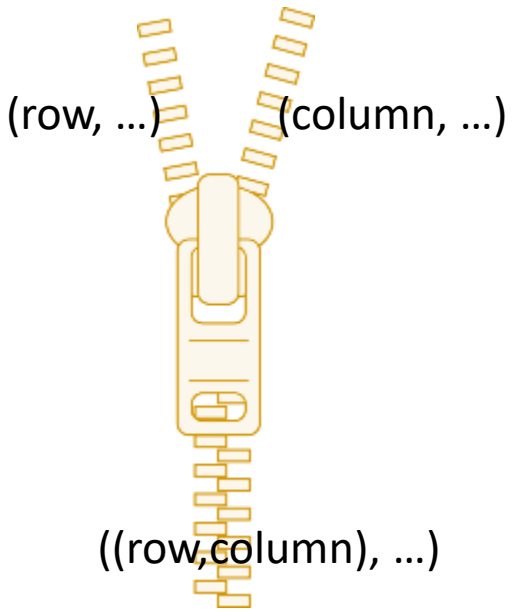
PS2 (pseudo-color)

PS2, only programming homework in CVE (NOTE: not difficult)

1. Search Min/Max
2. Generate look up color table
3. Apply look up color table

NOTE: OpenCV image is 2 dimensional matrix with [y][x] order (or [row][column])



- Min Max Search
- OpenCV
  - there is a way, but when multiple points, it becomes complicated, and I do not recommend to use them even you are allowed.
- Numpy
  - np.amax(), np.amin()
  - list(zip(*np.where()))
    - Iterative operator of zip()
    - np.where() generate separate array ((row, …), (col, …))
    - https://numpy.org/doc/stable/reference/generated/numpy.where.html
- Double loop
  - Single loop: Search max, min
    - If value is larger than max, new list is created with its location
    - Else if value is same as max, add location to the list
  - Double loop: (common)

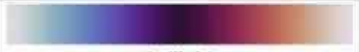
Generate color look up table

- OpenCV (easiest, but **you should not use** in this assignment)
  - cv2.applyColorMap(gray, colormap_type)
  - https://docs.opencv.org/4.x/d3/d50/group__imgproc__colormap.html
    - COLORMAP_JET(2): close to target
  - You can use right program in checking your program
- Numpy
  - np.linespace()
    - https://numpy.org/doc/stable/reference/generated/numpy.linspace.html
- for loop (Good enough)

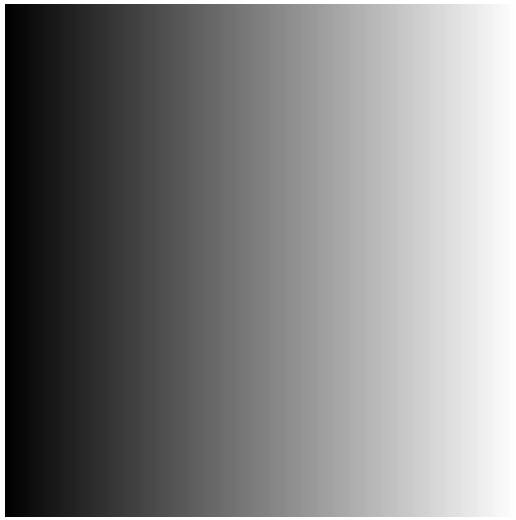
```
import cv2
import numpy as np
import argparse

parser = argparse.ArgumentParser(description='Pseudo Color generation via OpenCV')
parser.add_argument('-i', '--input', help='Path to input image.', default='x-ray.png')
parser.add_argument('-t', '--type',help='Type of colormap (0-21).', type=int, default='2')
args = parser.parse_args()

window_name = 'pseudo-color using OpenCV'

image = cv2.imread(args.input, 0)
new_image = cv2.applyColorMap(image, args.type)

cv2.namedWindow(window_name, cv2.WINDOW_NORMAL)
cv2.imshow(window_name, new_image)
cv2.waitKey()
```

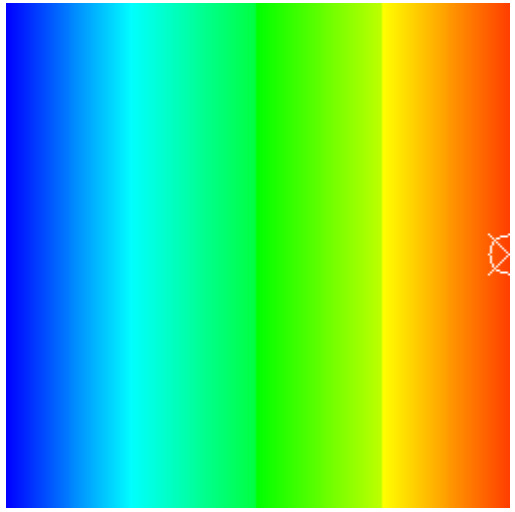
OpenCV colormap type (same as matlab)

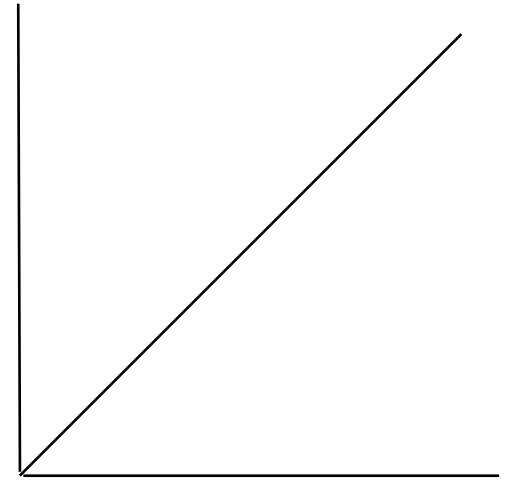
COLORMAP_AUTUMN Python: cv.COLORMAP_AUTUMN	
COLORMAP_BONE Python: cv.COLORMAP_BONE	
COLORMAP_JET Python: cv.COLORMAP_JET	
COLORMAP_WINTER Python: cv.COLORMAP_WINTER	
COLORMAP_RAINBOW Python: cv.COLORMAP_RAINBOW	
COLORMAP_OCEAN Python: cv.COLORMAP_OCEAN	
COLORMAP_SUMMER Python: cv.COLORMAP_SUMMER	
COLORMAP_SPRING Python: cv.COLORMAP_SPRING	
COLORMAP_COOL Python: cv.COLORMAP_COOL	
COLORMAP_HSV Python: cv.COLORMAP_HSV	
COLORMAP_PINK Python: cv.COLORMAP_PINK	

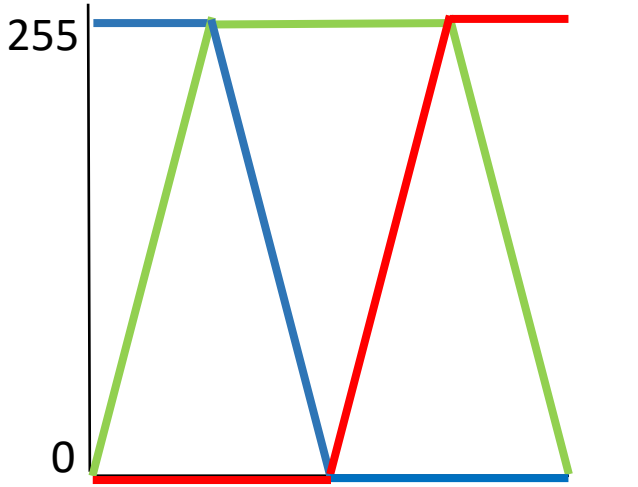
COLORMAP_HOT Python: cv.COLORMAP_HOT	
COLORMAP_PARULA Python: cv.COLORMAP_PARULA	
COLORMAP_MAGMA Python: cv.COLORMAP_MAGMA	
COLORMAP_INFERNO Python: cv.COLORMAP_INFERNO	
COLORMAP_PLASMA Python: cv.COLORMAP_PLASMA	
COLORMAP_VIRIDIS Python: cv.COLORMAP_VIRIDIS	
COLORMAP_CIVIDIS Python: cv.COLORMAP_CIVIDIS	
COLORMAP_TWILIGHT Python: cv.COLORMAP_TWILIGHT	
COLORMAP_TWILIGHT_SHIFTED Python: cv.COLORMAP_TWILIGHT_SHIFTED	
COLORMAP_TURBO Python: cv.COLORMAP_TURBO	
COLORMAP_DEEPGREEN Python: cv.COLORMAP_DEEPGREEN	

pseudo-color: color map generation









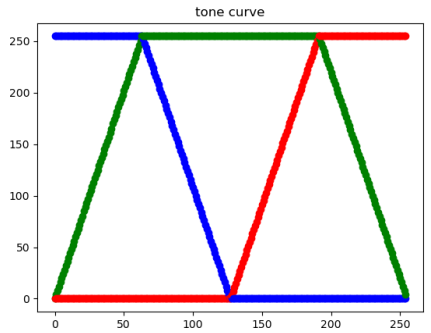
You need to check the input range of intensity and generate R/G/B tone line segment

Simple gray scale image makes it easy

```
import cv2
import numpy as np

img = np.empty((256,256), np.uint8)

for i in range(256):
    img[:,i] = i

cv2.imwrite("gray.png", img)
```

Check tone curve

```
import matplotlib.pyplot as plt

for i in range(0, 255):
    plt.scatter(i, table[0,i,0], c='blue')
    plt.scatter(i, table[0,i,1], c='green')
    plt.scatter(i, table[0,i,2], c='red')

plt.title("tone curve")
plt.show()
```



matplotlib output

Apply look up table

- OpenCV (fastest, but **you should not use** in this assignment)
  - cv2.LUT(image, table)
  - You can make sure your program works properly

This ASSIGNMET PROCEDURE

- Create output image data
  - OpenCV Image MAT is same as numpy.ndarray(shape, np.uint8)
- Apply look up table
  - Double for loop: simple but slow (C/C++ style)
  - Use numpy array technique to speed up