سهند نوعی 9923087 - تمرین صفر

لینک کولب من: https://colab.research.google.com/drive/107B5AjTAMgvZOtQPXT3ZFaYpWGEL8zuq#scrollTo=nnY3mPQTf-bu

قبل از هر کاری، با استفاده از مسیری که در خط پایین ذکر شده، یک کپی از این نوتبوک در گوگل درایو خودتان بسازید و تمرین را در آن نسخه حل کنید.

File --> Save a copy in Drive

Question 0: Run the cell below without modifying it.

بدون اعمال هیچ تغییری در بلوک زیر، آن را اجرا کنید.

Downloading the necessary data for this homework

```
#@title Downloading the necessary data for this homework
!wget --no-check-certificate 'https://drive.google.com/uc?export=download&id=1kAzfR-tVf3Oq91MJFkkLa2wqXSoQvPHH' -O 'HW0_data.zip'
!unzip HW0_data.zip
```

```
--2023-10-09 12:42:49--  https://drive.google.com/uc?export=download&id=1kAzfR-tVf3Oq91MJFkkLa2wqXSoQvPHH
Resolving drive.google.com (drive.google.com)... 172.253.115.101, 172.253.115.102, 172.253.115.113, ...
Connecting to drive.google.com (drive.google.com)|172.253.115.101|:443... connected.
HTTP request sent, awaiting response... 303 See Other
Location: https://doc-0o-2g-docs.googleusercontent.com/docs/securesc/ha0ro937gcuc7l7deffksulhg5h7mbp1/ov88rnv639gq4g1egmfrh2lpecp4s87g/1
Warning: wildcards not supported in HTTP.
--2023-10-09 12:42:50--  https://doc-0o-2g-docs.googleusercontent.com/docs/securesc/ha0ro937gcuc7l7deffksulhg5h7mbp1/ov88rnv639gq4g1egmf
Resolving doc-0o-2g-docs.googleusercontent.com (doc-0o-2g-docs.googleusercontent.com)... 142.251.16.132, 2607:f8b0:4004:c17::84
Connecting to doc-0o-2g-docs.googleusercontent.com (doc-0o-2g-docs.googleusercontent.com)|142.251.16.132|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 133870 (131K) [application/x-zip-compressed]
Saving to: ‘HW0_data.zip’

HW0_data.zip        100%[===================>] 130.73K  --.-KB/s    in 0.02s

2023-10-09 12:42:50 (7.33 MB/s) - ‘HW0_data.zip’ saved [133870/133870]

Archive:  HW0_data.zip
  inflating: chest-xray.png
```

The Imports

```
###
import numpy as np
import sys
import matplotlib.pyplot as plt
import cv2
###
```

Question 1: (20%)

Q1 - Part 1 (10%)

```
#@title Q1 - Part 1 (10%)
###
random_array = np.random.uniform(10, 54000, size=80)
random_array
###
```

```
array([41490.47704393, 35720.48723249, 26597.41675942, 32883.95530275,
       33382.96444654, 13358.22535807,  4676.74856224, 20693.01131063,
       41148.23291658, 37227.16149643, 28477.25445886,  2893.27146536,
       43396.36387135, 53090.19741949, 17482.70904854, 11126.88679649,
          76.5480986 , 23005.65734128, 35667.56738567,  9136.94836455,
       47211.70244871, 20085.99195897, 38681.40685438, 49174.35734929,
       30985.13882282, 39469.73387431, 50179.84879807, 17641.07308743,
       53946.5750659 , 21672.19213336, 24057.35601387, 32964.89975238,
       14492.04212384, 32994.31225439, 38002.16097078, 15762.44869782,
       29380.86988929, 42852.70602327,  2708.42143646, 26414.73860874,
       43637.5630819 , 16630.87791523, 17589.28278732, 44838.67197955,
       25926.75593046, 34418.66477268, 39207.2276376 , 20277.58181394,
       48875.30428918,  9280.36226762, 47477.59501794, 19901.51268906,
       47106.69262226, 34407.03262716, 25789.64924364, 29793.06750867,
       11716.74379133, 37709.10305514,  2964.5457593 , 29079.93825422,
       46558.37120265, 20099.74514027, 47453.23772394,  5143.16024611,
        6451.83099642, 19222.67623841, 15356.09106249,  3753.52832113,
       42979.89717593, 45176.06594347,  8891.83244403, 38885.3051667 ,
        3257.66346524, 19067.20762837,  8292.16853956, 21349.46136404,
       38711.82398426, 23620.63725998, 10474.8792333 , 15910.32465331])
```

Q1 - Part 2 (5%)

```
#@title Q1 - Part 2 (5%)
###
type(random_array), type(random_array[0])
###
```

```
(numpy.ndarray, numpy.float64)
```

Q1 - Part 3 (5%)

```
#@title Q1 - Part 3 (5%)
###
rounded = np.round(random_array, decimals=0)
rounded
###
```

```
array([41490., 35720., 26597., 32884., 33383., 13358.,  4677., 20693.,
       41148., 37227., 28477.,  2893., 43396., 53090., 17483., 11127.,
          77., 23006., 35668.,  9137., 47212., 20086., 38681., 49174.,
       30985., 39470., 50180., 17641., 53947., 21672., 24057., 32965.,
       14492., 32994., 38002., 15762., 29381., 42853.,  2708., 26415.,
       43638., 16631., 17589., 44839., 25927., 34419., 39207., 20278.,
       48875.,  9280., 47478., 19902., 47107., 34407., 25790., 29793.,
       11717., 37709.,  2965., 29080., 46558., 20100., 47453.,  5143.,
        6452., 19223., 15356.,  3754., 42980., 45176.,  8892., 38885.,
        3258., 19067.,  8292., 21349., 38712., 23621., 10475., 15910.])
```

Q1 - Part 4 (30%)

```
#@title Q1 - Part 4 (30%)
###
print("Integer types:")
print(f"int8: Min:{np.iinfo(np.int8).min}, Max:{np.iinfo(np.int8).max}, range:{np.iinfo(np.int8).max - np.iinfo(np.int8).min}, Size:{sys.gets
print(f"uint8: Min:{np.iinfo(np.uint8).min}, Max:{np.iinfo(np.uint8).max}, range:{np.iinfo(np.uint8).max - np.iinfo(np.uint8).min}, Size:{sys
print(f"int16: Min:{np.iinfo(np.int16).min}, Max:{np.iinfo(np.int16).max}, range:{np.iinfo(np.int16).max - np.iinfo(np.int16).min}, Size:{sys
print(f"uint16: Min:{np.iinfo(np.uint16).min}, Max:{np.iinfo(np.uint16).max}, range:{np.iinfo(np.uint16).max - np.iinfo(np.uint16).min}, Size
print(f"int32: Min:{np.iinfo(np.int32).min}, Max:{np.iinfo(np.int32).max}, range:{np.iinfo(np.int32).max - np.iinfo(np.int32).min}, Size:{sys
print(f"int64: Min:{np.iinfo(np.int64).min}, Max:{np.iinfo(np.int64).max}, range:{np.iinfo(np.int64).max - np.iinfo(np.int64).min}, Size:{sys
print(f"int: Size of int(1) is {sys.getsizeof(int(1))} and int(10000000000) is {sys.getsizeof(int(10000000000))} 👉 So the default size of ir
      " accommodate larger integer values based on the input.\n\n")
# print(f"float: Min:{np.finfo(np.float).min}, Max:{np.finfo(np.float).max}, range:{np.finfo(np.float).max - np.finfo(np.float).min}")
print("Float types:")
print(f"float32: Min:{np.finfo(np.float32).min}, Max:{np.finfo(np.float32).max}, range:{np.finfo(np.float32).max - np.finfo(np.float32).min},
print(f"float64: Min:{np.finfo(np.float64).min}, Max:{np.finfo(np.float64).max}, range:{np.finfo(np.float64).max - np.finfo(np.float64).min},
print(f"float: Size:{sys.getsizeof(float(1))}\n\n")

print(f"Since our array has integer numbers in range of 40 to 54000, np.int16 is enough for that:")
print(f"rounded array elements type before changing:{type(rounded[0])}")
rounded = np.int16(rounded)
print(f"rounded array elements type after changing:{type(rounded[0])}")
###
```

```
Integer types:
int8: Min:-128, Max:127, range:255, Size:25
uint8: Min:0, Max:255, range:255, Size:25
int16: Min:-32768, Max:32767, range:65535, Size:26
uint16: Min:0, Max:65535, range:65535, Size:26
int32: Min:-2147483648, Max:2147483647, range:4294967295, Size:28
```

```
int64: Min:-9223372036854775808, Max:9223372036854775807, range:18446744073709551615, Size:32
int: Size of int(1) is 28 and int(10000000000) is 32 👉 So the default size of int is the same as int32 but it can expand as needed to


Float types:
float32: Min:-3.4028234663852886e+38, Max:3.4028234663852886e+38, range:inf, Size:28
float64: Min:-1.7976931348623157e+308, Max:1.7976931348623157e+308, range:inf, Size:32
float: Size:24


Since our array has integer numbers in range of 40 to 54000, np.int16 is enough for that:
rounded array elements type before changing:<class 'numpy.float64'>
rounded array elements type after changing:<class 'numpy.int16'>
<ipython-input-32-9c76a7ddc8e6>:14: RuntimeWarning: overflow encountered in float_scalars
  print(f"float32: Min:{np.finfo(np.float32).min}, Max:{np.finfo(np.float32).max}, range:{np.finfo(np.float32).max - np.finfo(np.float32
<ipython-input-32-9c76a7ddc8e6>:15: RuntimeWarning: overflow encountered in double_scalars
  print(f"float64: Min:{np.finfo(np.float64).min}, Max:{np.finfo(np.float64).max}, range:{np.finfo(np.float64).max - np.finfo(np.float64
```

Q1 - Part 5 (5%)

```
#@title Q1 - Part 5 (5%)
###
r1 = rounded
# First way
r1 = rounded.reshape(8, 10)
print(f"r1.shape = {r1.shape}")
print(f"r1 = {r1}")
# Second way
rounded = np.reshape(rounded, (8, 10))
print(f"np.shape(rounded) = {np.shape(rounded)}")
print(f"rounded = {rounded}")
###
```

```
r1.shape = (8, 10)
r1 = [[-24046 -29816  26597 -32652 -32153  13358   4677  20693 -24388 -28309]
 [ 28477   2893 -22140 -12446  17483  11127     77  23006 -29868   9137]
 [-18324  20086 -26855 -16362  30985 -26066 -15356  17641 -11589  21672]
 [ 24057 -32571  14492 -32542 -27534  15762  29381 -22683   2708  26415]
 [-21898  16631  17589 -20697  25927 -31117 -26329  20278 -16661   9280]
 [-18058  19902 -18429 -31129  25790  29793  11717 -27827   2965  29080]
 [-18978  20100 -18083   5143   6452  19223  15356   3754 -22556 -20360]
 [  8892 -26651   3258  19067   8292  21349 -26824  23621  10475  15910]]
np.shape(rounded) = (8, 10)
rounded = [[-24046 -29816  26597 -32652 -32153  13358   4677  20693 -24388 -28309]
 [ 28477   2893 -22140 -12446  17483  11127     77  23006 -29868   9137]
 [-18324  20086 -26855 -16362  30985 -26066 -15356  17641 -11589  21672]
 [ 24057 -32571  14492 -32542 -27534  15762  29381 -22683   2708  26415]
 [-21898  16631  17589 -20697  25927 -31117 -26329  20278 -16661   9280]
 [-18058  19902 -18429 -31129  25790  29793  11717 -27827   2965  29080]
 [-18978  20100 -18083   5143   6452  19223  15356   3754 -22556 -20360]
 [  8892 -26651   3258  19067   8292  21349 -26824  23621  10475  15910]]
```

Q1 - Part 6 (5%)

```
#@title Q1 - Part 6 (5%)
###
print(f"min = {np.min(rounded)}")
print(f"max = {np.max(rounded)}")
###
```

```
min = -32652
max = 30985
```

Q1 - Part 7 (15%)

```
#@title Q1 - Part 7 (15%)
###
rounded_int8 = np.int8(rounded)
print(rounded_int8)
###
```

```
[[  18 -120  -27  116  103   46   69  -43  -68  107]
 [  61   77 -124   98   75  119   77  -34   84  -79]
 [ 108  118   25   22    9   46    4  -23  -69  -88]
 [  -7  -59 -100  -30  114 -110  -59  101 -108   47]
 [ 118   -9  -75   39   71  115   39   54  -21   64]
 [ 118  -66    3  103  -66   97  -59   77 -107 -104]
 [ -34 -124   93   23   52   23   -4  -86  -28  120]
 [ -68  -27  -70  123  100  101   56   69  -21   38]]
```

Q1 - Part 7 Explanation:

{بله تغییر در اکثر اعداد دیده شد. زیرا بازه int8 بین -128 تا 127 هست اما اعداد ما بین 10 تا 54000 هست بنابراین 8بیت سمت چپ در هنگام تبدیل آرایه int16 به int8 میس می‌شوند و قسمتی از داده از بین می‌رود.}

Q1 - Part 8 (15%)

```
#@title Q1 - Part 8 (15%)
###
C_two = tuple(rounded[:, 1])
print(type(C_two))
print(C_two)

R_three = rounded[2, 1:]
print(type(R_three))
print(R_three)
###
```

```
<class 'tuple'>
(-29816, 2893, 20086, -32571, 16631, 19902, 20100, -26651)
<class 'numpy.ndarray'>
[ 20086 -26855 -16362  30985 -26066 -15356  17641 -11589  21672]
```

Q1 - Part 9 (10%)

```
#@title Q1 - Part 9 (10%)
###
my_dict = dict(zip(C_two, R_three))
print(my_dict)
###
```

```
{-29816: 20086, 2893: -26855, 20086: -16362, -32571: 30985, 16631: -26066, 19902: -15356, 20100: 17641, -26651: -11589}
```

Question 2: (25%)

Q2 - Part 1 (100%)

```
#@title Q2 - Part 1 (100%)
###
# 1
# 1
def func(dims: tuple, seed: int):
  # 2
  if len(dims) !=2:
    return f"'dims' must be (2, ) but you've passed {len(dims), }"
  elif type(dims) != tuple:
    return f"'dims' type must be 'tuple' but you've passed {type(dims)}"
  elif type(seed) != int:
    return f"'seed' type must be 'int' but you've passed {type(seed)}"
  # 3
  result = np.zeros(dims, dtype=int)
  for i in range(dims[0]):
    for j in range(dims[1]):
      if (i, j) == (0, 0):
        result[0, 0] = seed
      else:
        if i - 1 < 0 and j - 1 >= 0:
          result[i, j] = result[i, j - 1]
        elif i - 1 >= 0 and j - 1 < 0:
          result[i, j] = - result[i - 1, j]
        else:
          result[i, j] = result[i, j - 1] - result[i - 1, j] - result[i - 1, j - 1]
  return result

func((3, 4), 1)
###
```

```
array([[ 1,  1,  1,  1],
       [-1, -3, -5, -7],
       [ 1,  5, 13, 25]])
```

Question 3: (30%)

Q3 - Part 1 (0%)

```
#@title Q3 - Part 1 (0%)
###
std_num = 9923087
###
```

Q3 - Part 2 (40%)

```
#@title Q3 - Part 2 (40%)
###
def circle_matrix(radius: np.uint8):
  if type(radius) not in [int, np.uint8]:
    return f"'radius' must be integer not {type(radius)}"
  if radius < 3:
    return f"'radius' must equal to or bigger than 3"
  radius = np.uint8(radius)


  center_i = center_j = radius
  side = 2 * radius + 1
  matrix = np.zeros((side, side), dtype=np.uint8)
  for i in range(side):
    for j in range(side):
      distance = np.sqrt((i - center_i) ** 2 + (j - center_j) ** 2)
      if distance <= radius:
        matrix[i, j] = 255
      else:
        matrix[i, j] = 0
  return matrix

circle_matrix(4)
###
```

```
array([[  0,   0,   0,   0, 255,   0,   0,   0,   0],
       [  0,   0, 255, 255, 255, 255, 255,   0,   0],
       [  0, 255, 255, 255, 255, 255, 255, 255,   0],
       [  0, 255, 255, 255, 255, 255, 255, 255,   0],
       [255, 255, 255, 255, 255, 255, 255, 255, 255],
       [  0, 255, 255, 255, 255, 255, 255, 255,   0],
       [  0, 255, 255, 255, 255, 255, 255, 255,   0],
       [  0,   0, 255, 255, 255, 255, 255,   0,   0],
       [  0,   0,   0,   0, 255,   0,   0,   0,   0]], dtype=uint8)
```

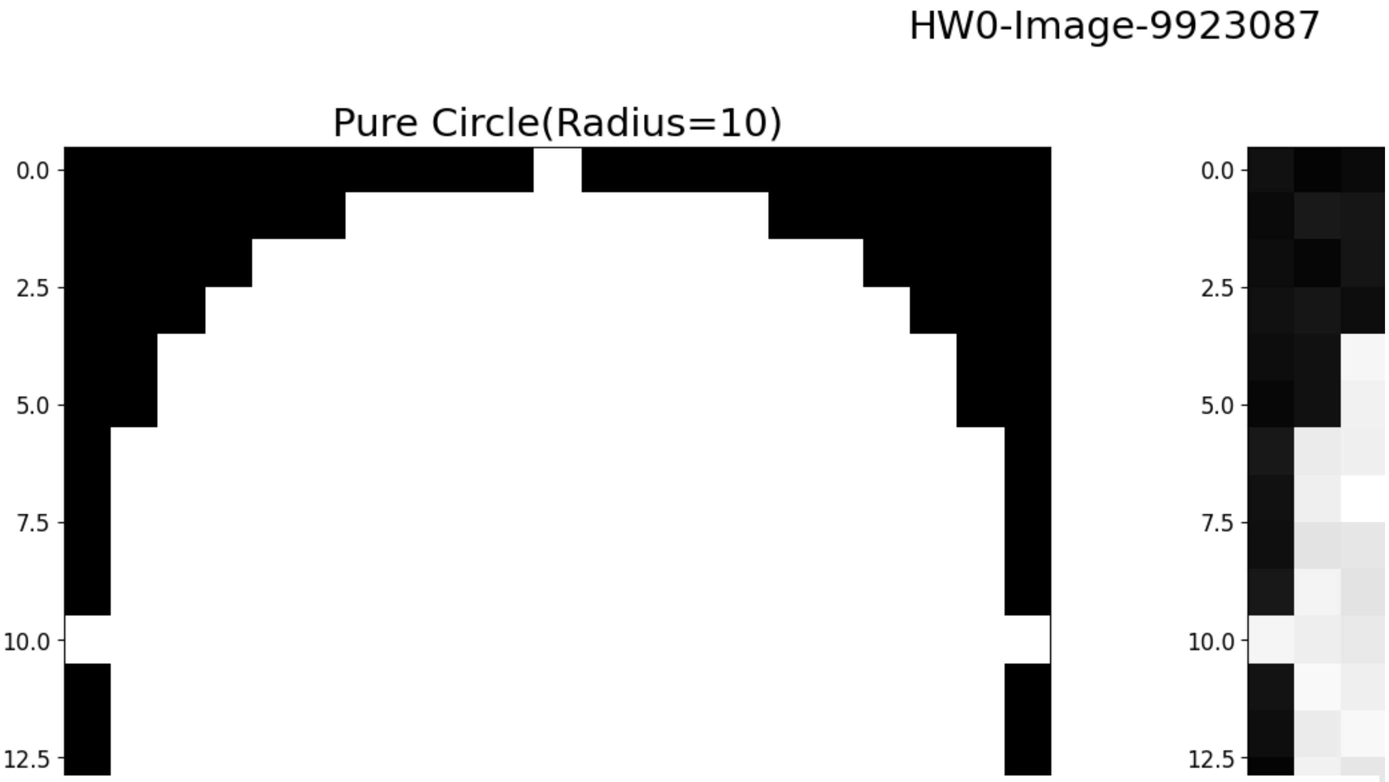
Q3 - Part 3 (35%)

```
#@title Q3 - Part 3 (35%)
###
def add_noise(matrix, noise_range: np.uint8):
  if matrix.ndim != 2:
    return "input matrix must be 2 dimensional"
  if noise_range <= 0:
    return "noise range must be positive"
  tmp = noise_range
  noise_range = np.uint8(noise_range)
  if tmp != noise_range:
    return "noise range must be between 0 to 255"

  noise_matrix = np.random.uniform(0, noise_range, size=matrix.shape)
  noise_matrix = np.uint8(np.floor(noise_matrix))
  final_noisy_matrix = matrix +  np.where(matrix == 0, noise_matrix, -noise_matrix)
  return final_noisy_matrix

noise_range = 20 + sum(list(map(int, str(std_num)))) % 15
add_noise(circle_matrix(3), noise_range)

###
```

```
    array([[  6,  14,   1, 252,  21,   2,   1],
           [  8, 233, 229, 228, 234, 228,   7],
           [  6, 228, 249, 252, 237, 235,  16],
           [231, 247, 235, 245, 233, 239, 243],
           [  8, 233, 252, 244, 229, 238,   4],
           [ 18, 237, 234, 249, 237, 228,   1],
           [ 21,  11,  14, 249,  26,  23,  15]], dtype=uint8)
```

Q3 - Part 4 (15%)

```
#@title Q3 - Part 4 (15%)
###
radius = 10
circle = circle_matrix(radius)
noisy_circle = add_noise(circle, noise_range)
# print(circle)
# print(noisy_circle)
circle_dim = circle.shape[0]
fig, axes = plt.subplots(1, 2, figsize=(circle_dim, circle_dim))
# print(type(axes))
# print(type(axes[0]))
# print(axes)
axes[0].imshow(circle, cmap="gray", vmin=0, vmax=255)
axes[0].set_title(f"Pure Circle(Radius={radius})", fontsize=circle_dim)
axes[1].imshow(noisy_circle, cmap="gray", vmin=0, vmax=255)
axes[1].set_title(f"Noisy Circle(Radius={radius})\nnoise range domain:{noise_range}", fontsize=circle_dim)
plt.suptitle(f"HW0-Image-{std_num}", fontsize=circle_dim)
plt.subplots_adjust(top=1.4)
for ax in axes:
  ax.tick_params(axis="x", labelsize=circle_dim / 1.75)
  ax.tick_params(axis="y", labelsize=circle_dim / 1.75)
# plt.tight_layout()
plt.show()

###
```



Q3 - Part 5 (10%)

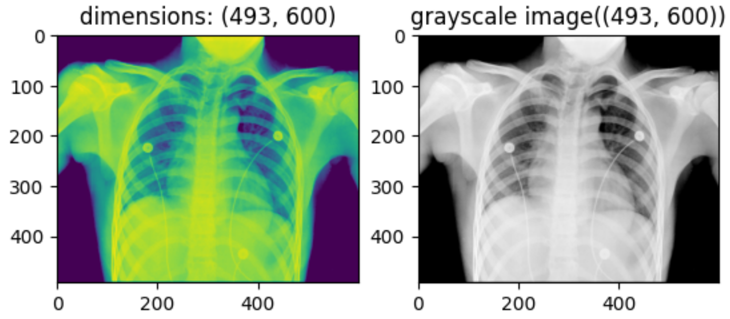
```
#@title Q3 - Part 5 (10%)
###
fig_3d = plt.figure()
circle_3d = fig_3d.add_subplot(121, projection="3d")
noisy_circle_3d = fig_3d.add_subplot(122, projection="3d")

x, y = np.meshgrid(np.arange(circle.shape[0]), np.arange(circle.shape[1]))
x_noisy, y_noisy = np.meshgrid(np.arange(noisy_circle.shape[0]), np.arange(noisy_circle.shape[1]))

z = circle.flatten()
z_noisy = noisy_circle.flatten()
# print(x)
# print(y)
# print(z.tolist())
# print(x.shape, y.shape, z.shape, circle.shape)

circle_3d.scatter(x, y, z, c='r', marker='.')
noisy_circle_3d.scatter(x_noisy, y_noisy, z_noisy, c='r', marker=".")
circle_3d.set_title("PURE CIRCLE")
noisy_circle_3d.set_title("NOISY CIRCLE")
plt.suptitle(f"HW0-Surface-{std_num}")
plt.show()
###
```

HW0-Surface-9923087

Question 4: (25%)

Q4 - Part 1 (15%)

```
#@title Q4 - Part 1 (15%)
###
r_image = plt.imread("chest-xray.png")
fig4, axes4 = plt.subplots(1, 2)
axes4[0].imshow(r_image)
axes4[0].set_title(f"dimensions: {r_image.shape}")

my_image = cv2.imread("chest-xray.png")
grayscale_image = cv2.cvtColor(my_image, cv2.COLOR_BGR2GRAY)

axes4[1].imshow(grayscale_image, cmap="gray")
axes4[1].set_title(f"grayscale image({grayscale_image.shape})")

plt.show()
###
```



Q4 - Part 2 (5%)

```
#@title Q4 - Part 2 (5%)
###
print(f"original image data type: {r_image.dtype}")
print(f"grayscale image data type: {grayscale_image.dtype}")
###
```

```
    original image data type: float32
    grayscale image data type: uint8
```

Q4 - Part 3 (10%)

```
#@title Q4 - Part 3 (10%)
###
print(f"original image memory usage: {r_image.nbytes / 10**6}MBs")
print(f"grayscale image memory usage: {grayscale_image.nbytes / 10**6}MBs")
###
```

```
    original image memory usage: 1.1832MBs
    grayscale image memory usage: 0.2958MBs
```

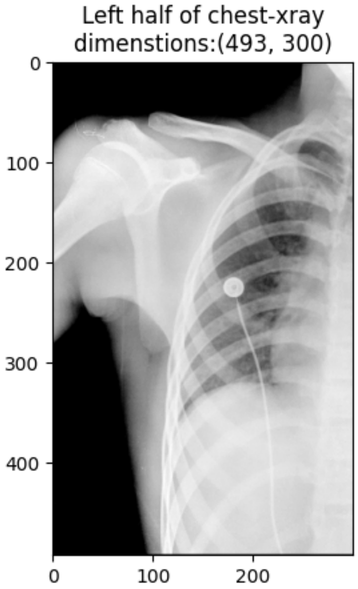
Q4 - Part 3 Explanation:

{از آنجایی که نوع داده هر پیکسل در تصویر اصلی از نوع float32 است و در تصویر خاکستری شده از نوع uint8، است حجم
اشغال شده توسط تصویر سیاه و سفید یک چهارم تصویر اصلی است.}

Q4 - Part 4 (15%)

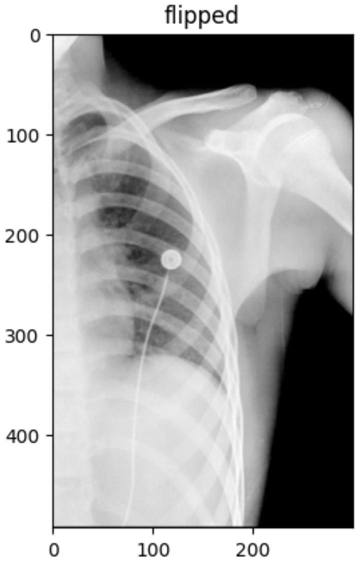
```
#@title Q4 - Part 4 (15%)
###
half = int(grayscale_image.shape[1] / 2)
```

```
left_half_gray = grayscale_image[:, :half]
plt.title(f"Left half of chest-xray\ndimenstions:{left_half_gray.shape}")
plt.imshow(left_half_gray, cmap="gray")
plt.show()
###
```



Q4 - Part 5 (10%)

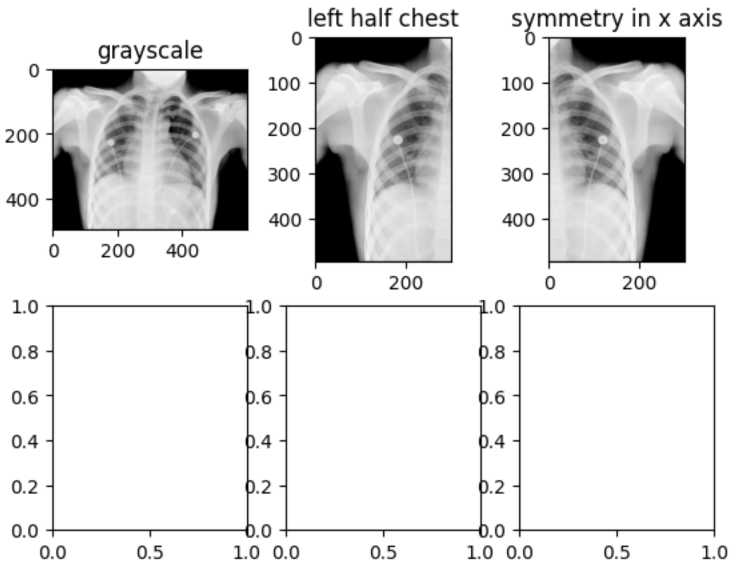
```
#@title Q4 - Part 5 (10%)
###
flipped = cv2.flip(left_half_gray, 1)
plt.imshow(flipped, cmap="gray")
plt.title("flipped")
plt.show()
###
```



Q4 - Part 6 (15%)

```
#@title Q4 - Part 6 (15%)
###
fig46, ax46 = plt.subplots(2, 3)
ax46[0, 0].imshow(grayscale_image, cmap="gray", vmin=0, vmax=255)
ax46[0, 0].set_title("grayscale")

ax46[0, 1].imshow(left_half_gray, cmap="gray", vmin=0, vmax=255)
ax46[0, 1].set_title("left half chest")
```

```
ax46[0, 2].imshow(flipped, cmap="gray")
ax46[0, 2].set_title("symmetry in x axis")
plt.show()
###
```



Q4 - Part 7 (30%)

```
#@title Q4 - Part 7 (30%)
###
fig47, ax47 = plt.subplots(2, 3, sharey="row")
ax47[0, 0].imshow(grayscale_image, cmap="gray", vmin=0, vmax=255)
ax47[0, 0].set_title("grayscale")

ax47[0, 1].imshow(left_half_gray, cmap="gray", vmin=0, vmax=255)
ax47[0, 1].set_title("left half chest")

ax47[0, 2].imshow(flipped, cmap="gray")
ax47[0, 2].set_title("symmetry in x axis")

color_range = 256
steps = 4
num_bins = color_range // steps
grayscale_image_hist, bins0 = np.histogram(grayscale_image, bins=num_bins, range=(0, color_range))
left_half_gray_hist, bins1 = np.histogram(left_half_gray, bins=num_bins, range=(0, color_range))
flipped_hist, bins2 = np.histogram(flipped, bins=num_bins, range=(0, color_range))
bar_width = 0.6
ax47[1, 0].bar(bins0[:-1], grayscale_image_hist, width=bar_width)
ax47[1, 1].bar(bins1[:-1], left_half_gray_hist, width=bar_width)
ax47[1, 2].bar(bins2[:-1], flipped_hist, width=bar_width)

print(sum(grayscale_image_hist))
plt.show()
###
```

```
295800
```

Q4 - Part 7 Explanation:

{از آنجایی که بخش زیادی از تصویر بدون رنگ یا مشکی است، بنابراین در نمودار هیستوگرام بیشترین سهم را intensity صفر دارد که تعداد پیکسل های با این شدت به بیش از 60هزار میرسد. در مقایسه نمودار هیستوگرام سه عکس میتوان گفت که flip کردن تاثیر در intensity ندارد بنابراین عکس دوم و سوم نمودار هیستوگرام مشابه دارند اما چون هر دوی این عکس ها نصف تعداد پیکسل های عکس اول را دارند، نمودار }

** توجه داشتید باشید کولب خود را بصورت viewer به اشتراک بگذارید. (بالا سمت راست دکمه Share قرار دارد که موقع فشردن آن یک صفحه باز می‌شود و گزینه‌ای که بصورت پیشفرض نوشته restricted را تغییر دهید)

** حتماً توجه کنید که در هنگام تحویل و آخرین ویرایش روی کد خود، خروجی همه‌ی بلاک‌ها، خروجی درست و نهایی همان بلاک باشد. (بطور مثال ممکن است تغییری در کد بدهید و رویت کنید که خروجی درست نیست و فقط تغییرتان را undo کرده ولی دوباره اجرایش نکنید و خروجی سلول همان خروجی دوم که جواب اشتباهی بود بماند).

** نکته مهم: لطفاً بعد از تحویل تمرین دیگر کد گوگل کولب خود را باز نکنید و حتی کوچکترین تغییری (حتی در حد ایجاد یک space) در آن ندهید.(چرا که تاریخ آخرین ویرایش آن تغییر کرده و برای مصحح محترم قابل احراز نیست که این کد شما چه زمانی نوشته شده است (از نظر موعد قابل پذیرش برای تحویل) و بخش کدنویسی آن تمرین از شما پذیرفته نخواهد شد)