# Homework 0 Questions

## Template Instructions

This document is a template with specific answer regions and a fixed number of pages. Given large class sizes and limited TA time, the template helps the course staff to grade efficiently and still focus on the content of your submissions. Please help us in this task:

- Make this document anonymous.
- Questions are in the orange boxes. Provide answers in the green boxes.
- Use the footer to check for correct page alignment.
- **Do NOT remove the answer box.**
- **Do NOT change the size of the answer box.**
- **Extra pages are not permitted unless otherwise specified.**
- **Template edits or page misalignment will lead to a 10 point deduction.**

## Gradescope Submission

- Compile this document to a PDF and submit it to Gradescope.
- Pages will be automatically assigned to the right questions on Gradescope.

## This Homework

- 7 questions **[4 + 4 + 8 + 6 + 5 + 3 + 3 = 33 points]**.
- Include code, images, and equations where appropriate.

# Declaration of Generative AI Use

## Reminder of Course Policy

- The use of GenAI tools (e.g., ChatGPT, Grammarly, Bard) for completing any part of this course is discouraged.
- Using these tools is not needed to be successful in the class and could be detrimental to your learning experience.
- If you use them, you must cite the tool you used and explain how you used it.
- If you do not cite the tool, it is an academic code violation.
- We will be using special tools to detect cases of unattributed GenAI use.

## Student Declaration

**Have you used generative AI tools to complete this assignment:**

YES ☐ NO ■

**If you answered YES above, describe what tools you used and what parts of the assignment you used them for below:**

*Example: I used ChatGPT to debug my convolution implementation*

**Q1: [4 points]**

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

- ☑ 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.
- ☐ Complete the Python tutorial.
- ☐ Set up the python environment and virtual environment.
- ☐ Set up an editing environment (e.g., VSCode), set it to use your python virtual environment, and know how to debug within it by setting breakpoints.

**Q2: [4 points]**

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

(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.

- ☐ Acceptable
- ☐ Violation

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

- ☐ Acceptable
- ☐ Violation

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

- ☐ 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.

- ☐ Acceptable
- ☐ Violation

**Q3:** **[8 points]** Computer vision is all around us, sometimes in surprising ways.

(a) **[2 points]**

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

(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.

(i) **[3 points]**

State one value that is affected if you use your superpower. Explain your reasoning. **[2–3 sentences]**

(ii) **[3 points]**

State one value that is affected if your archenemy used your superpower. Explain your reasoning. **[2–3 sentences]**

**Q4:** **[6 points]** Here is an image: grizzlypeak-grayscale.png (in the images folder)

(a) (i) **[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.

Using another for loop and other appropriate modifications, convert the following code to work on grizzlypeak-color.png (also in the images folder).

TODO: Modify the following code:

```
from skimage import io

A = io.imread('grizzlypeak-grayscale.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
```

(ii) **[1 points]** Let's find the time it takes to run this operation once. Because a single short task on multitasking computers often takes variable time, we can record how long code execution takes on a small number of images and then average the execution time.

Record how long it takes to execute 10 times. How long does it take to change the pixel values once? Please include your code.

*Note:* When measuring the time, please ignore the file loading. You should only time the image modification itself.

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

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

(b) **[3 points]** Let's say we want to run our image modification 1000 times. If we used our current implementation, it would be slow. Why might that be? We think we can find a faster solution.

(i) **[2 point]**

Using logical indexing (see the Python tutorial), find a faster solution to remove the haze.

```
    # TODO: Your code here
```

(ii) **[1 point]**

Now, using your optimized solution, record how long it takes to change the pixel values 1000 times. How long does the new code take to change the pixel values on one image? How much faster is the new version per image, as a multiplicative factor (eg: 2×, 5×)? Please include your code.

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

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

**Q5:** **[5 points]** Your friend tries to write code that reads in an image and displays it.

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

I = io.imread('./images/gigi.jpg').astype(float32)
plt.imshow(I)
plt.show()
```

However, when they run the code, the image is almost entirely white. Confused, they come to you for help.

(a) **[1 point]**

Why is the image being displayed incorrectly?
*Hint: What data values does plt.imshow() expect?*

(b) **[2 points]** You and your friend think of two possible ways to display the image correctly.

(i) **[1 point]**

Write a possible fix that uses the float data type.

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

(ii) **[1 point]**

Write a possible fix that uses the 8-bit unsigned integer data type.

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

(c) **[2 points]** Your friend comes to you and wants help darkening an image.

Read in 'gigi.jpg', darken the pixel intensities by an amount equivalent to 20% of the total brightness range of the *data type as stored in the image file*, and display the new image. Include your code and a screenshot of the darkened image.

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

**Q6: [3 points]**

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

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('./images/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:** **[3 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.

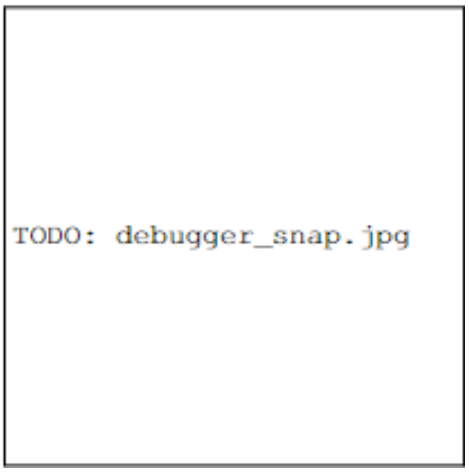
```
import numpy as np
from numpy import random as r

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, set a breakpoint before it, 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
```

**Only Q7 should be on this page**

## Feedback? (Optional)

We appreciate your feedback on how to improve the course. You can provide anonymous feedback through this form which can be accessed using your Brown account (your identity will not be collected). If you have urgent non-anonymous comments/questions, please email the instructor.