Getting Started with Image Preprocessing in Python

```
#importing required Libraries
import numpy as np
import tensorflow
import keras
import os
import glob
from skimage import io
import skimage
import random
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

Introduction

1. image resizing
2. Image slicing
3. RGB channels
4. Motion Difference
5. Point processing techniques

```
import os
dataset_pth = '/kaggle/input/animals10/raw-img/'
```

```
# accessing an image file from the dataset classes
image = io.imread('/kaggle/input/animals10/raw-img/cavallo/OIP---MGqQIhmz3OEPYP-46_xwHaFj.jpeg')

# plotting the original image
i, (im1) = plt.subplots(1)
i.set_figwidth(15)
im1.imshow(image)
```

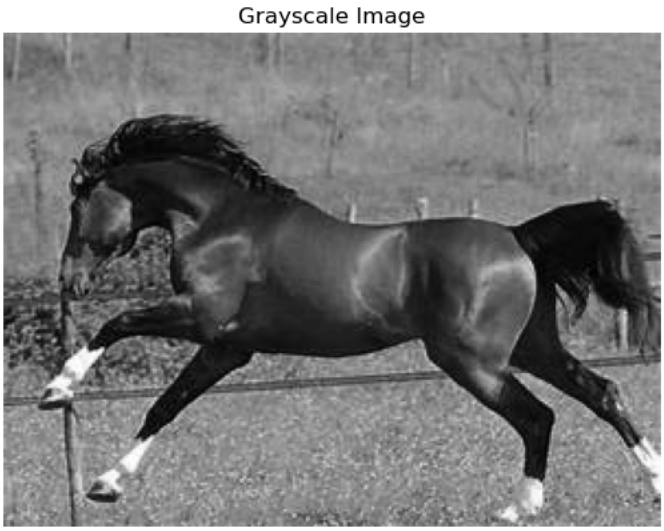
```
<matplotlib.image.AxesImage at 0x7840bc828690>
```


```
# Print the shape of the image
print("Shape of the image:", image.shape)  # MxNx3 for RGB
```

```
Shape of the image: (225, 300, 3)
```

```
# Convert to grayscale
if image.shape[2] == 3:  # Check if the image is RGB
    grayscale_image = np.mean(image, axis=2).astype(np.uint8)  # Simple average for grayscale
    plt.imshow(grayscale_image, cmap='gray')
    plt.axis('off')
```

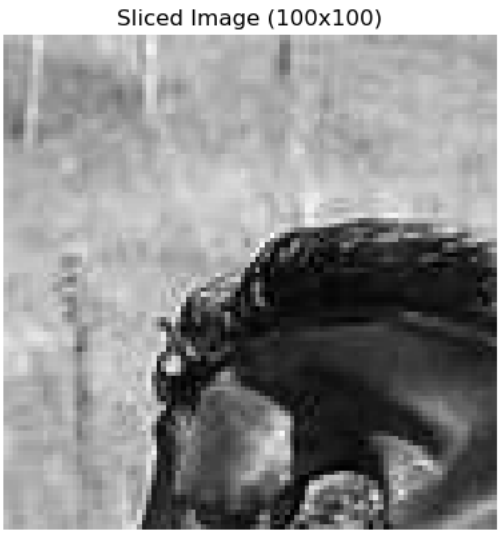
```
plt.title('Grayscale Image')
plt.show()

# Print the shape of the grayscale image
print("Shape of the grayscale image:", grayscale_image.shape)
```




```
Shape of the grayscale image: (225, 300)
```

Slicing

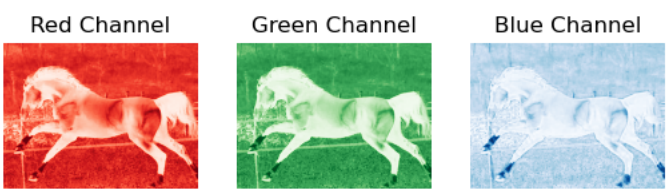
```
# Slicing the first 100x100 pixels
sliced_image = grayscale_image[:100, :100]
plt.imshow(sliced_image,cmap='gray')
plt.axis('off')
plt.title('Sliced Image (100x100)')
plt.show()
```



Split the RGB channels

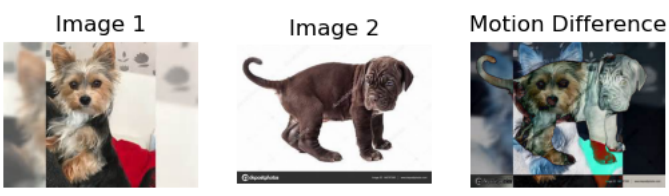
```
# Split the channels
red_channel = image[:, :, 0]
green_channel = image[:, :, 1]
blue_channel = image[:, :, 2]

# Display channels
plt.subplot(1, 3, 1)
plt.imshow(red_channel, cmap='Reds')
plt.axis('off')
plt.title('Red Channel')
```

```
plt.subplot(1, 3, 2)
plt.imshow(green_channel, cmap='Greens')
plt.axis('off')
plt.title('Green Channel')

plt.subplot(1, 3, 3)
plt.imshow(blue_channel, cmap='Blues')
plt.axis('off')
plt.title('Blue Channel')

plt.show()
```



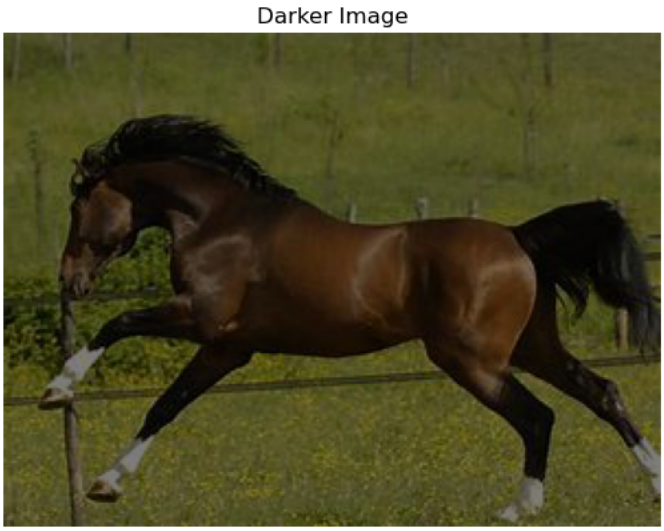
Logic operations on images (substractions)

```
from skimage.transform import resize

# Load two images
image1 = plt.imread('/kaggle/input/animals10/raw-img/cane/OIF-e2bexWrojgtQnAPPcUfOWQ.jpeg')
image2 = plt.imread('/kaggle/input/animals10/raw-img/cane/OIP---A27bIBcUgX1qkbpZOPswHaFS.jpeg')



# Resize image2 to match the shape of image1
image2_resized = resize(image2, (image1.shape[0], image1.shape[1]), anti_aliasing=True)

# Subtract images
motion_difference = np.abs(image1.astype(np.int16) - (image2_resized * 255).astype(np.int16))

# Display the result
plt.subplot(1, 3, 1)
plt.imshow(image1)
plt.axis('off')
plt.title('Image 1')

# Display the result
plt.subplot(1, 3,2)
plt.imshow(image2)
plt.axis('off')
plt.title('Image 2')
plt.subplot(1, 3, 3)

plt.imshow(motion_difference.astype(np.uint8))
plt.axis('off')
plt.title('Motion Difference')

plt.show()
```



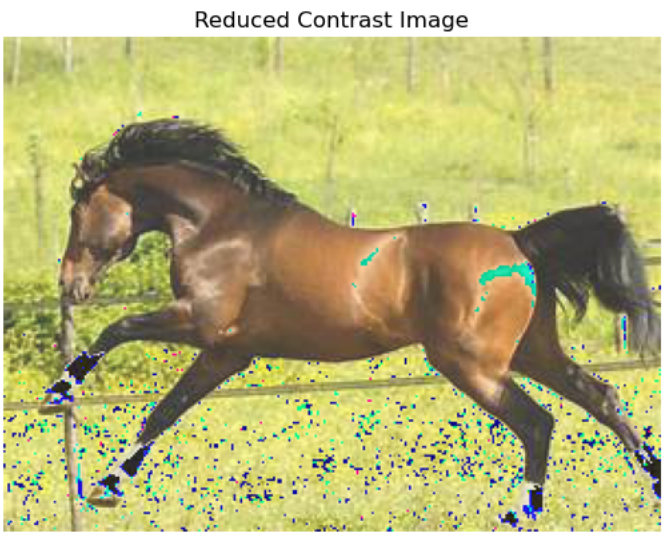
Darker Image

```
darker_image = (image * 0.5).astype(np.uint8)
plt.imshow(darker_image)
plt.axis('off')
plt.title('Darker Image')
plt.show()
```



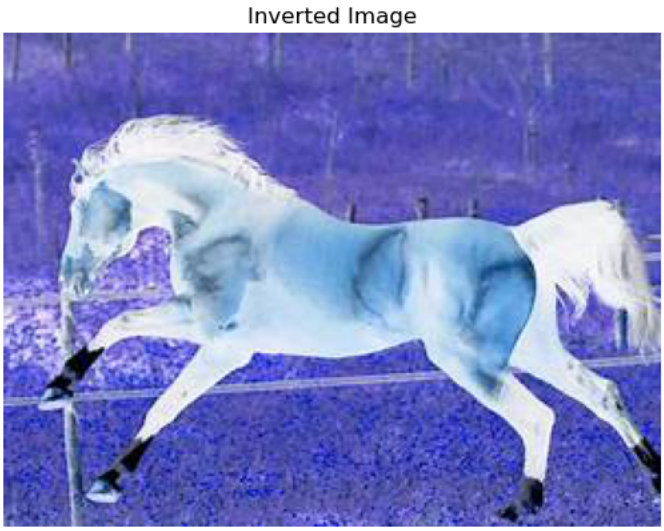
Reduced Contract

```
reduced_contrast_image = np.clip(image + 50, 0, 255).astype(np.uint8)  # Increase pixel values
plt.imshow(reduced_contrast_image)
plt.axis('off')
plt.title('Reduced Contrast Image')
plt.show()
```



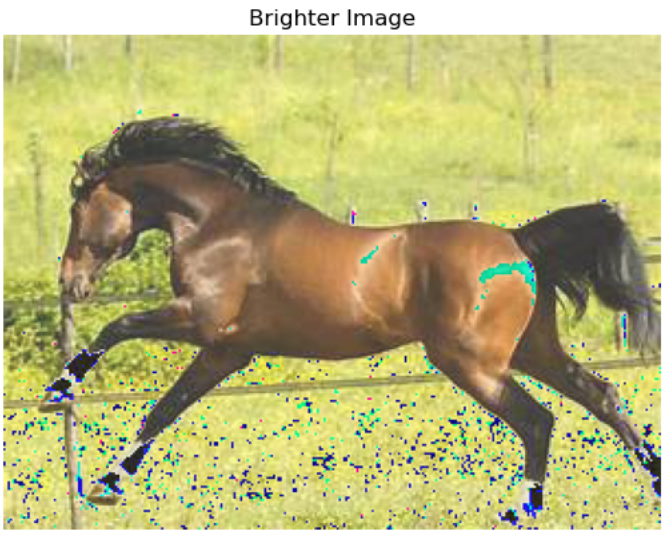
Inverted Image

```
inverted_image = 255 - image
plt.imshow(inverted_image)
plt.axis('off')
plt.title('Inverted Image')
plt.show()
```



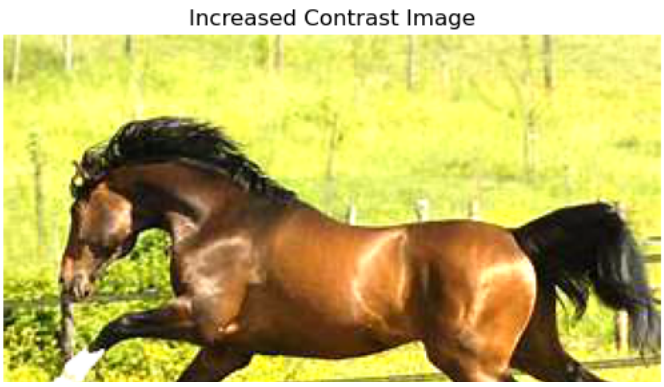
Brighter Image

```
brighter_image = np.clip(image + 50, 0, 255).astype(np.uint8)  # Increase pixel values
plt.imshow(brighter_image)
plt.axis('off')
plt.title('Brighter Image')
plt.show()
```



Increased contrast

```
increased_contrast_image = np.clip(image * 1.5, 0, 255).astype(np.uint8)  # Increase pixel values
plt.imshow(increased_contrast_image)
plt.axis('off')
plt.title('Increased Contrast Image')
plt.show()
```


Increased Contrast Image