Temat: Obsługa obrazów.

1. **Podstawowe operacje na obrazach z wykorzystaniem bibliotek OpenCV i Matplotlib.**
   - Czytanie obrazów.
   - Modyfikowanie obrazów.
   - Wyświetlanie obrazów.
   - Zapisywanie obrazów.
   - Konwersja obrazów.

Uwagi:

- Przy próbie załadowania pliku, który nie istnieje funkcja zwróci **None**
- Aktualnie OpenCV wspiera następujące formaty plików: **.bmp**, **.jpeg**, **.jpg**, **.png**, **.tiff** i **.tif**
- W ćwiczeniu do wczytywania plików obrazów będzie wykorzystywana funkcja `cv2.imread`, która ma następujące argumenty:
  - nazwa pliku z obrazem do wczytania
  - flagi do specyfikowania trybów wczytywania obrazu

- Zauważ, że OpenCV czyta obrazy w trybie **BGR** a nie w trybie **RGB**. Oznacza to, że kolejność kanałów staje się następująca: `blue, green, red`. Wszystkie funkcje OpenCV przetwarzają obrazy w tym trybie.

- Podczas wczytywania obrazów używane są trzy flagi:
  - **cv2.IMREAD_COLOR**: jest to domyślna flaga, następuje wczytywanie obrazu jako obrazu kolorowego (uwaga w trybie BGR !!!)
  - **cv2.IMREAD_GRAYSCALE**: czytanie obrazu w formacie skali szarości, tzn. dokonuje konwersji obrazu kolorowego do obrazu w skali szarości.
  - **cv2.IMREAD_UNCHANGED**: czytanie obrazu takim, jaki jest. Oznacza to, że jeśli obrazek jest obrazem PNG z przezroczystym tłem, zostanie odczytany jako obrazek BGRA, gdzie A określa kanał alfa – odpowiedzialny za przezroczystość. Jeśli ta flaga nie jest używana, obraz zostanie odczytany jako obraz BGR. Zwróć uwagę, że BGR odnosi się do kanałów niebieskiego, zielonego i czerwonego obrazu. A, czyli kanał alfa, odpowiada za przejrzystość. Dlatego obraz z przezroczystym tłem będzie odczytywany jako BGRA, a nie jako BGR. Należy również zauważyć, że OpenCV domyślnie używa trybu BGR i dlatego omawiamy tutaj tryb BGRA, a nie tryb RGBA.
- Do rozdzielenia kanałów używana jest funkcja **cv2.split**, która pobiera obraz trzykanałowy i zwraca listę trzech kanałów: `blue, green` i `red`.
- Do scalenia kanałów używana jest funkcja **cv2.merge**, która używa jeden argument w postaci zestawu składającego się z trzech kanałów `blue, green, red` i zwraca obraz scalony.
- W zależności od wykorzystywanego środowiska przetwarzania kodu Pythona mogą być używane funkcje, które wspomagają proces obsługi obrazów:
  - **cv2.imshow** – do otwarcia okna z obrazem
  - **cv2.waitKey** – określa jak długo sterowanie ma pozostać na danym oknie

- **cv2.destroyAllWindows** – zamyka wszystkie otwarte okna

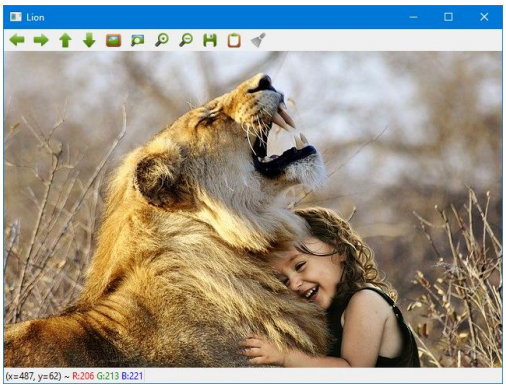
- Do zapisu obrazów służy funkcja **cv2.imwrite** z dwoma argumentami:
  - nazwą pliku do zapisu
  - obraz do przechowania

Zadanie 1. Wczytanie, przetwarzanie, wyświetlanie i zapisywanie obrazów

- Import bibliotek `numpy` i `matplotlib`
- Wczytać kolejno obrazy z plików testowych jako obrazy kolorowe lub w skali szarości.
- Dokonać sprawdzenia rozdzielczości obrazów.
- Wykonać modyfikację piksela o współrzędnych [0,0] nadając mu wartość=0, następnie wyświetlić obraz jako tablicę `numpy` i sprawdzić wartość intensywności zmodyfikowanego piksela.
- Wyświetlić obraz przy pomocy biblioteki `matplotlib`
- Zapisać obraz w nowym pliku pod tą samą nazwą ale uzupełnioną dopiskiem **_bis**.

A. Używanie biblioteki OpenCV do wyświetlania obrazów.

```
import cv2

img = cv2.imread("images/lion.jpg")

cv2.imshow("Lion",img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

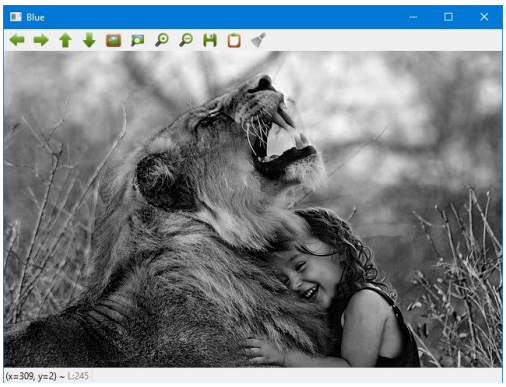


- Rozdzielenie kanałów

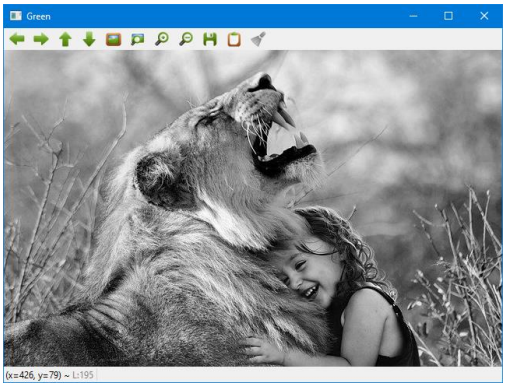
```
# Rozdzielenie kanałów
blue, green, red = cv2.split(img)
```

- Wyświetlenie poszczególnych kanałów pozyskanych w powyższym kroku

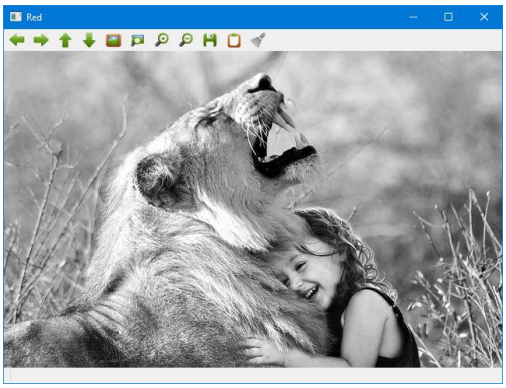
```
cv2.imshow("Blue",blue)
cv2.waitKey(0)
cv2.destroyAllWindows()
```




```
cv2.imshow("Green",green)
cv2.waitKey(0)
cv2.destroyAllWindows()
```




```
cv2.imshow("Red",red)
cv2.waitKey(0)
cv2.destroyAllWindows()
```




```
# Zapis wszystkich kanałów
cv2.imwrite("Blue.png",blue)
cv2.imwrite("Green.png",green)
cv2.imwrite("Red.png",red)
```

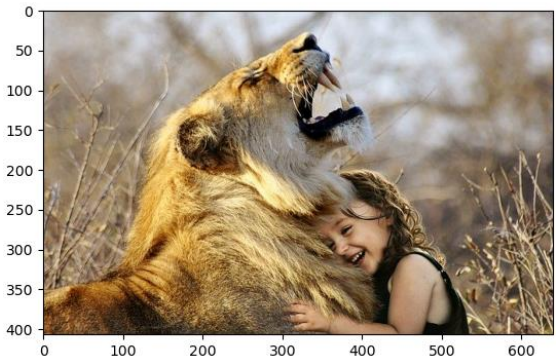
B. Użycie biblioteki `matplotlib` do wyświetlania obrazów.

- Biblioteka `matplotlib` jest zwykle używana w data science i widzeniu komputerowym do celów wizualizacji.
- Poniżej znajdują się przykłady jak można wykorzystać `matplotlib` do wyświetlania obrazów, które zostały wczytane lub przetwarzane przy użyciu biblioteki `OpenCV`.
- Należy pamiętać, że `matplotlib` zakłada tryb obrazów kolorowych jako RGB, podczas gdy `OpenCV` przechowuje je w trybie BGR.
- Konieczna jest zatem konwersja obrazów z jednego trybu do drugiego.
- Stosowane są dwa sposoby konwersji obrazów BGR na obrazy RGB:
    - przy użyciu funkcji **cv2.cvtColor** z flagą **cv2.COLOR_BGR2RGB**
    - przez odwrócenie kolejności kanałów – zrobione np. przez konstrukcję [:,:,::-1], gdzie ::-1 na ostatniej pozycji odpowiada za odwrócenie kanałów

  Oba powyższe sposoby są dozwolone zamiennie.
- Typowy przebieg wyświetlania obrazów z biblioteką `matplotlib` obejmuje polecenia:

```
import matplotlib.pyplot as plt
plt.imshow(img)
plt.show()
```


```
import matplotlib.pyplot as plt
plt.imshow(img[:,:,::-1])
plt.show()
```



- Obraz w skali szarości:

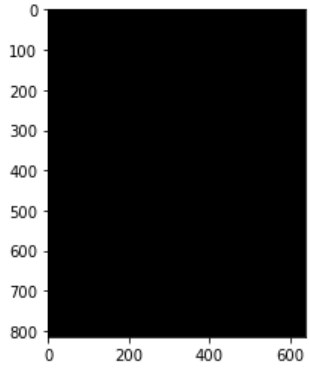
```
image = cv2.imread("images/lion.jpg", cv2.IMREAD_GRAYSCALE)
plt.imshow(img, cmap="gray")
```

Zadanie 2. Tworzenie efektu wody

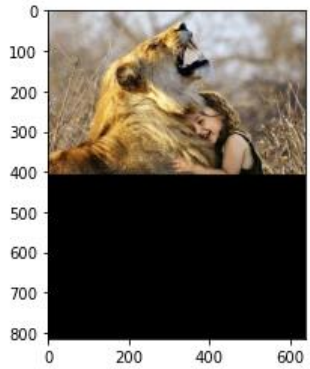
- W tym ćwiczeniu zaimplementujemy filtr wodny, który odpowiada za pionowe odwracanie obiektu unoszącego się na wodzie.
- Efekt filtra widoczny jest na poniższym obrazku:

Czynności do wykonania:
1. Wczytanie obrazu
2. Pionowe przerzucenie
3. Dołączenie oryginalnego obrazu z odwróconym
4. Wyświetlenie i zapisanie obrazu

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
img = cv2.imread("images/lion.jpg")

plt.imshow(img[:,:,::-1])
plt.show()

# Odczyt kształtu obrazu
img.shape

# Utwórz nową tablicę o podwójnym wymiarze
# Wysokość stanie się dwa razy większa
# Szerokość i liczba kanałów będą takie same
imgNew = np.zeros((814,640,3),dtype=np.uint8)

# Wyświetlenie obrazu
plt.imshow(imgNew[:,:,::-1])
plt.show()
```

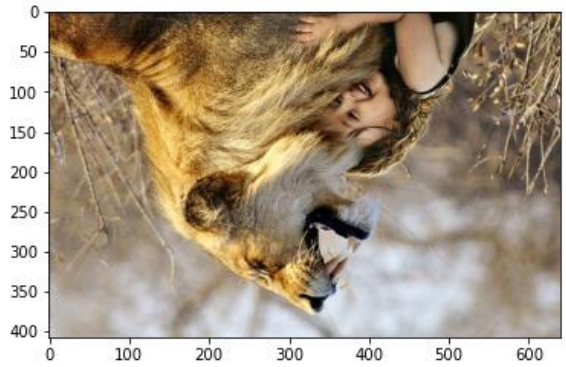



```
# Skopiuj oryginalny obraz do
# górnej połowy nowego obrazu
imgNew[:407][:] = img

# Wyświetl obraz
plt.imshow(imgNew[:,:,::-1])
plt.show()
```

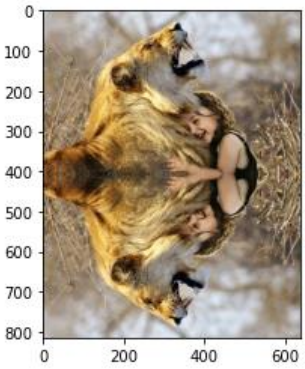


- Następnie odwróćmy w pionie oryginalny obraz.
- Możemy czerpać inspirację z tego, jak odwróciliśmy kanały za pomocą ::-1 na ostatniej pozycji.
- Ponieważ odwrócenie obrazu w pionie jest równoznaczne z odwróceniem kolejności wierszy na obrazie, użyjemy ::-1 na pierwszej pozycji:

```
# Odwrócenie wierszy obrazu
imgInverted = img[::-1,:,:]

# Wyświetlenie obrazu
plt.imshow(imgInverted[:,:,::-1])
plt.show()
```



```
# Skopiuj odwrócony obraz do
# dolnej połowy nowego obrazu
imgNew[407:][:] = imgInverted

# Wyświetl obraz
plt.imshow(imgNew[:,:,::-1])
plt.show()
```




```
# Zapisz obraz
cv2.imwrite("WaterEffect.png",imgNew)
```