
plt.show()
```

Q6: [2 + 1 points]

The debugger within VSCode is an important tool you can use to discover potential bugs in the code that you right.

Imagine our task is to create a crop of an image that starts at the center of the image and extends to the lower right corner of the image. If all goes well, we should only see content from the lower right region of the original image.

Image: gigi.jpg (in images folder)

```
from skimage import io
import matplotlib.pyplot as plt

origImage = io.imread('gigi.jpg')
(height, width, channels) = origImage.shape
startCropX = width % 2
startCropY = height % 2
croppedImage = origImage[startCropY:, startCropX:]

plt.imshow(croppedImage)
plt.show()
```

Create a new python file in the same directory as the image, and copy in the above code block. Then, open the file in VSCode, and execute the code within a debugging session by pressing F5 (or 'Run → Start Debugging'). At the prompt, we wish to 'Debug the currently active Python file'.
The output is not currently what we want, so let's stop execution and then identify the bug in this program:

1. First, set a breakpoint at line 7 and then re-execute the code within a debugging session.
2. Inspect the 'startCropX' variable either by looking at the left-hand Variables panel, or by mouse hovering over the variable in the text editor. What should it be?
3. Execute line 7 of code by 'stepping over' the current line (F10, or 'Run → Step Over). We should now be about to execute line 8.
4. Inspect 'startCropY' and verify its correctness.

At this point, you might have an idea of how to fix the code. But, before stopping execution and editing the file, let's test out our hypothesis in the 'Debug Console' during debugging.

1. Switch to the Debug Console by pressing CTRL-SHIFT-Y (or 'View → Debug Console') — you should see it in the bottom right of the display screen.
2. *This is an interactive Python console with access to working memory.* As a test, print out the value of 'width'. Perform a mathematical operation on 'width'.

3. Assign the right value to ‘startCropX’ within the Debug Console. Notice how the value updated in the Variables panel.
4. Do the same for ‘startCropY’.
5. From this point, execute the rest of the code by Continuing beyond our current paused position in the code. Press F5 to Continue (or ‘Run → Continue’).

Re-execute the debugger, and capture a screenshot showing your use of the Debug Console and inspection of a variable. Also, write the correct code below.


```
# TODO: paste your code here
```

Q7: [2 + 1 points] This program should print out the maximum value in the matrix obtained by multiplying a random non-square matrix with its transpose.

Here, we're using some numpy functions that may be new to us, but they each have self-explanatory names.

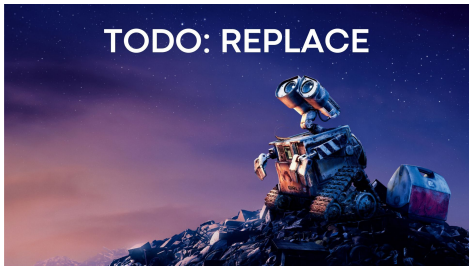
```
import numpy as np
from numpy import random as r

mat_1 = r.rand(200,150)
mat_2 = mat_1
np.transpose(mat_2)
mat_3 = np.matmul(mat_1, mat_2)
mat_max = np.max(mat_3)

print("Max value:", mat_max)
```

This time, when we execute the code, it will raise an exception.

> Run the code in a debugging session, note the exception, and inspect the variables. Form a hypothesis for the error, and use the Debug Console to test that it prevents the exception.

Hint: Remember rules about matrix multiplication. What should the dimensions of each matrix be? Use the debugger to notice how the shapes of the images do or do not change.

Capture a screenshot of your session showing us the issue and paste the correct code.




```
    # TODO: paste your code here
```

```
##################################################
# YOU MAY USE THIS ADDITIONAL SPACE

# WARNING: IF YOU DON'T END UP USING THIS SPACE
# KEEP THESE COMMENTS TO MAINTAIN PAGE ALIGNMENT
##################################################
```

Only A7 should be on this page

Feedback? (Optional)

Please help us make the course better. If you have any feedback for this assignment, we'd love to hear it!