Lab 3-4 ISI2 174748 Michał Broda

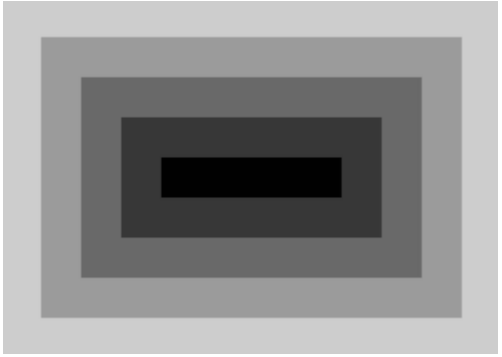
Zadanie 1

Funkcja rysuj ramki szare przyjmuje wartości:

- w = szerokość
- h = wysokość
- grub = grubość ramek
- zmiana_koloru = przeskok koloru w każdej ramce

```
def rysuj_ramki_szare(w, h, grub, zmiana_koloru):
    t = (h, w, 3)
    kolor = [255, 255, 255]
    tab = np.ones(t, dtype=np.uint8)
    ile = min(w, h) // (2 * grub)
    for i in range(ile):
        kolor[0] = kolor[0] - zmiana_koloru
        kolor[1] = kolor[1] - zmiana_koloru
        kolor[2] = kolor[2] - zmiana_koloru
        # wymiary ramki
        x0, y0 = i * grub, i * grub
        x1, y1 = w - i * grub, h - i * grub
        for y in range(y0, y1):
            for x in range(x0, x1):
                if x - x0 < grub or x1 - x <= grub or y - y0 < grub
or y1 - y <= grub:
                    tab[y, x] = kolor

    return Image.fromarray(tab)

rysuj_ramki_szare(250, 180, 20, 50).show()
rysuj_ramki_szare(250, 180, 20, 50).save("rysuj_ramki_szare.bmp")
```

Funkcja utworzyła dany obraz:

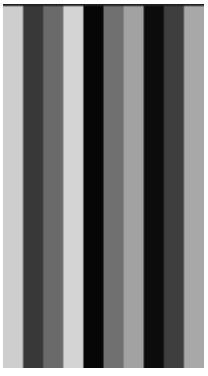


Funkcja rysuj pasy pionowe szare przyjmuje wartości:

- w = szerokość
- h = wysokość
- grub = grubość pasów
- zmiana_koloru = przeskok koloru w każdym pasie

```
def rysuj_pasy_pionowe_szare(w, h, grub, zmiana_koloru):
    t = (h, w,3)
    tab = np.ones(t, dtype=np.uint8)
    ile = int(w/grub)
    kolor1 = [50, 50, 50]
    kolor2 = [150, 150, 150]
    for k in range(ile):
        kolor = kolor1 if k % 2 == 0 else kolor2
        for g in range(ile):
            i = k * grub + g
            for j in range(h):
                tab[j,i] = kolor
        zmien_kolor(kolor1, zmiana_koloru)
        zmien_kolor(kolor2, zmiana_koloru)
    tab = tab * 255
    return Image.fromarray(tab)


rysuj_pasy_pionowe_szare(100, 180, 10, 50).show()
```

Funkcja utworzyłą obraz:



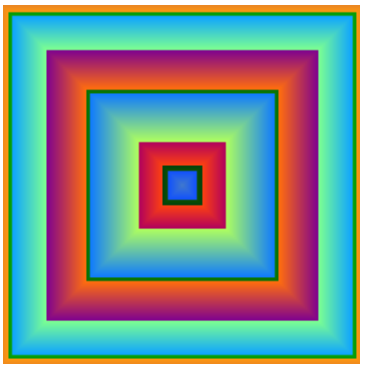
Zadanie 2

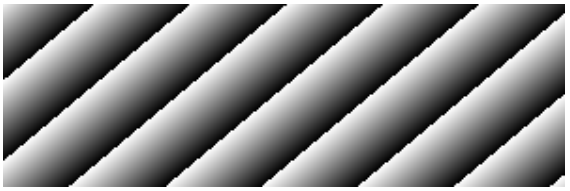
Funkcja negatyw przyjmuje jedną wartość

- obraz = obraz który chcemy zmienić

```
def negatyw(obraz):
    tab = np.asarray(obraz)
    if len(tab.shape) == 2:
        h, w = tab.shape
    elif len(tab.shape) == 3:
        h, w, c = tab.shape
    tab_neg = tab.copy()
    if obraz.mode == "1":
        for i in range(h):
            for j in range(w):
                tab_neg[i, j] = 1 - tab[i, j]
        return Image.fromarray(tab_neg)
    elif obraz.mode == "L" or obraz.mode == "RGB":
        for i in range(h):
            for j in range(w):
                tab_neg[i, j] = 255 - tab[i, j]
        return Image.fromarray(tab_neg)
    else:
        return "Zły format obrazu"


negatyw(gwiazdka).show()
negatyw(rysuj_ramki_kolorowe(200, [20,120,220], 6, 5, -6)).show()
negatyw(rysuj_po_skosie_szare(100, 300, 6, 5)).show()
```

Funkcja tworzy obrazy:





Zadanie 3

Funkcja koloruj w paski przyjmuje wartości:

- obraz = obraz do kolorowania
- grub = grubość pasków
- zmiana_kolor = przeskok koloru w każdym pasku

```
def koloruj_w_paski(obraz, grub, zmiana_kolor):
    t_obraz = np.asarray(obraz)
    h, w = t_obraz.shape
    t =(h, w, 3)
    kolor = [120, 240, 50]
    tab = np.ones(t, dtype=np.uint8)
    ile = int(h/ grub)
    for k in range(ile):
        for g in range(grub):
            i = k * grub + g
            for j in range(w):
                if t_obraz[i, j] == False:
                    tab[i, j] = kolor
                else:
                    tab[i, j] = [255, 255, 255]
        zmien_kolor(kolor, zmiana_kolor)
    return Image.fromarray(tab)

koloruj_w_paski(inicjaly, 10, 50).show()
koloruj_w_paski(inicjaly, 10, 50).save("koloruj_paski.png")
koloruj_w_paski(inicjaly, 10, 50).save("koloruj_paski.jpg")
```

Funkcja tworzy obraz:

Format png:

Format jpg:

Obrazy różnią się od siebie dlatego że format PNG używa konwersji bezstratnej a format JPG używa konwersji stratnej by zaoszczędzić na pamięci, format JPG także nie obsługuje kanału alfa w przeciwieństwie do PNG

Zadanie 4

Wartości w typie uint8 zawijają się zgodnie z zasadą arytmetyki modulo 256:

Dla wartości:

a) 328 uzyskamy 72
b) -24 uzyskamy 232

Zadanie 5

Korzystając z funkcji rysuj pasy pionowe szare 3 razy otrzymujemy kanały R, G, B i używając funkcji rysuj pasy kolorowe tworzymy obraz

```
def rysuj_pasy_pionowe_szare(w, h, grub, kolor_tlo, kolor_pas):
    t = (h, w)
    tab = np.ones(t, dtype=np.uint8)
    ile = int(w/grub)
    tab[:] = kolor_pas
    for k in range(ile):
        for g in range(grub):
            i = k * grub + g
            for j in range(h):
                if k % 2 == 0:
                    tab[j, i] = kolor_tlo
    return tab

red = rysuj_pasy_pionowe_szare(300, 200, 10, 150, 255)
green = rysuj_pasy_pionowe_szare(300, 200, 18, 150, 255)
blue = rysuj_pasy_pionowe_szare(300, 200, 26, 150, 255)

obrazek = np.zeros((200, 300, 3), dtype=np.uint8)


def rysuj_pasy_kolorowe():
    for i in range(300):
        for j in range(200):
            obrazek[j, i, 0] = (obrazek[j, i, 0] + red[j, i]) % 255
            obrazek[j, i, 1] = (obrazek[j, i, 1] + green[j, i]) %
255
            obrazek[j, i, 2] = (obrazek[j, i, 2] + blue[j, i]) %
255
    return obrazek
```

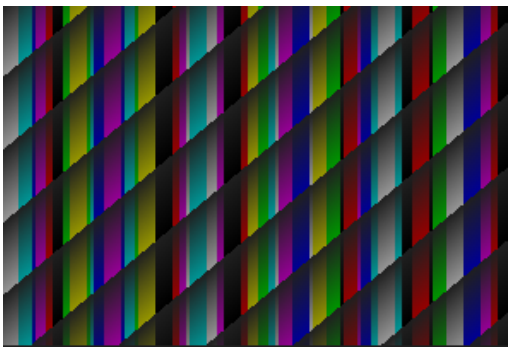
```
Image.fromarray(rysuj_pasy_kolorowe()).show()
Image.fromarray(rysuj_pasy_kolorowe()).save("obraz6.png")
```
Funkcja tworzy następujący obraz:

Zadanie 6

Z obrazu z zadania 5 i z tablicy kanału alfa otrzymanej z funkcji rysuj po skosie szare tworzymy obraz z trybie RGBA

```
def rysuj_po_skosie_szare(h, w, a, b):
    t = (h, w)
    tab = np.zeros(t, dtype=np.uint8)
    for i in range(h):
        for j in range(w):
            tab[i, j] = (a*i + b*j) % 256
    return Image.fromarray(tab)

canal_a = rysuj_po_skosie_szare(200, 300, 6, 5)
rgb = rysuj_pasy_kolorowe()

a_ext = np.expand_dims(canal_a, axis=-1)

combined = np.concatenate((rgb, a_ext), axis=-1)
obraz7 = Image.fromarray(combined)
obraz7.show()
obraz7.save("obraz7.png")
```

Otrzymany obraz:

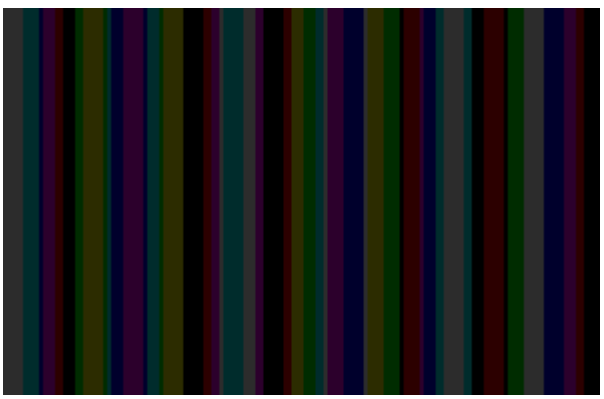


Zadanie 7

Konwersujemy obraz rgb do cmyk

```
def rgb_to_cmyk(rgb_array):
    rgb = rgb_array.astype(float) / 255
    r, g, b = rgb[..., 0], rgb[..., 1], rgb[..., 2]
    k = 1 - np.max(rgb, axis=2)
    c = (1 - r - k) / (1 - k + 1e-8)
    m = (1 - g - k) / (1 - k + 1e-8)
    y = (1 - b - k) / (1 - k + 1e-8)
    c[np.isnan(c)] = 0
    m[np.isnan(m)] = 0
    y[np.isnan(y)] = 0
    cmyk = np.stack((c, m, y, k), axis=2) * 255
    return cmyk.astype(np.uint8)


t_cmyk = rgb_to_cmyk(rysuj_pasy_kolorowe())
image_cmyk = Image.fromarray(t_cmyk, mode="CMYK")
image_cmyk.show()
image_cmyk.save("obraz8.tiff")
```

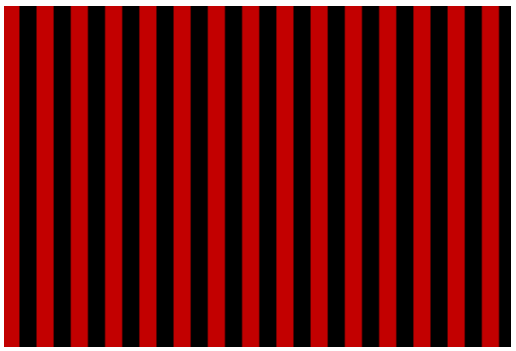
Tak wygląda obraz po konwersji:



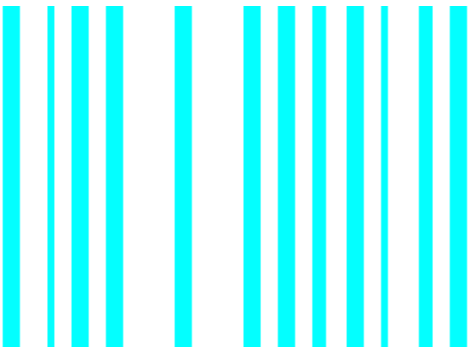
Korzystając z funkcji kanal_r i kanal_c wyciągami do obrazu kanał r z obrazu z zadania 5 i kanał c z obrazu z zadania 7

```
def kanal_r(obraz):
    tab = np.zeros((200, 300, 3), dtype=np.uint8)
    for i in range(300):
        for j in range(200):
            tab[j, i, 0] = obraz[j, i, 0]
    return tab


Image.fromarray(kanal_r(rysuj_pasy_kolorowe())).show()
Image.fromarray(kanal_r(rysuj_pasy_kolorowe())).save("r.png")


def kanal_c(obraz):
    tab = np.zeros((200, 300, 4), dtype=np.uint8)
    for i in range(300):
        for j in range(200):
            tab[j, i, 0] = obraz[j, i, 0]
    return tab


Image.fromarray(kanal_c(t_cmyk), mode="CMYK").show()
Image.fromarray(kanal_c(t_cmyk), mode="CMYK").save("c.tiff")
```

Kanał R:



Kanał C:



Obrazy widocznie różnią się od siebie kolorami oraz układem pasów
Kanał R(red) ma czerwone pasy na czarnym tle

Kanał C(cyan) ma cyjanowe pasy na białym tle