IMAGE PROCESSING SECTIONS

```python
from __future__ import print_function
import cv2

# read the image
image = cv2.imread('image/ziad.jpg')

# save the image in a new path "newimage.jpg"
cv2.imwrite("newimage.jpg", image)

# numpy array for the image details
print("height: {} pixels".format(image.shape[0]))
print("width: {} pixels".format(image.shape[1]))
print("channels: {}".format(image.shape[2]))

# display the image
cv2.imshow("Image", image)
cv2.waitKey(0)
```

CHAPTER 3

```python
from __future__ import print_function
import cv2

image = cv2.imread('image/ziad.jpg')

# to get color of a pixel as BGR
(b, g, r) = image[0, 20]
print('red: {} , green: {}, blue: {}'.format(r, g, b))

# to take a part of pixels from an image
corner = image[0:100 , 0:100]

# to change a color of an image part, give it a color
#                       B  G  R
image[0:100 , 0:100] = (0, 0, 255)
cv2.imshow("Corner", image)
cv2.waitKey(0)
```

CHAPTER 4

- **To draw line:**

```python
import cv2
import numpy as np

# numpy array with zeros values and range 300 x 300
canvas = np.zeros((300, 300, 3), dtype='uint8')

# to draw line in the canvas numpy array
cv2.line(canvas, (0, 0), (300, 300), (0, 0, 255), 4)
cv2.line(canvas, (0, 300), (300, 0), (0, 255, 0), 4)
cv2.line(canvas, (150, 0), (150, 300), (255, 0, 0), 4)
cv2.imshow("Line", canvas)
cv2.waitKey(0)

```

CHAPTER 5

- To draw rectangle:

```python
import cv2
import numpy as np

# numpy array with zeros values and range 300 x 300
canvas = np.zeros((300, 300, 3), dtype='uint8')

# to draw rectangle in the canvas numpy array
aqua = (0,255,255)
cv2.rectangle(canvas, (100, 100), (200, 150), aqua, 6)

cv2.imshow("Line", canvas)
cv2.waitKey(0)

```

CHAPTER 5

- To draw circle:

```python
import cv2
import numpy as np

canvas = np.zeros((300, 300, 3), dtype = "uint8")

# center of the circle
(centerX, centerY) = (canvas.shape[1] // 2, canvas.shape[0] // 2)
white = (255, 255, 255)

# loop to draw circles ===> range of radius += 25
for r in range(0, 175, 25):
    cv2.circle(canvas, (centerX, centerY), r, white, 3)

cv2.imshow("Canvas", canvas)
cv2.waitKey(0)
```

CHAPTER 5

```python
import cv2
import numpy as np

canvas = np.zeros((300, 300, 3), dtype='uint8')

# loop to draw 25 circles
for i in range(0,25):
    # to put random radius between 5 and 199 pixels
    radius=np.random.randint(5,high=200)

    # to put random color RGB
    color=np.random.randint(0,high=256,size =(3,)).tolist()

    # to put random circle center point
    pt=np.random.randint(0,high=300,size =(2,))

    cv2.circle(canvas,tuple(pt),radius,color,-1)


cv2.imshow("Canvas",canvas)
cv2.waitKey(0)
```

CHAPTER 5

- To draw random circle with random RGB and Center point:

CHAPTER 6

To apply translation:

- **Negative values** of tx will shift the image to the left and **Positive values** will shift the image to the right.
- **Negative value** of ty will shift the image up and **positive values** will shift the image down

```python
import cv2
import numpy as np

# to display the original image
image = cv2.imread('image/ziad.jpg')
cv2.imshow("Original", image)
cv2.waitKey(0)

# shifting (translating)
#                       tx          ty
m = np.float32([[1, 0, 25], [0, 1, 50]])
# to apply this transformation to the image
shifted = cv2.warpAffine(image, m, (image.shape[1], image.shape[0]))
cv2.imshow("Shifted Down and Right", shifted)
cv2.waitKey(0)

m = np.float32([[1, 0,-50], [0, 1,-90]])
shifted = cv2.warpAffine(image, m, (image.shape[1], image.shape[0]))
cv2.imshow("Shifted Up and Left", shifted)
cv2.waitKey(0)

```

CHAPTER 6

To apply rotation:

- The scale :
- Value = 1.0: means the same dimensions of the image are used.
- Value = 2.0: the image would be doubled in size.
- Value = 0.5: halves the size of the image
- The rotation degree:
- Negative value: clockwise
  مع عقارب الساعة

- Positive value: vise verse
  العكس صحيح

```python
import cv2
import numpy as np

image = cv2.imread('image/ziad.jpg')
cv2.imshow("Original", image)

(h, w) = image.shape[:2]
center = (w // 2, h // 2)

# getRotationMatrix2D(Rotation point, Rotation degree, scale)
M = cv2.getRotationMatrix2D(center, 45, 1.0)

# to apply rotation
rotated = cv2.warpAffine(image, M, (w, h))
cv2.imshow("Rotated by 45 Degrees", rotated)
cv2.waitKey(0)

M = cv2.getRotationMatrix2D(center,-90, 1.0)
rotated = cv2.warpAffine(image, M, (w, h))
cv2.imshow("Rotated by-90 Degrees", rotated)
cv2.waitKey(0)

```

CHAPTER 6

- To apply resize width:

```python
import cv2
import numpy as np

image = cv2.imread('image/ziad.jpg')
cv2.imshow("Original", image)

# r = new width / original width => resized ratio
r = 150.0 / image.shape[1]

# dim = (new_width, new_height)
dim = (150, int(image.shape[0] * r))

# cv2.INTER_AREA is a good choice for shrinking (resizing smaller).
resized = cv2.resize(image, dim, interpolation = cv2.INTER_AREA)
cv2.imshow("Resized (Width)", resized)
cv2.waitKey(0)
```

CHAPTER 6

- **To resize height:**

```python
import cv2
import numpy as np

image = cv2.imread('image/ziad.jpg')
cv2.imshow("Original", image)

# r = new height / original height => resized ratio
r = 50.0 / image.shape[0]

# dim = (new_width, new_height)
dim = (int(image.shape[1] * r), 50)

# cv2.INTER_AREA is a good choice for shrinking (resizing smaller).
resized = cv2.resize(image, dim, interpolation = cv2.INTER_AREA)
cv2.imshow("Resized (Height)", resized)
cv2.waitKey(0)
```

CHAPTER 6

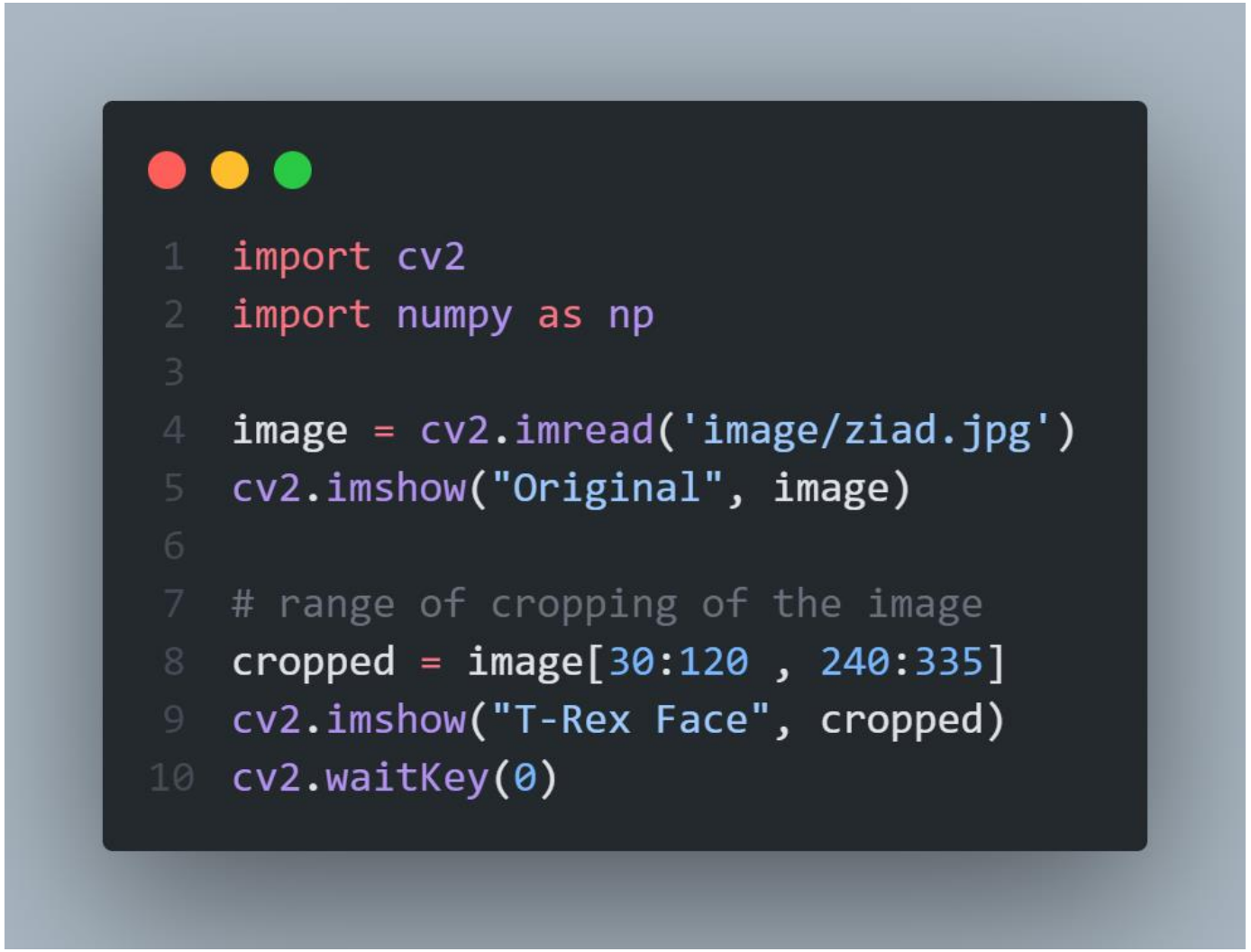
- **To apply flipping:**

```python
import cv2
import numpy as np

image = cv2.imread('image/ziad.jpg')
cv2.imshow("Original", image)

# to filp horizontally == 1
flipped = cv2.flip(image, 1)
cv2.imshow("Flipped Horizontally", flipped)
cv2.waitKey(0)

# to flip vertically ==== 0
flipped = cv2.flip(image, 0)
cv2.imshow("Flipped Vertically", flipped)
cv2.waitKey(0)

# to flip both horizontally and vertically == -1
flipped = cv2.flip(image,-1)
cv2.imshow("Flipped Horizontally & Vertically", flipped)
cv2.waitKey(0)
```

CHAPTER 6

cropping:

- Start y: The starting y coordinate. y = 30.
- End y: The ending y coordinate. y = 120.
- Start x: The starting x coordinate. x = 240.
- End x: The ending x-axis coordinate. x = 335.



```python
import cv2
import numpy as np

image = cv2.imread('image/ziad.jpg')
cv2.imshow("Original", image)

# range of cropping of the image
cropped = image[30:120 , 240:335]
cv2.imshow("T-Rex Face", cropped)
cv2.waitKey(0)
```

CHAPTER 6

- **Arithmetic:**

```python
from __future__ import print_function
import cv2
import numpy as np

image = cv2.imread('image/ziad.jpg')
cv2.imshow("Original", image)

# add 100 to every pixel in the image
M = np.ones(image.shape, dtype = "uint8") * 100
added = cv2.add(image, M)
cv2.imshow("Added", added)
cv2.waitKey(0)

# subtract 50 from every pixel in the image
M = np.ones(image.shape, dtype = "uint8") * 50
subtracted = cv2.subtract(image, M)
cv2.imshow("Subtracted", subtracted)
cv2.waitKey(0)
```

CHAPTER 6

- Bitwise:

Let's quickly review our binary operations:

1. **AND:** A bitwise AND is true if and only if both pixels are greater than zero.
2. **OR:** A bitwise OR is true if either of the two pixels are greater than zero.
3. **XOR:** A bitwise XOR is true if and only if *either* of the two pixels are greater than zero, but not both.
4. **NOT:** A bitwise NOT inverts the "on" and "off" pixels in an image.

```python
from __future__ import print_function
import cv2
import numpy as np

rectangle = np.zeros((300, 300), dtype = "uint8")
cv2.rectangle(rectangle, (25, 25), (275, 275), 255,-1)

circle = np.zeros((300, 300), dtype = "uint8")
cv2.circle(circle, (150, 150), 150, 255,-1)

bitwiseAnd = cv2.bitwise_and(rectangle, circle)
cv2.imshow("AND", bitwiseAnd)
cv2.waitKey(0)

bitwiseOr = cv2.bitwise_or(rectangle, circle)
cv2.imshow("OR", bitwiseOr)
cv2.waitKey(0)

bitwiseXor = cv2.bitwise_xor(rectangle, circle)
cv2.imshow("XOR", bitwiseXor)
cv2.waitKey(0)

bitwiseNot = cv2.bitwise_not(circle)
cv2.imshow("NOT", bitwiseNot)
cv2.waitKey(0)
```

CHAPTER 6

```python
import cv2
import numpy as np

image = cv2.imread('image/ziad.jpg')
cv2.imshow("Original", image)

mask = np.zeros(image.shape[:2], dtype = "uint8")
(cX, cY) = (image.shape[1] // 2, image.shape[0] // 2)
cv2.rectangle(mask, (cX- 75, cY- 75), (cX + 75 , cY + 75), 255,-1)
cv2.imshow("Mask", mask)
cv2.waitKey(0)

masked = cv2.bitwise_and(image, image, mask = mask)
cv2.imshow("Mask Applied to Image", masked)
cv2.waitKey(0)
```

- Masking rectangle:

CHAPTER 6

```python
import cv2
import numpy as np

image = cv2.imread('image/ziad.jpg')
cv2.imshow("Original", image)

mask = np.zeros(image.shape[:2], dtype = "uint8")
(cX, cY) = (image.shape[1] // 2, image.shape[0] // 2)
cv2.circle(mask, (cX, cY), 100, 255,-1)
masked = cv2.bitwise_and(image, image, mask = mask)
cv2.imshow("Mask", mask)
cv2.imshow("Mask Applied to Image", masked)
cv2.waitKey(0)
```

- Masking circle:

CHAPTER 6

- **Splitting and merging channels:**

```python
import cv2
import numpy as np

image = cv2.imread('image/ziad.jpg')
cv2.imshow("Original", image)

# split image to its colors
(B, G, R) = cv2.split(image)

cv2.imshow("Red", R)
cv2.imshow("Green", G)
cv2.imshow("Blue", B)
cv2.waitKey(0)

# merging
merged = cv2.merge([B, G, R])
cv2.imshow("Merged", merged)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

CHAPTER 6

Spliting:

```python
import cv2
import numpy as np

image = cv2.imread('image/ziad.jpg')
cv2.imshow("Original", image)

# # split image to its colors
(B, G, R) = cv2.split(image)

zeros = np.zeros(image.shape[:2], dtype = "uint8")
cv2.imshow("Red", cv2.merge([zeros, zeros, R]))
cv2.imshow("Green", cv2.merge([zeros, G, zeros]))
cv2.imshow("Blue", cv2.merge([B, zeros, zeros]))
cv2.waitKey(0)
```

CHAPTER 6

- **Color spaces:**

```python
import cv2
import numpy as np

image = cv2.imread('image/ziad.jpg')
cv2.imshow("Original", image)

gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
cv2.imshow("Gray", gray)

hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
cv2.imshow("HSV", hsv)

lab = cv2.cvtColor(image, cv2.COLOR_BGR2LAB)
cv2.imshow("L*a*b*", lab)
cv2.waitKey(0)
```

CHAPTER 10

- **Laplacian and Sobel:**

```python
import cv2
import numpy as np

image = cv2.imread('image/ziad.jpg')
image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
cv2.imshow("GrayScale", image)

lap = cv2.Laplacian(image, cv2.CV_64F)
lap = np.uint8(np.absolute(lap))
cv2.imshow("Laplacian", lap)
cv2.waitKey(0)

sobelX = cv2.Sobel(image, cv2.CV_64F, 1, 0)
sobelY = cv2.Sobel(image, cv2.CV_64F, 0, 1)
sobelX = np.uint8(np.absolute(sobelX))
sobelY = np.uint8(np.absolute(sobelY))
sobelCombined = cv2.bitwise_or(sobelX, sobelY)
cv2.imshow("Sobel X", sobelX)
cv2.imshow("Sobel Y", sobelY)
cv2.imshow("Sobel Combined", sobelCombined)
cv2.waitKey(0)
```

CHAPTER 10

```python
import cv2
import numpy as np

image = cv2.imread('image/ziad.jpg')

image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
image = cv2.GaussianBlur(image, (5, 5), 0)
cv2.imshow("Blurred", image)

canny = cv2.Canny(image, 30, 150)
cv2.imshow("Canny", canny)
cv2.waitKey(0)
```

- Canny:

THANKS