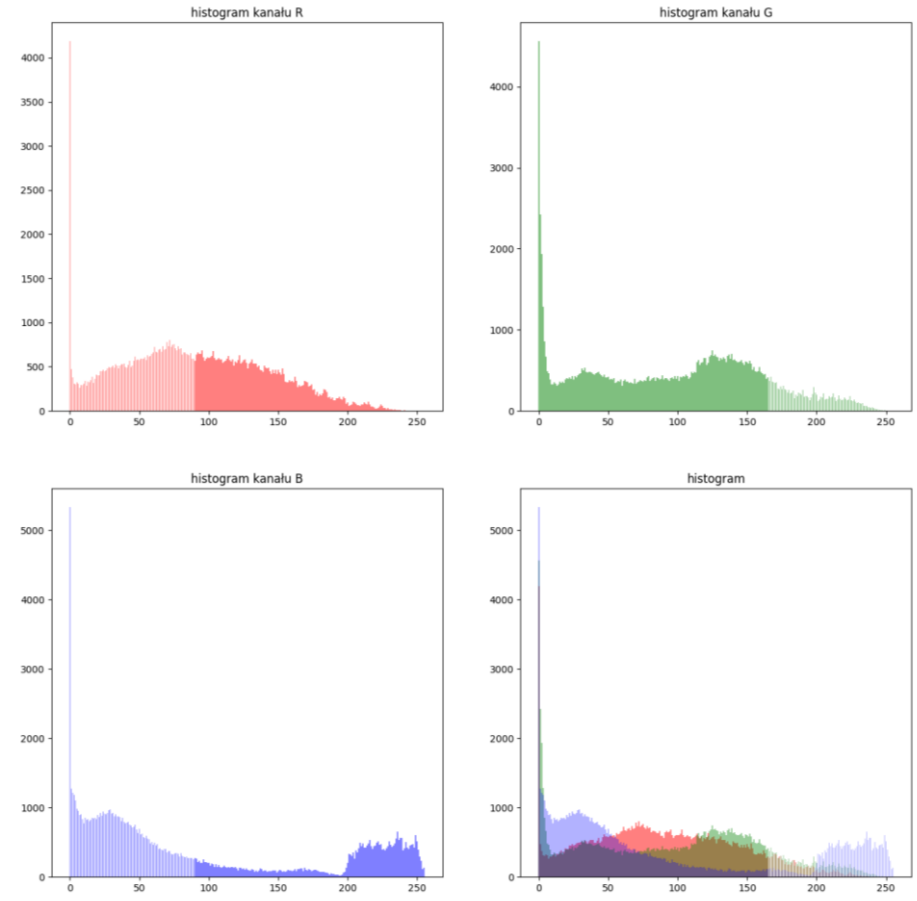
Krzysztof Krupicki - raport lab5-6

1. Pobierz statystyki obrazu im i je skomentuj
    a. przedstaw histogram obrazu im na diagramie plt, przedstaw histogramy wszystkich jego kanałów na diagramach plt.

```
def rysuj_histogram_RGB(obraz):
    hist = obraz.histogram()
    plt.figure(figsize=(16, 16))
    plt.subplot( *args: 2, 2, 1)  # ile obrazów w pionie, ile w poziomie, numer obrazu
    plt.title("histogram kanału R")
    plt.bar(range(256), hist[:256], color='r', alpha=0.5)
    plt.subplot( *args: 2, 2, 2)
    plt.title("histogram kanału G")
    plt.bar(range(256), hist[256:2 * 256], color='g', alpha=0.5)
    plt.subplot( *args: 2, 2, 3)
    plt.title("histogram kanału B")
    plt.bar(range(256), hist[2 * 256:], color='b', alpha=0.5)
    plt.subplot( *args: 2, 2, 4)
    plt.title("histogram")
    plt.bar(range(256), hist[:256], color='r', alpha=0.5)
    plt.bar(range(256), hist[256:2 * 256], color='g', alpha=0.4)
    plt.bar(range(256), hist[2 * 256:], color='b', alpha=0.3)
    plt.subplots_adjust(wspace=0.2, hspace=0.2)
    plt.savefig('histogramy.png')


rysuj_histogram_RGB(im)
```



Krzysztof Krupicki - raport lab5-6

b. Ile jest pikseli o wartości 155 na każdym z kanałów?

```
im_histogram = im.histogram()
print(f'Kanał R: {im_histogram[155]}')
print(f'Kanał G: {im_histogram[155 + 256]}')
print(f'Kanał B: {im_histogram[155 + 2 * 256]}')
print('\n')
```

```
Kanał R: 329
Kanał G: 542
Kanał B: 72
```

c. Napisz funkcję zlicz_piksele(obraz, kolor), która zlicza, ile jest pikseli w danym kolorze. Ile jest pikseli o wartości [155,155,155] w obrazie im?

```
def zlicz_piksele(obraz, kolor):
    t = np.asarray(obraz)
    liczba_pikseli = np.sum(np.all(t == np.array(kolor), axis=-1))
    print(f'Łącznie pikseli o wartości {kolor}: {liczba_pikseli}')


zlicz_piksele(im, kolor: [155, 155, 155])
```

```
Łącznie pikseli o wartości [155, 155, 155]: 0
```

2. Zapisz obraz im w formacie jpg a potem wczytaj jako im_jpg.

a. Porównaj statystyki obrazów im oraz im_jpg. Dla czego te obrazy się różnią?

```
===== Statystyki obrazu PNG =====
extrema  [(0, 255), (0, 255), (0, 255)]
count  [100000, 100000, 100000]
mean  [89.19494, 95.61805, 90.70041]
rms  [103.81432088108076, 114.93160735846341, 124.96983720082218]
median  [86, 101, 52]
stddev  [53.11944934199149, 63.768823763634686, 85.97031950523332]
===== Statystyki obrazu JPG =====
extrema  [(0, 255), (0, 255), (0, 255)]
count  [100000, 100000, 100000]
mean  [89.25336, 95.75403, 91.18454]
rms  [103.46635791405824, 114.82711556945075, 124.48258046811209]
median  [85, 101, 55]
stddev  [52.33665014796419, 63.37532807614569, 84.74250707282857]
```

Odpowiedź:

Obrazy różnią się ponieważ zapisując obraz w formacie JPG następuje kompresja stratna, tracimy część danych obrazu.

Krzysztof Krupicki - raport lab5-6

b. Zastosuj ImageChops.difference, aby otrzymać różnicę tych obrazów. Pobierz statystyki różnicy i je skomentuj

```
===== Statystyki obrazu różnic =====
extrema  [(0, 66), (0, 42), (0, 104)]
count  [100000, 100000, 100000]
mean  [3.91044, 2.61936, 7.02085]
rms  [5.984981202978001, 3.9471736723888906, 10.662925020837388]
median  [3, 2, 4]
stddev  [4.53083425059889, 2.952817839014117, 8.025312160751131]
```

Odpowiedź:
Ekstrema nie zmieniły się (mamy pełny zakres barw).
Średnia różni się minimalnie, największa różnica w kanale niebieskim, co oznacza że utraciliśmy najwięcej niebieskiej barwy.
Mediana przesunęła się w kanale niebieskim o 3 wartości, wprowadziła lekką zmianę barwy.
Mniejsze odchylenie standardowe w JPG oznacza rozmycie szczegółów.
Minimalny spadek RMS, czyli mamy drobną utratę szczegółów i kontrastu.

c. Jak zmienią się statystyki, gdy jeszcze dwa razy zapiszesz obraz im_jpg w formacie jpg?

```
===== Statystyki różnic PNG vs 3xJPG =====
extrema  [(0, 75), (0, 51), (0, 113)]
count  [100000, 100000, 100000]
mean  [4.61485, 3.12475, 8.61772]
rms  [7.224781657600457, 4.765258649853122, 13.013960196650364]
median  [3, 2, 5]
stddev  [5.558833463731396, 3.597725314347942, 9.751823521865026]
```

Odpowiedź:
Po zapisaniu jeszcze dwa razy jako JPG:
Średnia rośnie, kolory stają się jaśniejsze - tracimy kontrast.
Mniejsze ochylenie standardowe, większe rozmycie szczegółów.
Niewielka zmiana mediany.
Dalszy spadek RMS to spadek kontrastu i energii obrazu.

Każde kolejne zapisanie pliku JPG powoduje kumulację strat.
Kompresja działa na już skompresowanych danych, więc pojawia się coraz większe rozmycie, kolory się lekko zmieniają, kontrast i ostrość maleją, statystyki (stddev, RMS) spadają systematycznie.

Krzysztof Krupicki - raport lab5-6

3. Wykonaj następujące polecenia dla obrazu im

a. Wczytaj tablicę obrazu i pobierz kanały t_r, t_g, t_b obrazu z tablicy obrazu, zapisz jako obrazy im_r, im_g, im_b

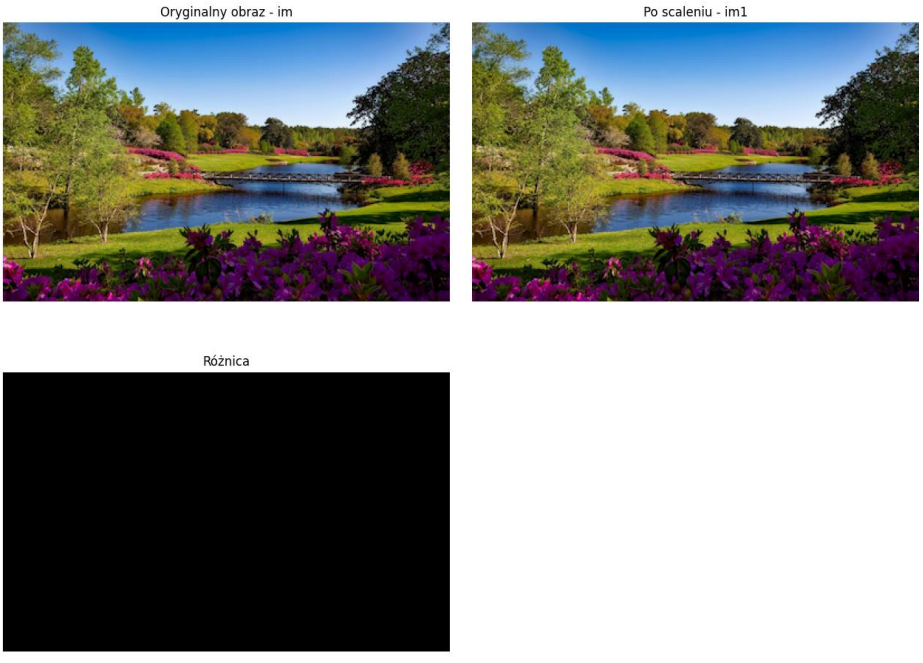
```
t = np.asarray(im)
t_r = t[:, :, 0]
t_g = t[:, :, 1]
t_b = t[:, :, 2]
im_r = Image.fromarray(t_r)
im_g = Image.fromarray(t_g)
im_b = Image.fromarray(t_b)
```

b. Utwórz obraz im1 przez scalenie metodą merge obrazów im_r, im_g, im_b i zastosuj ImageChops.difference(im, im1) do porównania otrzymanego obrazu z obrazem wejściowym.

```
im1 = Image.merge( mode: "RGB",  bands: (im_r, im_g, im_b))
diff = ImageChops.difference(im, im1)
```

c. Umieść na jednej figurze plt (fig1.png) obrazy im, im1 i wynik porównania

```
plt.figure(figsize=(16, 12))
plt.subplot( *args: 2, 2, 1)
plt.title("Oryginalny obraz - im")
plt.axis('off')
plt.imshow(im)
plt.subplot( *args: 2, 2, 2)
plt.title("Po scaleniu - im1")
plt.axis('off')
plt.imshow(im1)
plt.subplot( *args: 2, 2, 3)
plt.title("Różnica")
plt.axis('off')
plt.imshow(diff)
plt.subplots_adjust(wspace=0.05, hspace=0.05)
plt.savefig('fig1.png')
plt.show()
```



Krzysztof Krupicki - raport lab5-6

d. Czy są jakieś różnice?

```
===== Statystyki różnic =====
extrema  [(0, 0), (0, 0), (0, 0)]
count  [100000, 100000, 100000]
mean  [0.0, 0.0, 0.0]
rms  [0.0, 0.0, 0.0]
median  [0, 0, 0]
stddev  [0.0, 0.0, 0.0]
```

Odpowiedź:

Nie widać różnic w obrazach, wszystkie statystyki w obrazie różnic to 0, czyli otrzymany obraz w wyniku złączenia trzech kanałów jest taki sam jak oryginał.

4. Napisz funkcję mieszaj_kanaly(obraz), która w sposób losowy miesza kanały r, g, b pobrane metodą split z danego obrazu oraz ich negatywy nr,ng,nb. Dopuszczalne jest losowanie z powtórzeniami tzn. może być b, r, g ale też b, b, g .

a. Zastosuj tę funkcje do obrazu im. Otrzymany obraz nazwij mix i zapisz w formacie png.

```python
def mieszaj_kanaly(obraz):
    r, g, b = obraz.split()
    nr = Image.fromarray(255 - np.array(r))
    ng = Image.fromarray(255 - np.array(g))
    nb = Image.fromarray(255 - np.array(b))
    kanaly = [r, g, b, nr, ng, nb]
    nazwy_kanalow = ["R", "G", "B", "negatyw R", "negatyw G", "negatyw B"]
    wybrane = random.choices(kanaly, k=3)
    mix = Image.merge( mode: "RGB", wybrane)
    wybrane_nazwy = [nazwy_kanalow[kanaly.index(k)] for k in wybrane]
    # print("Wybrane kanały:", wybrane_nazwy)
    return mix


mix = mieszaj_kanaly(im)
mix.show()
mix.save('mix.png')
```

Krzysztof Krupicki - raport lab5-6

b. Napisz funkcję rozpoznaj_mix(obraz, mix), która dla danych obrazów w trybie RGB obraz i mix rozpoznaje w jaki sposób mix powstał z obrazu (zmiana kolejności kanałów).

```
def rozpoznaj_mix(obraz, mix):
    r, g, b = obraz.split()
    nr = Image.fromarray(255 - np.array(r, dtype=np.uint8))
    ng = Image.fromarray(255 - np.array(g, dtype=np.uint8))
    nb = Image.fromarray(255 - np.array(b, dtype=np.uint8))

    oryginalne_kanaly = [r, g, b, nr, ng, nb]
    nazwy_kanalow = ["R", "G", "B", "negatyw R", "negatyw G", "negatyw B"]
    wynik = []

    for mix_kanal in mix.split():
        for nazwa, k in zip(nazwy_kanalow, oryginalne_kanaly):
            if np.all(np.array(mix_kanal) == np.array(k)):
                wynik.append(nazwa)
                break
    print('Rozpoznano:')
    print(f'R -> {wynik[0]}')
    print(f'G -> {wynik[1]}')
    print(f'B -> {wynik[2]}')


rozpoznaj_mix(im, mix)
```

```
Rozpoznano:
R -> G
G -> R
B -> negatyw G
```

5. Dlaczego polecenie r, g, b = im.split() nie działa, gdy im = Image.open('beksinski1.png')?

```
im = Image.open('beksinski.png')
im1 = Image.open('beksinski1.png')
print('===== beksinski.png =====')
print(im.mode)
print('===== beksinski1.png =====')
print(im1.mode)
# r, g, b = im1.split()
```

```
===== beksinski.png =====
RGB
===== beksinski1.png =====
RGBA
```

Odpowiedź:

Polecenie nie działa, ponieważ mamy 4 kanały (obraz RGBA), a podajemy tylko 3 zmienne, kanał 4 jest od alpha, czyli przeźroczystość.

Krzysztof Krupicki - raport lab5-6

6. Napisz funkcję ocen_czy_identyczne(obraz1, obraz2), która sprawdza czy dwa obrazy są identyczne. Funkcja musi uwzględniać tryb obrazu, rozmiar obrazu, wartości pikseli. Na wyjściu powinien pojawić się komunikat „obrazy identyczne” lub „obrazy nie są identyczne, bo …” (w miejsce … wstawić uzasadnienie np. „obrazy mają różne tryby”).

   a. Zastosuj funkcję do porównania obrazu beksinski.png z obrazami beksinski1.png, beksinski2.png, beksinski3.png

```
def ocen_czy_identyczne(obraz1, obraz2):
    t_obraz1 = np.asarray(obraz1)
    t_obraz2 = np.asarray(obraz2)
    roznice_pikseli = []
    if len(t_obraz1.shape) == 3:
        obraz1_w, obraz1_h, obraz1_d = t_obraz1.shape
        obraz2_w, obraz2_h, obraz2_d = t_obraz2.shape
    else:
        obraz1_w, obraz1_h = t_obraz1.shape
        obraz2_w, obraz2_h = t_obraz2.shape
    min_x = min(obraz1_w, obraz2_w)
    min_y = min(obraz1_h, obraz2_h)
    for h in range(min_y):
        for w in range(min_x):
            for k in range(3):
                if t_obraz1[w, h, k] != t_obraz2[w, h, k]:
                    roznice_pikseli.append((w, h))
    if obraz1.mode != obraz2.mode:
        print("Obrazy nie są identyczne bo, obrazy mają różne tryby")
    elif obraz1.size != obraz2.size:
        print("Obrazy nie są identyczne bo, obrazy mają różne rozmiary")
    elif obraz1.format != obraz2.format:
        print("Obrazy nie są identyczne bo, obrazy mają różne formaty")
    elif len(roznice_pikseli) > 0:
        print("Obrazy nie są identyczne bo, obrazy mają inne wartości pikseli:")
        for wsp in roznice_pikseli[:5]:
            print(wsp)
    else:
        print("Obrazy są identyczne.")


im_beksinski = Image.open('beksinski.png')
im_beksinski1 = Image.open('beksinski1.png')
im_beksinski2 = Image.open('beksinski2.png')
im_beksinski3 = Image.open('beksinski3.png')
print('beskinski vs beksinski1')
ocen_czy_identyczne(im_beksinski, im_beksinski1)
print('\nbeskinski vs beksinski2')
ocen_czy_identyczne(im_beksinski, im_beksinski2)
print('\nbeskinski vs beksinski3')
ocen_czy_identyczne(im_beksinski, im_beksinski3)
```

Krzysztof Krupicki - raport lab5-6

```
beskinski vs beksinski1
Obrazy nie są identyczne bo, obrazy mają różne tryby

beskinski vs beksinski2
Obrazy nie są identyczne bo, obrazy mają różne rozmiary

beskinski vs beksinski3
Obrazy nie są identyczne bo, obrazy mają inne wartości pikseli:
(148, 51)
(149, 51)
(150, 51)
(151, 51)
(152, 51)
```

7. Napisz funkcję pokaz_roznice(obraz_wejsciowy), która tworzy nowy obraz obraz_wynikowy (w trybie obrazu wejściowego) na którym zwiększone są wartości przez skalowanie tzn. na każdym kanale zmieniamy wartości elementów tablicy obrazu według reguły (wartosc_elementu/ max_wartosc_kanalu) *255

   a. W zadaniu 2c w Lab 5 powstał obraz po trzykrotnym zapisie w formacie jpg. Nazwij ten obraz im_jpg3

   b. Utwórz obraz diff zawierający różnice między obrazami im z poprzednich ćwiczeń i im_jpg3

   c. Umieść na jednej figurze plt (fig2.png) obrazy im ,_jpg3 , diff oraz obraz otrzymany z funkcji pokaz_roznice(diff)

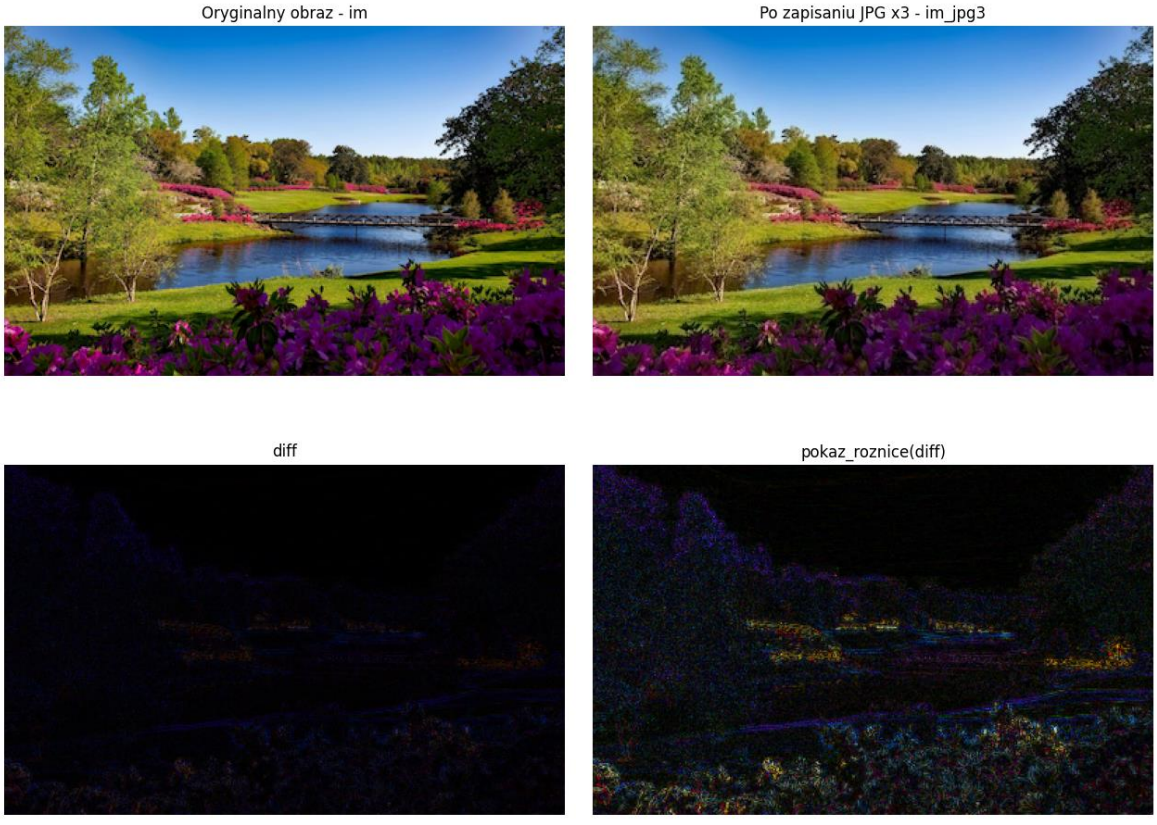
```python
def pokaz_roznice(obraz_wejsciowy):
    t_obraz = np.asarray(obraz_wejsciowy)
    if len(t_obraz.shape) == 3:
        w, h, d = t_obraz.shape
    else:
        w, h = t_obraz.shape
        d = 1
    t_wynik = np.copy(t_obraz)

    for k in range(d):
        max_wartosc_kanal = np.max(t_obraz[:, :, k])
        if max_wartosc_kanal != 0:
            t_wynik[:, :, k] = (t_obraz[:, :, k] / max_wartosc_kanal) * 255
        else:
            t_wynik[:, :, k] = 0
    obraz_wynik = Image.fromarray(t_wynik)
    return obraz_wynik


im_jpg3 = Image.open('im_jpg3.jpg')
im = Image.open('im.png')
diff = ImageChops.difference(im, im_jpg3)
diff.save('diff.png')
diff_pokaz = pokaz_roznice(diff)
diff_pokaz.save('diff_pokaz.png')
```

Krzysztof Krupicki - raport lab5-6

```
plt.figure(figsize=(16, 12))
plt.subplot( *args: 2, 2, 1)
plt.title("Oryginalny obraz - im")
plt.axis('off')
plt.imshow(im)
plt.subplot( *args: 2, 2, 2)
plt.title("Po zapisaniu JPG x3 - im_jpg3")
plt.axis('off')
plt.imshow(im_jpg3)
plt.subplot( *args: 2, 2, 3)
plt.title("diff")
plt.axis('off')
plt.imshow(diff)
plt.subplot( *args: 2, 2, 4)
plt.title("pokaz_roznice(diff)")
plt.axis('off')
plt.imshow(diff_pokaz)
plt.subplots_adjust(wspace=0.05, hspace=0.05)
plt.savefig('fig2.png')
```



Krzysztof Krupicki - raport lab5-6

8. Napisz funkcję wstaw_inicjaly(obraz_bazowy, obraz_wstawiany, m, n, kolor). Obraz_wstawiany jest obrazem w trybie 1, na którym są inicjały w kolorze czarnym na białym tle. W miejscu (m, n) w obrazie bazowym wstawiamy inicjały w kolorze kolor (tam, gdzie były białe piksele obraz bazowy się nie zmienia, a kolor pojawia się tylko w miejscu czarnych pikseli). Gdyby obraz_wstawiany miał wyjść poza ramy obrazu bazowego należy go przyciąć (jak w funkcji wstaw_obraz_w_obraz(obraz_bazowy, obraz_wstawiany, m, n) z Lab2).

a. Zastosuj funkcję do obrazu im wybranego na poprzednich ćwiczeniach i wstaw własne inicjały w 3 różnych kolorach:

1. w prawym górnym rogu,
2. w lewym dolnym rogu,
3. w połowie wysokości tak, żeby było widać tylko pierwszą literę inicjałów.
4. obraz wynikowy zapisz jako obraz_inicjaly.png

```
def wstaw_inicjaly(obraz_bazowy, obraz_wstawiany, m, n, kolor):
    t = np.asarray(obraz_bazowy, dtype=np.uint8)
    t_obraz_bazowy = np.copy(t)
    t_obraz_wstawiany = np.asarray(obraz_wstawiany)

    h_ob, w_ob, d = t_obraz_bazowy.shape
    h_ow, w_ow = t_obraz_wstawiany.shape
    n_k = min(h_ob, n + h_ow)
    m_k = min(w_ob, m + w_ow)
    n_p = max(0, n)
    m_p = max(0, m)

    for h in range(n_p, n_k):
        for w in range(m_p, m_k):
            if t_obraz_wstawiany[h - n, w - m] == 0:
                t_obraz_bazowy[h, w] = kolor
    return Image.fromarray(t_obraz_bazowy)


inicjaly = Image.open('inicjaly.bmp')
im_wstawione = wstaw_inicjaly(im, inicjaly, 400 - 100,  n: 0,  kolor: [255, 0, 255])
im_wstawione = wstaw_inicjaly(im_wstawione, inicjaly,  m: 0, 250 - 50,  kolor: [255, 255, 0])
im_wstawione = wstaw_inicjaly(im_wstawione, inicjaly, 400 - 50, (250 - 50) // 2,  kolor: [0, 255, 255])
im_wstawione.save('obraz_inicjaly.png')
```

Krzysztof Krupicki - raport lab5-6

9. Obraz zakodowany1.bmp powstał tak, że zastosowałam funkcję ukryj_kod(obraz, im_kod), gdzie obraz jest wczytanym obrazem jesien.jpg, a im_kod jest wczytanym obrazem kod.bmp.

a. Napisz funkcję odkoduj(obraz1, obraz2), która wczytuje dwa obrazy, a na wyjściu podaje obraz w trybie L ilustrujący różnice w ten sposób, że piksele różne wyświetlają się na biało a piksele równe na czarno. Na przykład, wczytujemy jesien.jpg oraz zakodowany1.bmp a wynikiem funkcji jest kod.bmp.

b. Zastosuj funkcję odkoduj do obrazów jesien.jpg oraz zakodowany1.bmp. Otrzymany obraz zapisz jako kod2.bmp.

```python
def odkoduj(obraz1, obraz2):
    t1 = np.asarray(obraz1)
    t2 = np.asarray(obraz2)
    """Wersja krótsza z użyciem np.any
    np.any zwraca tablice ktora zawiera wartości bool na każdej współrzędnej
    """
    roznice = np.any(t1 != t2, axis=2)
    roznice = roznice.astype(np.uint8) * 255
    """Wersja dłuższa z przechodzeniem petlą po każdej współrzędnej"""
    h, w, d = t1.shape
    kod = np.zeros( shape: (h, w), dtype=np.uint8)

    for i in range(h):
        for j in range(w):
            r1, g1, b1 = t1[i, j]
            r2, g2, b2 = t2[i, j]

            if r1 != r2 or g1 != g2 or b1 != b2:
                kod[i, j] = 255
            else:
                kod[i, j] = 0
    return Image.fromarray(kod, mode="L")


zakodowany = Image.open('zakodowany1.bmp')
jesien = Image.open('jesien.jpg')
odkodowany = odkoduj(zakodowany, jesien)
odkodowany.save('kod2.bmp')
```