### 1.3.5 Drawing rectangle circle

**1. Drawing rectangle**

**Rectangle(img，pt1，pt2，color，thickness=None，lineType=None，shift=None)**

**Parameter Description**

- img: Canvas or carrier image.
- pt1, pt2： required parameters. Represents the vertices and diagonal vertices, that is the upper left corner and lower right corner of the rectangle.
- color: required parameters, be used to set color of rectangle.
- thickness: optional parameter, be used to set the width of the rectangle side, when the value is negative, it means to fill the rectangle.
- lineType： optional parameter, be used to set the type of line segment, 8 (8 adjacent connection line-default), 4 (4 adjacent connection line) and cv2.LINE_AA are anti-aliased.

**2. Drawing circle**

**cv2.circle(img, center, radius, color[,thickness[,lineType]])**

**Parameter Description**

- img: canvas or carrier image
- center: the coordinates of the center of the circle, the format: (50,50)
- radius: radius
- color: color
- Thickness: be used to set the width of the circle side, when the value is -1, it means to fill the circle.
- lineType: type of line segment. The default is 8, the connection type description is shown in the table below.

| Parameter | Description |
|---|---|
| cv2.FILLED | fill |
| cv2.LINE_4 | 4 Connection Type |
| cv2.LINE_8 | 8 Connection Type |
| cv2.LINE_AA | Anti-aliasing, this parameter will make the line smoother |

**3. Draw ellipse**

**cv2.ellipse(img, center, axes, angle, StartAngle, endAngle, color[,thickness[,lineType]])**

**Parameter Description**

- center: Center point of ellipse(x, x)
- axes: hort radius and long radius, (x, x)
- angle: the angle of counterclockwise rotation
- StartAngle: The angle of the starting angle of the arc
- endAngle: The angle of the end angle of the arc

- Img, color: canvas or carrier image and color.

# The fifth parameter refers to the angle at which drawing starts counterclockwise, and the sixth parameter refers to the angle at which drawing ends counterclockwise
# If the fourth or fifth parameters are added with symbols, the opposite direction is indicated, that is, the clockwise direction.

**4. Draw polygon**

**cv2.polylines(img,[pts],isClosed, color[,thickness[,lineType]])**

- pts: Vertex of polygon
- isClosed: Whether closed or not. (True/False)
- Other parameters refer to the parameters of drawing circle

*Path:/home/dofbot/Dofbot\4.opencv\3.IP_Draw_text_line_segments\ 05_Drawing rectangle_circle.ipynb*

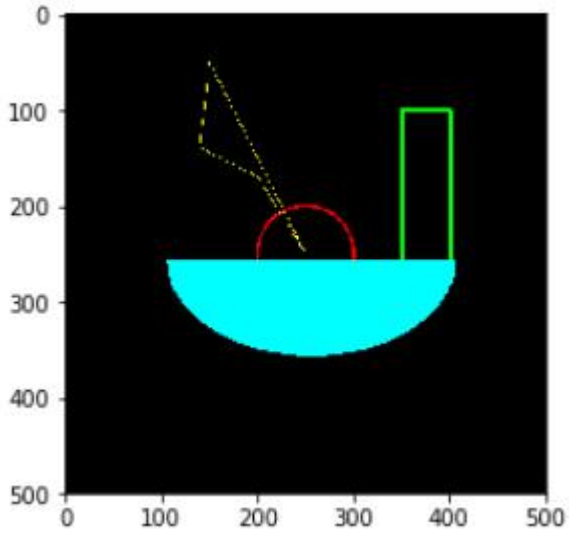
```
import cv2
import numpy as np
newImageInfo = (500,500,3)
dst = np.zeros(newImageInfo,np.uint8)
# 1 2 upper left corner 3 lower right corner 4 5 fill -1 >0 line w
cv2.rectangle(dst,(350,100),(400,270),(0,255,0),3)
# 2 center 3 r
cv2.circle(dst,(250,250),(50),(255,0,0),2)
# 2 center 3 axis 4 angle 5 begin 6 end 7
cv2.ellipse(dst, (256,256), (150,100), 0, 0, 180, (0,255,255), -1)

points = np.array([[150,50], [140,140], [200,170], [250,250], [150,50]], np.int32)
#print(points.shape)
points = points.reshape((-1,1,2))
#print(points.shape)
cv2.polylines(dst,[points],True,(255,255,0))
# cv2.imshow('dst',dst)
      # cv2.waitKey(0)
```

```
import matplotlib.pyplot as plt

dst = cv2.cvtColor(dst, cv2.COLOR_BGR2RGB)
plt.imshow(dst)
plt.show()
```

After running the following program, a picture will be displayed in the jupyterLab control interface, as shown below.


0
100
200
300
400
500
0
100
200
300
400
500