Homework 1 Questions

Instructions

- Compile and read through the included Python tutorial.
- 2 questions.
- Include code.
- Feel free to include images or equations.
- Please make this document anonymous.
- **Please use only the space provided and keep the page breaks.** Please do not make new pages, nor remove pages. The document is a template to help grading.
- If you really need extra space, please use new pages at the end of the document and refer us to it in your answers.

Submission

- Please zip your folder with **hw1_student id_name.zip** (ex: hw1_20201234_Peter.zip)
- Submit your homework to KLMS.
- An assignment after its original due date will be degraded from the marked credit per day: e.g., A will be downgraded to B for one-day delayed submission.

Questions

Q1: We wish to set all pixels that have a brightness of 10 or less to 0, to remove sensor noise. However, our code is slow when run on a database with 1000 grayscale images.

Image: grizzlypeakg.png

```
import cv2
import numpy as np
A = cv2.imread('grizzlypeakg.png',0)
m1, n1 = A.shape
for i in range(m1):
   for j in range(n1):
      if A[i,j] <= 10:
         A[i,j] = 0
```

Q1.1: How could we speed it up?

A1.1: Your answer here.

We can use logical indexing to improve performance.

```
import cv2
import numpy as np
A = cv2.imread('grizzlypeakg.png',0)
B = A <= 10
A[B] = 0
```

Q1.2: What factor speedup would we receive over 1000 images? Please measure it.

Ignore file loading; assume all images are equal resolution; don't 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.

A1.2: Your answer here.

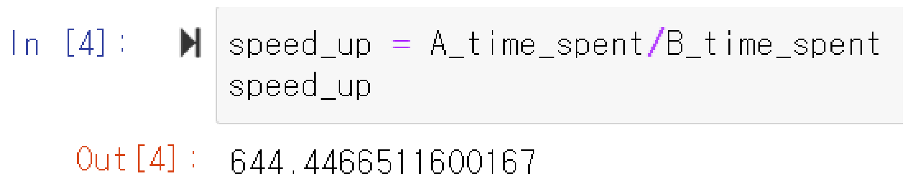
I tried to convert all 1000 images, but it took too much time and I couldn't. So, the result was obtained assuming that the ratio of time to convert 1000 images is the same as the ratio of time to convert 1 image.

```
import cv2
import numpy as np
import time

A = cv2.imread('grizzlypeakg.png',0)
m, n = A.shape
A_start_time = time.time()
for i in range(m):
  for j in range(n):
    if A[i,j] <= 10:
A[i,j] = 0
A_time_spent = time.time() - A_start_time

B = cv2.imread('grizzlypeakg.png',0)
B_start_time = time.time()
B_log = B <= 10
B[B_log] = 0
B_time_spent = time.time() - B_start_time

speed_up = A_time_spent/B_time_spent
speed_up
```



```
In [4]:  speed_up = A_time_spent/B_time_spent
         speed_up

Out[4]:  644.4466511600167
```

Figure 1: Result of speed ratio of gray image

The speed ratio is about 644.

Q1.3: How might a speeded-up version change for color images? Please measure it.

Image: grizzlypeak.jpg

A1.3: Your answer here.

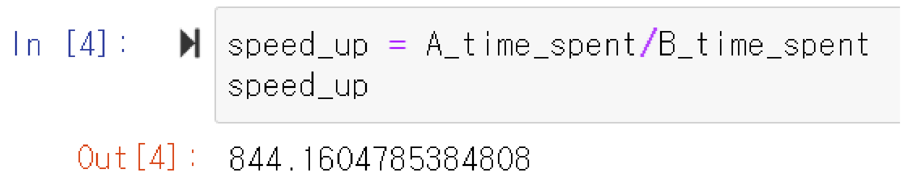
This is similar to A1.2

```
import cv2
import numpy as np
import time

A = cv2.imread('grizzlypeak.jpg')
l, m, n = A.shape
A_start_time = time.time()
for i in range(l):
  for j in range(m):
    for k in range(n):
      if A[i,j,k] <= 10:
        A[i,j,k] = 0
A_time_spent = time.time() - A_start_time

B = cv2.imread('grizzlypeak.jpg')
B_start_time = time.time()
B_log = B <= 10
B[B_log] = 0
B_time_spent = time.time() - B_start_time

speed_up = A_time_spent/B_time_spent
speed_up
```



```
In [4]:  speed_up = A_time_spent/B_time_spent
         speed_up

Out[4]:  844.1604785384808
```

Figure 2: Result of speed ratio of color image

The speed ratio is about 844.

Q2: We wish to reduce the brightness of an image but, when trying to visualize the result, we see a brightness-reduced scene with some weird "corruption" of color patches.

Image: gigi.jpg

```
import cv2
import numpy as np
I = cv2.imread('gigi.jpg').astype(np.uint8)
I = I - 40
cv2.imwrite('result.png', I)
```

Q2.1: What is incorrect with this approach? How can it be fixed while maintaining the same amount of brightness reduction?

A2.1: Your answer here.

We have to add a few lines of code.

```
import cv2
import numpy as np
I = cv2.imread('gigi.jpg').astype(np.uint8)
I_log = I < 40
I = I - 40
I[I_log] = 0
cv2.imwrite('result.png', I)
```

Q2.2: Where did the original corruption come from? Which specific values in the original image did it represent?

A2.2: Your answer here.

All values of I have values between 0 and 255. So, values less than 40 are out of range after subtracting by 40. Originally, they should have negative values, but they are changed to high values, because they are out of the range of bits to be expressed. For example, 39 becomes 255, 38 becomes 254, ..., 0 becomes 216. High values are displayed brightly, so they appear partially bright in the picture.