Accessing Pixels and Drawing Shapes

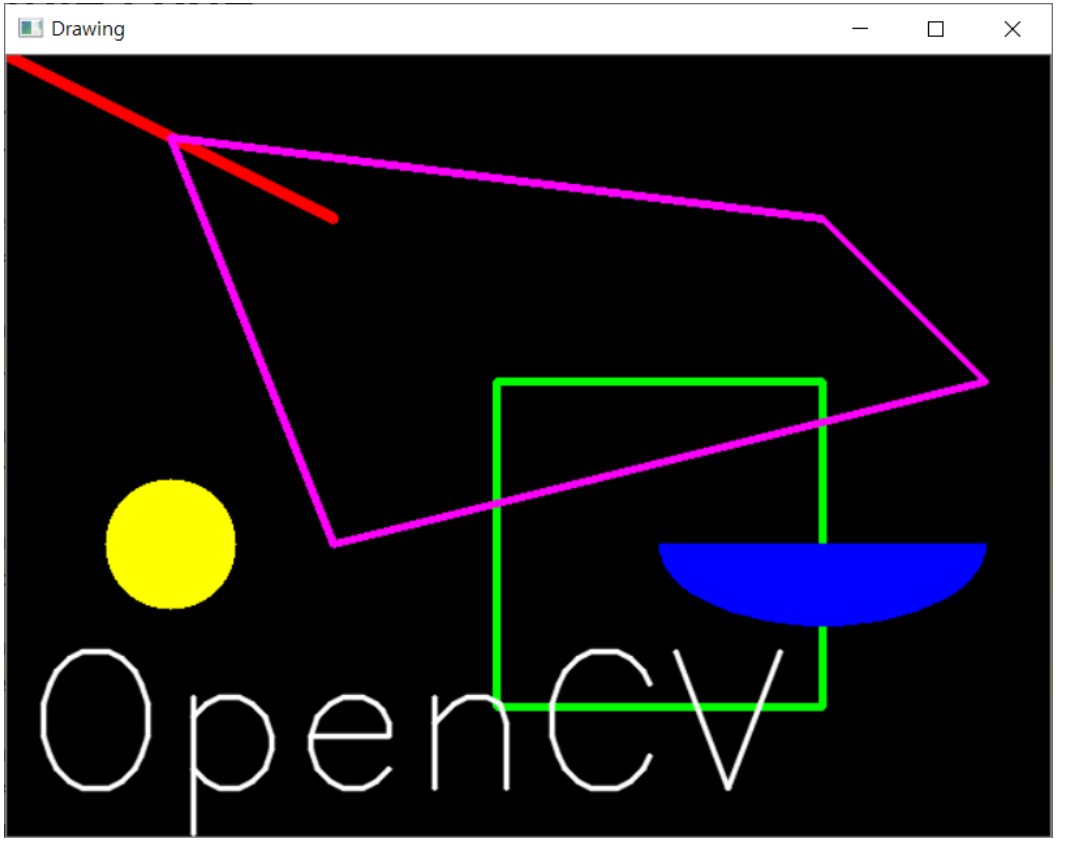
학습목표

학습목표

1. 이미지 픽셀의 값을 읽고 쓰는 방법 방법을 이해한다.
2. 이미지에 다양한 모양의 도형을 그리는 방법을 이해한다.
3. 이미지에 글자를 출력하는 방법을 이해한다.

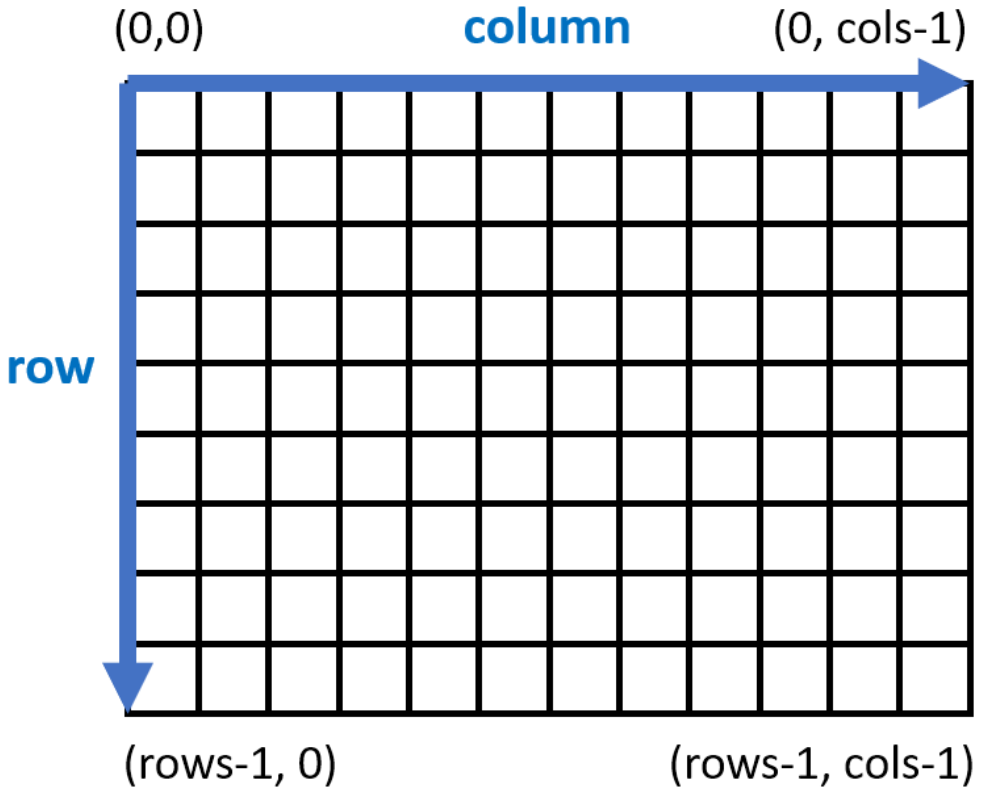
- 왜? 인식결과 등을 이미지 위에 표시하기 위해서!

학습목표



Accessing Pixels

Array Index



Pixel 값에 access할 때는 array index를 이용한다!

ndarray.item()[1]

```
value = ndarray.item(args)
```

```
>>> np.random.seed(123)
>>> x = np.random.randint(9, size=(3, 3))
>>> x
array([[2, 2, 6],
       [1, 3, 6],
       [1, 0, 1]])

>>> x[1][2]
6
>>> x[2,1]
0

>>> x.item(3)
1
>>> x.item((0, 1))
2
```

1. https://numpy.org/doc/stable/reference/generated/numpy.ndarray.item.html

ndarray.itemset()[1]

```
ndarray.itemset(args)
```

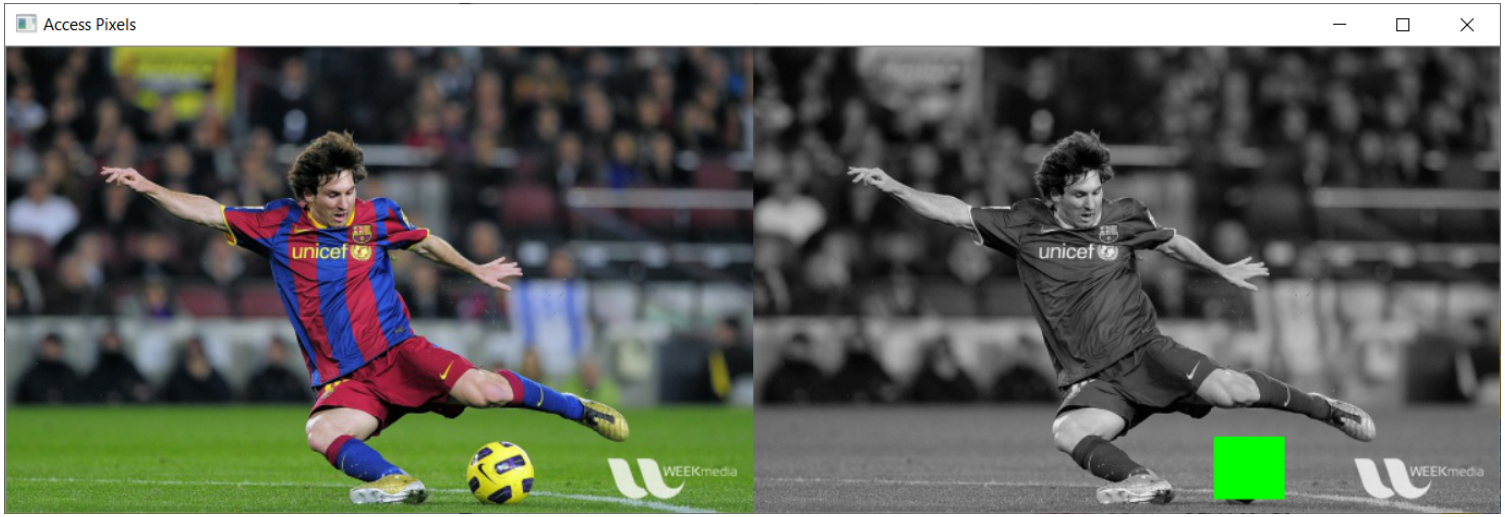
```
>>> np.random.seed(123)
>>> x = np.random.randint(9, size=(3, 3))
>>> x
array([[2, 2, 6],
       [1, 3, 6],
       [1, 0, 1]])

>>> x[2][1] = 3
>>> x[2,0] = 4

>>> x.itemset(4, 0)
>>> x.itemset((2, 2), 9)

>>> x
array([[2, 2, 6],
       [1, 0, 6],
       [4, 3, 9]])
```

1. https://numpy.org/doc/stable/reference/generated/numpy.ndarray.itemset.html

Example: Masking the Ball

```
import numpy as np
import cv2

# Load a color image
img_color = cv2.imread('messi5.jpg')

# Get the image size
rows, cols = img_color.shape[:2]

# Create a gray image with the same size
img_gray = np.zeros((rows, cols), np.uint8)

# Iterate over the whole image
for row in range(rows):
    for col in range(cols):
        # Read the pixel from the color image
        B = img_color.item(row, col, 0)
        G = img_color.item(row, col, 1)
        R = img_color.item(row, col, 2)

        # Convert it to gray
        gray = int(0.299*R + 0.587*G + 0.114*B)
```

Result: Masking the Ball



Double for-loop vs. `ndarray` slicing

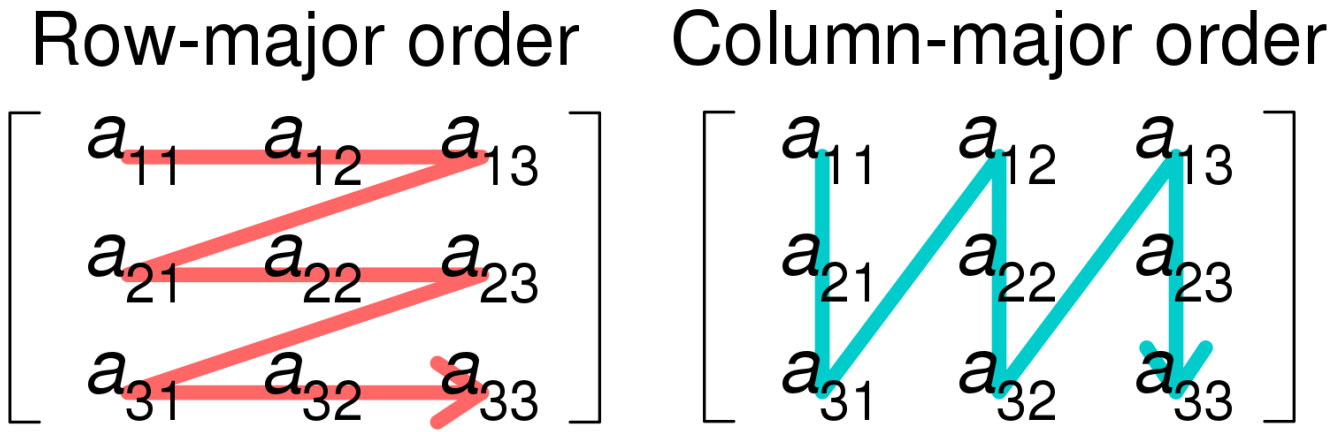
```
# Double for-loop
for row in range(286, 332):
   for col in range(338, 390):
      # Write the pixel in the result image
      img_result.itemset(row, col, 0,   0) # B
      img_result.itemset(row, col, 1, 255) # G
      img_result.itemset(row, col, 2,   0) # R

# NumPy ndarray slicing
mask = []
img_result[286:331, 338:389, 0] = 0
img_result[286:331, 338:389, 1] = 255
img_result[286:331, 338:389, 2] = 0
```

Which Iteration is Better?

```
# Iterate over the whole image
for row in range(rows):
    for col in range(cols):
        # Read the pixel from a gray image
        gray = img_gray.item(row, col)
```

```
# Iterate over the whole image
for col in range(cols):
    for row in range(rows):
        # Read the pixel from a gray image
        gray = img_gray.item(row, col)
```

Column-major vs. Row-major



https://en.wikipedia.org/wiki/Row-_and_column-major_order

Column-major vs. Row-major

Matrix Representation

0	1	2
3	4	5
6	7	8

Row-major

0	1	2	3	4	5	6	7	8

Column-major

0	3	6	1	4	7	2	5	8

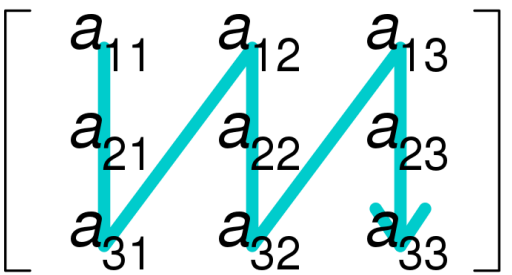
메모리에 어떤 순서로 저장하느냐의 문제!

Column-major vs. Row-major

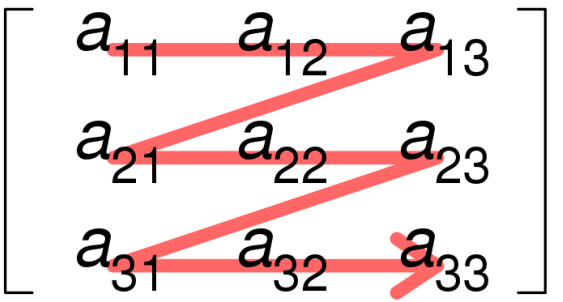
- Column-major
- OpenGL
- 왜?

- Row-major
- OpenCV
- 왜?

Column-major order

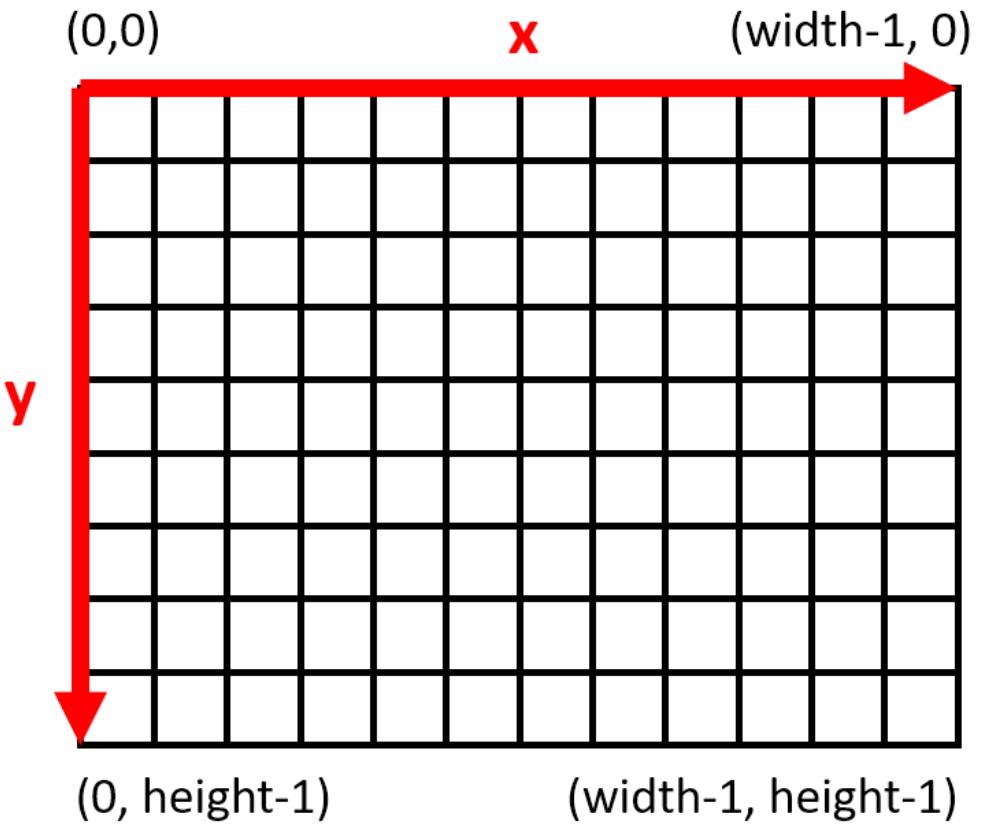


Row-major order



Drawing Shapes

Image Coordinates



도형을 그릴때는 Image Coordinates를 이용한다! 왜?

`cv.line()`[1]

```
img = cv.line(img, pt1, pt2, color[, thickness[, lineType[,
shift]]])
```

- 두 점을 잇는 직선을 그린다.
  - `img`: image
  - `pt1`: first point of the line segment
  - `pt2`: second point of the line segment
  - `color`: line color
  - `thickness`: line thickness
  - `lineType`[2]: line type
  - `shift`: number of fractional bits in the point coords

1. https://docs.opencv.org/4.4.0/d6/d6e/group__imgproc__draw.html#ga7078a9fae8c7e7d13d24dac2520ae4a2
2. https://docs.opencv.org/4.4.0/d6/d6e/group__imgproc__draw.html#gaf076ef45de481ac96e0ab3dc2c29a777

`cv.rectangle()`[1]

```
img = cv.rectangle(img, pt1, pt2, color[, thickness[, lineType[, shift]]])
```

- 두 점을 대각선의 끝점으로 하는 사각형을 그린다.
  - `img`: image
  - `pt1`: first point of the line segment
  - `pt2`: second point of the line segment
  - `color`: line color
  - `thickness`: line thickness
  - `lineType`[2]: line type
  - `shift`: number of fractional bits in the point coords

1. https://docs.opencv.org/4.4.0/d6/d6e/group__imgproc__draw.html#ga07d2f74cadcf8e305e810ce8eed13bc9
2. https://docs.opencv.org/4.4.0/d6/d6e/group__imgproc__draw.html#gaf076ef45de481ac96e0ab3dc2c29a777

`cv.circle()`[1]

```
img = cv.circle(img, center, radius, color[, thickness[, lineType[, shift]]])
```

- 원을 그린다.
  - `img`: image
  - `center`: center of the circle
  - `radius`: radius of the circle
  - `color`: line color
  - `thickness`: line thickness
  - `lineType`[2]: line type
  - `shift`: number of fractional bits in the point coords

1. https://docs.opencv.org/4.4.0/d6/d6e/group__imgproc__draw.html#ga28b2267d35786f5f890ca167236cbc69
2. https://docs.opencv.org/4.4.0/d6/d6e/group__imgproc__draw.html#gaf076ef45de481ac96e0ab3dc2c29a777

`cv.ellipse()`[1]

```
img = cv.ellipse(img, center, axes, angle, startAngle, endAngle,
color[, thickness[, lineType[, shift]]])
```

- 타원을 그린다.
  - `img`: image
  - `center`: center of the ellipse
  - `axes`: half of the size of the main axes
  - `angle`: rotation angle in degrees
  - `startAngle`: starting angle of the slliptic arc in degrees
  - `endAngle`: ending angle of the slliptic arc in degrees
  - `color`: line color
  - `thickness`: line thickness
  - `lineType`[2]: line type
  - `shift`: number of fractional bits in the point coords

1. https://docs.opencv.org/4.4.0/d6/d6e/group__imgproc__draw.html#gaf10604b069374903dbd0f0488cb43670
2. https://docs.opencv.org/4.4.0/d6/d6e/group__imgproc__draw.html#gaf076ef45de481ac96e0ab3dc2c29a777

`cv.polylines()`[1]

```
img = cv.polylines(img, pts, isClosed, color[, thickness[,
lineType[, shift]]])
```

- 점들을 잇는 다각형을 그린다.
  - `img`: image
  - `pts`: array of points
  - `isClosed`: whether the polylines are closed or not
  - `color`: line color
  - `thickness`: line thickness
  - `lineType`[2]: line type
  - `shift`: number of fractional bits in the point coords

1. https://docs.opencv.org/4.4.0/d6/d6e/group__imgproc__draw.html#ga1ea127ffbbb7e0bfc4fd6fd2eb64263c
2. https://docs.opencv.org/4.4.0/d6/d6e/group__imgproc__draw.html#gaf076ef45de481ac96e0ab3dc2c29a777

`cv.putText()`[1]

```
img = cv.putText(img, text, org, fontFace, fontScale, color[,
thickness[, lineType[, bottomLeftOrigin]]])
```

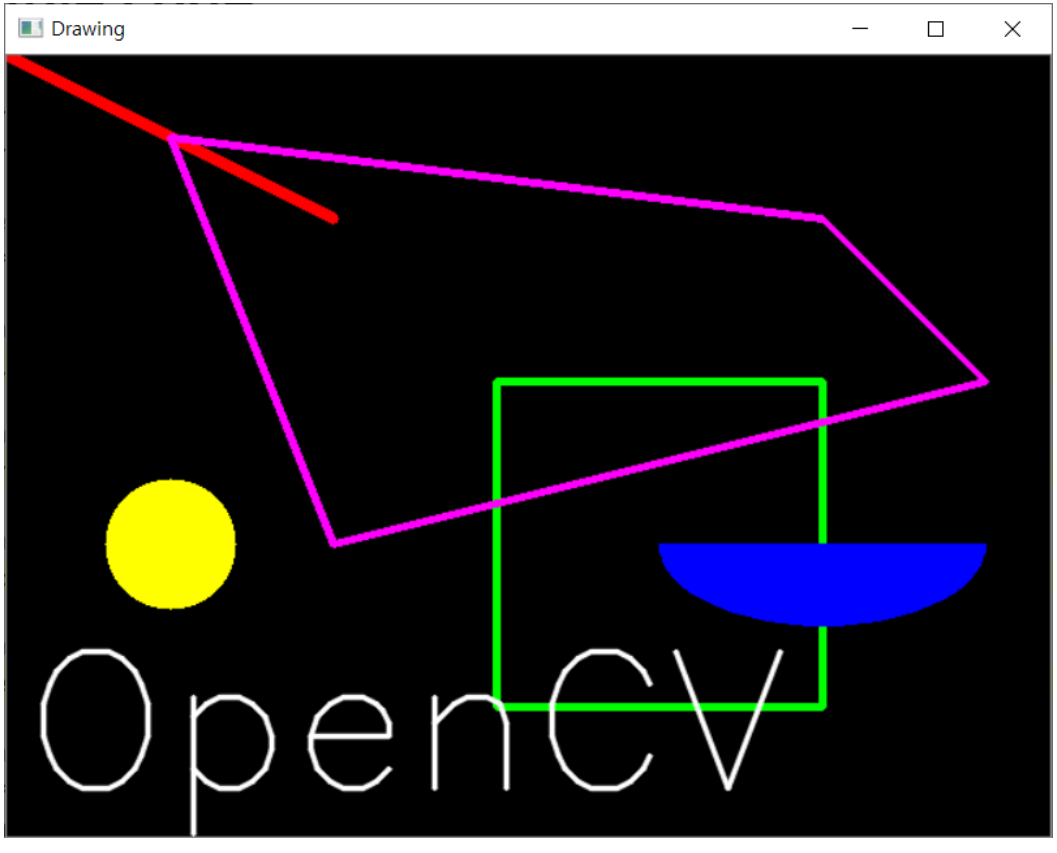
- 글자를 그린다.
  - `img`: image
  - `text`: text string to be drawn
  - `org`: bottom-left corner of the text string in the image
  - `fontFace`[2]: Font type
  - `fontScale`: font scale factor that is multiplied by the font-specific base size
  - `color`: line color
  - `thickness`: line thickness
  - `lineType`[3]: line type
  - `bottomLeftOrigin`: when true, the image data origin is at the bottom-left corner. Otherwise, it is at the top-right corner

1. https://docs.opencv.org/4.4.0/d6/d6e/group__imgproc__draw.html#ga5126f47f883d730f633d74f07456c576
2. https://docs.opencv.org/4.4.0/d6/d6e/group__imgproc__draw.html#ga0f9314ea6e35f99bb23f29567fc16e11
3. https://docs.opencv.org/4.4.0/d6/d6e/group__imgproc__draw.html#gaf076ef45de481ac96e0ab3dc2c29a777

Drawing Example

```
import numpy as np
import cv2

# Create a color image (black)
img = np.zeros((480, 640, 3), np.uint8)

# Draw a line
cv2.line(img, (0, 0), (200, 100), (0, 0, 255), 5)

# Draw a rectangle
cv2.rectangle(img, (300, 200), (500, 400), (0, 255, 0), 3)

# Draw a circle
cv2.circle(img, (100, 300), 40, (0, 255, 255), -1)

# Draw an ellipse
cv2.ellipse(img, (500, 300), (100, 50), 0, 0, 180, 255, -1)

# Draw a polygon
pts = np.array([[100, 50], [200, 300], [600, 200], [500, 100]],
np.int32)
pts = pts.reshape((-1,1,2))
```

Drawing Result



Push Code to GitHub

References

References

- OpenCV Python Tutorials
  - Core Operations
    - Basic Operations on Images
  - GUI Features in OpenCV
    - Drawing Functions in OpenCV