2024 BMED223 :: Hands-on instructions

- **Filename: week11_studentID_studentName.zip**
- **Deadline: 2024/05/20 23:59**
- **No excuse for late submission**. Prepare for submission in advance.
- There is no limit on the number of submissions, but grading will be based on the final file.
- Use comments (#, #%%, etc.) to separate problems.
- **Please write a class in *prob4.py*** that is separated from *main.py* file.
- **And import the *prob4.py* into the *main.py* file** and output the answer to the problem.
- Please **compress all written python files** to the above name and submit.

서동휘 andy1200@korea.ac.kr
박영주 lime2514@korea.ac.kr

1. Matrix Manipulation and Vectorized Operations

a. Make a 2D matrix **A** with the contents listed below.

1 2 3

4 5 6

7 8 9

b. Print the size, data type, and number of bytes of **A**.

c. Print the value of center element of **A**.

d. Print the last column of **A** assuming that you don't know the size of **A**.

e. Print the last row of **A** assuming that you don't know the size of **A**.

f. Transpose A and copy the last row of the transposed **A** into a vector(1d array) **A2** then print it. Change the value of A2[0] to 777 and display A and A2. Do you observe 777 in A as well? If yes, figure out the reason and correct the problem.

g. With A in (a), append a row vector [10 11 12] at the end of last row of **A**.

h. Take 3x2 sub-array from **A** and move it into a new array **B** and print **B**. Change the value of B[0] to 777 and display A and B. Do you observe 777 in A as well? If yes, figure out the reason and correct the problem.

i. Create 8x8 array **C** and fill it with random numbers. The minimum value of **C** should be greater than or equal to 0, and its maximum value should be less than or equal to 1. Then, multiply 255 to C and convert the data type so that each element of the array be unsigned integer.

j. Print the shape, maximum and minimum values of **C**.

k. Create a new array **D** so that it has the same size of array **C**, then fill the array **D** with zeros.

l. Without using for- or while- loop, find the elements in **C** whose values are greater than or equal to 128, and move those elements into **D**. (hint: vectorization)

m. Write a single line of codes to perform the Problem 11 and 12 at once. (hint: vectorization)

2. **Functions with numpy.ndarray class**

Make a 2D matrix **u2** by using array **v2** as follows.

```
v2      = [[ 3    5   -2],
           [5   -1    0],
           [2    4   -3]]
```

a. If a value in **v2** is great than 0, move it to **u2** at the same location. Otherwise, change the value to 0. You MUST use numpy.where() function

b. Make a customized function *thres()* which runs in exactly same was as (a)
   - *thres()* MUST have two input parameter.
     The one of input parameter is a array like **v2** in (a) and the other is threshold value.
   - *thres()* MUST return the output matrix like **u2** in (a)

c. Run (a) again by using *thres()* function and print the output.

3. Simple Image Processing

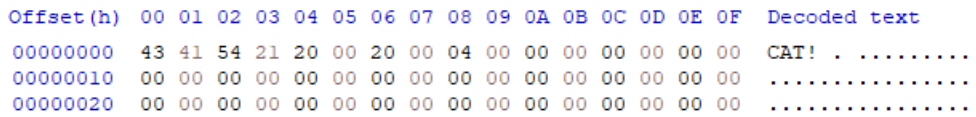
a. Image Negative
   - 1) Use *open()* function in PIL.Image library to read *'chest_xray.jpg'*.
   - 2) Convert the data type of *'chest_xray'* image into numpy array and name it *'ori_image'*
   - 3) Print the image size, min value, and max value.
   - 4) Make a 2d zero array *'rev_image'* which has the same size as *'ori_image'*.
   - 5) For each pixel, calculate **(max value) – (pixel value)** and put the results into *'rev_image'* array.

b. Image thresholding
   - 1) Make a 2d zero array *'thres_image'* which has the same size as *'ori_image'*.
   - 2) Change the pixel values of *'thres_image'* by the following rule:
     If a pixel in *'ori_image'* is lower than the mean value, put 0 at the same pixel location.
     Otherwise, move the pixel values from *'ori_image'* to *'thres_image'*.
     (hint: You can use *thres()* function from the problem 2.)
   - 3) Display *'ori_image'*, *'rev_image'* and '*thres_image'* within the same window.
     (hint : matplotlib.pyplot.imshow(img, cmap ='gray') and pyplot.subplots)

4. Simple Binary File Handling

a. Load the binary file '*cat.bin*', which includes the header information detailed below. All data saved using the little endian method.

<Figure : Binary values in *'cat.bin'*>



```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text
00000000   43 41 54 21 20 00 20 00 04 00 00 00 00 00 00 00  CAT! . .........
00000010   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000020   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

<Table : Bytes information of *'cat.bin'*>

bytes location	Data information
0 – 3	Header signature
4 – 5	Row size of the image (int - size 2)
6 - 7	Column size of the image (int – size 2)
8 – 11	Byte size per pixel (int – size 4)
12 – (before footer)	Data
4 bytes in end	Footer

You need to create a ***class bin_img_processing*** to process the image data.

b. Save the data information as attributes of the class.
(including the header signature, row size, pixel information, etc.)

c. Create methods of ***class bin_img_procesing***:

- Method 1 : Display the image using Matplotlib.
- Method 2 : Show the numpy array of the image data.
- Method 3 : Normalize all the image pixel values and save them in a new array.

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$$ (x : pixel value of the image)

d. Save the normalized image data as float type in a new binary file.