Task02 - 艺术画笔见乾坤

一、概述

1.matplotlib的三层api

1. matplotlib.backend_bases.FigureCanvas 代表了绘图区，所有的图像都是在绘图区完成的。
2. matplotlib.backend_bases.Renderer 代表了渲染器，可以近似理解为画笔，控制如何在 FigureCanvas 上画图。
3. **matplotlib.artist.Artist**代表了具体的图表组件，即调用了Renderer的接口在Canvas上作图。

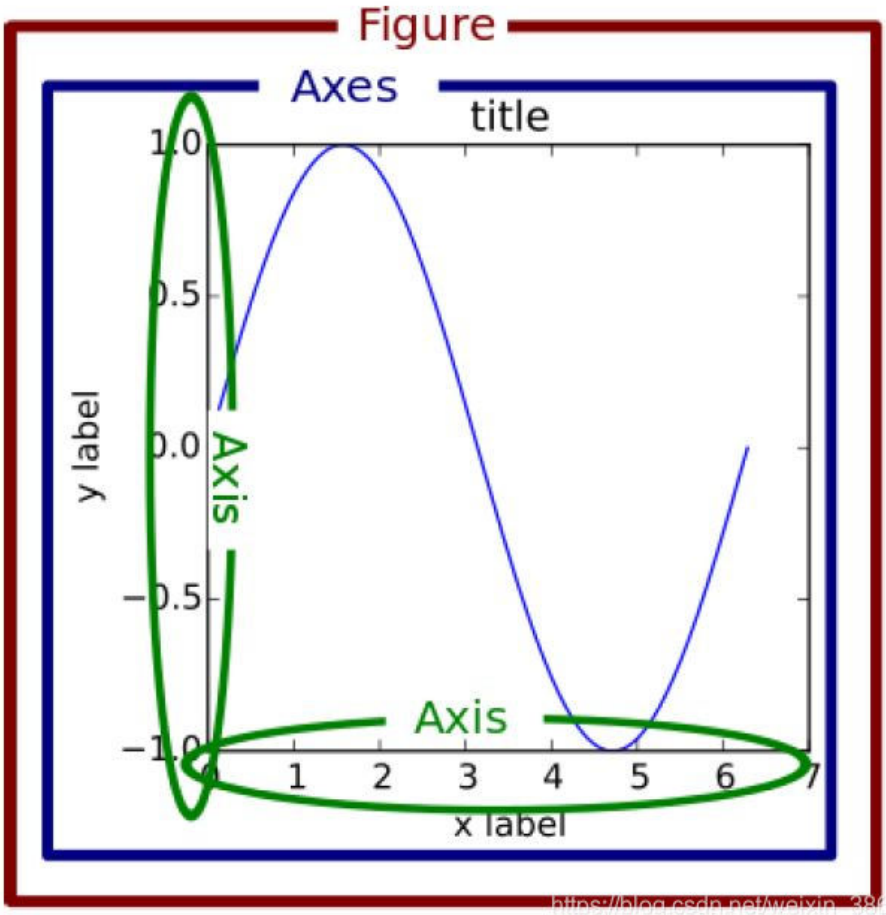
前两者处理程序和计算机的底层交互的事项，第三项Artist就是具体的调用接口来做出我们想要的图，比如图形、文本、线条的设定。所以通常来说，**我们95%的时间，都是用来和matplotlib.artist.Artist类打交道的。**

2. Artist的分类

Artist有两种类型：**primitives** 和 **containers。**

1. primitive是基本要素，它包含一些我们要在绘图区作图用到的标准图形对象，如曲线Line2D，文字text，矩形Rectangle，图像image等。
2. container是容器，即用来装基本要素的地方，包括图形figure、坐标系Axes和坐标轴Axis。

他们之间的关系如下图所示：



3. matplotlib标准用法

matplotlib的标准使用流程为：

1. 创建一个Figure实例
2. 使用Figure实例创建一个或者多个Axes或Subplot实例
3. 使用Axes实例的辅助方法来创建primitive

ps:Axes是一种容器，它可能是matplotlib API中最重要的类，并且我们大多数时间都花在和它打交道上。

一个流程示例及说明如下：

```
t = np.arange(0.0, 1.0, 0.01)
s = np.sin(2_np.pi_t)
line, = ax.plot(t, s, color='blue', lw=2)
line, = ax1.plot(t, s, color='blue', lw=2)
```

二、自定义你的Artist对象

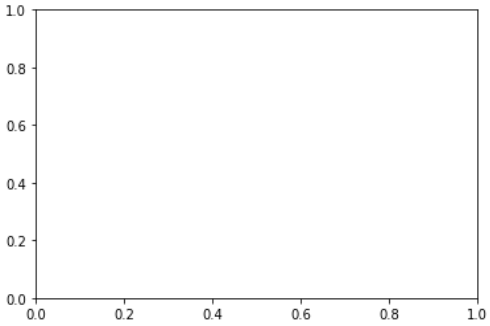
1. Artist属性

- Figure.patch属性：是一个Rectangle，代表了图表的矩形框，它的大小就是图表的大小， 并且可以通过它设置figure的背景色和透明度。
- Axes.patch属性：也是一个Rectangle，代表了绘图坐标轴内部的矩形框（白底黑边）， 通过它可以设置Axes的颜色、透明度等。

每个matplotlib Artist都有以下属性：

1. alpha属性：透明度。值为0—1之间的浮点数
2. axes属性：返回这个Artist所属的axes，可能为None
3. figure属性：该Artist所属的Figure，可能为None
4. label：一个text label
5. visible：布尔值，控制Artist是否绘制

```
# .patch
plt.figure().patch
plt.axes().patch
```



2. 属性调用方式

Artist对象的所有属性都通过相应的 get_* 和 set_* 函数进行读写。

- 一次设置一个属性：如下面将alpha属性设置为当前值的一半，

```
a = o.get_alpha()
o.set_alpha(0.5*a)
```

- 一次设置多个属性：用set方法，

```
o.set(alpha=0.5, zorder=2)
```

- 获取属性：可使用 matplotlib.artist.getp(o,"alpha")，如果指定属性名，则返回对象的该属性值；如果不指定属性名，则返回对象的所有的属性和值。

```
import matplotlib
# Figure rectangle的属性
matplotlib.artist.getp(fig.patch)
```

三、基本元素 - primitives

1. 2DLines

其中常用的的参数有：

- xdata:需要绘制的line中点的在x轴上的取值，若忽略，则默认为range(1,len(ydata)+1)
- ydata:需要绘制的line中点的在y轴上的取值
- linewidth:线条的宽度
- linestyle:线型
- color:线条的颜色
- marker:点的标记
- markersize:标记的size

1.1 如何设置Line2D的属性

- 直接在plot()函数中设置

```
import matplotlib.pyplot as plt
x = range(0,5)
y = [2,5,7,8,10]
plt.plot(x,y, linewidth=10) # 设置线的粗细参数为10
```

- 通过获得线对象，对线对象进行设置

```
line, = plt.plot(x, y, '-')
line.set_antialiased(False) # 关闭抗锯齿功能
```

- 获得线属性，使用setp()函数设置

```
plt.setp(lines, color='r', linewidth=10)
```

1.2 如何绘制lines

- 1) 绘制直线line
  - pyplot方法绘制

```
import matplotlib.pyplot as plt
x = range(0,5)
y = [2,5,7,8,10]
plt.plot(x,y)
```

- Line2D对象绘制

```
import matplotlib.pyplot as plt
from matplotlib.lines import Line2D
fig = plt.figure()
ax = fig.add_subplot(111)
line = Line2D(x, y)
ax.add_line(line)
ax.set_xlim(min(x), max(x))
ax.set_ylim(min(y), max(y))
plt.show()
```

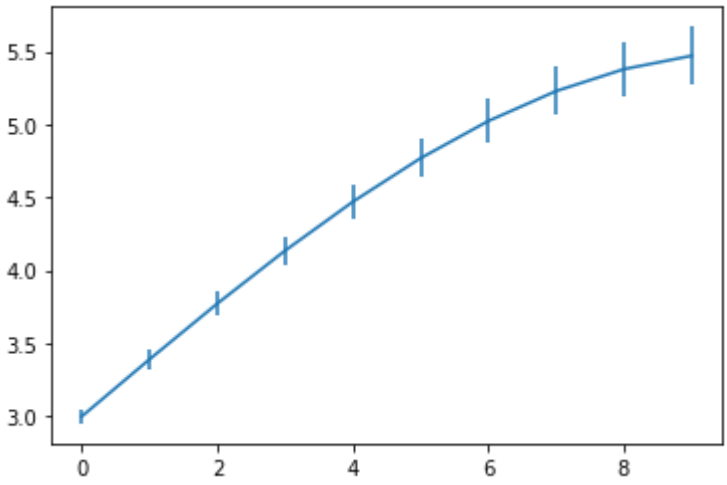
- 2) errorbar绘制误差折线图

pyplot里有个专门绘制误差线的功能，通过errorbar类实现，它的构造函数： ```python matplotlib.pyplot.errorbar(x, y, yerr=None, xerr=None, fmt='', ecolor=None, elinewidth=None, capsize=None, barsabove=False, lolims=False, uplims=False, xlolims=False, xuplims=False, errorevery=1, capthick=None, *, data=None, **kwargs) ``` 其中最主要的参数是前几个:

1. x：需要绘制的line中点的在x轴上的取值
2. y：需要绘制的line中点的在y轴上的取值
3. yerr：指定y轴水平的误差
4. xerr：指定x轴水平的误差
5. fmt：指定折线图中某个点的颜色，形状，线条风格，例如'co–'
6. ecolor：指定error bar的颜色
7. elinewidth：指定error bar的线条宽度

绘制errorbar:

```python import numpy as np import matplotlib.pyplot as plt fig = plt.figure() x = np.arange(10) y = 2.5 * np.sin(x / 20 * np.pi) yerr = np.linspace(0.05, 0.2, 10) plt.errorbar(x, y + 3, yerr=yerr, label='both limits (default)') ```


```

2. patches

matplotlib.patches.Patch类是二维图形类。它的基类是matplotlib.artist.Artist，它的构造函数：

```
Patch(edgecolor=None, facecolor=None, color=None,
linewidth=None, linestyle=None, antialiased=None,
hatch=None, fill=True, capstyle=None, joinstyle=None,
**kwargs)
```

2.1 Rectangle-矩形

Rectangle矩形类在官网中的定义是： 通过锚点xy及其宽度和高度生成。
Rectangle本身的主要比较简单，即xy控制锚点，width和height分别控制宽和高。它的构造函数：

```
class matplotlib.patches.Rectangle(xy, width, height, angle=0.0, **kwargs)
```

在实际中最常见的矩形图是hist直方图和bar条形图。

- 1) hist-直方图

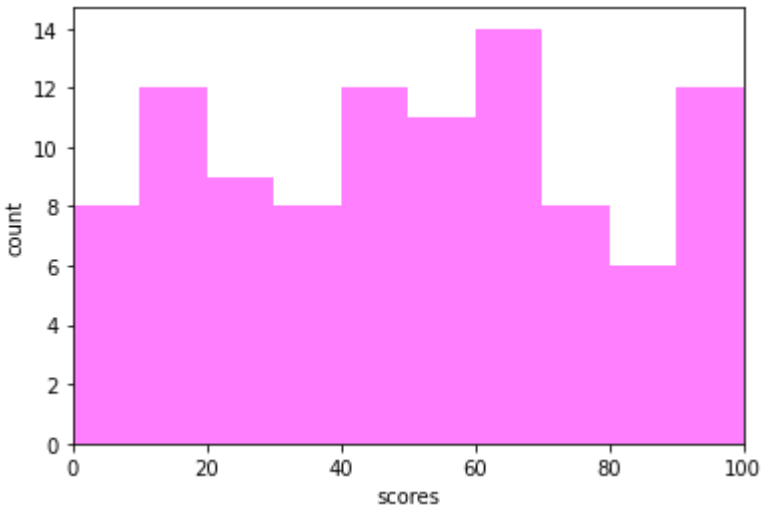
```
matplotlib.pyplot.hist(x,bins=None,range=None, density=None, bottom=None, histtype='bar', align='mid', log=False, color
```

下面是一些常用的参数：

1. x: 数据集，最终的直方图将对数据集进行统计
2. bins: 统计的区间分布
3. range: tuple, 显示的区间，range在没有给出bins时生效
4. density: bool，默认为false，显示的是频数统计结果，为True则显示频率统计结果，这里需要注意，频率统计结果=区间数目/(总数*区间宽度)，和normed效果一致，官方推荐使用density
5. histtype: 可选{'bar', 'barstacked', 'step', 'stepfilled'}之一，默认为bar，推荐使用默认配置，step使用的是梯状，stepfilled则会对梯状内部进行填充，效果与bar类似
6. align: 可选{'left', 'mid', 'right'}之一，默认为'mid'，控制柱状图的水平分布，left或者right，会有部分空白区域，推荐使用默认
7. log: bool，默认False,即y坐标轴是否选择指数刻度
8. stacked: bool，默认为False，是否为堆积状图

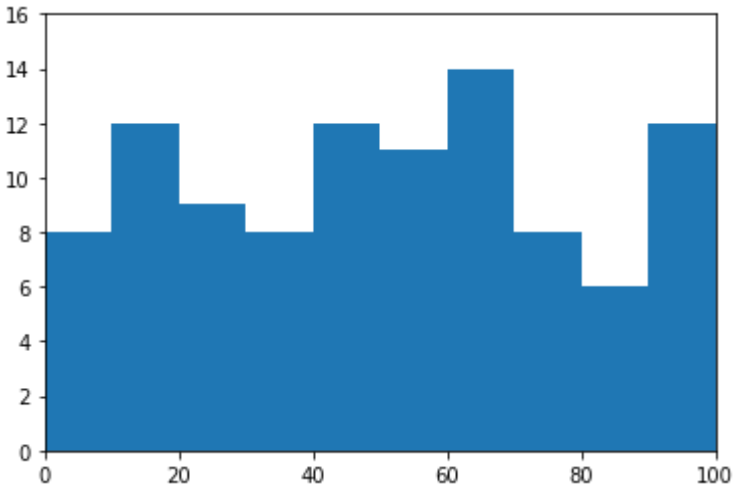
- a) pyplot的hist绘制直方图:

```
import matplotlib.pyplot as plt
import numpy as np
x=np.random.randint(0,100,100) #生成[0-100)之间的100个数据,即 数据集
bins=np.arange(0,101,10) #设置连续的边界值，即直方图的分布区间[0,10),[10,20)...
plt.hist(x,bins,color='fuchsia',alpha=0.5)#alpha设置透明度，0为完全透明
plt.xlabel('scores')
plt.ylabel('count')
plt.xlim(0,100)#设置x轴分布范围 plt.show()
```



- b) Rectangle矩形类绘制直方图:

```
import pandas as pd
import re
df = pd.DataFrame(columns = ['data'])
df.loc[:,'data'] = x
df['fenzu'] = pd.cut(df['data'], bins=bins, right = False,include_lowest=True)
df_cnt = df['fenzu'].value_counts().reset_index()
df_cnt.loc[:,'mini'] = df_cnt['index'].astype(str).map(lambda x:re.findall('\[(.*)\,',x)[0]).astype(int)
df_cnt.loc[:,'maxi'] = df_cnt['index'].astype(str).map(lambda x:re.findall('\,(.*)\)',x)[0]).astype(int)
df_cnt.loc[:,'width'] = df_cnt['maxi']- df_cnt['mini']
df_cnt.sort_values('mini',ascending = True,inplace = True)
df_cnt.reset_index(inplace = True,drop = True)
#用Rectangle把hist绘制出来
import matplotlib.pyplot as plt
fig = plt.figure()
ax1 = fig.add_subplot(111)
#rect1 = plt.Rectangle((0,0),10,10)
#ax1.add_patch(rect)
#ax2 = fig.add_subplot(212)
for i in df_cnt.index:
    rect =  plt.Rectangle((df_cnt.loc[i,'mini'],0),df_cnt.loc[i,'width'],df_cnt.loc[i,'fenzu'])
#rect2 = plt.Rectangle((10,0),10,5)
    ax1.add_patch(rect)
#ax1.add_patch(rect2)
ax1.set_xlim(0, 100)
ax1.set_ylim(0, 16)
plt.show()
```



- 2) bar-柱状图

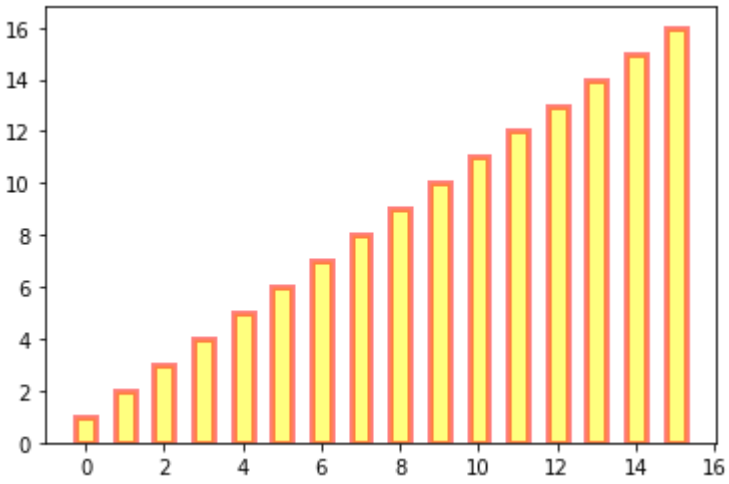
```
matplotlib.pyplot.bar(left, height, alpha=1, width=0.8, color=, edgecolor=, label=, lw=3)
```

下面是一些常用的参数：

1. left：x轴的位置序列，一般采用range函数产生一个序列，但是有时候可以是字符串
2. height：y轴的数值序列，也就是柱形图的高度，一般就是我们需要展示的数据；
3. alpha：透明度，值越小越透明
4. width：为柱形图的宽度，一般这是为0.8即可；
5. color或facecolor：柱形图填充的颜色；
6. edgecolor：图形边缘颜色
7. label：解释每个图像代表的含义，这个参数是为legend()函数做铺垫的，表示该次bar的标签

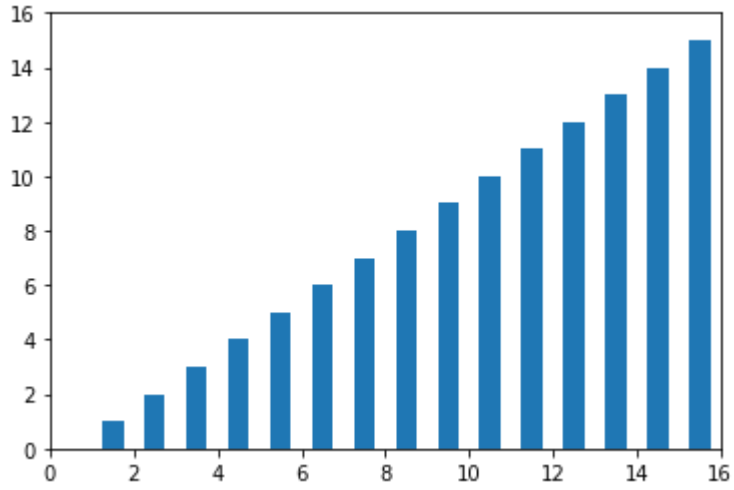
- a) 使用pyplot的bar绘制柱状图

```
import matplotlib.pyplot as plt
y = range(1,17)
plt.bar(np.arange(16), y, alpha=0.5, width=0.5, color='yellow', edgecolor='red', label='The First Bar', lw=3)
```



- b) Rectangle矩形类绘制柱状图

```
#import matplotlib.pyplot as plt
fig = plt.figure()
ax1 = fig.add_subplot(111)
for i in range(1,17):
    rect =  plt.Rectangle((i+0.25,0),0.5,i)
    ax1.add_patch(rect)
ax1.set_xlim(0, 16)
ax1.set_ylim(0, 16)
plt.show()
```



2.2 Polygon-多边形

matplotlib.patches.Polygon类是多边形类。其基类是matplotlib.patches.Patch，它的构造函数：

```
class matplotlib.patches.Polygon(xy, closed=True, **kwargs)
```

xy是一个N×2的numpy array，为多边形的顶点。
closed为True则指定多边形将起点和终点重合从而显式关闭多边形。

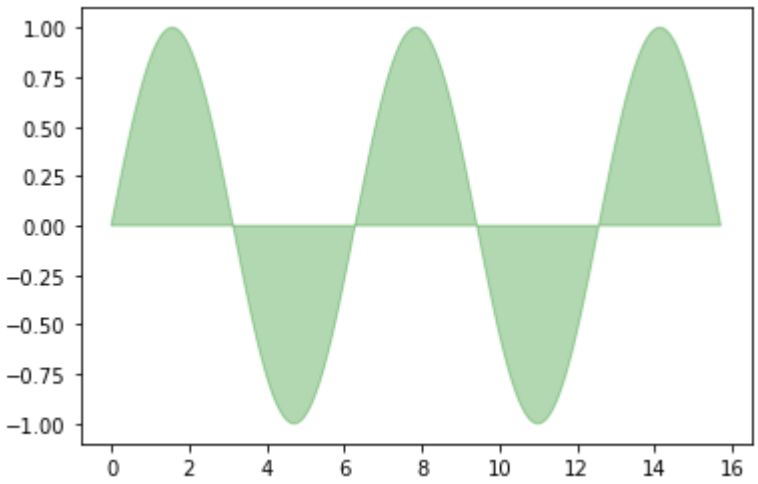
matplotlib.patches.Polygon类中常用的是fill类，它是基于xy绘制一个填充的多边形，它的定义：

```
matplotlib.pyplot.fill(*args, data=None, **kwargs)
```

参数说明：关于x、y和color的序列，其中color是可选的参数，每个多边形都是由其节点的x和y位置列表定义的，后面可以选择一个颜色说明符。您可以通过提供多个x、y、[颜色]组来绘制多个多边形。

- 1) fill绘制图形:

```
import matplotlib.pyplot as plt
x = np.linspace(0, 5 * np.pi, 1000)
y1 = np.sin(x)
plt.fill(x, y1, color = "g", alpha = 0.3)
```



2.3 Wedge-契形

matplotlib.patches.Polygon类是多边形类。其基类是matplotlib.patches.Patch，它的构造函数：

```
class matplotlib.patches.Wedge(center, r, theta1, theta2, width=None, **kwargs)
```

一个Wedge-契形 是以坐标x,y为中心，半径为r，从θ1扫到θ2(单位是度)。
如果宽度给定，则从内半径r -宽度到外半径r画出部分楔形。wedge中比较常见的是绘制饼状图。

matplotlib.pyplot.pie语法：

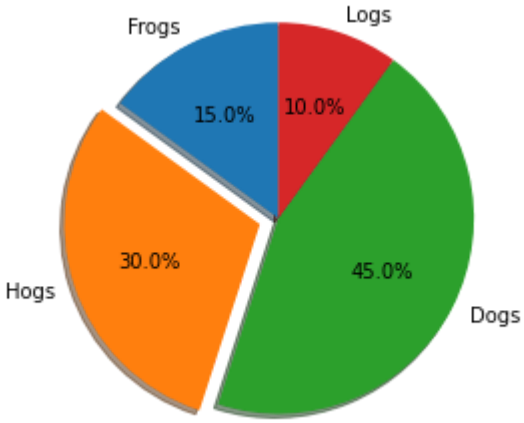
```
matplotlib.pyplot.pie(x, explode=None, labels=None, colors=None, autopct=None,
 pctdistance=0.6, shadow=False, labeldistance=1.1, startangle=0, radius=1,
  counterclock=True, wedgeprops=None, textprops=None, center=0, 0, frame=False,
   rotatelabels=False, *, normalize=None, data=None)
```

制作数据x的饼图，每个楔子的面积用x/sum(x)表示。
其中最主要的参数是前4个：

1. x：契型的形状，一维数组。
2. explode：如果不是等于None，则是一个len(x)数组，它指定用于偏移每个楔形块的半径的分数。
3. labels：用于指定每个契型块的标记，取值是列表或为None。
4. colors：饼图循环使用的颜色序列。如果取值为None，将使用当前活动循环中的颜色。
5. startangle：饼状图开始的绘制的角度。

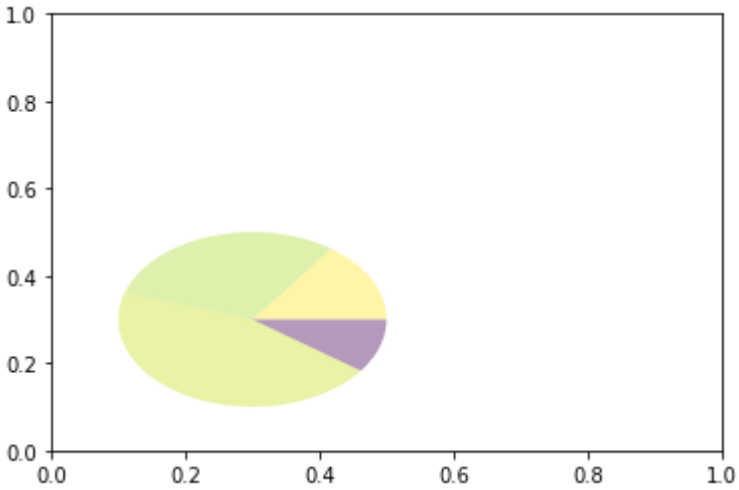
- 1) pie-饼图
  - a) pie绘制饼状图:

```
import matplotlib.pyplot as plt
labels = 'Frogs', 'Hogs', 'Dogs', 'Logs'
sizes = [15, 30, 45, 10]
explode = (0, 0.1, 0, 0)
fig1, ax1 = plt.subplots()
ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%', shadow=True, startangle=90)
ax1.axis('equal') # Equal aspect ratio ensures that pie is drawn as a circle.
plt.show()
```



- b) wedge绘制饼图:

```
import matplotlib.pyplot as plt
from matplotlib.patches import Circle, Wedge
from matplotlib.collections import PatchCollection
fig = plt.figure()
ax1 = fig.add_subplot(111)
theta1 = 0
sizes = [15, 30, 45, 10]
patches = []
patches += [
    Wedge((0.3, 0.3), .2, 0, 54),             # Full circle
    Wedge((0.3, 0.3), .2, 54, 162),  # Full ring
    Wedge((0.3, 0.3), .2, 162, 324),              # Full sector
    Wedge((0.3, 0.3), .2, 324, 360),  # Ring sector
]
colors = 100 * np.random.rand(len(patches))
p = PatchCollection(patches, alpha=0.4)
p.set_array(colors)
ax1.add_collection(p)
plt.show()
```



3. collections

collections类是用来绘制一组对象的集合，collections有许多不同的子类，如RegularPolyCollection, CircleCollection, Pathcollection, 分别对应不同的集合子类型。其中比较常用的就是散点图，它是属于PathCollection子类，scatter方法提供了该类的封装，根据x与y绘制不同大小或颜色标记的散点图。 它的构造方法：

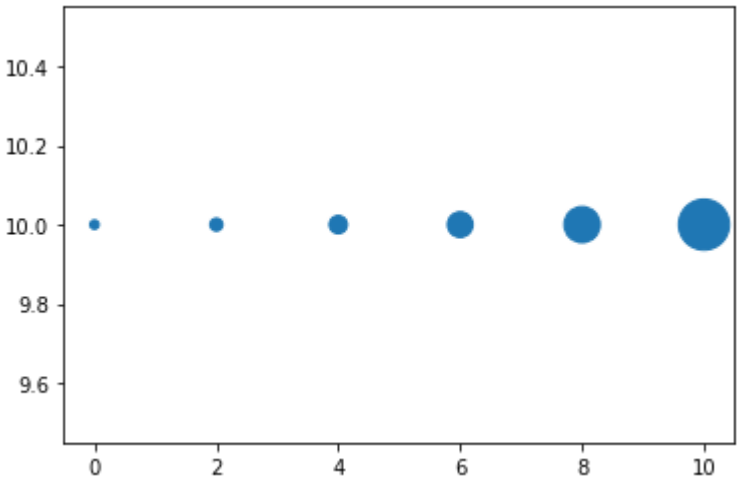
```
Axes.scatter(self, x, y, s=None, c=None, marker=None, cmap=None, norm=None, vmin=None, vmax=None, alpha=None, linewidth
```

其中最主要的参数是前5个：

1. x：数据点x轴的位置
2. y：数据点y轴的位置
3. s：尺寸大小
4. c：可以是单个颜色格式的字符串，也可以是一系列颜色
5. marker: 标记的类型

- a) scatter绘制散点图:

```
x = [0,2,4,6,8,10]
y = [10]*len(x)
s = [20*2**n for n in range(len(x))]
plt.scatter(x,y,s=s)
plt.show()
```



4. images

images是matplotlib中绘制image图像的类，其中最常用的imshow可以根据数组绘制成图像，它的构造函数：

```
class matplotlib.image.AxesImage(ax, cmap=None, norm=None, interpolation=None, origin=None, extent=None, filternorm=Tru
```

imshow根据数组绘制图像:

```
matplotlib.pyplot.imshow(X, cmap=None, norm=None, aspect=None, interpolation=None, alpha=None, vmin=None, vmax=None, or
```

使用imshow画图时首先需要传入一个数组，数组对应的是空间内的像素位置和像素点的值，interpolation参数可以设置不同的差值方法，具体效果如下。

```
import matplotlib.pyplot as plt
import numpy as np
methods = [None, 'none', 'nearest', 'bilinear', 'bicubic', 'spline16',
           'spline36', 'hanning', 'hamming', 'hermite', 'kaiser', 'quadric',
           'catrom', 'gaussian', 'bessel', 'mitchell', 'sinc', 'lanczos']
```

```
grid = np.random.rand(4, 4)

fig, axs = plt.subplots(nrows=3, ncols=6, figsize=(9, 6),
                        subplot_kw={'xticks': [], 'yticks': []})
for ax, interp_method in zip(axs.flat, methods):
    ax.imshow(grid, interpolation=interp_method, cmap='viridis')
    ax.set_title(str(interp_method))

plt.tight_layout()
plt.show()
```