专业技能实训

授课教师：王洪松

邮箱：hongsongwang@seu.edu.cn

matplotlib.pyplot画图

数据可视化: Matplotlib

- 可视化库：matplotlib
  - 图中元素的确定均由Python代码完成
    - 点、线、轴
    - 标题、图例
    - 布局（子图）
  - 图片可直接输出到控制台，也可保存到本地图片

- 样例图库
  - https://matplotlib.org/2.0.2/gallery.html

- Gallery
  - Lines, bars, and markers
  - Shapes and collections
  - Statistical plots
  - Images, contours, and fields
  - Pie and polar charts
  - Color
  - Text, labels, and annotations
  - Ticks and spines
  - Axis scales
  - Subplots, axes, and figures
  - Style sheets
  - Specialty plots
  - Showcase
  - API
  - pylab examples
  - mplot3d toolkit
  - axes_grid toolkit
  - widgets
  - Miscellaneous examples

数据可视化: Matplotlib

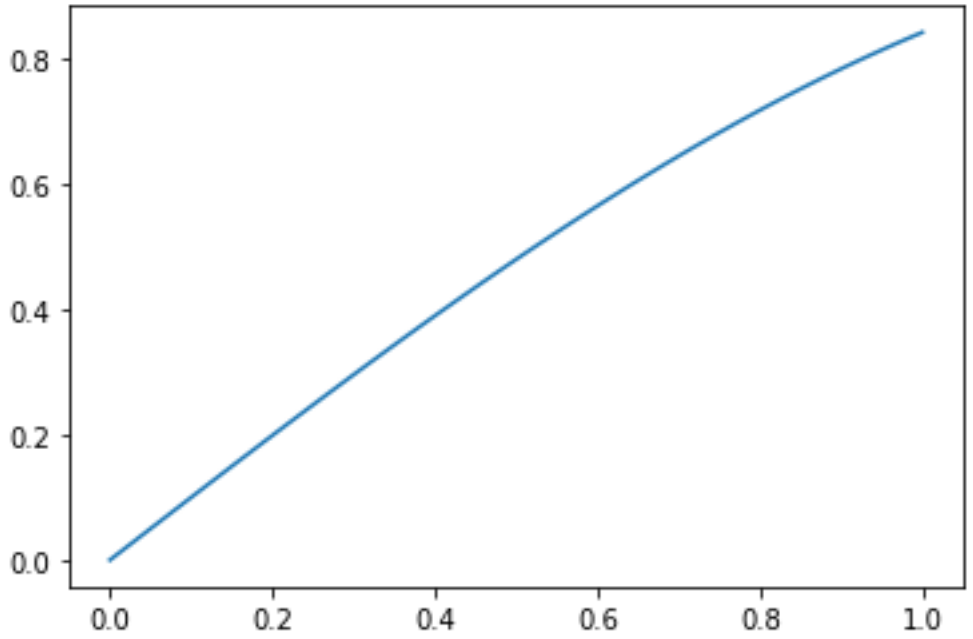
- matplotlib.pyplot
  - 提供和Matlab相似的绘图API，方便用户快速绘制2D图表
- pylab模块
  - 包含numpy和pyplot中常用的绘图函数，方便用户快速进行计算和绘图，常用于Ipython中的交互操作使用

```
import matplotlib.pyplot as plt
```

数据可视化: Matplotlib

- 快速绘图：点、线、轴

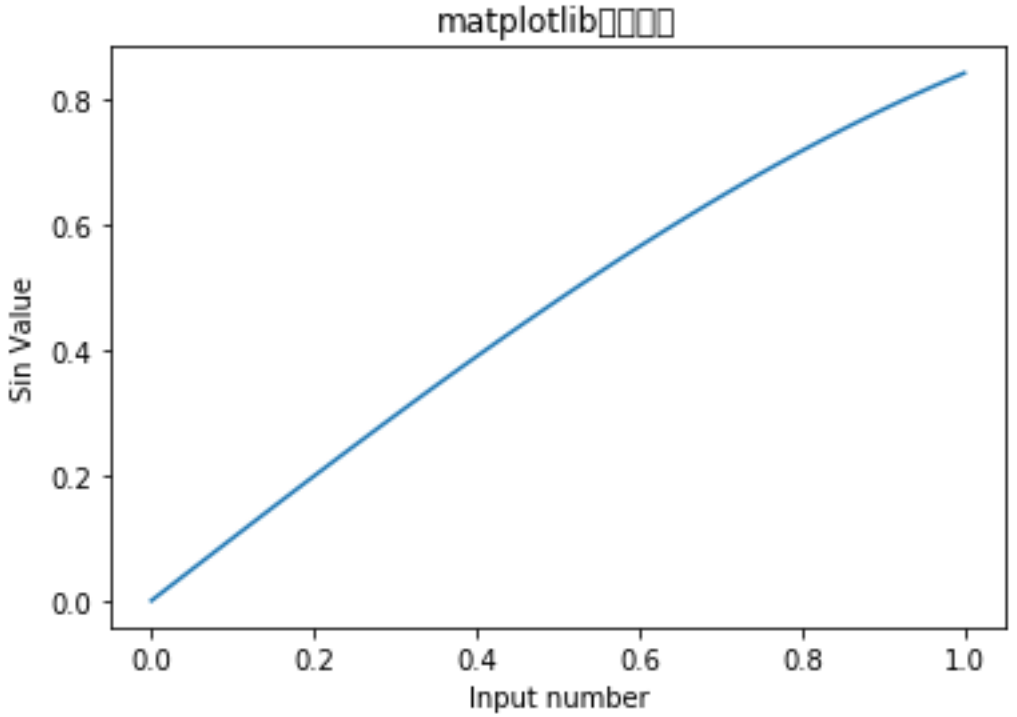
```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(0, 1, 101)
y= np.sin(x)
plt.plot(x,y)
plt.show()
```



数据可视化: Matplotlib

- 快速绘图：标题

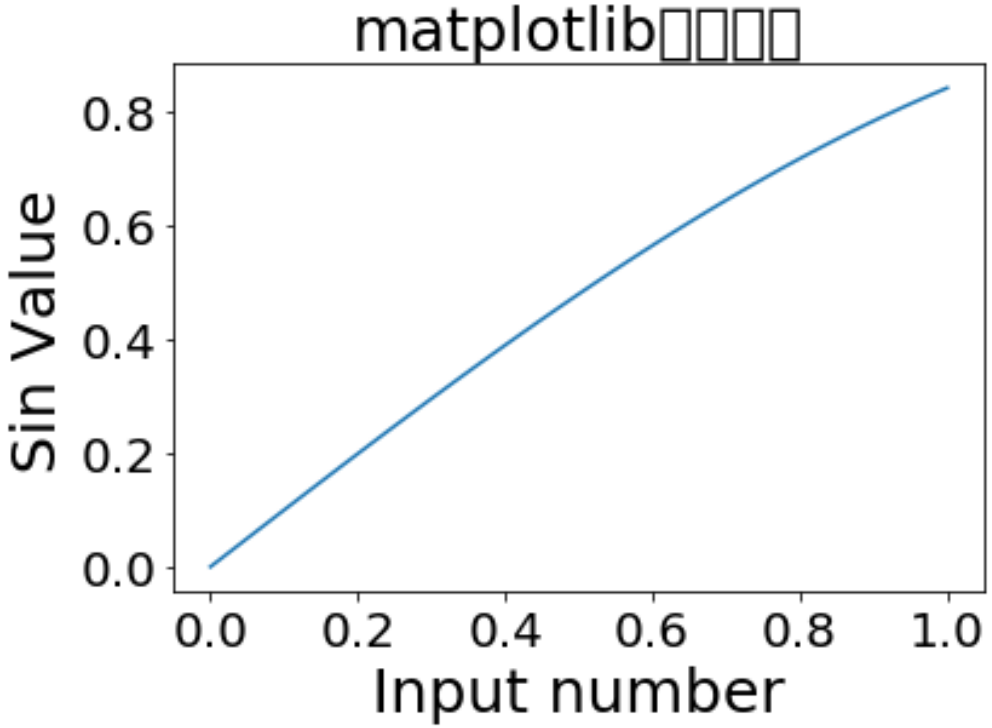
```
x = np.linspace(0, 1, 101)
y = np.sin(x)
plt.plot(x, y)
plt.xlabel('Input number')
plt.ylabel('sin Value')
plt.title('matplotlib快速绘图')
plt.show()
```



数据可视化: Matplotlib

- 快速绘图：标题字号

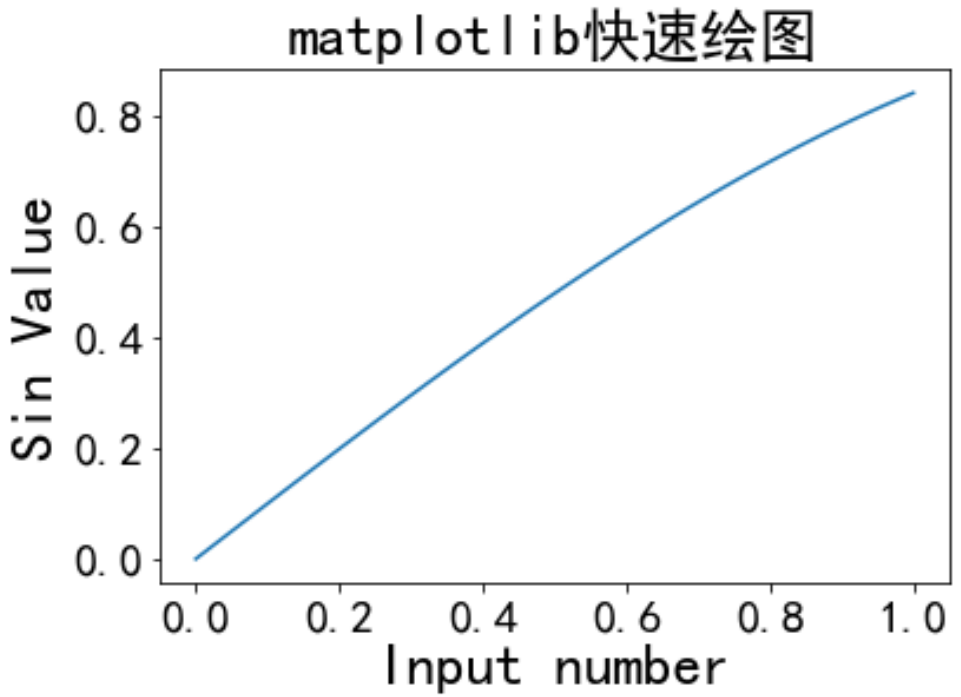
```
x = np.linspace(0, 1, 101)
y = np.sin(x)
plt.plot(x, y)
plt.xlabel('Input number', fontsize=25)
plt.ylabel('sin Value', fontsize=25)
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
plt.title('matplotlib快速绘图', fontsize=25)
plt.show()
```



数据可视化: Matplotlib

- 快速绘图：标题中文显示

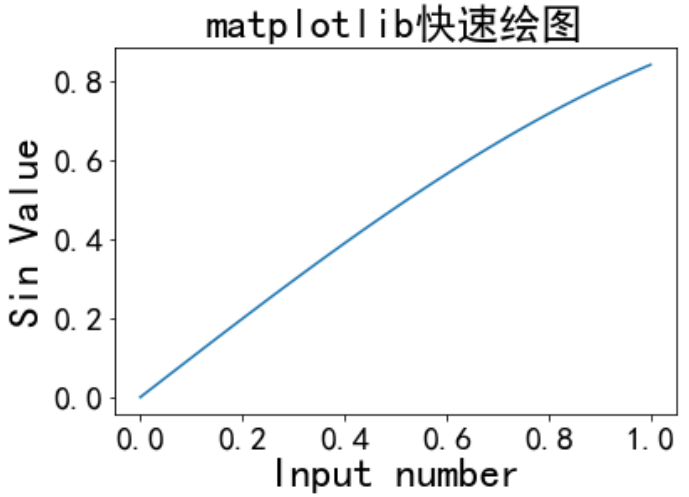
```
x = np.linspace(0, 1, 101)
y = np.sin(x)
plt.plot(x, y)
plt.xlabel('Input number', fontsize=25)
plt.ylabel('sin Value', fontsize=25)
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
plt.title('matplotlib快速绘图', fontsize=25)
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['font.serif'] = ['SimHei']
plt.show()
```



数据可视化: Matplotlib

- 快速绘图：标题中文显示

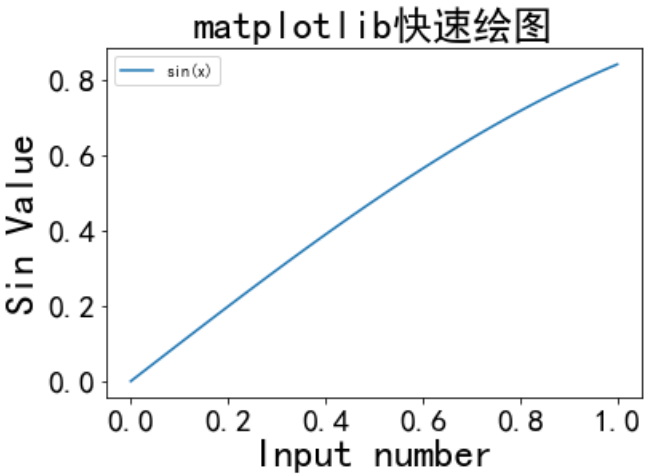
```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.font_manager as fm

myfont = fm.FontProperties(fname=r'C:/Windows/Fonts/simhei.ttf', size = 25)
x = np.linspace(0, 1, 101)
y = np.sin(x)
plt.plot(x, y)
plt.xlabel('Input number', fontsize=25)
plt.ylabel('sin Value', fontsize=25)
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
plt.title('matplotlib快速绘图', fontproperties=myfont)
plt.show()
```

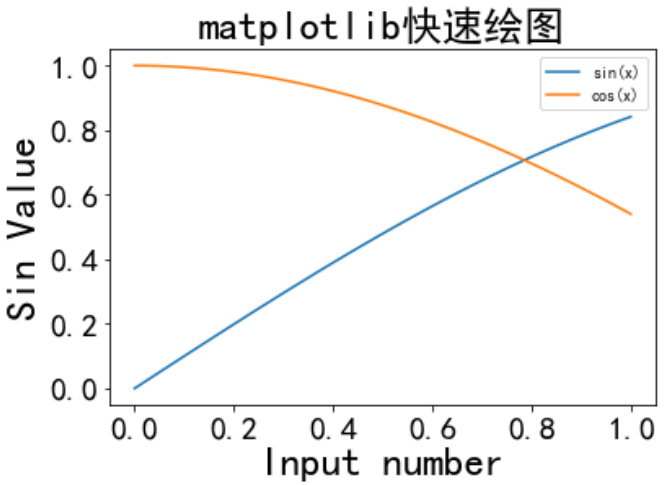


数据可视化: Matplotlib

- 快速绘图：图例

```
myfont = fm.FontProperties(fname=r'C:/Windows/Fonts/simhei.ttf',size = 25)
x = np.linspace(0, 1, 101)
y = np.sin(x)
z = np.cos(x)
plt.plot(x, y, label = 'sin(x)')
plt.plot(x, z, label = 'cos(x)')
plt.xlabel('Input number',fontsize=25)
plt.ylabel('Sin Value',fontsize=25)
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
plt.title(u'matplotlib快速绘图', fontproperties=myfont)
plt.legend()
plt.show()
```

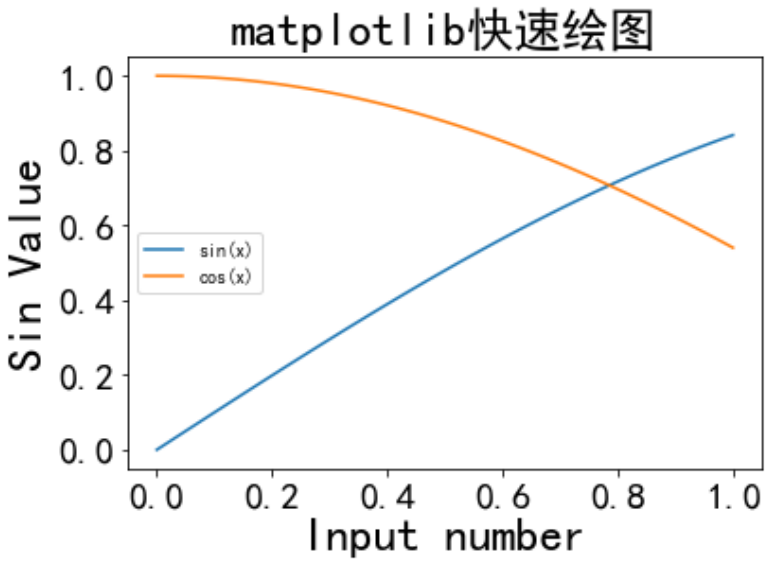




数据可视化: Matplotlib

- 快速绘图：图例位置

```
myfont = fm.FontProperties(fname=r'C:/Windows/Fonts/simhei.ttf',size = 25)
x = np.linspace(0, 1, 101)
y = np.sin(x)
z = np.cos(x)
plt.plot(x, y, label = 'sin(x)')
plt.plot(x, z, label = 'cos(x)')
plt.xlabel('Input number',fontsize=25)
plt.ylabel('Sin Value',fontsize=25)
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
plt.title(u'matplotlib快速绘图', fontproperties=myfont)
#plt.legend(loc='String or Number', bbox_to_anchor=(num1, num2))
plt.legend(loc=6)
plt.show()
```



数据可视化: Matplotlib

- 快速绘图：图例位置 plt.legend(loc= ‘String or Number’)

String	Number
best	**0**
upper right	1
upper left	2
lower left	3
lower right	4
right	5
center left	6
center right	7
lower center	8
upper center	9
center	10

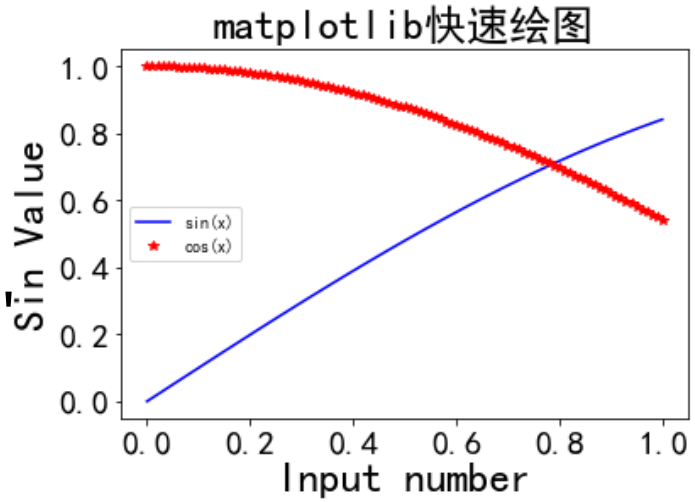
bbox_to_anchor=(num1, num2)

num1 用于控制 legend 的左右移动，值越大越向右边移动。
num2 用于控制 legend 的上下移动，值越大，越向上移动。

数据可视化: Matplotlib

- 快速绘图：颜色图标控制

```
myfont = fm.FontProperties(fname=r'C:/Windows/Fonts/simhei.ttf', size = 25)
x = np.linspace(0, 1, 101)
y = np.sin(x)
z = np.cos(x)
plt.plot(x, y, c='b',label ='sin(x)')
plt.plot(x, z, 'r*', label = 'cos(x)’ ) # cannot use c='r*'
plt.xlabel('Input number', fontsize=25)
plt.ylabel('sin Value', fontsize=25)
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
plt.title('matplotlib快速绘图', fontproperties=myfont)
plt.legend(loc=6)
plt.show()
```



数据可视化: Matplotlib

颜色（color 简写为 c）：
蓝色： 'b' (blue)
绿色： 'g' (green)
红色： 'r' (red)
蓝绿色(墨绿色)： 'c' (cyan)
红紫色(洋红)： 'm' (magenta)
黄色： 'y' (yellow)黑色： 'k' (black)
白色： 'w' (white)
灰度表示： e.g. 0.75 ([0,1]内任意浮点数)
RGB表示法： e.g. '#2F4F4F' 或 (0.18, 0.31, 0.31)

线型（linestyle 简写为 ls）：
实线： '-'
虚线： '--'
虚点线： '-.'
点线： ':'
点： '.'

数据可视化: Matplotlib

点型（标记marker）：

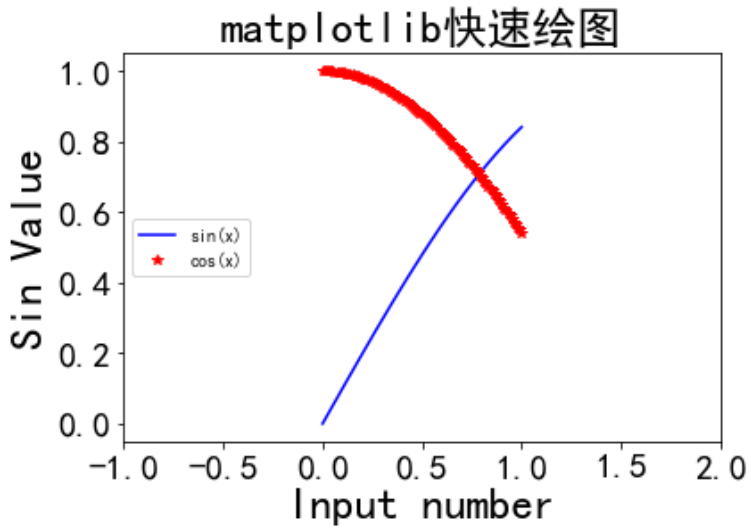
像素： ',’
圆形： 'o’
上三角： '^’
下三角： 'v’
左三角： '<‘
右三角： '>’
方形： 's’
加号： '+’ 叉
形： 'x’
棱形： 'D’
细棱形： 'd’
六角形： 'h’
旋转六角形： 'H’
五角形： 'p’
垂直线： '|’
水平线： '_

标记大小（markersize 简写为 ms）
markersize： 实数
标记边缘宽度（markeredgewidth 简写为 mew）
markeredgewidth：实数
实数标记边缘颜色（markeredgecolor 简写为 mec）
markeredgecolor：
透明度（alpha）
alpha： [0,1]之间的浮点数
线宽（linewidth）
linewidth： 实数

数据可视化: Matplotlib

- 快速绘图：坐标轴

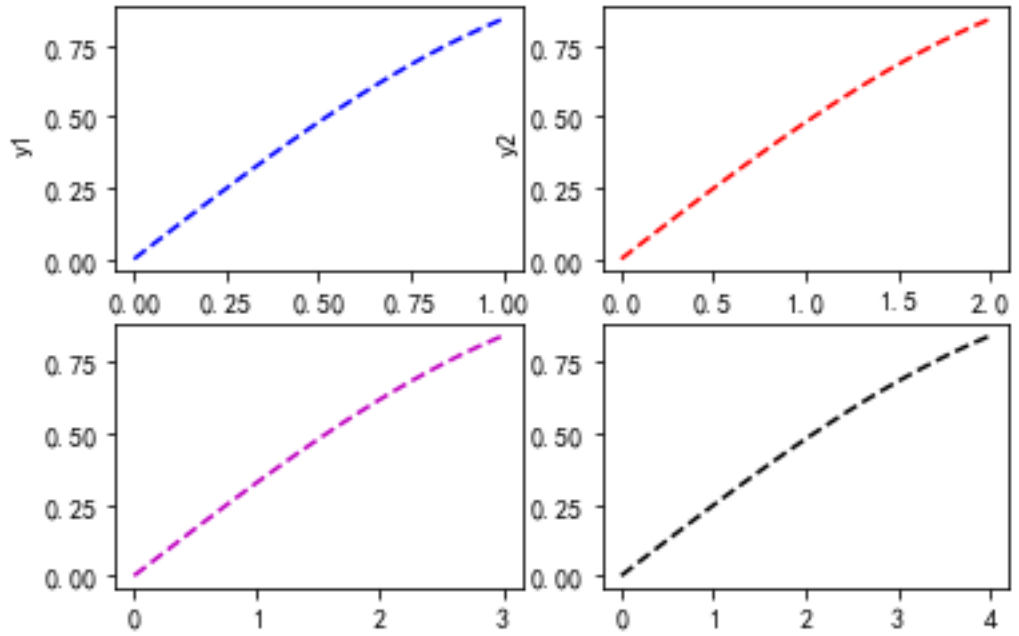
```
myfont = fm.FontProperties(fname=r'C:/Windows/Fonts/simhei.ttf',size = 25)
x = np.linspace(0, 1, 101)
y = np.sin(x)
z = np.cos(x)
plt.plot(x, y, c='b',label = 'sin(x)')
plt.plot(x, z, 'r*',label = 'cos(x)')
plt.xlabel('Input number',fontsize=25)
plt.ylabel('Sin Value',fontsize=25)
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
plt.title(u'matplotlib快速绘图', fontproperties=myfont)
plt.legend(loc=6)
plt.rcParams['axes.unicode_minus']=False #显示负号
plt.xlim(-1,2) #设置x轴最小和最大数
plt.show()
```



数据可视化: Matplotlib

- 快速绘图：布局

```
#要生成两行两列，这是第一个图plt.subplot('行','列','编号')
plt.subplot(2,2,1)
plt.plot(x,y,'b--')
plt.ylabel('y1')
plt.subplot(2,2,2) #两行两列,这是第二个图
plt.plot(2*x,y,'r--')
plt.ylabel('y2')
plt.subplot(2,2,3)#两行两列,这是第三个图
plt.plot(3*x,y,'m--')
plt.subplot(2,2,4)#两行两列,这是第四个图
plt.plot(4*x,y,'k--')
plt.show()
```



数据可视化: Matplotlib

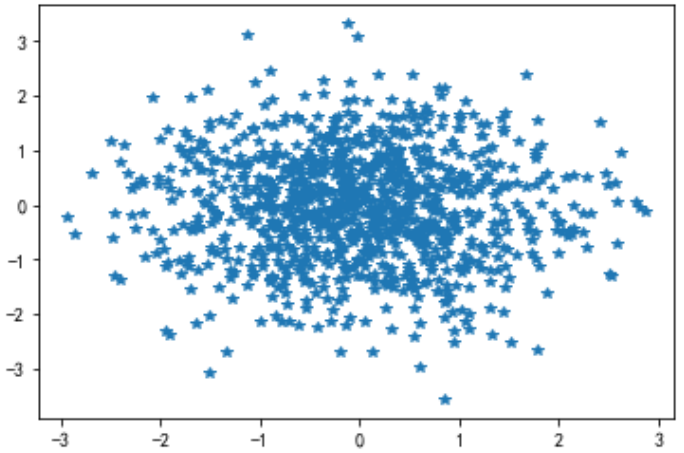
- 图片保存 plt.savefig(‘file.type’ , dpi=xxx)
  - 保存图片时可写明绝对路径
  - 保存图片的格式根据后缀名确定
  - 可以设置dpi参数

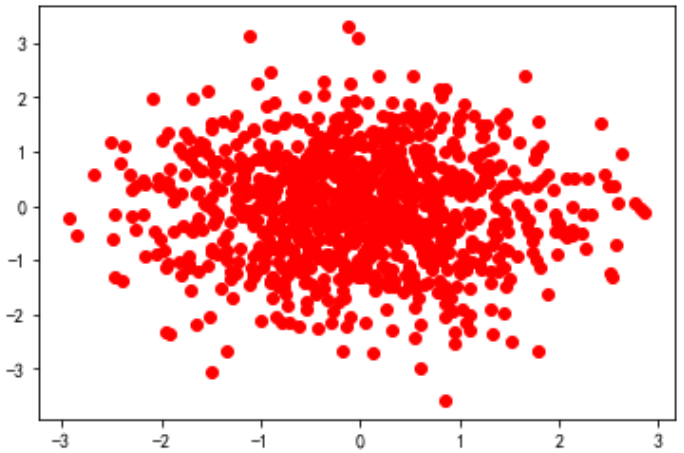
- fig = plt.figure(figsize=(a, b), dpi=dpi)
  - figsize 设置图形的大小，a 为图形的宽， b 为图形的高，单位为英寸
  - dpi 为设置图形每英寸的点数

数据可视化: Matplotlib

- 快速绘图：散点图（Scatter plot）

```
x = np.random.randn(1000)
y = np.random.randn(1000)
plt.plot(x, y, '*')
plt.show()

plt.scatter(x, y, c='r')
plt.show()
```

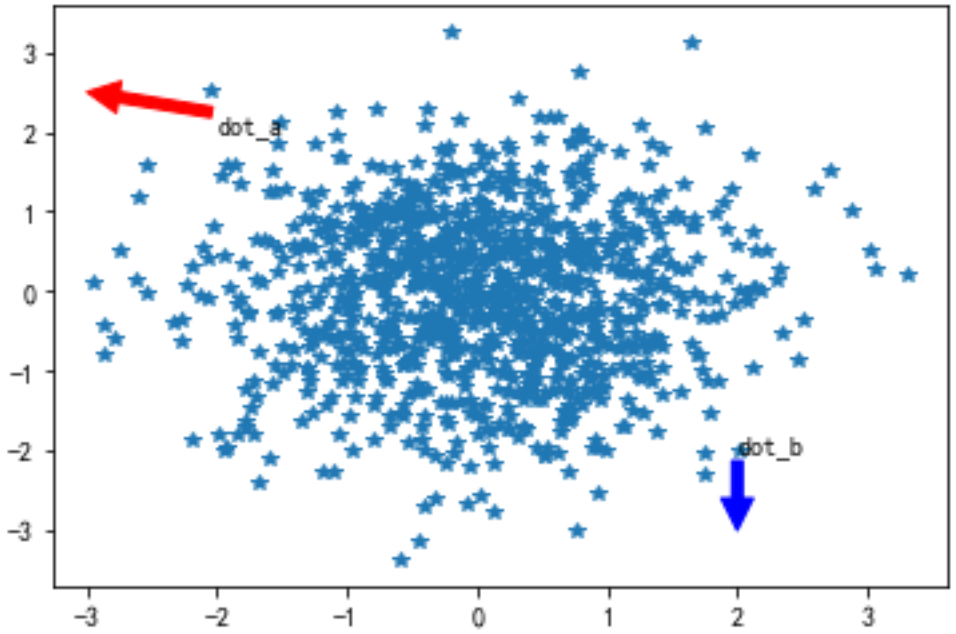




数据可视化: Matplotlib

- 快速绘图：图中标注

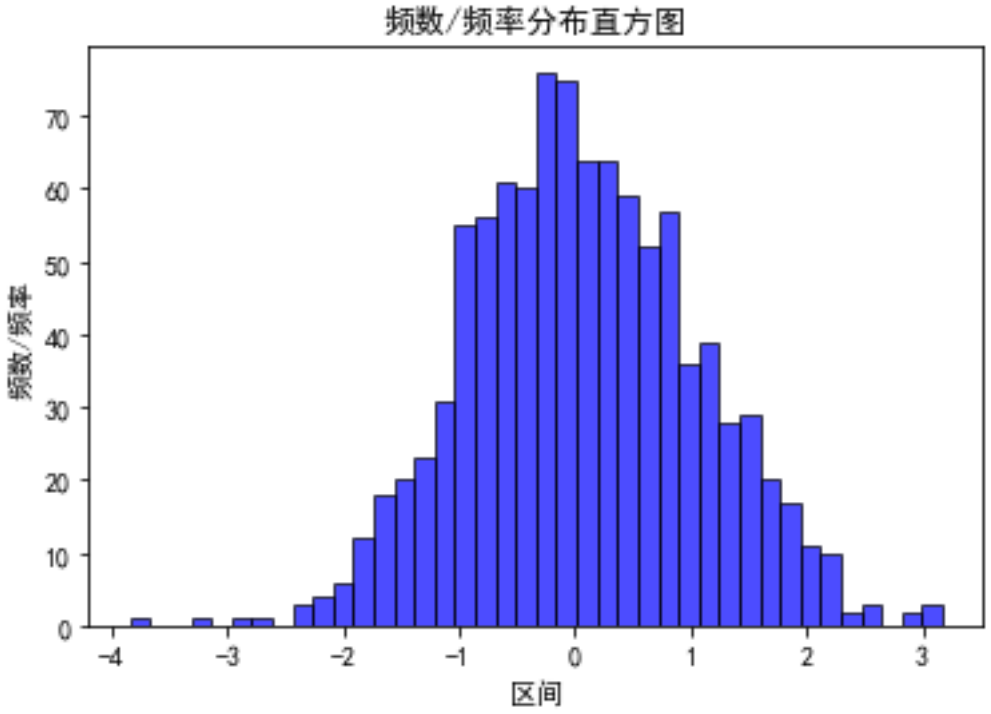
```
x = np.random.randn(1000)
y = np.random.randn(1000)
plt.plot(x, y, '*')
plt.annotate('dot_a', xy=(-3, 2.5),xytext=(-2, 2),arrowprops={'color':'red'})
plt.annotate('dot_b', xy=(2, -3),xytext=(2, -2),arrowprops={'color':'blue'})
plt.show()
```



数据可视化: Matplotlib

- 快速绘图：直方图

```
x = np.random.randn(1000)
"""
绘制直方图
data:必选参数，绘图数据
bins:直方图的长条形数目，可选项，默认为10
normed:是否将得到的直方图向量归一化，可选项，默认为0，
        代表不归一化，显示频数。normed=1，表示归一化，显示频率。
facecolor:长条形的颜色
edgecolor:长条形边框的颜色
alpha:透明度
"""
plt.hist(x, bins=40, normed=0, facecolor="blue", \
         edgecolor="black", alpha=0.7)
# 显示横轴标签
plt.xlabel("区间")
# 显示纵轴标签
plt.ylabel("频数/频率")
# 显示图标题
plt.title("频数/频率分布直方图")
plt.show()
```

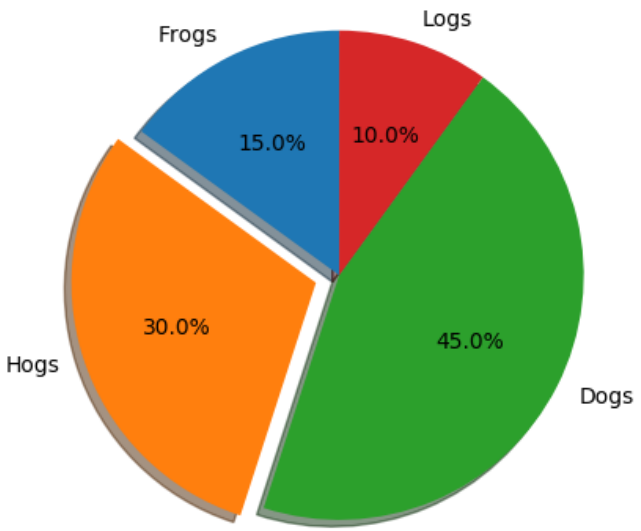


数据可视化: Matplotlib

- 参考官网Gallery

pie_and_polar_charts example code: pie_demo_features.py

(Source code, png, pdf)




```
# Pie chart, where the slices will be ordered
# and plotted counter-clockwise:
labels = 'Frogs', 'Hogs', 'Dogs', 'Logs'
sizes = [15, 30, 45, 10]
# only "explode" the 2nd slice (i.e. 'Hogs')
explode = (0, 0.1, 0, 0)

fig1, ax1 = plt.subplots()
ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
        shadow=True, startangle=90)
# Equal aspect ratio ensures that pie is drawn as a circle.
ax1.axis('equal')
plt.show()
```

OpenCV基础

什么是OpenCV

- OpenCV: Open Source Computer Vision software library
  - 英特尔于1999年创建
- 跨平台
  - Windows | Linux | Android | Mac OS | iOS …
- BSD许可下免费
  - Commercial & non-commercial applications
- 用C/C++实现的，支持多种编程语言的API

OpenCV：OpenCV的配置

- 下载最新的基于 Python 的 OpenCV 库
  https://pypi.org/project/opencv-python/

  pip install *.whl
  或者
  pip install opencv-python （Main modules package）
  pip install opencv-contrib-python （Full package）
  pip install opencv-contrib-python –i 镜像地址

  国内常用镜像地址：
  阿里云：https://mirrors.aliyun.com/pypi/simple/
  清华大学：https://pypi.tuna.tsinghua.edu.cn/simple

- OpenCV Tutorials
  https://docs.opencv.org/4.x/d9/df8/tutorial_root.html

OpenCV主要包

- **core** ▶ Base data structures & core routines
- **imgproc** ▶ Image processing routines
  - Linear / nonlinear image filtering
  - Geometric image transforms
  - Shape descriptors
  - Basic image operators
  - Histograms
  - Basic feature detection
- **video** ▶ Video analysis routines
  - Motion estimation
  - Motion segmentation
  - Background subtraction
  - Object tracking

OpenCV主要包

- **calib3D** ▶ Basic multiple-view geometry algorithms
  - Single / stereo camera calibration
  - Object pose estimation
  - Stereo correspondence
  - 3D reconstruction
- **features2D** ▶ 2D image features routines
  - Feature detectors
  - Descriptor extractors
  - Descriptor matchers
  - Object categorization
- **objdetect** ▶ Object detection routines
  - Detection of objects and instances of predefined classes
    e.g. faces | eyes | mugs | people | cars | …

OpenCV主要包

- **highgui** ▶ High-level GUI
  - Simple GUI capabilities
- **imgcodecs** ▶ I/O for image files
  - Standard image codecs
- **videoio** ▶ I/O for video files | image sequences | cameras
  - Video capturing & codecs

    incl. OpenNI-compatible depth sensors (Kinect | XtionPRO | ...)

OpenCV主要包：其他辅助模块

- **FLANN** ▶ Fast Library for Approximate Nearest Neighbors
  Clustering & search in multidimensional spaces
- **cv2** ▶ OpenCV-Python bindings
- **photo** ▶ Computational photography
- **stitching** ▶ Image stitching
- **superres** ▶ Image super resolution
- **viz** ▶ 3D visualizer
- …

OpenCV主要数据结构

- Point, Point2f：二维点对象（x, y）
- Size：二维尺寸结构（int width, height）
- Rect：二维矩形结构 （int x, y, width, height）
- RotatedRect：带角度的矩形对象
- Mat：图像对象，通道数 1: grayscale, 3: BGR

图像处理基本操作

- 读取图像

  retval = cv2.imread(filename[, flags])，支持各种静态图像格式

  filename 表示要读取的图像的完整文件名

  flags 是读取标记，该标记用来控制读取文件的类型

  当图像的路径是错的，OpenCV 也不会提醒你，但img是None

值	含义	数值
cv2.IMREAD_UNCHANGED	保持原格式不变	−1
cv2.IMREAD_GRAYSCALE	将图像调整为单通道的灰度图像	0
cv2.IMREAD_COLOR	将图像调整为 3 通道的 BGR 图像。该值是默认值	1
cv2.IMREAD_ANYDEPTH	当载入的图像深度为 16 位或者 32 位时，就返回其对应的深度图像；否则，将其转换为 8 位图像	2
cv2.IMREAD_ANYCOLOR	以任何可能的颜色格式读取图像	4

图像处理基本操作

- 显示图像
  cv2.namedWindow(winname)创建指定名称的窗口
  cv2.imshow(winname, mat)显示图像
  cv2.destroyWindow(winname)释放（销毁）指定窗口
  cv2.destroyAllWindows() 释放（销毁）所有窗口
  winname 是窗口名称，mat 是要显示的图像
- 保存图像
  retval = cv2.imwrite(filename, img[, params])
  retval 是返回值，如果保存成功返回True，否则返回False。
  filename 是要保存的目标文件的完整路径名，包含文件扩展名。
  img 是被保存图像的名称，params 是保存类型参数，是可选的。

图像处理基本操作

问题：图像太大，imshow显示不全?

图像很大，甚至超过了电脑屏幕，
并且无法调整大小

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

解决：

```
cv2.namedWindow('demo', 0)
cv2.imshow("demo", img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

或者规定显示图像大小，使用
cv2.resizeWindow(winname, width, height)

```
cv2.namedWindow('demo', 0)
cv2.resizeWindow('demo', 600, 500)
cv2.imshow("demo", img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

其它基本操作：按键

retval = cv2.waitKey([delay])用来等待按键

当用户按下键盘后，该语句会被执行，并获取返回值。

retval 表示返回值；如果没有按键被按下，则返回–1；如果有按键被按下，则返回该按键的 ASCII 码。

delay 表示等待键盘触发的时间，单位是 ms；当该值是负数或者零时，表示无限等待；该值默认为 0。

在实际使用中，可以通过函数 cv2.waitKey()获取按下的按键， 并针对不同的键做出不同的反应，从而实现交互功能。

其它基本操作：从摄像头或者文件捕获视频

创建一个 VideoCapture 对象，参数可以是设备的索引号，或者是一个视频完整文件名。

```
import numpy as np
import cv2

cap = cv2.VideoCapture(0)

while(True):
    # Capture frame-by-frame
    ret, frame = cap.read()

    # Our operations on the frame come here
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Display the resulting frame
    cv2.imshow('frame',gray)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# When everything done, release the capture
cap.release()
cv2.destroyAllWindows()
```

设备索引号就是在指定要使用的摄像头。一般的笔记本电脑都有内置摄像头，所以参数就是 0。可以通过设置成 1 或者其它的来选择别的摄像头。

cap.read() 返回一个布尔（True/False）。如果帧读取的是正确的，就是 True。可以通过检查它的返回值来查看视频文件是否已经到了结尾

其它基本操作：从摄像头或者文件捕获视频

有时 cap 可能不能成功的初始化摄像头设备。这种情况下上面的代码会报错。可以使用 cap.isOpened()，来检查是否成功初始化了。如果返回值是 True，那就没有问题。否则就要使用函数 cap.open()。

你可以使用函数 cap.get(propId) 来获得视频的一些参数信息。这里 propId 可以是 0 到 18 之间的任何整数。每一个数代表视频的一个属性。其中的一些值可以使用 cap.set(propId,value) 来修改， value 就是你想要设置成的新值。

可以使用 cap.get(3) 和 cap.get(4) 来查看每一帧的宽和高。 默认情况下得到的值是 640*480。但是可以使用 ret=cap.set(3,320) 和ret=cap.set(4,240) 来把宽和高改成 320*240。

其它基本操作：从摄像头或者文件捕获视频

cap.set(propId,value)，propId代表视频的属性值

- **CV_CAP_PROP_POS_MSEC** Current position of the video file in milliseconds.
- **CV_CAP_PROP_POS_FRAMES** 0-based index of the frame to be decoded/captured next.
- **CV_CAP_PROP_POS_AVI_RATIO** Relative position of the video file: 0 - start of the film, 1 - end of the film.
- **CV_CAP_PROP_FRAME_WIDTH** Width of the frames in the video stream.
- **CV_CAP_PROP_FRAME_HEIGHT** Height of the frames in the video stream.
- **CV_CAP_PROP_FPS** Frame rate.
- **CV_CAP_PROP_FOURCC** 4-character code of codec.
- **CV_CAP_PROP_FRAME_COUNT** Number of frames in the video file.
- **CV_CAP_PROP_FORMAT** Format of the Mat objects returned by retrieve() .
- **CV_CAP_PROP_MODE** Backend-specific value indicating the current capture mode.
- **CV_CAP_PROP_BRIGHTNESS** Brightness of the image (only for cameras).
- **CV_CAP_PROP_CONTRAST** Contrast of the image (only for cameras).
- **CV_CAP_PROP_SATURATION** Saturation of the image (only for cameras).
- **CV_CAP_PROP_HUE Hue** of the image (only for cameras).
- **CV_CAP_PROP_GAIN** Gain of the image (only for cameras).
- **CV_CAP_PROP_EXPOSURE** Exposure (only for cameras).
- **CV_CAP_PROP_CONVERT_RGB** Boolean flags indicating whether images should be converted to RGB.
- **CV_CAP_PROP_WHITE_BALANCE** Currently unsupported
- **CV_CAP_PROP_RECTIFICATION** Rectification flag for stereo cameras (note: only supported by DC1394 v 2.x backend currently)

其它基本操作：保存视频

```
import numpy as np
import cv2

cap = cv2.VideoCapture(0)

# Define the codec and create VideoWriter object
fourcc = cv2.cv.FOURCC(*'XVID')
out = cv2.VideoWriter('output.avi',fourcc, 20.0, (640,480))

while(cap.isOpened()):
    ret, frame = cap.read()
    if ret==True:
        frame = cv2.flip(frame,0)

        # write the flipped frame
        out.write(frame)

        cv2.imshow('frame',frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    else:
        break

# Release everything if job is finished
cap.release()
out.release()
cv2.destroyAllWindows()
```

- 创建一个 VideoWriter 的对象，确定一个输出文件的名字。
- 指定 FourCC 编码。
- 频率和帧的大小也都需要确定。

FourCC 用来确定视频的编码格式。
以 MJPG 为例：
cv2.cv.FOURCC('M','J','P','G') 或者
cv2.cv.FOURCC(*'MJPG')

OpenCV 中的绘图函数

- 画线 cv2.line(img,pt1,pt2)
  起点和终点
- 画圆cv2.circle(img, center, radius)
  中心点坐标和半径大小
- 画矩形 cv2.rectangle(img, pt1, pt2)
  左上角顶点和右下角顶点的坐标

cv2.line(img,(0,0),(511,511),(255,0,0),5)

cv2.circle(img,(447,63), 63, (0,0,255), -1)

cv2.rectangle(img,(384,0),(510,128),(0,255,0),3)

需要设置下面这些通用参数

- img：要绘制图形的那幅图像。
- color：形状的颜色。以 RGB 为例，（255,0,0）代表蓝色。对于灰度图只需要传入灰度值。
- thickness：线条的粗细。如果给一个闭合图形设置为 -1，那么这个图形就会被填充。默认值是 1。
- linetype：线条的类型，8 连接，抗锯齿等。默认情况是 8 连接。cv2.LINE_AA 为抗锯齿，这样看起来会非常平滑。

OpenCV 中的绘图函数

- 画椭圆 cv2.ellipse(img, center, axes, angle, startAngle, endAngle)

中心点的位置坐标，长轴和短轴的长度，椭圆沿逆时针方向旋转的角度，椭圆弧演顺时针方向起始的角度和结束角度，如果是 0 很 360，就是整个椭圆。

cv2.ellipse(img,(256,256),(100,50),0,0,180,(255,0,0),-1)

- 画多边形 cv2.polylines(img, pts, isClosed)

每个顶点的坐标。用这些点的坐标构建一个维度为N*1*2（或N*2）的数组，N表示点的数目，数据类型必须为 int32。

pts=np.array([[10,5],[20,30],[70,20],[50,10]], np.int32)

pts=pts.reshape((-1,1,2)) (可省略)

img = cv2.polylines(img,[pts],True, ,(255,0,0),2)

OpenCV 中的绘图函数

- 在图片上添加文字

  cv2.putText(img, text, org, fontFace, fontScale)

  主要参数：绘制的位置、字体类型、字体的大小、文字的一般属性如颜色，粗细，线条的类型等。

font=cv2.FONT_HERSHEY_SIMPLEX

cv2.putText(img,'OpenCV',(10,500), font, 4,(255,255,255),2)

所有的绘图函数的返回值都是None，所以不能使用 img =cv2.line(...)

OpenCV鼠标事件

- 鼠标事件回调函数

  cv2.setMouseCallback(winname, MouseCallback func)

  MouseCallback参数：int event, int x, int y, flags, param

创建一个鼠标事件回调函数，但鼠标事件发生是他就会被执行。
鼠标事件可以是鼠标上的任何动作，比如左键按下，左键松开，左键双击等。

在双击过的地方绘制一个圆圈

```
def draw_circle(event,x,y,flags,param):
    if event==cv2.EVENT_LBUTTONDBLCLK:
        cv2.circle(img,(x,y),100,(255,0,0),-1)

# 创建图像与窗口并将窗口与回调函数绑定
img=np.zeros((512,512,3),np.uint8)
cv2.namedWindow('image')
cv2.setMouseCallback('image',draw_circle)
while(1):
    cv2.imshow('image',img)
    if cv2.waitKey(20)&0xFF==27:
        break
cv2.destroyAllWindows()
```

OpenCV鼠标事件

- 鼠标事件标记

EVENT_LBUTTONDBLCLK = 7左键双击
EVENT_LBUTTONDOWN = 1左键击下
EVENT_LBUTTONUP = 4 左键弹起
EVENT_MBUTTONDBLCLK = 9中键双击
EVENT_MBUTTONDOWN = 3中键击下
EVENT_MBUTTONUP = 6中键弹起

EVENT_FLAG_ALTKEY = 32摁住Alt
EVENT_FLAG_CTRLKEY = 8摁住Ctrl

EVENT_FLAG_LBUTTON = 1摁住左键
EVENT_FLAG_MBUTTON = 4摁住中键
EVENT_FLAG_RBUTTON = 2摁住右键
EVENT_FLAG_SHIFTKEY = 16摁住Shift

EVENT_MOUSEHWHEEL = 11滚动条向左，flags>0。向右，flags<0
EVENT_MOUSEMOVE = 0鼠标移动
EVENT_MOUSEWHEEL = 10滚动条向上，flags>0。向下，flags<0
EVENT_RBUTTONDBLCLK = 8中键双击
EVENT_RBUTTONDOWN = 2中键击下
EVENT_RBUTTONUP = 5 中键弹起

OpenCV鼠标事件

程序示例：拖动鼠标绘图

```
import cv2
import numpy as np

# 创建回调函数
def draw_circle(event,x,y,flags,param):
    global ix,iy,drawing
    # 当按下左键是返回起始位置坐标
    if event==cv2.EVENT_LBUTTONDOWN:
        drawing=True
        ix,iy=x,y
    # 当鼠标左键按下并移动是绘制图形。 event 可以查看移动， flag 查看是否按下
    if event==cv2.EVENT_MOUSEMOVE and flags==cv2.EVENT_FLAG_LBUTTON:
        if drawing==True:
            # 绘制圆圈，小圆点连在一起就成了线， 3 代表了笔画的粗细
            cv2.circle(img,(x,y),3,(0,0,255),-1)
        # 当鼠标松开停止绘画。
        if event==cv2.EVENT_LBUTTONUP:
            drawing=False

drawing=False
ix,iy=-1,-1
img=np.zeros((512,512,3),np.uint8)
cv2.namedWindow('image')
cv2.setMouseCallback('image',draw_circle)

while(1):
    cv2.imshow('image',img)
    # 按ESC退出
    if cv2.waitKey(1)&0xFF == 27:
        break
```

滚动条

- 滚动条（Trackbar）在 OpenCV 中是非常方便的交互工具，它依附于特定的窗口而存在。通过调节滚动条能够设置、获取指定范围内的特定值。

  cv2.createTrackbar(trackbarname, winname, value, count, onChange)

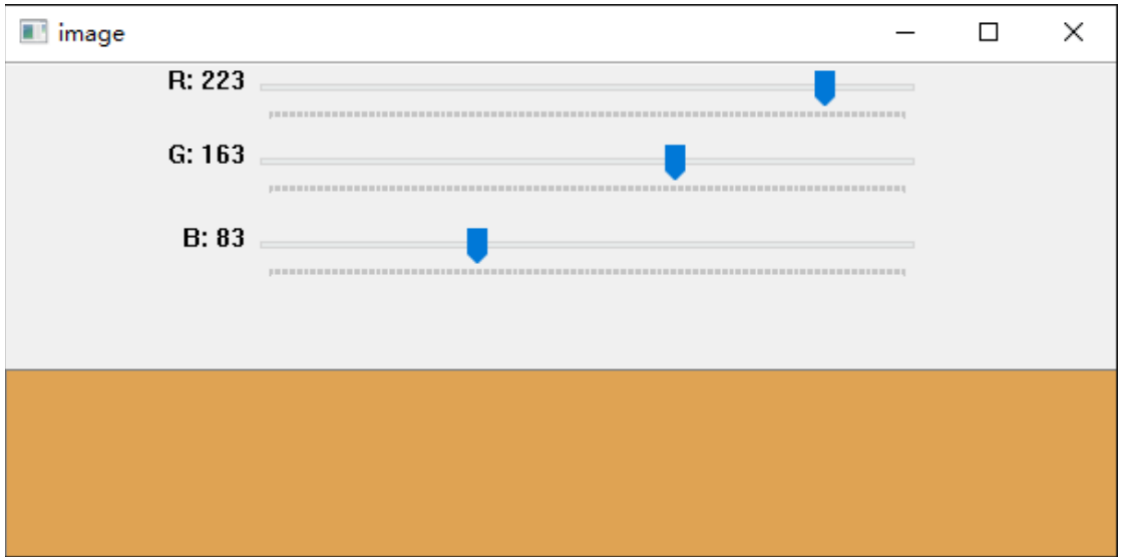
  trackbarname 为滚动条的名称。
  winname 为滚动条所依附窗口的名称。
  value 为初始值，该值决定滚动条中滑块的位置。
  count 为滚动条的最大值。通常情况下， 其最小值是 0。
  onChange 为回调函数。一般情况下，将滚动条改变后要实现的操作写在回调函数内。

- 获取滚动条的值

  retval=getTrackbarPos(trackbarname, winname)

滚动条

```
import cv2
import numpy as np
def changeColor(x):
    r = cv2.getTrackbarPos('R', 'image')
    g = cv2.getTrackbarPos('G', 'image')
    b = cv2.getTrackbarPos('B', 'image')
    img[:] = [b, g, r]
img = np.zeros((200, 700, 3), np.uint8)
cv2.namedWindow('image')
cv2.createTrackbar('R', 'image', 0, 255, changeColor)
cv2.createTrackbar('G', 'image', 0, 255, changeColor)
cv2.createTrackbar('B', 'image', 0, 255, changeColor)
while (1):
    cv2.imshow('image', img)
    if cv2.waitKey(1) & 0xFF == 27: # ESC
        break
cv2.destroyAllWindows()
```



用滚动条实现调色板

专业技能实训：练习题

- 用橡皮擦擦除图像
  - 输入图像，用滚动条控制橡皮擦（鼠标）大小。
  - 橡皮擦（鼠标）拖动，擦除图像中鼠标所拖动区域的像素点，被擦除的像素设为白色；停止拖动鼠标，该擦除过程停止。
  - 在滚动条下方，同时显示原图和处理后的图像；前两步可反复运行，直到输出满意的处理图像为止。

- 在鼠标点击的位置插入文字
  - 输入图像和文字内容。
  - 在鼠标点击的位置，在图像中插入文字。
  - 用多个滚动条分别控制所插入文字的颜色、大小等。