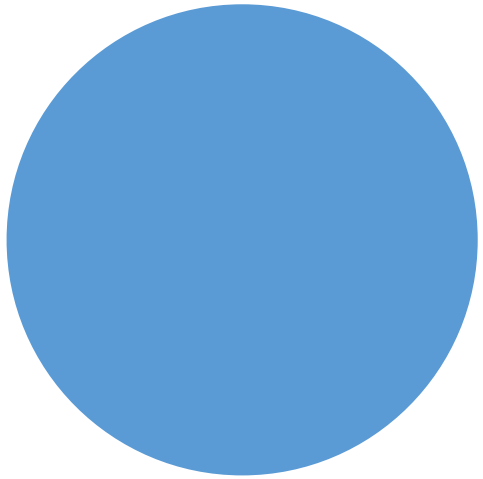
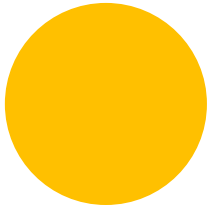



PYTHON:Matplotlib

2019.11.28

~/PPT/11.28.Matplotlib.pptx

http://bigsec.net/b52/scipydoc/matplotlib_intro.html

教程1

https://liam.page/2014/09/11/matplotlib-tutorial-zh-cn/

教程2

https://matplotlib.org/gallery.html

大量例子

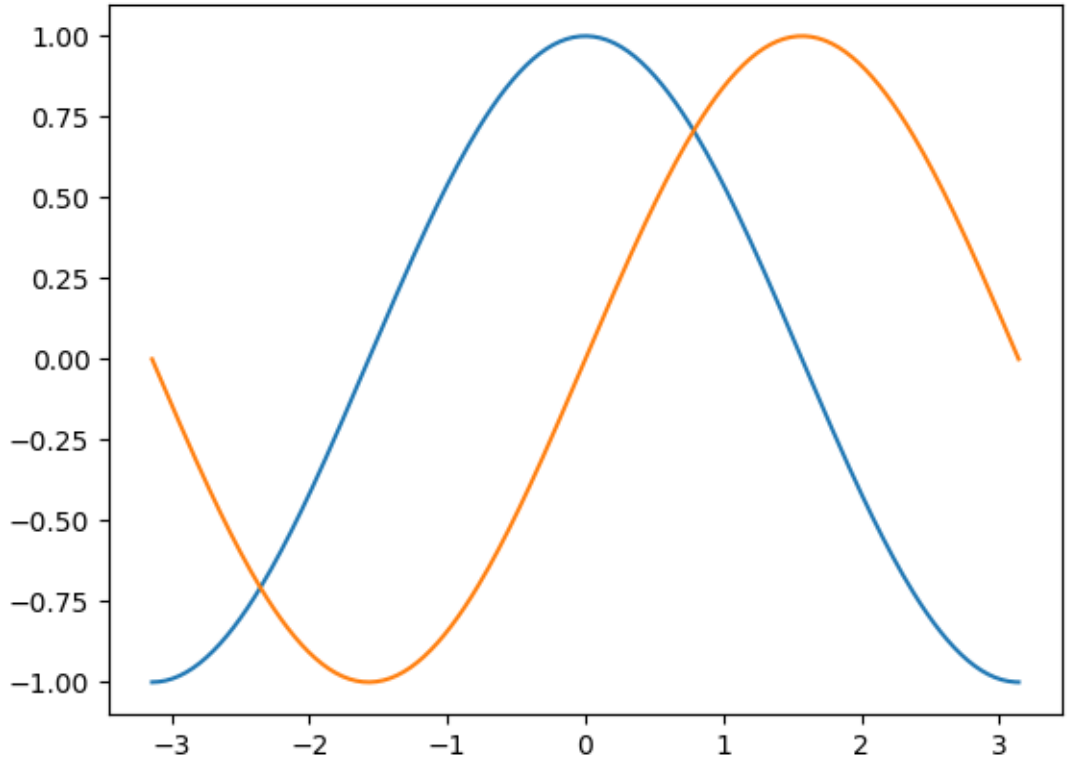
https://matplotlib.org/api/_as_gen/matplotlib.pyplot.html

查询函数用法

课上讲到的东西大概只占全部功能的10%，细节功能在实际运用中根据需求从实例查找

Matplotlib

```
from pylab import *
import numpy as np
import matplotlib.pyplot as plt
#建立数组
X = np.linspace(-np.pi, np.pi, 256,endpoint=True)
C,S = np.cos(X), np.sin(X)
#画图
plt.plot(X,C)
plt.plot(X,S)
#显示图片
plt.show()
```



快速开始

https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.plot.html?highlight=plot#matplotlib.pyplot.plot

```
#firstpic.py
import numpy as np
import matplotlib.pyplot as plt
#建立数组
X = np.linspace(-np.pi, np.pi, 256,endpoint=True)
C,S = np.cos(X), np.sin(X)
#画图
plt.plot(X,C)
plt.plot(X,S)
#保存图片
plt.show()
plt.savefig('mypic')
```

plt.show会打开一个图片窗口，关闭后就关闭了图片随后的savefig会保存一个空图像

默认存为.png，也可以指定为.jpg或.pdf文件

终端：python firstpic.py

python 命令行：run firstpic.py

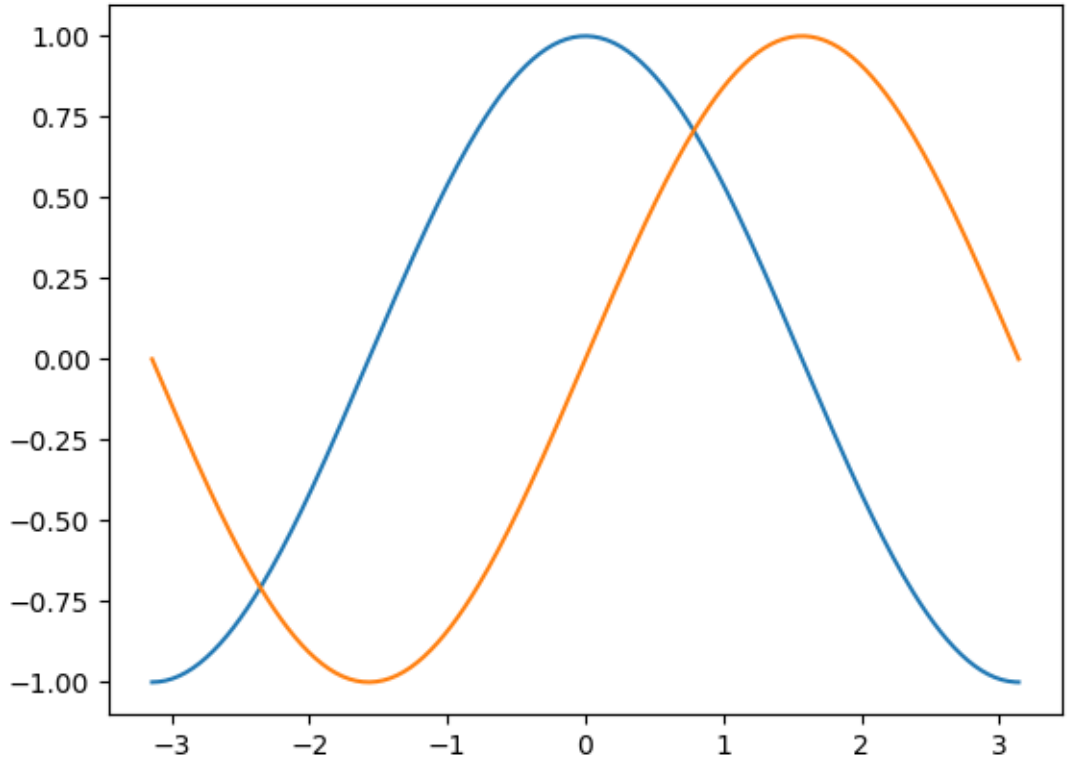
脚本运行

```
from pylab import *
import numpy as np
import matplotlib.pyplot as plt
#建立数组
X = np.linspace(-np.pi, np.pi, 256,endpoint=True)
C,S = np.cos(X), np.sin(X)
#画图
plt.ion()
plt.plot(X,C)
plt.plot(X,S)
#显示图片
plt.show()
plt.ioff()
```

打开互动模式

不需要plt.show()显示图像

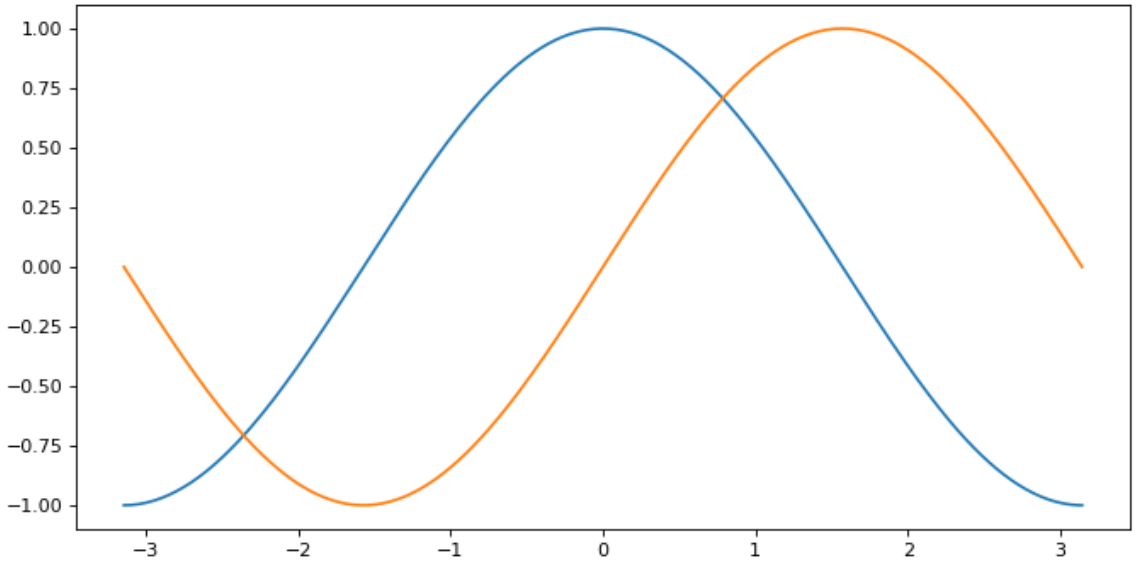
关闭互动模式



互动模式

```
import numpy as np
import matplotlib.pyplot as plt
X = np.linspace(-np.pi, np.pi, 256,endpoint=True)
C,S = np.cos(X), np.sin(X)
plt.figure(figsize=(10,5), dpi=80)
plt.plot(X,C)
plt.plot(X,S)
plt.show()
```

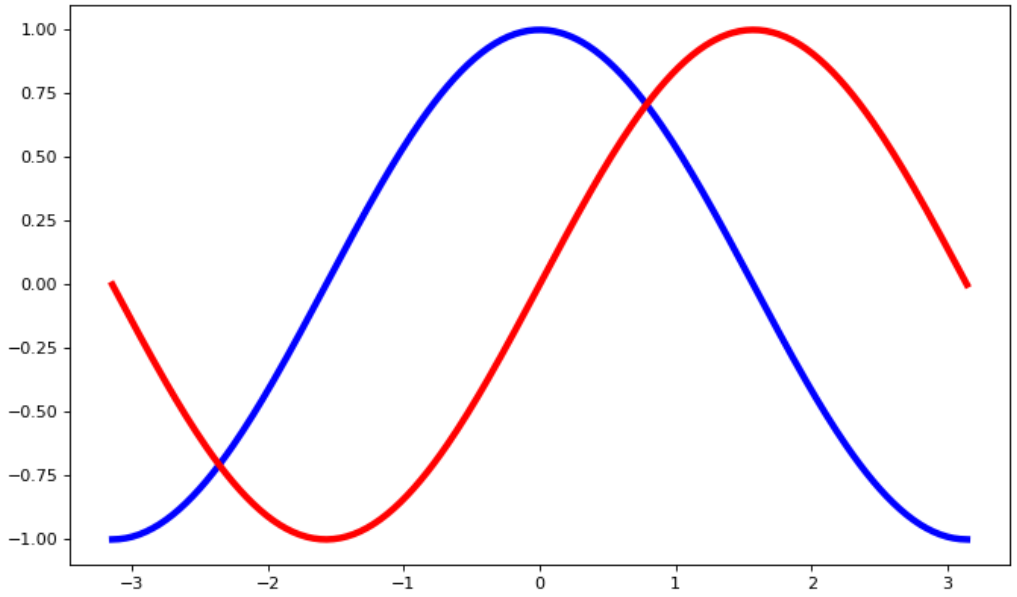
单位inch



改变图片尺寸

```
import numpy as np
import matplotlib.pyplot as plt
X = np.linspace(-np.pi, np.pi, 256,endpoint=True)
C,S = np.cos(X), np.sin(X)
plt.figure(figsize=(10,6), dpi=80)
plt.plot(X,C,color="blue",linewidth=4,linestyle="-")
plt.plot(X,S,color="red",linewidth=4,linestyle="-")
plt.show()
```

颜色，c　　线宽，lw　　线型，ls



改变线型

还可以用：**1.RGB值, (0.1,0.2,0.5), #0f0f0f ； 2.'C0','C1'表示当前颜色循环值中的第1,2,…个颜色**

color 或 c

缩写	颜色
'b'	blue
'g'	green
'r'	red
'c'	cyan
'm'	magenta
'y'	yellow
'k'	black
'w'	white

颜色

linestype 或 ls

	description
'-'	solid line style
'--'	dashed line style
'-.'	dash-dot line style
':'	dotted line style

marker　　**markersize 或 ms: 点的大小**

character	description
'.'	point marker
','	pixel marker
'o'	circle marker
'v'	triangle_down marker
'^'	triangle_up marker
'<'	triangle_left marker
'>'	triangle_right marker
'1'	tri_down marker
'2'	tri_up marker
'3'	tri_left marker
'4'	tri_right marker
's'	square marker
'p'	pentagon marker
'*'	star marker
'h'	hexagon1 marker
'H'	hexagon2 marker
'+'	plus marker
'x'	x marker
'D'	diamond marker
'd'	thin_diamond marker
'|'	vline marker
'_'	hline marker

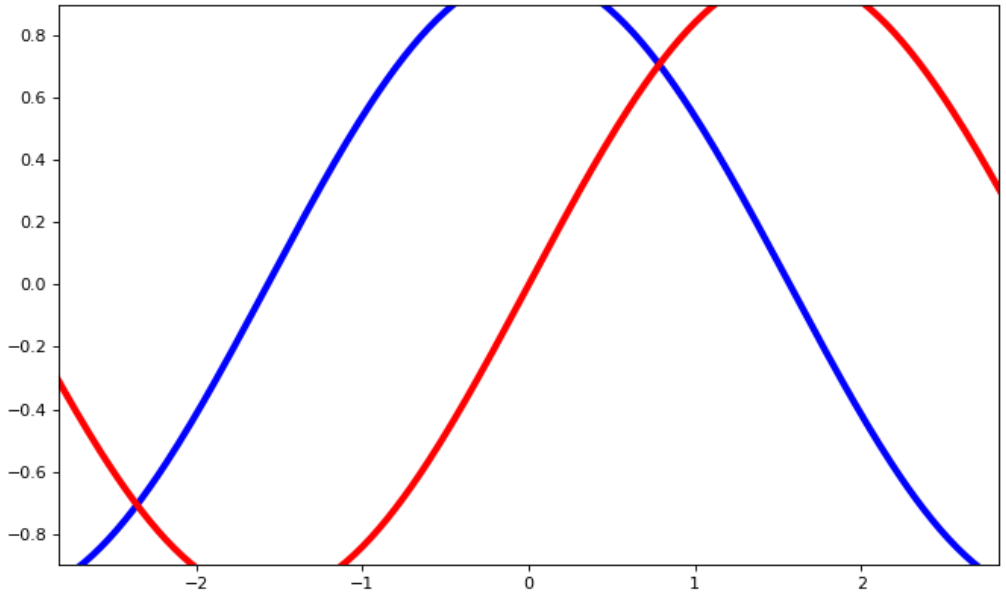
组合字符串

```
'b'    # blue markers with default shape
'or'   # red circles
'-g'   # green solid line
'--'   # dashed line with default color
'^k:'  # black triangle_up markers connected by a dotted line
```

```
plt.plot(X,Y,'--or')
```

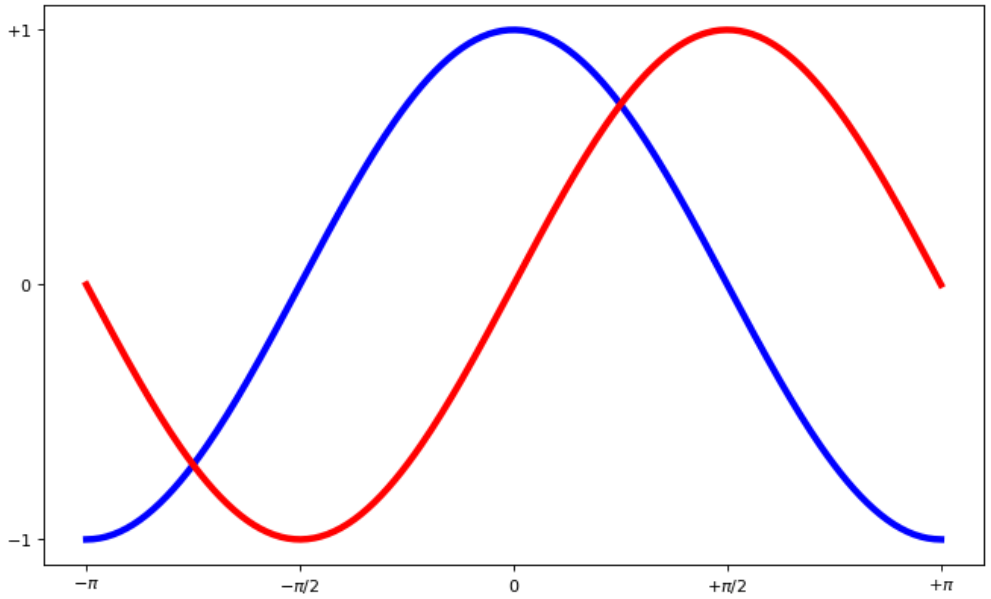
线型列表

```
import numpy as np
import matplotlib.pyplot as plt
X = np.linspace(-np.pi, np.pi, 256,endpoint=True)
C,S = np.cos(X), np.sin(X)
plt.figure(figsize=(10,6), dpi=80)
plt.plot(X,C,color="blue",linewidth=4,linestyle="-")
plt.plot(X,C,color="red",linewidth=4,linestyle="-")
plt.xlim(X.min()*0.9, X.max()*0.9)
plt.ylim(C.min()*0.9, C.max()*0.9)
plt.show()
```



改变边界

```
import numpy as np
import matplotlib.pyplot as plt
X = np.linspace(-np.pi, np.pi, 256,endpoint=True)
C,S = np.cos(X), np.sin(X)
plt.figure(figsize=(10,6), dpi=80)
plt.plot(X,C,color="blue",linewidth=4,linestyle="-")
plt.plot(X,C,color="red",linewidth=4,linestyle="-")
plt.xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi],
      [r'$-\pi$', r'$-\pi/2$', r'$0$', r'$+\pi/2$',
r'$+\pi$'])

plt.yticks([-1, 0, +1],
      [r'$-1$', r'$0$', r'$+1$'])
plt.show()
```



刻度位置

刻度对应标签，可忽略

`xticks(ticks=None, labels=None, **kwargs)`

fontsize, rotation,
……

改变刻度

text字符串支持LaTex的公式输入

尝试输入：plt.text(0, 0, "$\Theta=\pi r^2/\sqrt{x}$")

LaTex的公式输入语法自行网上查找

```
import numpy as np
import matplotlib.pyplot as plt
X = np.linspace(-np.pi, np.pi, 256,endpoint=True)
C,S = np.cos(X), np.sin(X)
plt.figure(figsize=(10,6), dpi=80)
plt.plot(X,C,color="blue",linewidth=4,linestyle="-")
plt.plot(X,C,color="red",linewidth=4,linestyle="-")
xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi],[r'$-\pi$',
r'$-\pi/2$', r'$0$', r'$+\pi/2$', r'$+\pi$'])
yticks([-1, 0, +1],[r'$-1$', r'$0$', r'$+1$'])
ax = plt.gca()
ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)
ax.xaxis.set_ticks_position('bottom')
ax.spines['bottom'].set_position(('data',0))
ax.yaxis.set_ticks_position('left')
ax.spines['left'].set_position(('data',0))
plt.show()
```

获取当前axes容器

上方和右边的轴线设为不可见

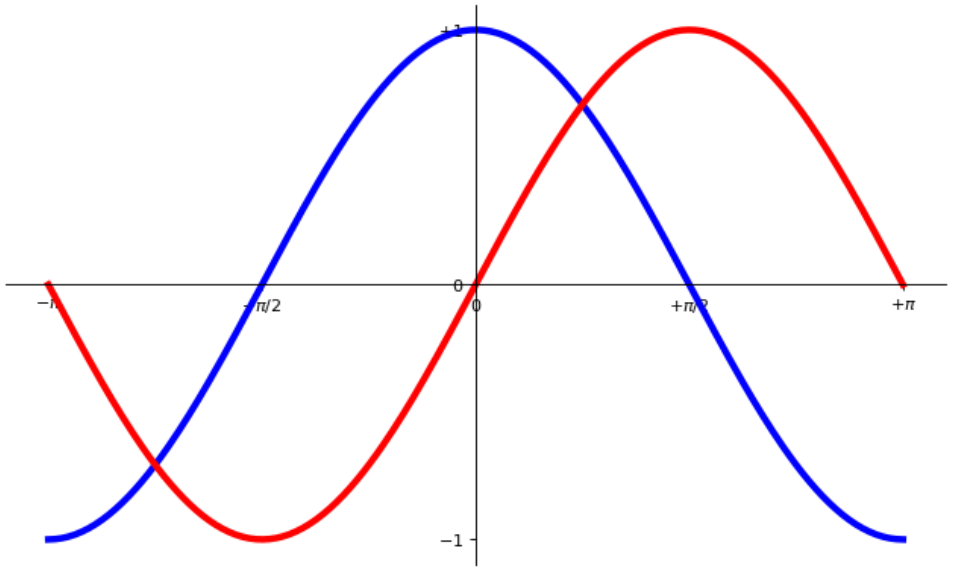
下方x轴刻度可见

把下方x轴线移动到数据的0处

左方y轴刻度可见

把左方y轴线移动到数据的0处

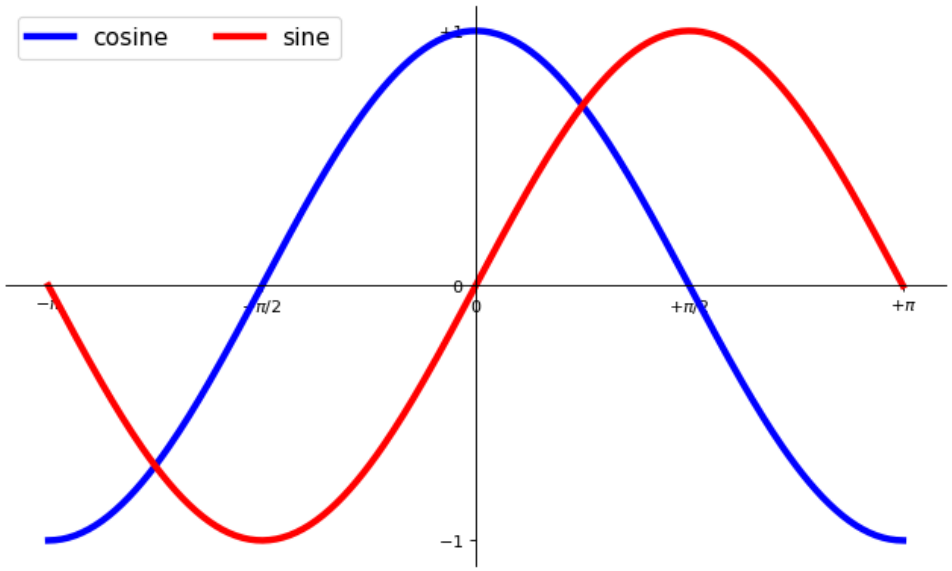
'outward', 'axes', 'data'



改变轴线(spine)位置

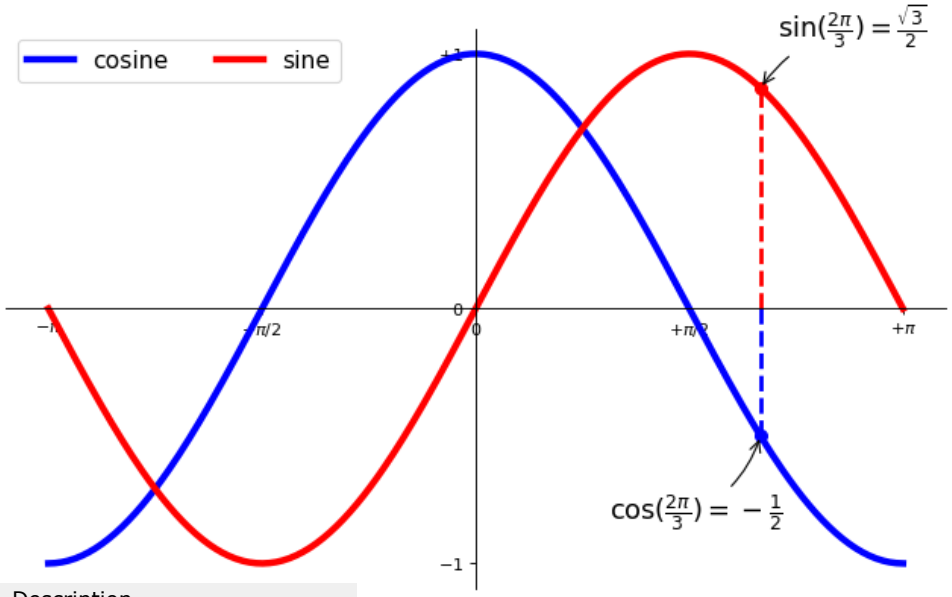
```
import numpy as np
import matplotlib.pyplot as plt
X = np.linspace(-np.pi, np.pi, 256,endpoint=True)
C,S = np.cos(X), np.sin(X)
plt.figure(figsize=(10,6), dpi=80)
plt.plot(X,C,color="blue",linewidth=4,linestyle="-",
label="cosine")
plt.plot(X,C,color="red",linewidth=4,linestyle="-",
label="sine")
.....
plt.legend(loc='upper left', fontsize=16, ncol=2)
```

全不写就采用默认值



加上图例

```
...
t = 2*np.pi/3
plt.plot([t,t],[0,np.cos(t)], color ='blue',
         linewidth=2.5, linestyle="--")
plt.scatter([t,],[np.cos(t),], 50, color ='blue')

plt.annotate(r'$\sin(\frac{2\pi}{3})=\frac{\sqrt{3}}{
2}$',
         xy=(t, np.sin(t)),
         xycoords='data',
         xytext=(+10, +30),
         textcoords='offset points',
         fontsize=16,
         arrowprops=dict(arrowstyle="->",
         connectionstyle="arc3,rad=.2"))

...
```



Value	Description
'figure points'	Points from the lower left of the figure
'figure pixels'	Pixels from the lower left of the figure
'figure fraction'	Fraction of figure from lower left
'axes points'	Points from lower left corner of axes
'axes pixels'	Pixels from lower left corner of axes
'axes fraction'	Fraction of axes from lower left
'data'	Use the coordinate system of the object being annotated (default)
'polar'	*(theta,r)* if not native 'data' coordinates

加上注释

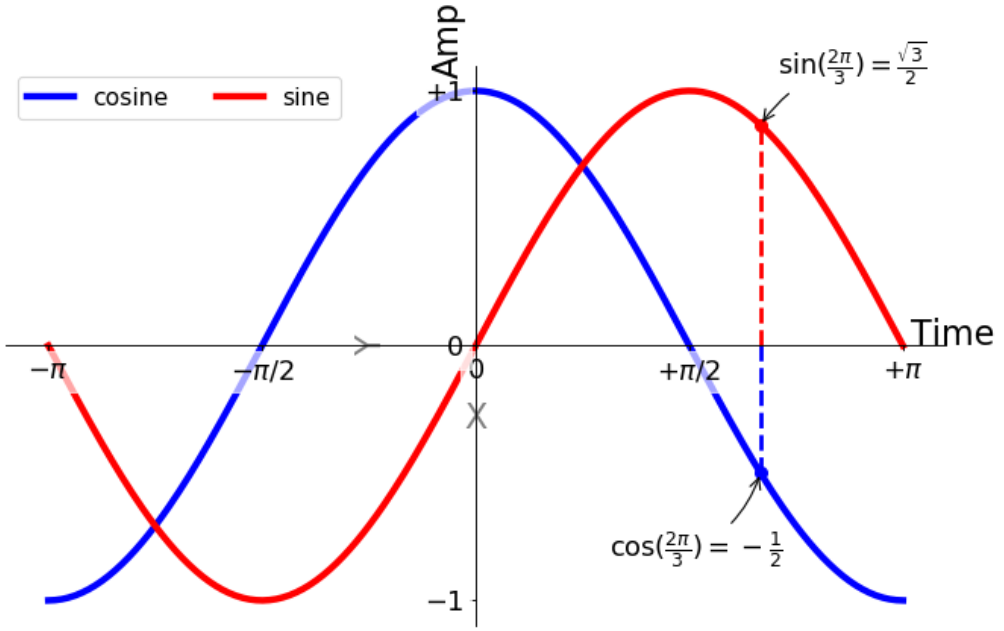
https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.annotate.html?highlight=annotate#matplotlib.pyplot.annotate

plt.text() 在图上任意位置添加文本

```
plt.xlabel("X axis", fontsize=20)
plt.ylabel("Y axis", fontsize=20)
```

labelpad

```
plt.text(3.2, 0, "Time", fontsize=20)
plt.text(-0.2, 1.2, "Amp", fontsize=20, rotation=90)
```



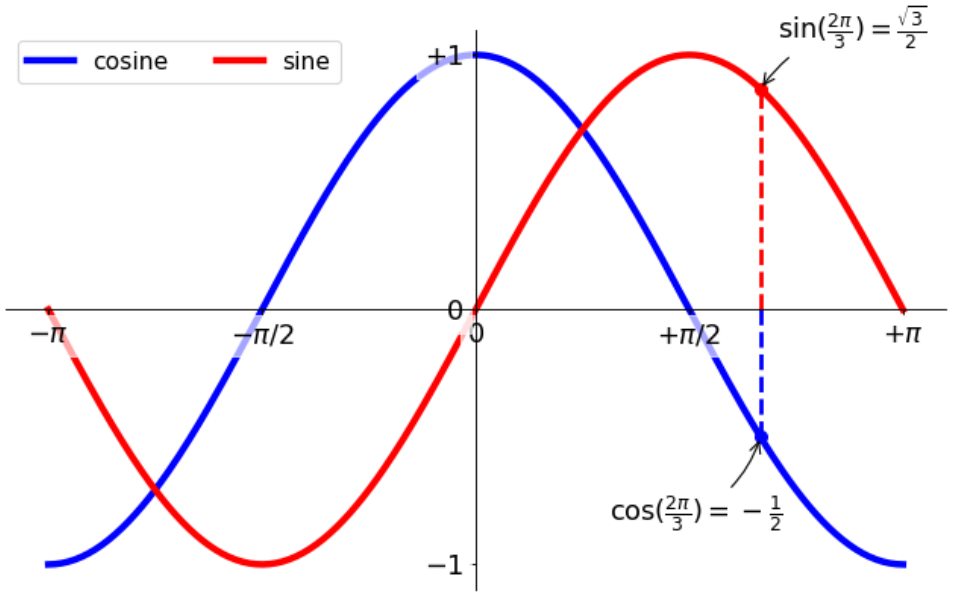
text字符串支持LaTex的公式输入

尝试输入：plt.text(0, 0, "$\Theta=\pi r^2/\sqrt{x}$")

LaTex的公式输入语法自行网上查找

坐标轴标签

```
...
for label in ax.get_xticklabels() + ax.get_yticklabels():
    label.set_fontsize(16)
    label.set_bbox(dict(facecolor='white',
edgecolor='None', alpha=0.65 ))
...
```



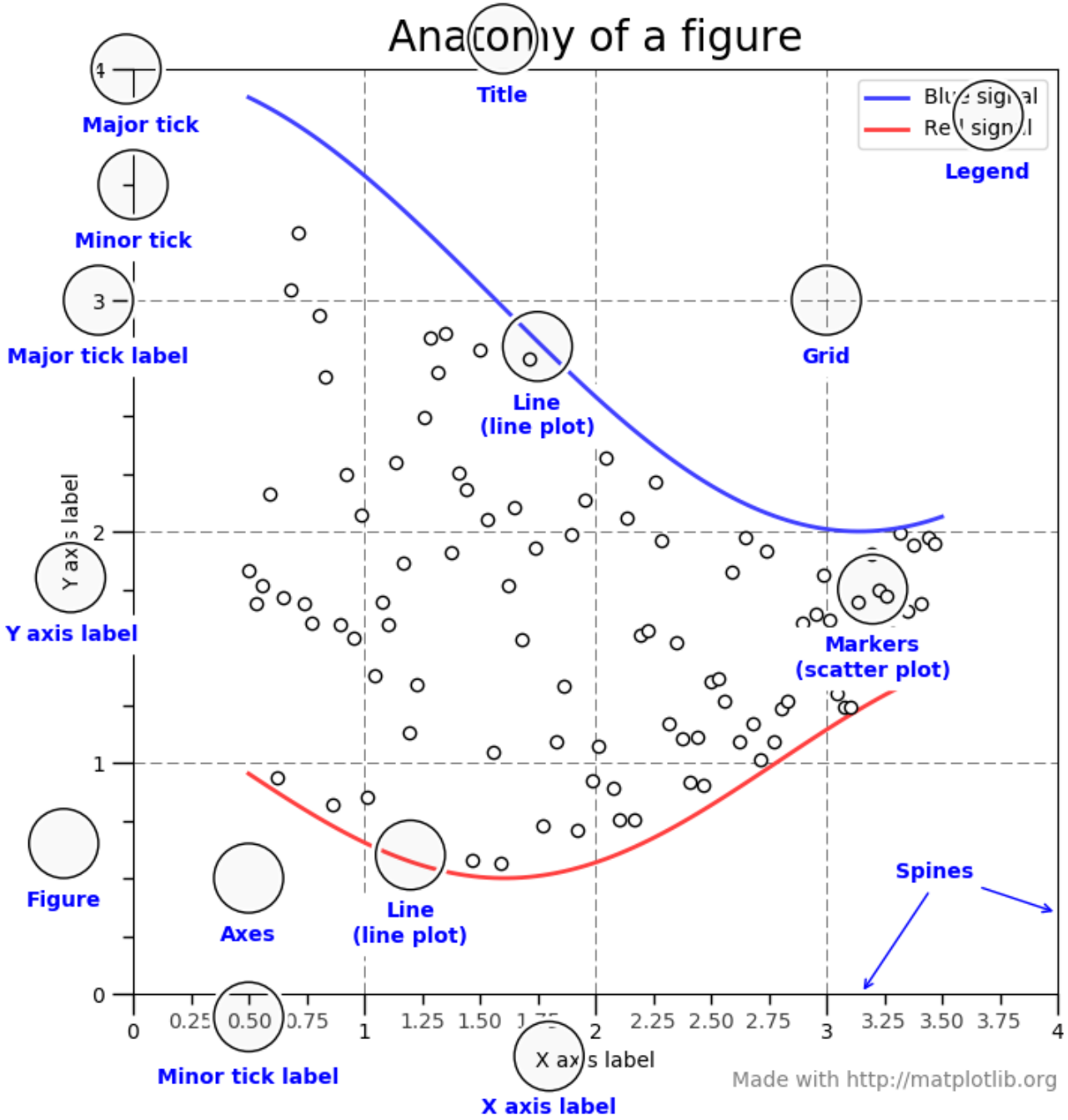
调整刻度外观

pyplot：
state-machine environment

用对象控制图中的每个元素

pylab提供了快速接口
不推荐

matplotlib对象



matplotlib API包含有三层

backend_bases.FigureCanvas

backend_bases.Renderer

artist.Artist

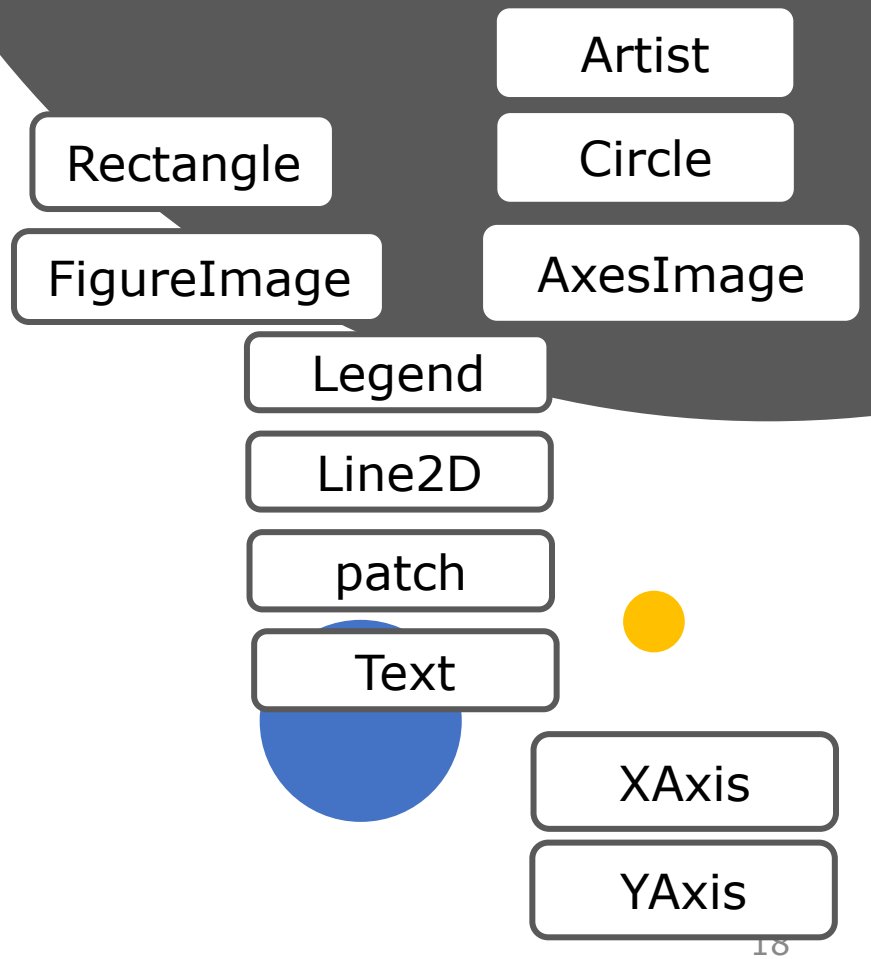
Artist.set_***
Artist.get_***

pyplot.getp(Artist, “atriname”)
pyplot.setp(Artist, “atriname”)

Figures
- axes
- patch
- images
- legends
- lines
- patches
- texts

Axes
- artists
- patch
- images
- legends
- lines
- patches
- texts
- xaxis
- yaxis

Artists分为简单类型和容器类型两种。简单类型的Artists为标准的绘图元件，例如Line2D、 Rectangle、 Text、 AxesImage 等等。而容器类型则可以包含许多简单类型的Artists，使它们组织成一个整体，例如Axis、 Axes、 Figure等。



对象、容器

显式控制作图流程

```
import matplotlib.pyplot as plt

fig=plt.figure()

ax1 = fig.add_axes([0.1,0.1,0.8,0.8])
ax2 = fig.add_subplot(111)

ax1.plot(…)
ax2.plot(…)

fig.savefig(“name.png”)
plt.show()
plt.close(fig)
```

1.新建一张图

2.建立坐标轴（子图）

3.画图

4.保存或直接显示

不需要对figure，Axes对象进行控制时这步可以省略

Axes.func()方法基本都有简便作图函数pyplot.func()

1.新建一张图

pyplot.figure()

```
matplotlib.pyplot.figure(num=None,
        figsize=None,
        dpi=None,
        facecolor=None,
        edgecolor=None,
        frameon=True,
        FigureClass=<class'matplotlib.figure.Figure' >,
        clear=False,
        **kwargs)
```

Figure容器

2.建立一个/多个 “图轴”

figure.add_axes()

figure.add_subplot()

Parameters:

rect : sequence of floatThe dimensions [left, bottom, width, height] of the new axes. All quantities are in fractions of figure width and height.

projection : {None, 'aitoff', 'hammer', 'lambert', 'mollweide', 'polar', 'rectilinear', str}, optionalThe projection type of the Axes. *str* is the name of a custom projection, see projections. The default None results in a 'rectilinear' projection.

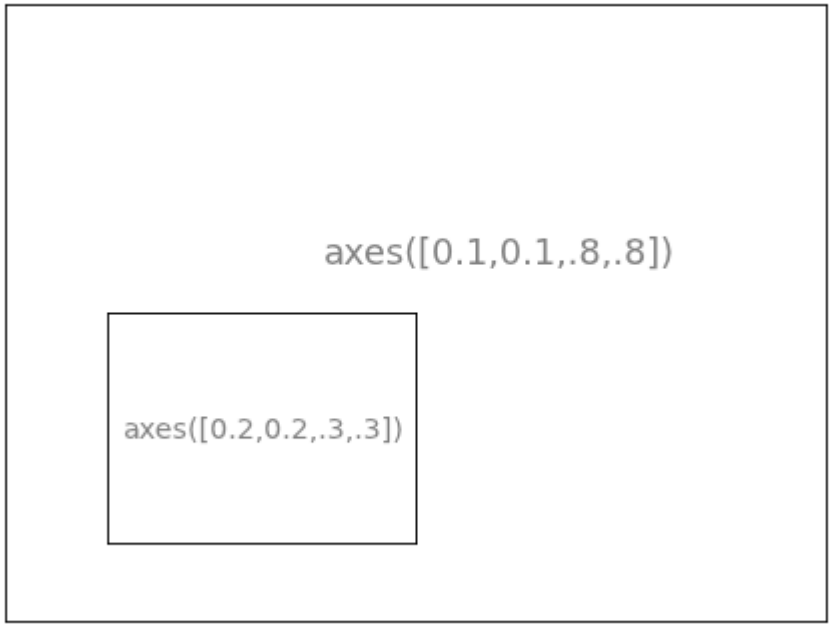
polar : boolean, optionalIf True, equivalent to projection='polar’.

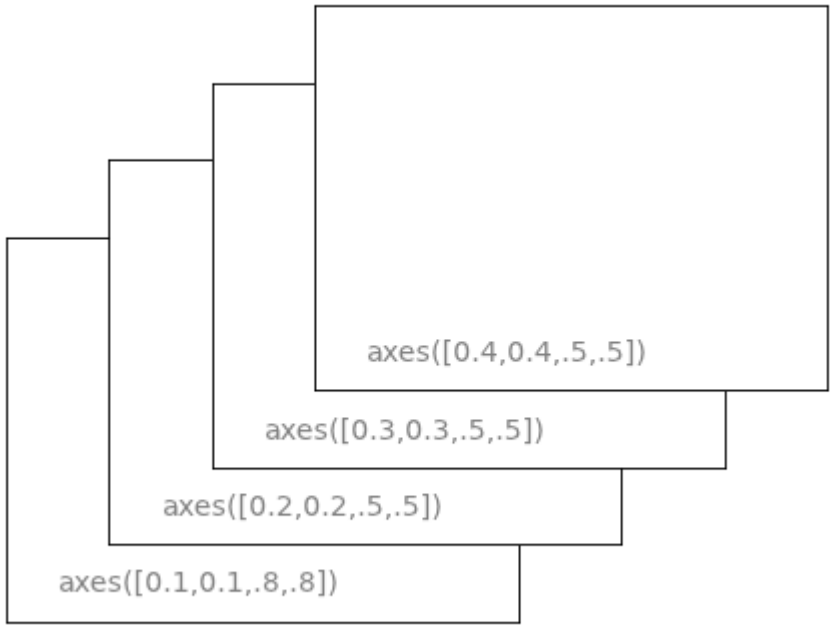
sharex, sharey : Axes, optionalShare the x or y axis with sharex and/or sharey. The axis will have the same limits, ticks, and scale as the axis of the shared axes.

label : strA label for the returned axes.

```
add_subplot(nrows, ncols, index, **kwargs)
add_subplot(pos, **kwargs)
add_subplot(axes)
```

Axes容器

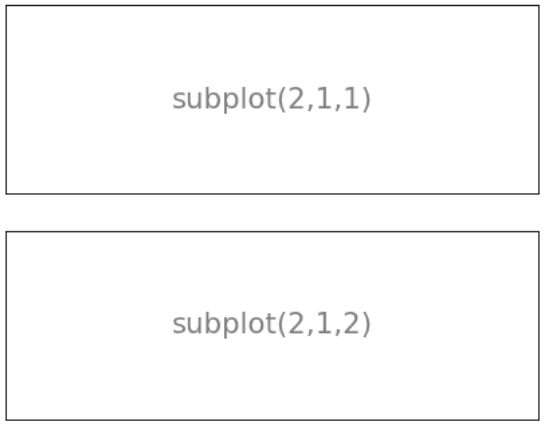


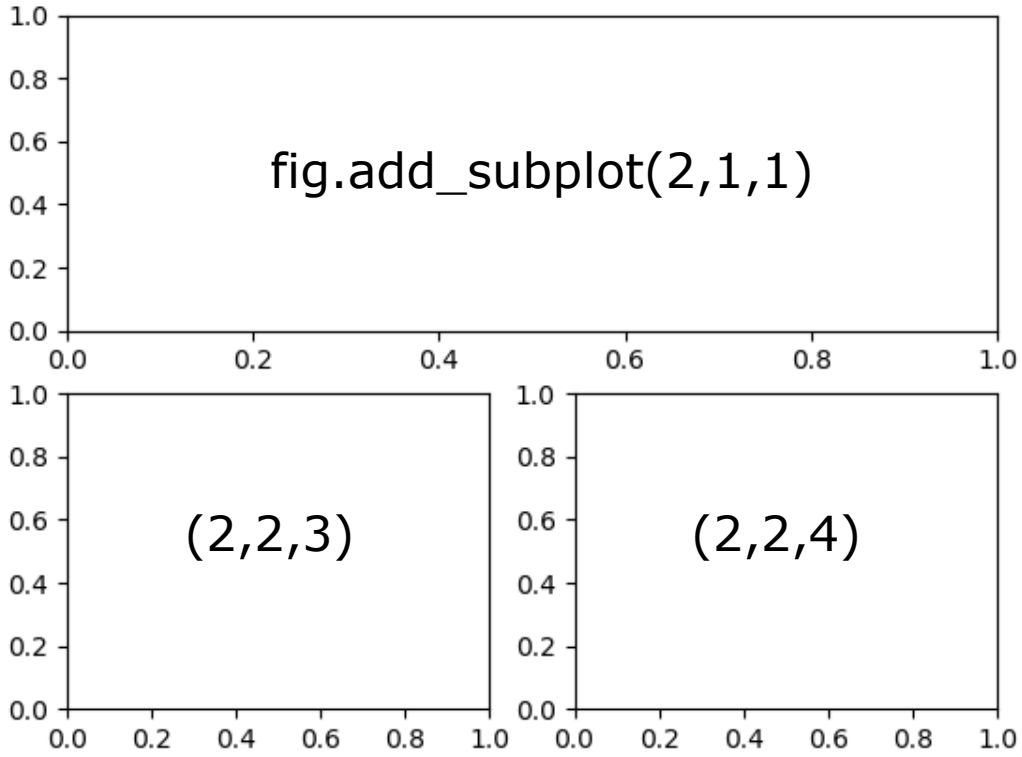


add_axes(left, bottom, width, height)

尺寸以figure的长宽为单位

后画的在上，zorder大的在上





add_subplot(nrows, ncols, index)

3.画图

Axes.xxx

Axes的方法	所创建的对象	添加进的列表	描述
annotate	Annotate	texts	添加注释
bars	Rectangle	patches	柱状图
errorbar	Line2D, Rectangle	lines,patches	带error bar的折线图
fill	Polygon	patches	填充折线与x轴之间空间的图
hist	Rectangle	patches	直方图
imshow	AxesImage	images	给定二维坐标和相应颜色后画图
legend	Legend	legends	显示图例
plot	Line2D	lines	折线图
scatter	PolygonCollection	Collections	散点图
text	Text	texts	添加文字

Axes方法（画图函数）

matplotlibrc文件 → 当前目录 → 用户配置目录 → 系统配置目录

matplotlib.rcdefaults()

恢复缺省配置（默认配置文件的配置）

```
import matplotlib
matplotlib.get_configdir()
```

查看当前用户配置目录路径

```
matplotlib.matplotlib_fname()
```

查看当前用户配置文件路径

```
matplotlib.rcParams
```

载入在本次运行中的配置参数

```
matplotlib.rcParams["lines.marker"] = "o"
```

将默认折线标志点设为圆圈

组名

控制变量名

```
matplotlib.rc("lines", marker="x", linewidth=2, color="red")
```

将默认折线标志点设为×，线宽设为2，颜色设为红色

全局默认参数

Axes.tick_params(self, axis='both', **kwargs)
…

```
fig = plt.figure()
ax = fig.add_subplot(111)
ax.tick_params(axis='x', labelsize=24, color='C4')
```

把x轴的刻度文字大小设为24，刻度颜色设为当前颜色循环中的第四个颜色

```
from matplotlib.ticker import (MultipleLocator,
                               FormatStrFormatter,
                               AutoMinorLocator)
ax.xaxis.set_minor_locator(MultipleLocator(0.1))
```

设置副刻度间隔为0.1

```
ax.set_xscale("log")
```

设置x轴刻度以log均匀分布

```
plt.xscale("log")
```

参数控制：刻度样式

Parameters:	**axis** : {'x', 'y', 'both'}, optionalWhich axis to apply the parameters to.
Other Parameters:	**axis** : {'x', 'y', 'both'}Axis on which to operate; default is 'both'. **reset** : boolIf *True*, set all parameters to defaults before processing other keyword arguments. Default is *False*. **which** : {'major', 'minor', 'both'}Default is 'major'; apply arguments to *which* ticks. **direction** : {'in', 'out', 'inout'}Puts ticks inside the axes, outside the axes, or both. **length** : floatTick length in points. **width** : floatTick width in points. **color** : colorTick color; accepts any mpl color spec. **pad** : floatDistance in points between tick and label. **labelsize** : float or strTick label font size in points or as a string (e.g., 'large'). **labelcolor** : colorTick label color; mpl color spec. **colors** : colorChanges the tick color and the label color to the same value: mpl color spec. **zorder** : floatTick and label zorder. **bottom, top, left, right** : boolWhether to draw the respective ticks. **labelbottom, labeltop, labelleft, labelright** : boolWhether to draw the respective tick labels. **labelrotation** : floatTick label rotation **grid_color** : colorChanges the gridline color to the given mpl color spec. **grid_alpha** : floatTransparency of gridlines: 0 (transparent) to 1 (opaque). **grid_linewidth** : floatWidth of gridlines in points. **grid_linestyle** : stringAny valid Line2D line style spec.

```
fig = plt.figure()
ax = fig.add_subplot(111)
line = ax.plot([1,2])
```

在图轴ax中添加了一条线段，并把该线段的入口返回给line变量

```
c = line.get_color()
```

返回线段对象的颜色

```
c = line.set_color('red')
```

把线段颜色设为红色

```
ax.lines[0].set_lw(5)
```

把线段宽度设为5，可以直接调用图轴中lines列表中的对象操作

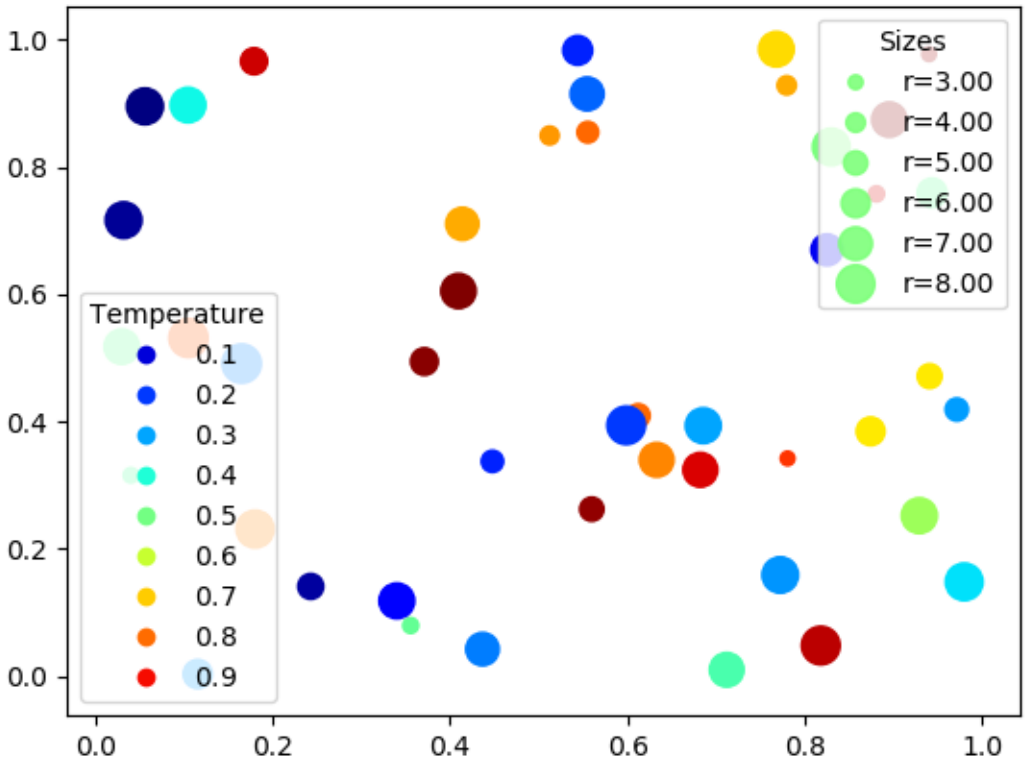
本页的功能基本不会在实际操作中用到，只是作为认识matiplotlib中对象元素的示例

```
ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)
ax.xaxis.set_ticks_position('bottom')
ax.spines['bottom'].set_position(('data',0))
```

参数控制：线段对象

```
N = 45
x, y = np.random.rand(2, N)
c = np.random.rand(N)
s = np.random.rand(N)*100

fig, ax = plt.subplots()

scatter = ax.scatter(x, y, c=c, s=s, cmap='jet')

legend1 = ax.legend(*scatter.legend_elements(),
                    loc="lower left", title="Temperature")
ax.add_artist(legend1)

kw = dict(prop="sizes", num=5, color=scatter.cmap(0.5),
fmt="r={x:.2f}", func=lambda s: np.sqrt(s / np.pi), alpha=0.9)


handles, labels = scatter.legend_elements(**kw)
legend2 = ax.legend(handles, labels, loc="upper right", title="Sizes")
```

如果不添加进axes中，第二个legend会覆盖第一个



散点图

用于生成散点图的图例：返回图例句柄（handle）和文字（label）

参数:
prop：string, optional, default "colors"
Can be "colors" or "sizes". In case of "colors", the legend handles will show the different colors of the collection. In case of "sizes", the legend will show the different sizes.

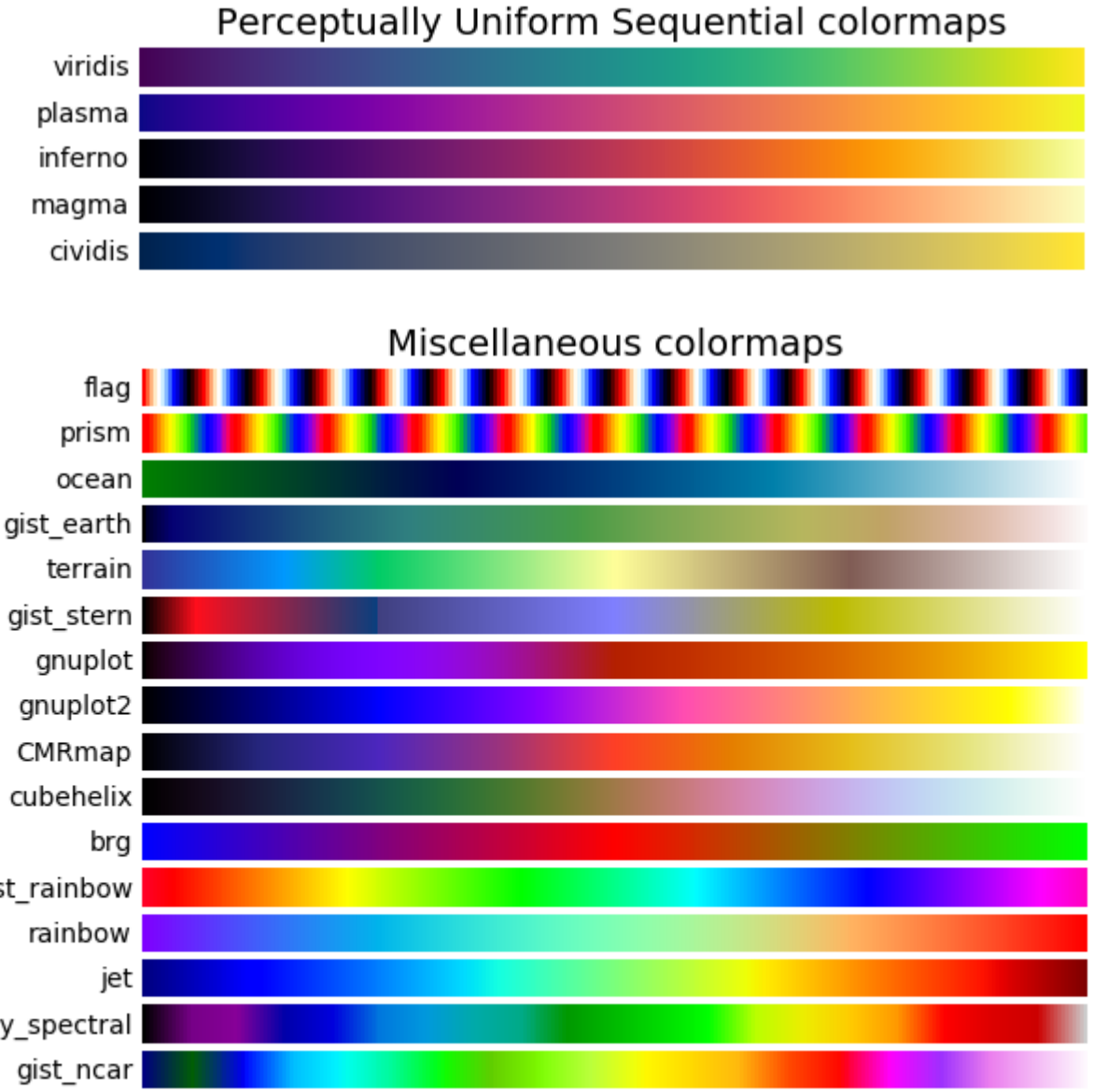
num：int, None, "auto" (default), array-like, or Locator,
optional Target number of elements to create. If None, use all unique elements of the mappable array. If an integer, target to use num elements in the normed range. If "auto", try to determine which option better suits the nature of the data. The number of created elements may slightly deviate from num due to a Locator being used to find useful locations. If a list or array, use exactly those elements for the legend. Finally, a Locator can be provided.

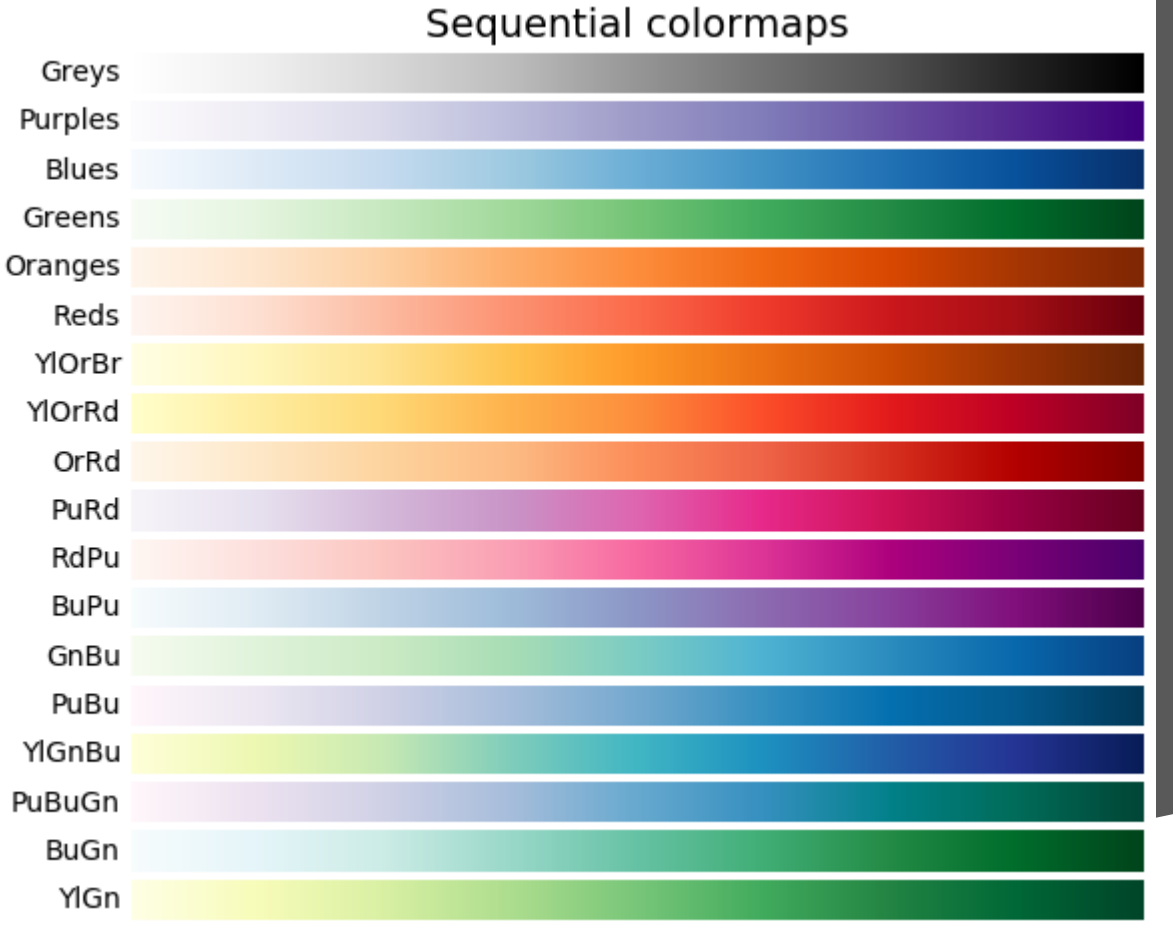
fmt：string, Formatter, or None (default)
The format or formatter to use for the labels. If a string must be a valid input for a StrMethodFormatter. If None (the default), use a ScalarFormatter.

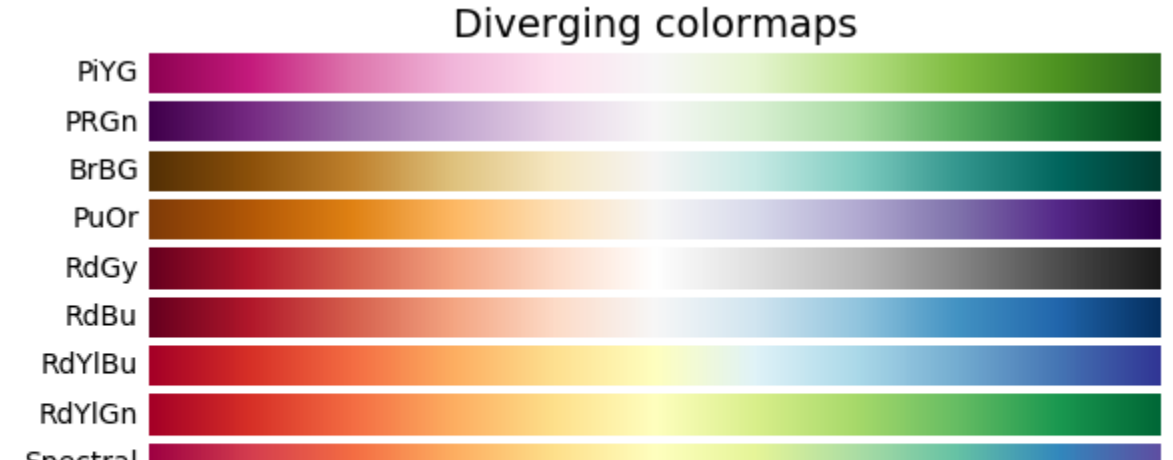
func：function, default lambda x: x
Function to calculate the labels. Often the size (or color) argument to scatter() will have been pre-processed by the user using a function s = f(x) to make the markers visible; e.g. size = np.log10(x). Providing the inverse of this function here allows that pre-processing to be inverted, so that the legend labels have the correct values; e.g. func = np.exp(x, 10).

kwargs：further parameters
Allowed kwargs are color and size. E.g. it may be useful to set the color of the markers if prop="sizes" is used; similarly to set the size of the markers if prop="colors" is used. Any further parameters are passed onto the Line2D instance. This may be useful to e.g. specify a different markeredgecolor or alpha for the legend handles.

scatter.legend_elements



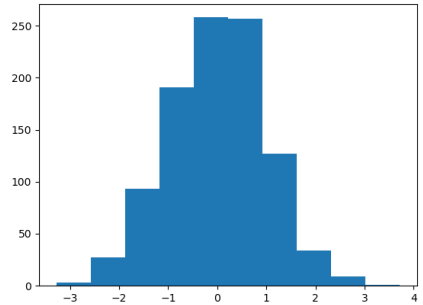




cmap

2019/11/28

```
x=np.random.randn(1000)
fig,ax=plt.subplot(111)
ax.hist(x)
plt.show()
```



```
Axes.hist(self, x,
          bins=None,
          range=None,
          density=None,
          weights=None,
          cumulative=False,
          bottom=None,
          histtype='bar',
          align='mid',
          orientation='vertical',
          rwidth=None,
          log=False,
          color=None,
          label=None,
          stacked=False,
          data=None,
          **kwargs)
```

x：输入数组，一维或二维。如果时N*M的二维数组，则对每一行的的M个数分布计算一次hist并一次性画出直方图

bins：整数或数组。整数时表示分bin的数量，数组时表示bin的边界。默认为10。

denstiy：True时画出概率密度。默认为False。

log：True时Y轴刻度以log增长。等效于axes.set_yscale('log')。

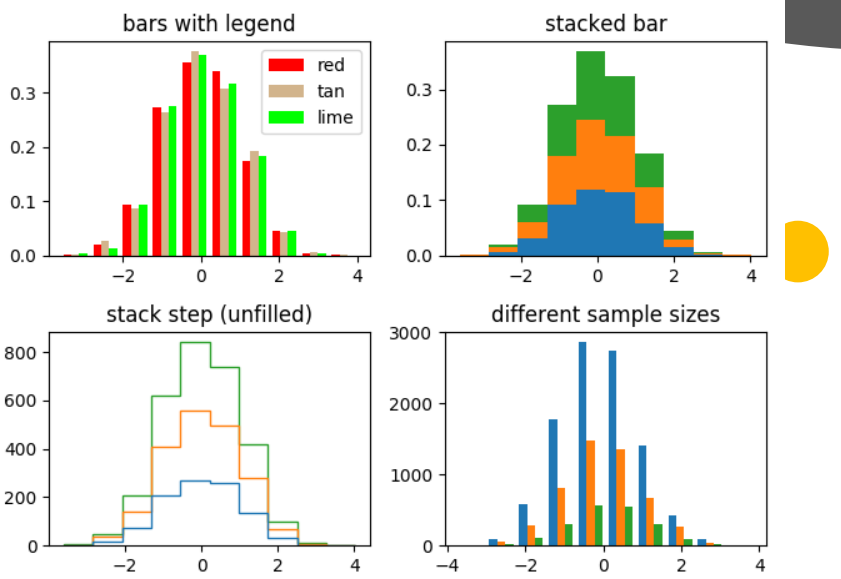
cumulative：True时画数量累积图。

histtype：柱状图的形式。'bar', 'barstacked', 'step', 'stepfilled'

rwidth：每个柱子占bin宽度的比例，范围0-1，默认为1

align：每个柱子与bin的对齐位置，'left', 'right', 'mid'，默认为'mid'

label：图例标签



直方图

Axes.contour([X, Y,] Z, [levels], **kwargs)

X坐标矩阵 | Y坐标矩阵 | Z值矩阵 | 等高线的划分数量（整数）或位置（递增数组）

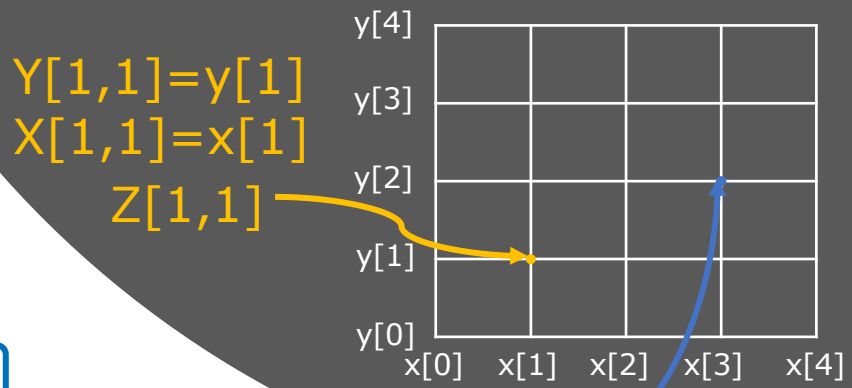
Axes.contourf([X, Y,] Z, [levels], **kwargs)

```
fig = plt.figure()
ax = fig.add_subplot(111)
CS = ax.contour(X, Y, Z,cmap='jet')
ax.clabel(CS, inline=1, fontsize=10)
fig.colorbar(CS, ax=ax, [cax=…])
ax.set_title('Simplest default with labels')
```

在等高线上标记线高

ax:依附的图轴；cax：画在该图轴内

```
delta = 0.025
x = np.arange(-3.0, 3.0, delta)
y = np.arange(-2.0, 2.0, delta)
X, Y = np.meshgrid(x, y)
Z = np.sin(X)**10 + np.cos(10+X*Y) * np.cos(X)
```



Z[3,2]
X[3,2]=x[3]
Y[3,2]=y[2]

等高线图

cmap：渐变颜色表

colors：单一颜色或者颜色数组

vmin,vmax：与渐变色标的0,1相对应的z值

```
ax.contour(X, Y, Z, cmap='jet', vmin=0, vmax=1)
```

linewidths：线宽，可以是单个数字，也可以是数组

linestyles：线型

想要更加细节设置，自行搜索图例读代码

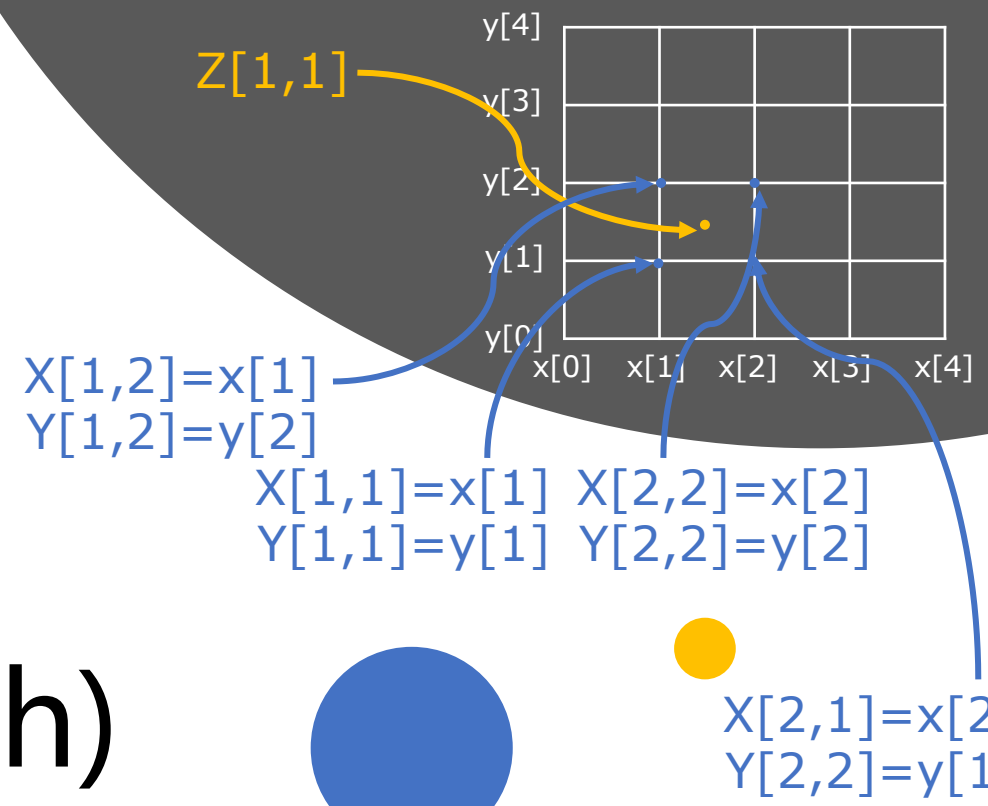
contour方法的其他参数

Axes.pcolormesh([X, Y,] Z, **kwargs)

x边界矩阵 | Y边界矩阵 | Z值矩阵

```
fig = plt.figure()
ax = fig.add_subplot(111)
PS = ax.pcolormesh(X, Y, Zmid,cmap='gist_rainbow')

fig.colorbar(PS, orientation='horizontal')
```

```
delta = 0.025
x = np.arange(-3.0, 3.0, delta)
y = np.arange(-2.0, 2.0, delta)
X, Y = np.meshgrid(x, y)
Xmid = X[:-1,:-1]+delta/2
Ymid = Y[:-1,:-1]+delta/2
Zmid = np.sin(Xmid)**10 +\
     np.cos(10+Xmid*Ymid) * np.cos(Xmid)
```



色彩混合图（color mesh)

做以下模拟

有10000个粒子，每个粒子质量为1，初始在一个100x100的二维方形范围内均匀分布。

粒子在二维平面内运动，运动方式为 随机位置变化+中心引力势场下的加速运动 。
每个粒子在下一秒的位置为以其目前位置为中心，向随机方向移动（0,1）之间的随机距离。
随机运动不会使粒子增加速度。
在盒子中心有一个质量为10的黑洞，会对粒子施加引力。粒子之间没有引力。

请计算经过10000秒后各个粒子的位置。

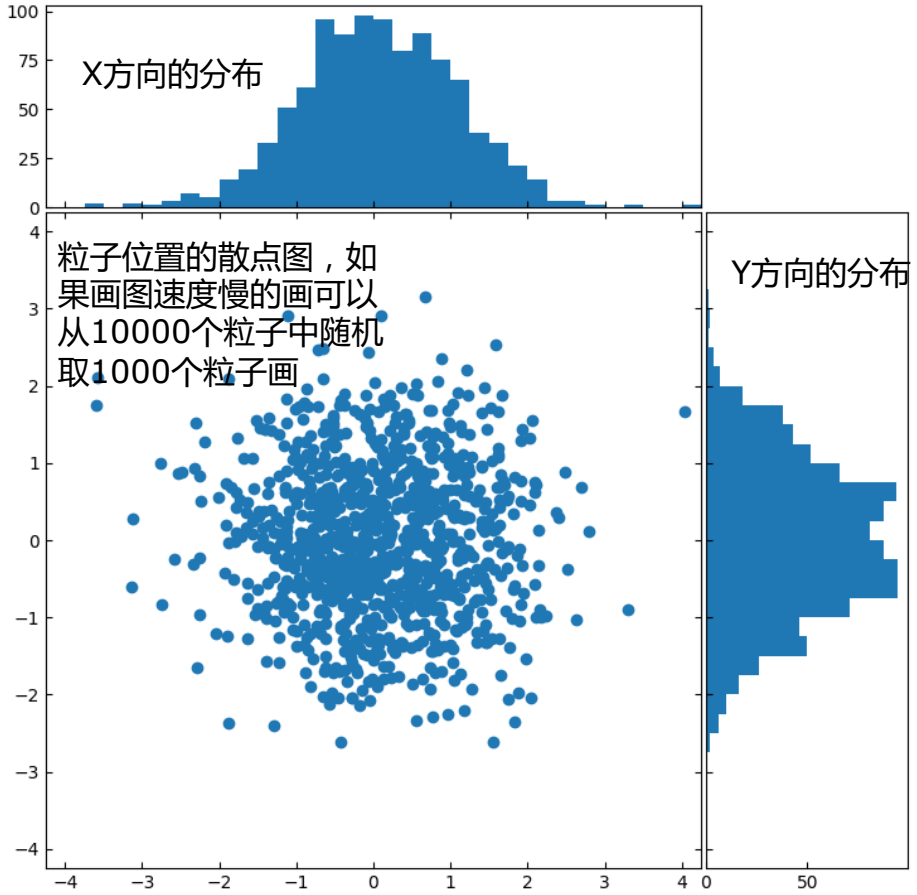
时间，质量，长度单位都为1，G也设为1。粒子运动没有边界。

采用最终的粒子信息，画以下两幅图（X,Y方向的范围限制在原始的100x100区域内）

1.粒子位置分布的散点+直方图。示意图在右侧，注意调整图中的样式（如散点大小，透明度）来改进图片的视觉效果。如何画横向直方图请自行搜索资料。

2.画出平面上粒子平均动能分布的等高线+color mesh图，画出color mesh图的color bar。

交所有的.py文件(用jupyter notebook画的把对应的代码导出到.py文件)。上交前务必测试程序能正常运行。



No+Name+p2

作业