```python
# Wybranie obrazka z folderu grafiki
https://drive.google.com/drive/folders/1_F2ncQA8--C-jghM7BjmsrUx1hTjucSL?usp=sharing
# - pobrać i zapisać do folderu projektu pod nazwą 'image.jpg'
import cv2
from PIL import Image
import numpy as np

# 1. Wczytanie grafiki z pliku i wyświetlenie obrazu

# a) funkcja do wyświetlania obrazu (opencv)
def show_image(img):
    cv2.imshow("image", img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

# b) funkcja, która wczyta obraz z pliku i go wyświetli (opencv)
def read_image_cv(path):
    img = cv2.imread(path, cv2.IMREAD_COLOR)
    print(img)
    print(img.shape)
    print(type(img))
    show_image(img)
    return img

# c) funkcja, która wczyta obraz z pliku i go wyświetli (pillow)
def read_image_PIL(path):
    im = Image.open(path)
    try:
        print(im)
    except:
        print(type(im))
    im.show()
    return im

# • Porównanie
print("Biblioteka OpenCV:")  # kolory w BGR
image = read_image_cv("image.jpg")

print("Biblioteka Pillow:")
read_image_PIL("image.jpg") # tworzy własny nowy  obiekt


# 2. Flip obrazka (odbicie) [do góry nogami]

# a) algorytm
def reverse_image(img):
    new_img = []
    for row in range(img.shape[0]):
        new_row = []
        for column in range(img.shape[1]):
            new_row.append(img[-1-row][column])
        new_img.append(new_row)
    return np.array(new_img)
```

```python
show_image(reverse_image(image))
# b) to samo z wykorzystaniem wycinków list
def reverse_short(img):
    img_reverse = img[::-1]
    return img_reverse

show_image(reverse_short(image))

# c) wbudowana funkcja OpenCV
show_image(cv2.flip(image,0))



# 3. Zmiana koloruna skalę szarości

# a) algorytm
def gray_scale(img):
    for row in range(img.shape[0]):
        for column in range(img.shape[1]):
            gray = int(sum(img[row][column])/3)
            img[row][column][0] = gray
            img[row][column][1] = gray
            img[row][column][2] = gray
    return np.array(img)

show_image(gray_scale(image))

# b) wbudowana funkcja OpenCV
show_image(cv2.cvtColor(image, cv2.COLOR_BGR2GRAY))


# 4. Filtr sepia
def sepia(img):
    for row in range(img.shape[0]):
        for column in range(img.shape[1]):
            B = img[row][column][0]
            G = img[row][column][1]
            R = img[row][column][2]
            img[row][column][2] = min(255, (0.393*R + 0.769*G + 0.189*B))
            img[row][column][1] = min(255, (0.349*R + 0.686*G + 0.168*B))
            img[row][column][0] = min(255, (0.272*R + 0.534*G + 0.131*B))
    return np.array(img, dtype=np.uint8)

show_image(image)
show_image(sepia(image))
```

```python
# 5. Zadanie samodzielne - change_colors
# Napisz funkcję change_colors, która przyjmuje 4 argumenty:

# • img -obraz w formacie np.array,
# • R_scale - współczynnik zmiany koloru czerwonego piksela,
# • G_scale - współczynnik zmiany koloru zielonego piksela,
# • B_scale - współczynnik zmiany koloru niebieskiego piksela.

# Funkcja powinna dla każdego piksela ustalać nową wartość koloru jako stara wartość
# razy współczynnik dla odpowiedniej barwy oraz zwrócić przefiltrowany obraz.
# Przetestuj funkcję wpisując różne współczynniki jako parametry, co dzieje się z
# obrazem? Jak myślicie funkcja generuje nowy obraz czy modyfikuje stary?

def change_colors(img,R_scale,G_scale,B_scale):
    for row in range(img.shape[0]):
        for column in range(img.shape[1]):
            img[row][column][2] = img[row][column][2] * R_scale
            img[row][column][1] = img[row][column][1] * G_scale
            img[row][column][0] = img[row][column][0] * B_scale
    return np.array(img, dtype= np.uint8)
show_image(change_colors(image,1.4,0,0))

# img2 = img.copy()
```