# Project 0 Questions

## Instructions

- Complete the setup checklist.
- 5 questions.
- Include code, images, or equations where appropriate.
- Please make this document anonymous.
- We recommend editing this .tex file in Overleaf to ensure the used packages are available. If you use pdfLatex, your distribution (e.g., MiKTeX) needs to have the packages installed..
- This assignment is **fixed length**, and the pages have been assigned for you in Gradescope. As a result, **please do NOT add any new pages**. We will provide ample room for you to answer the questions. If you *really* wish for more space, please add a page *at the end of the document*.
- **We do NOT expect you to fill up each page with your answer.** Some answers will only be a few sentences long, and that is okay.
- When you are finished, compile this document to a PDF and submit it directly to Gradescope.

## Setup Checklist (Graded)

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 Piazza.

☑ 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.

## Questions

**Q1:** Please find and read the course collaboration policy on the course website and mark whether each of the following scenarios violates the policy.

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

(a) Another CSCI1430 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/Piazza.

- ■ Acceptable
- □ Violation

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

- □ Acceptable
- ■ Violation

(c) Googling 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

**Q2:** Computer vision is all around us, sometimes in surprising ways.

(a) If you could have any computer vision related superpower—there are no limitations—what would it be? (1-2 sentences)

(b) How would you use your superpower? (2-3 sentences)

(c) Find a recent example (past 6 months or so) of a controversial real world use of computer vision. What caused the controversy? (3-4 sentences)

Please cite a source. Extra credit for unique examples!

**A2:** Your answer here.

(a) I would like to have a superpower of automatically generating a 3D model once I see the appearance of a target object.

(b) I am superfan of Lego. If I have that superpower, I can convert everything that I am interested in into a 3D model and make a Lego based on that. By using this superpower, I can bring more fun to other kids' lives as well.

(c) In mid-June 2020, Boston prohibited the use of facial surveillance technology, and bar any city official from receiving facial tracking by demanding it via third parties.

The racial bias caused by this computer vision technology leads to the controversy. Some people may be misidentified by this technology. In addition, it may have a chilling effect on civil liberties, and may be used to essentially chill free speech.

Source:

https://www.wbur.org/news/2020/06/23/boston-facial-recognition-ban

https://www.eff.org/document/ordinance-banning-face-surveillance-technology-boston

**Q3.1:** Here is an image: grizzlypeakg.png

We wish to set all pixels that have a value of 50 or less to 0, to remove some of the lower-intensity haze around the bright lights. However, our code only works on single-channel grayscale images.

```
from skimage import io

A = io.imread('grizzlypeakg.png')
(height,width) = A.shape
for i in range(height):
    for j in range(width):
        if A[i,j] <= 50 :
            A[i,j] = 0
```

How could we convert it to handle color images using another for loop? Please include your code.

*Image:* grizzlypeak.jpg is a color version of the same image.

**A3.1:** Your written answer here. (1-2 sentences)

```
from skimage import io

A = io.imread('grizzlypeak.jpg')
(height, width, channel) = A.shape
for i in range(height):
    for j in range(width):
        for c in range(channel):
            if A[i, j, c] <= 50:
                A[i, j, c] = 0
```

**Q3.2:** Next, 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.

(a) Please time your code execution for a single image!

*However, let's not assume that the time taken for one image* $\times 1000$ *will equal* 1000 *image computations, as 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.*

*When measuring the time, you can ignore file loading. To do so, you should 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. You can also assume that all images are of equal resolution to our initial image—we only need to use grizzlypeak.jpg for this question.*

(b) 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, write code for a sped-up version.

(c) Time the execution of your faster code—use an appropriate number of images to get a reliable estimate. How much faster is the new version, as a multiplicative factor? (eg., $2\times$, $5\times$.)

**A3.2:** (a) Your written answer here. (1-2 sentences)
The average code execution for a single image is about 15.402 seconds. This result is obtained by computing the same image 10 times and taking its average.

(b)

```
from skimage import io
import time

sum = 0
for i in range(10):
    A = io.imread('grizzlypeak.jpg')
    start = time.time()
    A[A<=50] = 0
    end = time.time()
    sum += (end - start)
average_time = sum / 10
print(average_time)
```

(c)
The average code execution for a single image using the faster code is about 0.007 second. So the new version is $2200\times$ times faster.

A3.2: (Extra Space)

**Q4:** We wish to reduce the brightness of an image by editing the values in its matrix. But, when trying to visualize the result, we see some "errors".

*Image:* gigi.jpg

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

I =  io.imread('gigi.jpg').astype(np.float32)
I = I - 50
plt.imshow( I )
plt.show()
```

What is incorrect with this approach? How can it be fixed while maintaining the same intended brightness reduction? Please include your code and result image.

**A4:** Your written answer here. (2-3 sentences)

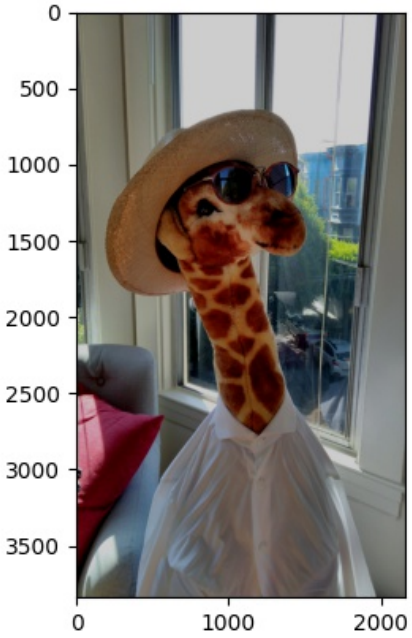
Some pixels may have an intensity that is less than 50. Thus, some negative intensity will come up after an intensity reduction of 50. Each element in the image array (obtained by io.imread() method) is of the type <class 'numpy.uint8'> that is an unsigned integer ranging from 0 to 255. Thus, there is a risk of Integer Overflow. It is a part of reasons that lead to "errors". So I set all the pixels with an intensity less than 50 to a pixel with the intensity of 50 first. After that, I did an intensity reduction on the whole image.

It is incorrect to use astype(np.float32) method that violates typical assumptions made in other functions about data type ranges. That can cause confusion while using built-in scikit-image processing functions.

While using matplotlib.pyplot.imshow, Matplotlib, it is assumed that the pixel values are in between 0 and 1 if the image is in floating-point format. So, when converting an image to floating-point format, we should normalize the image so that all its entries are between 0 and 1. It prevents potential confusion.

```
from skimage import io, img_as_float32
import matplotlib.pyplot as plt
import numpy as np

I =  io.imread('gigi.jpg')
I[I<50] = 50
I = I - 50
I = img_as_float32(I)
plt.imshow(I)
plt.show()
```

Result image here:


0
500
1000
1500
2000
2500
3000
3500
0
1000
2000

**A4:** (Extra Space)

**Q5.1: Debugging—Control Flow** Next, please show us that you can use the debugger within VSCode—we'd like to make sure everyone is set up and can use it correctly.

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

```
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.

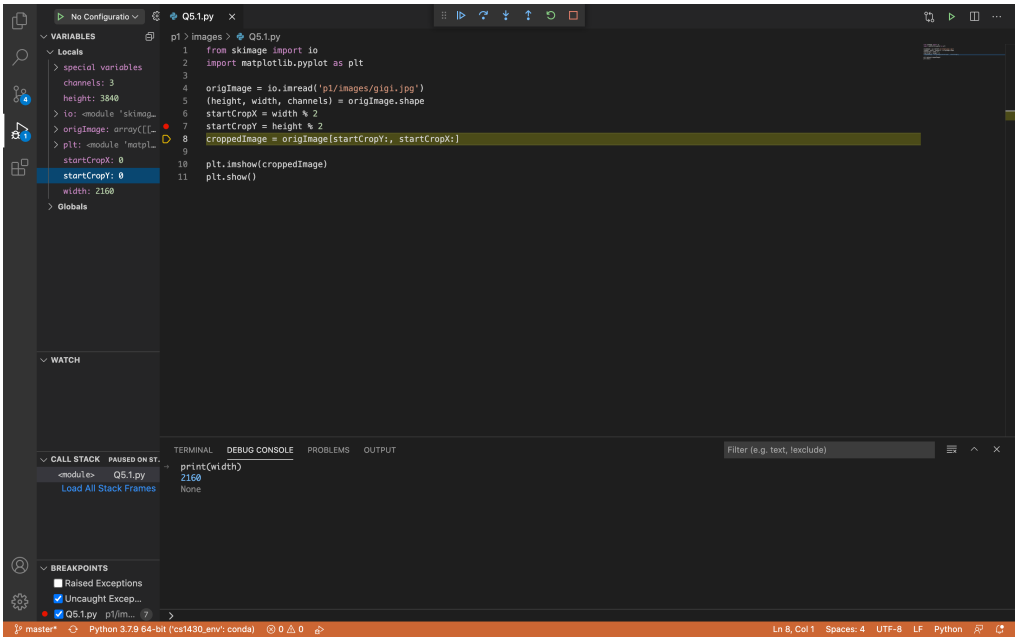
At this point, you might have an idea of how to fix the code. But, before stopping execution and editing the file, let's test out our hypothesis in the 'Debug Console' during debugging.

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’).

Do we now see the correct output?

**A5.1:** (a) Re-execute the debugger, and capture a screenshot showing your use of the Debug Console and inspection of a variable. Please include it below.




(b) Describe what caused the bug. You can write the corrected lines as your answer below.

In order to get a crop of an image that starts at the center of the image and extends to the lower right corner of the image, we need to find the center of the image and include all the pixels in the lower right region of the original image. So instead of using the Modulo operation to get startCropX and startCropY, we should use Integer Division operation.

```
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()
```

**A5.1:** (Extra space)

**Q5.2: Debugging—Exceptions** 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.

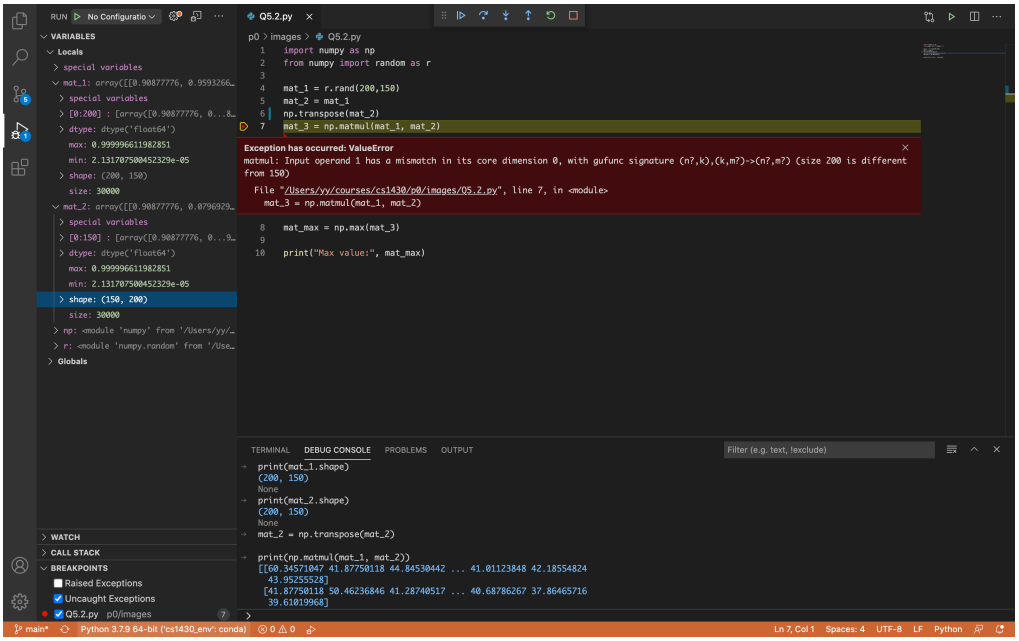
```
import numpy as np
from numpy import random as r

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)
```

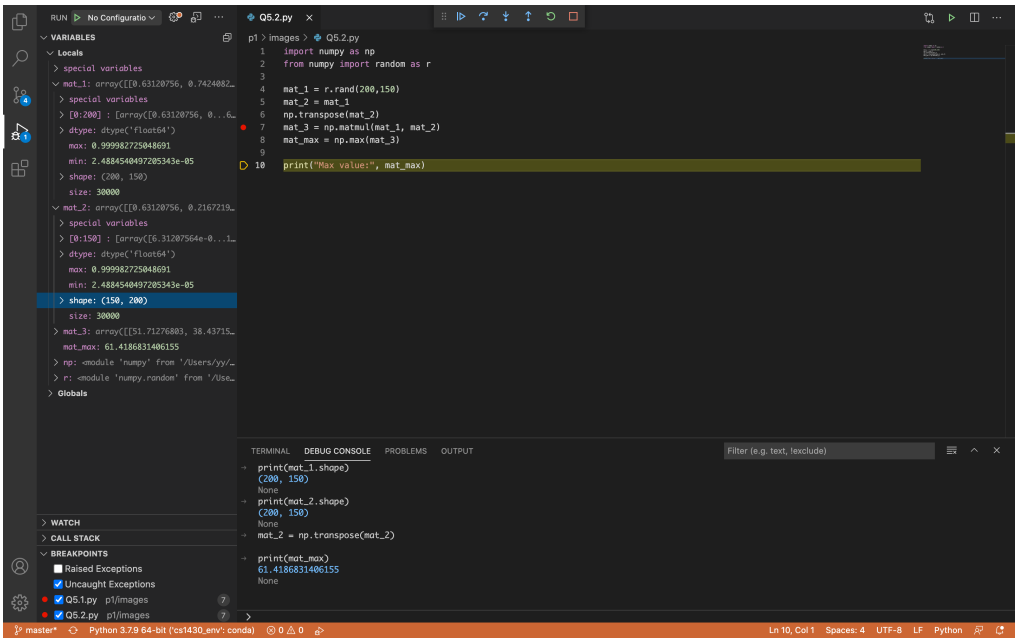
This time, when we execute the code, it will exception.

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. Capture a screenshot of your session showing us the issue and your fix.

*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.*

**A5.2:** (a) Please include an image of your debugging session here.






(b) Describe what caused the bug. You can write the corrected lines as your answer below.

The transposed value of the matrix $mat\_2$ is not stored. So both matrices have a shape of $(200, 150)$. The last dimension of $mat\_1$ is not the same size as the second-to-last dimension of $mat\_2$, which violates rules of matrix multiplication, and leads to the $ValueError$.

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

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

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

**A5.2** (Extra space)