نام و شماره دانشجویی : فاطمه توکلی، 400131016

نام درس : پردازش رقمی تصویر

نام استاد : دکتر رحمتی

تمرین شماره 01

.1

تمامی پیکیج های مورد استفاده در این سئوال:

```
import sys
from PIL import Image
from math import log10, sqrt
import cv2
import numpy as np
import math
import imageio
```

a. متد خواسته شده به صورت زیر پیاده سازی می شود:

```
def immirror(img) :
    width  = img.size[0]
    height = img.size[1]
    for y in range(height):
        for x in range(width//2):
            left = img.getpixel((x, y))
            right = img.getpixel((width - 1 - x, y))
            img.putpixel((width - 1 - x, y), left)
            img.putpixel((x, y), right)
    return img
```

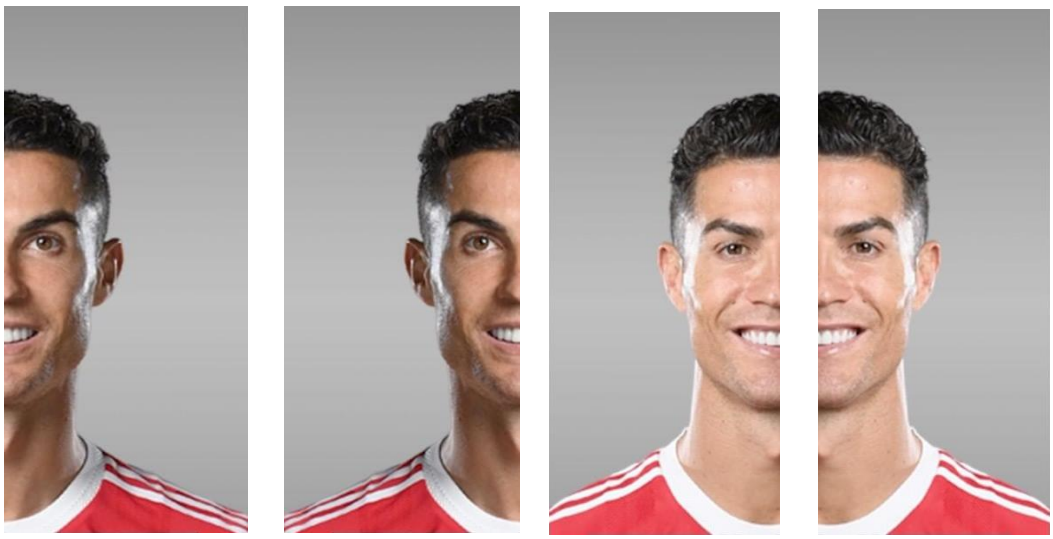
نتیجه :

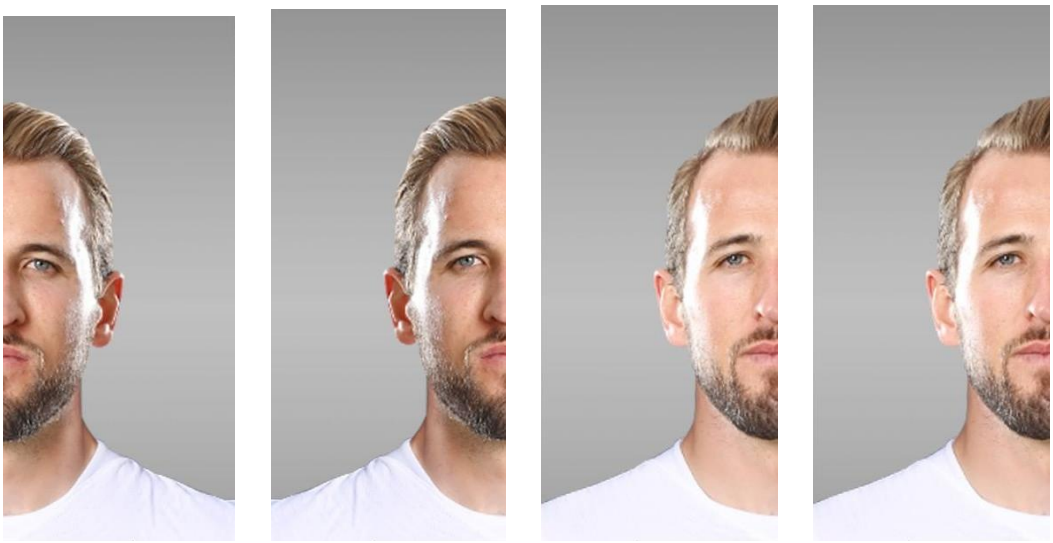
b. ابتدا تصویر به دو قسمت تبدیل کرده و سپس هرکدام از قسمت راست و چپ صورت را به عنوان ورودی به imgmirror میدهیم تا همه حالات گفته شده را داشته باشیم:

```
import imageio
img = imageio.imread("cristiano_ronaldo.png")
height, width, channel = img.shape
width_cutoff = width // 2
R = img[:, :width_cutoff]
L = img[:, width_cutoff:]
imageio.imwrite("cristiano_ronaldor.png", R)
imageio.imwrite("cristiano_ronaldol.png", L)
```

```
image1 = Image.open("cristiano_ronaldor.png")
immirror(image1)
image1.save("cristiano_ronaldor_i.png")
image2 = Image.open("cristiano_ronaldol.png")
immirror(image2)
image2.save("cristiano_ronaldol_i.png")
```

نتیجه:





نتیجه بقیه عکس ها در پوشه هر عکس در فایل zip می باشد.

c.عکس ها ترکیب کرده و به متد های پیاده شده به عنوان ورودی می دهیم:

```
def impsnr(img, ref):
    mse = np.mean((img - ref) ** 2)
    if(mse == 0):
        return 100
    max_pixel = 255.0
    psnr = 20 * log10(max_pixel / sqrt(mse))
    return psnr

def main():
    original = cv2.imread("original_image.png")
    compressed = cv2.imread("compressed_image.png", 1)
    value = PSNR(original, compressed)
    print(f"PSNR value is {value} dB")

if __name__ == "__main__":
    main()
```

```
def ssim(img1, img2):
    C1 = (0.01 * 255)**2
    C2 = (0.03 * 255)**2
    img1 = img1.astype(np.float64)
    img2 = img2.astype(np.float64)
    kernel = cv2.getGaussianKernel(11, 1.5)
    window = np.outer(kernel, kernel.transpose())
    mu1 = cv2.filter2D(img1, -1, window)[5:-5, 5:-5]  # valid
    mu2 = cv2.filter2D(img2, -1, window)[5:-5, 5:-5]
    mu1_sq = mu1**2
    mu2_sq = mu2**2
    mu1_mu2 = mu1 * mu2
    sigma1_sq = cv2.filter2D(img1**2, -1, window)[5:-5, 5:-5] - mu1_sq
    sigma2_sq = cv2.filter2D(img2**2, -1, window)[5:-5, 5:-5] - mu2_sq
    sigma12 = cv2.filter2D(img1 * img2, -1, window)[5:-5, 5:-5] - mu1_mu2
    ssim_map = ((2 * mu1_mu2 + C1) * (2 * sigma12 + C2)) / ((mu1_sq + mu2_sq + C1) *
                                                            (sigma1_sq + sigma2_sq + C2))
    return ssim_map.mean()


def  imssim(img, ref):
    if not img.shape == ref.shape:
        raise ValueError('Input images must have the same dimensions.')
    if img.ndim == 2:
        return ssim(img, ref)
    elif img.ndim == 3:
        if img.shape[2] == 3:
            ssims = []
            for i in range(3):
                ssims.append(ssim(img, ref))
            return np.array(ssims).mean()
        elif img.shape[2] == 1:
            return ssim(np.squeeze(img), np.squeeze(ref))
    else:
        raise ValueError('Wrong input image dimensions.')
```

```
Ronaldo :
PSNR value  for Right, Right_inverse  is 34.24794001503599 dB
PSNR value  for left, left_inverse is 34.34185528762685 dB
PSNR value  for Right_inverse, Left_inverse is 31.537565997912466 dB
```

```
ronaldo
immsim value  for Right, Right_inverse  is 0.8537958069619882 dB
immsim value  for Left, Left_inverse is 0.8537802394765147 dB
immsim value  for Right_inverse, Left_inverse is 0.711103640085596
```

```
erling_haaland:
PSNR value  for Right, Right_inverse  is 34.24794001503599 dB
PSNR value  for left, left_inverse is 34.34185528762685 dB
PSNR value  for Right_inverse, Left_inverse is 31.537565997912466 dB

erling_haaland:
immsim value  for Right, Right_inverse  is 0.8537958069619882 dB
immsim value  for Left, Left_inverse is 0.8537802394765147 dB
immsim value  for Right_inverse, Left_inverse is 0.711103640085596


harry_kane:
PSNR value  for Right, Right_inverse  is 31.58824845013133 dB
PSNR value  for left, left_inverse is 31.72275787486541 dB
PSNR value  for Right_inverse, Left_inverse is 31.60146064469846 dB

harry_kane:
immsim value  for Right, Right_inverse  is 0.7153548125267667 dB
immsim value  for Left, Left_inverse is 0.7205117371371886 dB
immsim value  for Right_inverse, Left_inverse is 0.7074963401423734


jose_mourinho:
PSNR value  for Right, Right_inverse  is 31.64696365479426 dB
PSNR value  for left, left_inverse is 31.464385063201277 dB
PSNR value  for Right_inverse, Left_inverse is 31.461975635530717 dB

jose_mourinho:
immsim value  for Right, Right_inverse  is 0.6651054354758014 dB
immsim value  for Left, Left_inverse is 0.6846706417261671 dB
immsim value  for Right_inverse, Left_inverse is 0.6545196105565081


jurgen_klopp:
PSNR value  for Right, Right_inverse  is 31.51114473777782 dB
PSNR value  for left, left_inverse is 31.135120283960077 dB
PSNR value  for Right_inverse, Left_inverse is 31.28911790269698 dB

jurgen_klopp:
immsim value  for Right, Right_inverse  is 0.663376084623444 dB
immsim value  for Left, Left_inverse is 0.6492603378218109 dB
immsim value  for Right_inverse, Left_inverse is 0.6453757503365726


kevin_de_bruyne:
PSNR value  for Right, Right_inverse  is 31.444254092247157 dB
PSNR value  for left, left_inverse is 31.36784105386237 dB
PSNR value  for Right_inverse, Left_inverse is 31.350258575493925 dB

kevin_de_bruyne:
immsim value  for Right, Right_inverse  is 0.7109926475834144 dB
immsim value  for Left, Left_inverse is 0.7008134326075431 dB
immsim value  for Right_inverse, Left_inverse is 0.7003548111190224


lionel_messi:
PSNR value  for Right, Right_inverse  is 31.63941788130814 dB
PSNR value  for left, left_inverse is 31.454469217608477 dB
PSNR value  for Right_inverse, Left_inverse is 31.483287120751683 dB

lionel_messi:
immsim value  for Right, Right_inverse  is 0.7199680829932156 dB
immsim value  for Left, Left_inverse is 0.7056535947318944 dB
immsim value  for Right_inverse, Left_inverse is 0.7045162571027047


ngolo_kante:
PSNR value  for Right, Right_inverse  is 31.293312932820303 dB
PSNR value  for left, left_inverse is 31.49364979332408 dB
PSNR value  for Right_inverse, Left_inverse is 31.434734681284855 dB

ngolo_kante:
immsim value  for Right, Right_inverse  is 0.6560670190418205 dB
immsim value  for Left, Left_inverse is 0.6763578431512025 dB
immsim value  for Right_inverse, Left_inverse is 0.6586679689903823
```

```
pep_guardiola:
PSNR value  for Right, Right_inverse  is 31.515931261656817 dB
PSNR value  for left, left_inverse is 31.431193755917064 dB
PSNR value  for Right_inverse, Left_inverse is 31.53356364843933 dB
```

```
pep_guardiola:
immsim value  for Right, Right_inverse  is 0.6888398758672264 dB
immsim value  for Left, Left_inverse is 0.6825249353588041 dB
immsim value  for Right_inverse, Left_inverse is 0.6845571030618441
```

```
zlatan_ibrahimovic:
PSNR value  for Right, Right_inverse  is 31.515931261656817 dB
PSNR value  for left, left_inverse is 31.431193755917064 dB
PSNR value  for Right_inverse, Left_inverse is 31.53356364843933 dB
```

```
zlatan_ibrahimovic:
immsim value  for Right, Right_inverse  is 0.6888398758672264 dB
immsim value  for Left, Left_inverse is 0.6825249353588041 dB
immsim value  for Right_inverse, Left_inverse is 0.6845571030618441
```

.2

تمامی پکیج های مورد استفاده در این سئوال:

```
: import pandas as pd
  import numpy as np
  from PIL import Image
  import cv2
  from matplotlib import pyplot as plt
```

.a

```
img1 = Image.open('P2//grayscale1.png')
img2 = Image.open('P2//grayscale2.png')
```

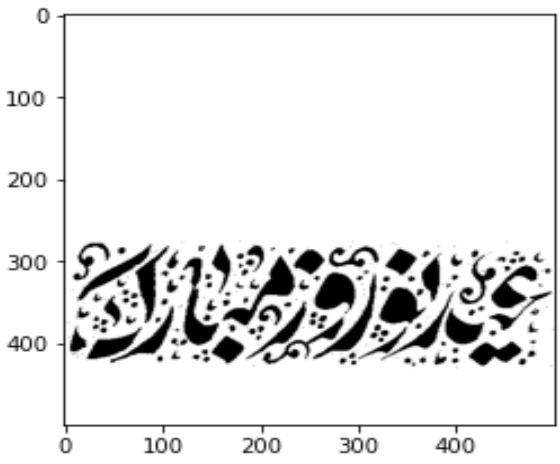
```
img1= np.array(img1)
img1.shape
```

```
(500, 500)
```

```
img2= np.array(img2)
img2.shape
```

```
(500, 500)
```

```
for i in range(0,len(img2)):
    new = np.zeros(img1.shape)
    new[500-i:] = img2[0:i]
    new[0:500-i] = img2[i:500]
    z = np.absolute(new - img1)
    print('stage :' + str(i))
    plt.figure()
    plt.imshow(z,cmap ='gray',vmin=0,vmax=1)
    plt.show()
```



b. عکس msg را به باینری تبدیل کرده و سپس به وسیله هر پیکسل عکس img را کد کرده و تغییرات جزئی در مقدار پیکسل ها میدهیم:

```
msg = cv2.imread('P2//msg.png', 2)
msg = cv2.resize(msg,(1600,1200))

ret, bin_img = cv2.threshold(msg, 127, 255, cv2.THRESH_BINARY)

# converting to its binary form
bw = cv2.threshold(msg, 127, 255, cv2.THRESH_BINARY)
cv2.imshow("Binary", bin_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
bin_img
```

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

```
main_img = cv2.imread('P2//img.png')
main_img = cv2.cvtColor(main_img, cv2.COLOR_BGR2RGB)
cv2.imshow("img",main_img )
cv2.waitKey(0)
cv2.destroyAllWindows()
```

کد کردن عکس img و تغییر مقدار پیکسل ها:

```
img_mid = np.absolute(main_img[:,:,0] - main_img[:,:,2])
img_mid.shape
```

```
(1200, 1600)
```

```
for i in range(0,1200):
    for j in range(0,1600):
        if bin_img[i,j]:
            if not (img_mid[i,j] % 2):
                if main_img[i,j,0] <255:
                    main_img[i,j,0] +=1
                else:
                    main_img[i,j,0] -= 1
        elif (img_mid[i,j] % 2):
            if main_img[i,j,0] < 255:
                main_img[i,j,0] += 1
            else:
                main_img[i,j,0] -= 1
```

```
plt.imshow(main_img)
```

```
<matplotlib.image.AxesImage at 0x184094ea610>
```

c. مانند شبیه کد داده شده عمل میکنیم:

```
img_mid = np.absolute(main_img[:,:,0] - main_img[:,:,2])
img_mid.shape
```

```
(1200, 1600)
```

```
img_mid = np.absolute(main_img[:,:,0] - main_img[:,:,2])
img_mid.shape
```

```
(1200, 1600)
```

```
new = np.zeros(bin_img.shape)
for i in range(0,1200):
    for j in range(0,1600):
        if img_mid[i,j] % 2 != 0:
            new[i,j] = 1
        else:
            new[i,j] = 0
plt.imshow(new , cmap ='gray',vmin=0,vmax=1)
```

```
<matplotlib.image.AxesImage at 0x1feefe64670>
```

3.

تمامی پکیج های مورد استفاده در این سئوال:

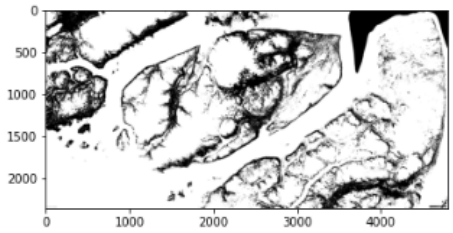
```
import numpy as np
import cv2
import matplotlib.pyplot as plt
import glob
```

a. با استفاده از threshold و ازمون خطا مقداری مناسب برای اینکه بتوانیم تفاوت یخ ها به دست بیاوریم محاسبه میکنیم:

```
mylius_erichsen_land_2000 = cv2.imread('./inputs/P3/mylius_erichsen_land_2000.jpg', 0)
mylius_erichsen_land_2020 = cv2.imread('./inputs/P3/mylius_erichsen_land_2020.jpg', 0)
```

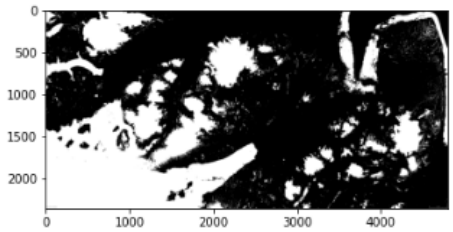
```
_, mylius_erichsen_land_2000 = cv2.threshold(mylius_erichsen_land_2000, 0, 1, cv2.THRESH_OTSU)
plt.imshow(mylius_erichsen_land_2000, cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x1d5d8ac7850>
```




```
_, mylius_erichsen_land_2020 = cv2.threshold(mylius_erichsen_land_2020, 0, 1, cv2.THRESH_OTSU)
plt.imshow(mylius_erichsen_land_2020, cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x1d5d96f0730>
```



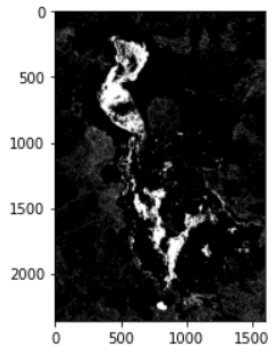

```
snows, melted = 0, 0
for i in range(mylius_erichsen_land_2000.shape[0]):
    for j in range(mylius_erichsen_land_2000.shape[1]):
        if mylius_erichsen_land_2000[i, j] == 1:
            snows += 1
            if mylius_erichsen_land_2020[i, j] == 0:
                melted += 1

print('Melted Percentage: {} %'.format(round(np.multiply(100, np.divide(melted, snows)), 2)))
```

```
Melted Percentage: 60.27 %
```

b.

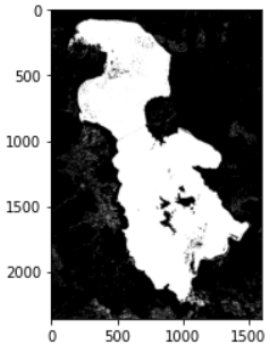
```
lake_urmia_2018 = lake_urmia_2018[:, 450: -550]
lake_urmia_2018 = cv2.inRange(lake_urmia_2018, (50, 108, 50), (120, 255, 255))
_, lake_urmia_2018 = cv2.threshold(lake_urmia_2018, 0, 1, cv2.THRESH_OTSU)
plt.imshow(lake_urmia_2018, cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x1d5de9f8490>
```



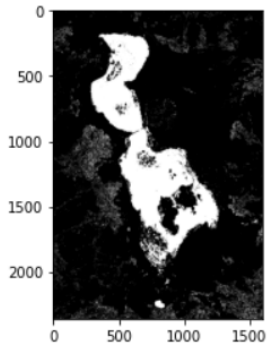
```
lake_urmia_2000 = lake_urmia_2000[:, 450: -550]
lake_urmia_2000 = cv2.inRange(lake_urmia_2000, (0, 60, 90), (100, 255, 255))
_, lake_urmia_2000 = cv2.threshold(lake_urmia_2000, 0, 1, cv2.THRESH_OTSU)
plt.imshow(lake_urmia_2000, cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x1d5de98dd80>
```




```
lake_urmia_2020 = lake_urmia_2020[:, 450: -550]
lake_urmia_2020 = cv2.inRange(lake_urmia_2020, (50, 100, 50), (110, 255, 255))
_, lake_urmia_2020 = cv2.threshold(lake_urmia_2020, 0, 1, cv2.THRESH_OTSU)
plt.imshow(lake_urmia_2020, cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x1d5dea470a0>
```




```
water_2000, water_2018, water_2020 = 0, 0, 0
for i in range(lake_urmia_2000.shape[0]):
    for j in range(lake_urmia_2000.shape[1]):
        if lake_urmia_2000[i, j] == 1:
            water_2000 += 1
        if lake_urmia_2018[i, j] == 1:
            water_2018 += 1
        if lake_urmia_2020[i, j] == 1:
            water_2020 += 1
```

```
scale = 20   # km
pixels = 314 # pixels for 10 km
```

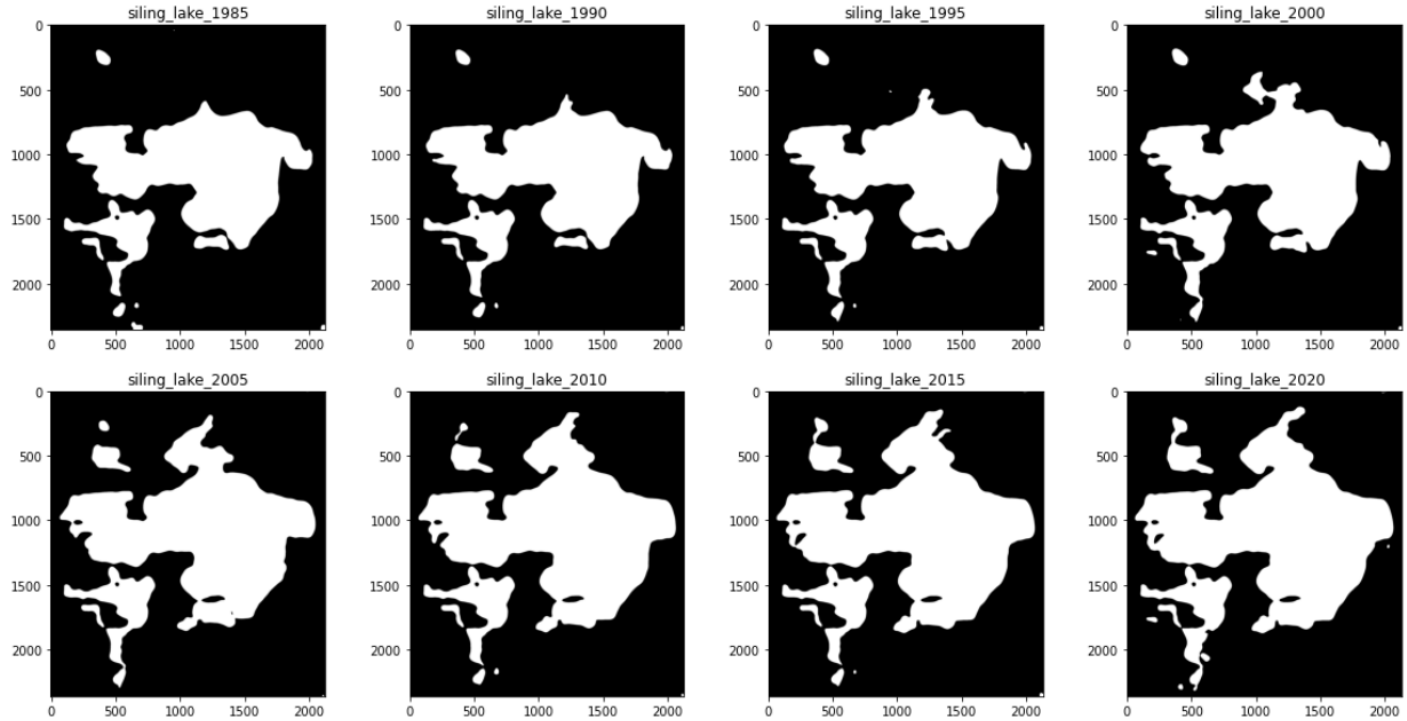
```
area_2000 = calculate_area(water_2000, scale, pixels)
volume_2000 = np.multiply(area_2000, 0.0028)
```

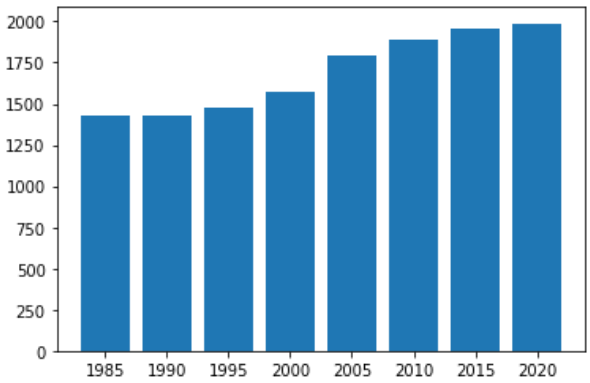
```
area_2018 = calculate_area(water_2018, scale, pixels)
volume_2018 = np.multiply(area_2018, 0.0006)
```

```
print('Volume of water evaporated between 2000 and 2018: \n{} km3'.format(round(np.subtract(volume_2000, volume_2018), 2)))
```

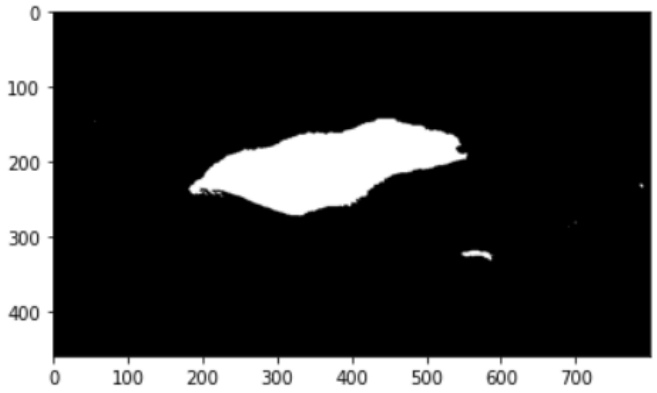
```
Volume of water evaporated between 2000 and 2018:
13.7 km3
```

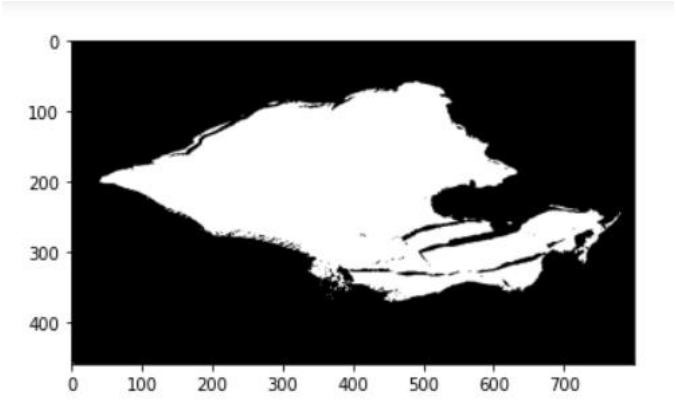
c.


siling_lake_1985
siling_lake_1990
siling_lake_1995
siling_lake_2000
siling_lake_2005
siling_lake_2010
siling_lake_2015
siling_lake_2020


2000
1750
1500
1250
1000
750
500
250
0
1985
1990
1995
2000
2005
2010
2015
2020

D.


0
100
200
300
400
0
100
200
300
400
500
600
700


0
100
200
300
400
0
100
200
300
400
500
600
700

```
glacier, melted = 0, 0
for i in range(furtwangler_glacier_2003.shape[0]):
    for j in range(furtwangler_glacier_2003.shape[1]):
        if furtwangler_glacier_2003[i, j] == 1:
            glacier += 1
            if furtwangler_glacier_2017[i, j] == 0:
                melted += 1

print('Melted Percentage: {} %'.format(round(np.multiply(100, np.divide(melted, glacier)), 2)))
```

```
Melted Percentage: 75.59 %
```

.4

تمامی پکیج های مورد استفاده در این سئوال:

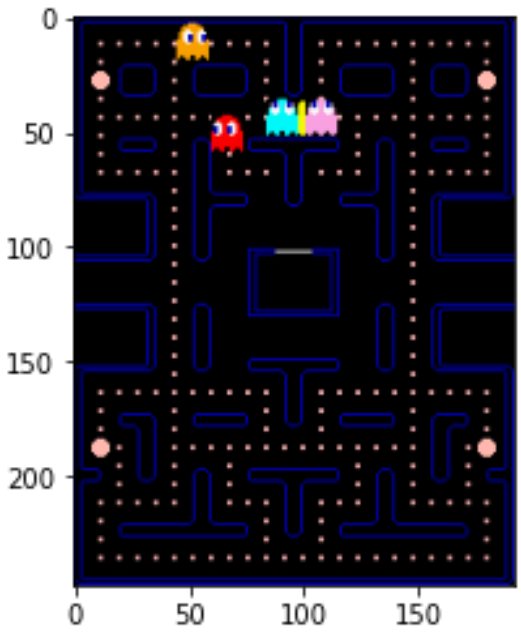
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import skimage.color
import random
```

a.

```
maze = cv2.imread('inputs/P4/maze.png')[:, :, ::-1]
fig, ax = plt.subplots()
plt.imshow(maze)
plt.show()
print(maze.shape)

dot = cv2.imread('inputs/P4/dot.png')
print(dot.shape)


gray_maze = skimage.color.rgb2gray(maze)


t_start = 0.72
t_stop = 0.77
binary_start_maze = (gray_maze > t_start)
binary_stop_maze = (gray_maze < t_stop)
binary_maze = np.logical_and(binary_start_maze, binary_stop_maze)

pacman_closed = cv2.imread('inputs/P4/pacman_closed.png')[:, :, ::-1]
move_type = ["move1_down.png",
             "move1_left.png",
             "move1_right.png",
             "move1_up.png",
             "move2_left.png",
             "move2_right.png",
             "move2_up.png",
             "move2_down.png"
             ]
ghost_names = ["blinky_", "clyde_", "inky_", "pinky_"]
```

```
def init_game(maze_img,item_imgs):
    selected_ghost_names = []
    for i in range(len(ghost_names)):
        selected_ghost_names.append(ghost_names[i] + move_type[random.randint(0, len(move_type) - 1)])

    # select random points
    x_pos = []
    y_pos = []
    for i in range(0, 5):
        x_idx, y_idx = np.where(binary_maze == True)  # list of all the indices with pixel value True
        rand_idx = np.random.choice(x_idx)   # randomly choose any element in the x_idx list
        x_pos.append(x_idx[rand_idx])
        y_pos.append(y_idx[rand_idx])
        # delete selected points to not select them in next iter
        binary_maze[x_idx[rand_idx]: x_idx[rand_idx]+8, y_idx[rand_idx]: y_idx[rand_idx] + 8] = False
        binary_maze[x_idx[rand_idx] - 8: x_idx[rand_idx], y_idx[rand_idx]: y_idx[rand_idx] + 8] = False
        binary_maze[x_idx[rand_idx]: x_idx[rand_idx] + 8, y_idx[rand_idx] - 8: y_idx[rand_idx]] = False
        binary_maze[x_idx[rand_idx] - 8: x_idx[rand_idx], y_idx[rand_idx] - 8: y_idx[rand_idx]] = False


    x, y, z = pacman_closed.shape
    x_lobe = int(x/2)
    y_lobe = int(y/2)
    maze[x_pos[0] - x_lobe: x_pos[0] + x - x_lobe, y_pos[0] - y_lobe: y_pos[0] + y - y_lobe, :] = pacman_closed[:, :, :]

    fig, ax = plt.subplots()
    plt.imshow(maze)
    plt.show()

    # set ghosts in selected points
    for i in range(len(selected_ghost_names)):
        ghost = cv2.imread('inputs/P4/' + selected_ghost_names[i])[:, :, ::-1]
        x, y, z = ghost.shape
        x_lobe = int(x / 2)
        y_lobe = int(y / 2)
        maze[x_pos[i+1] - x_lobe: x_pos[i+1] + x - x_lobe, y_pos[i+1] - y_lobe: y_pos[i+1] + y - y_lobe, :] = \
            ghost[:, :, :]

    fig, ax = plt.subplots()
    plt.imshow(maze)
    plt.show()

game_img = init_game(binary_maze,ghost_names)
```

خروجی:



.5

a.

بازتاب نور از شی روی retina تصویر میشود و شامل دو نوع است :Cone که حساس به رنگ است و جزئیات ار درک میکند و Rod که سطوح روشنایی را درک میکند و بینایی خود و درک خود از محیط با شرایط مختلف تطبیق دهد.

b.

با دوربین 50MP ابعادش 8700*5800 است و عکس با دوربین 12MP دارای ابعاد 3000*4000 است و f نشان دهنده میزان نوری است که لنز میتواند عبور دهد در نتیجه حجم عکس های Samsung بیشتر است

c.

تبدیل RGB به grayscale به اینصورت است که از سه کانال قرمز، ابی و سبز میانگین میگیریم و تصویر در مقیاس خاکستری مورد نظر به دست می آوریم ، عکس اینکار در صورتی که نسبت ترکیب هرکدام از کانال هار درهر تصویر بدانیم ممکن است ولی به صورت کلی تمامی کانال ها را به صورت یکسان در می آورد که ترکیب درستی از عکس واقعی نیست.

d.

resampling تکنیک دستکاری یک تصویر و تبدیل آن به شکل دیگری است. این دستکاری می تواند به دلایل مختلفی از جمله تغییر وضوح، تغییر جهت، چرخش، تغییر نقاط نمونه برداری و غیره می باشد.

Sub_sampling روشی است که با انتخاب زیر مجموعه ای از داده های اصلی، اندازه داده ها را کاهش می دهد. زیرمجموعه با انتخاب یک پارامتر n مشخص می شود و مشخص می کند که هر nامین نقطه داده باید استخراج شود. هدف این است که یک تصویر بگیرید و ابعاد آن را کاهش دهید تا در نتیجه تصویر کوچکتری داشته باشید.