Matplotlib

Pyplot

Pyplot 是 Matplotlib 的子库，提供了和 MATLAB 类似的绘图 API。

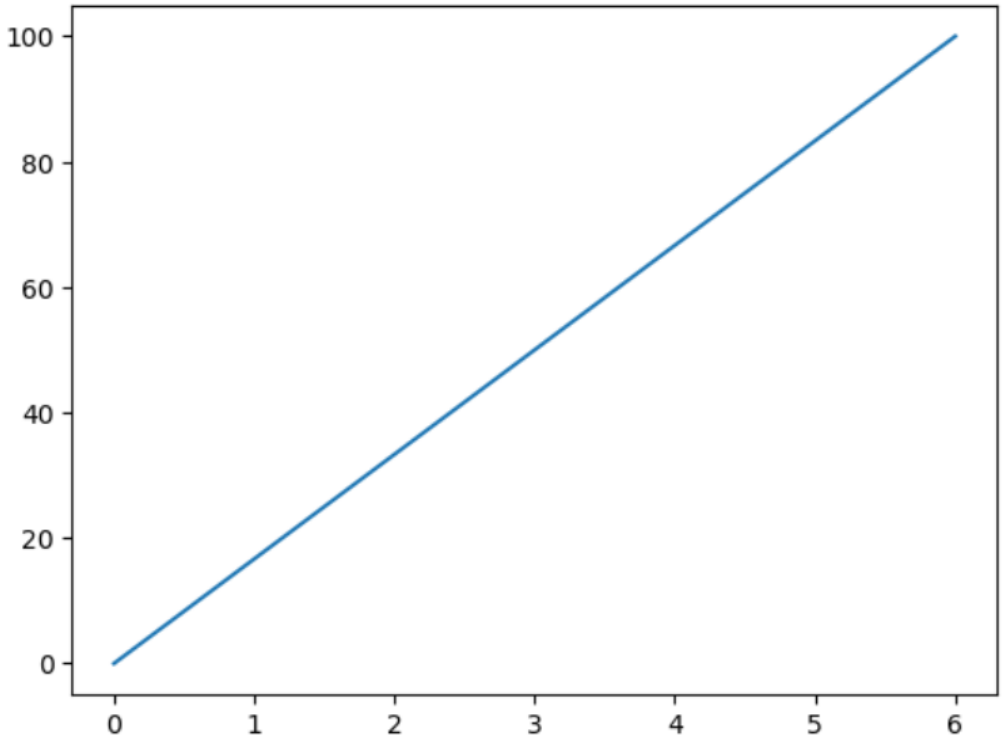
Pyplot 是常用的绘图模块，能很方便让用户绘制 2D 图表。

常用的 pyplot 函数：

- `plot()`：用于绘制线图和散点图
- `scatter()`：用于绘制散点图
- `bar()`：用于绘制垂直条形图和水平条形图
- `hist()`：用于绘制直方图
- `pie()`：用于绘制饼图
- `imshow()`：用于绘制图像
- `subplots()`：用于创建子图

绘制直线

```
xpoints = np.array([0, 6])  # 设置x的点
ypoints = np.array([0, 100])  # 设置对应的y的点

plt.plot(xpoints, ypoints)
plt.show()
```



plot() 用于画图它可以绘制点和线，语法格式如下：

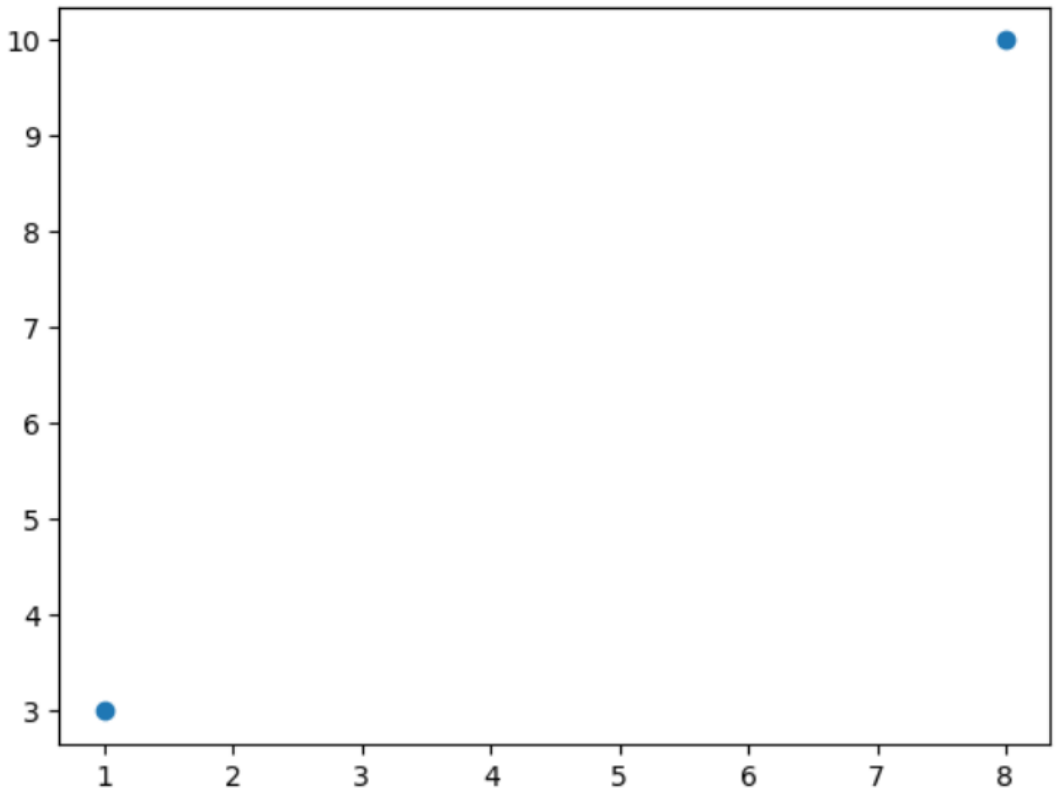
```
# 画单条线
plot([x], y, [fmt], *, data=None, **kwargs)
# 画多条线
plot([x], y, [fmt], [x2], y2, [fmt2], ..., **kwargs)
```

> **参数说明：**
>
> - **x, y：** 点或线的节点，`x` 为 `x` 轴数据，`y` 为 `y` 轴数据，数据可以列表或数组。
> - **fmt：** 可选，定义基本格式（如颜色、标记和线条样式）。
> - **kwargs：** 可选，用在二维平面图上，设置指定属性，如标签，线的宽度等。

> **颜色字符：** `b` 蓝色，`m` 洋红色，`g` 绿色，`y` 黄色，`r` 红色，`k` 黑色，`w` 白色，`c` 青绿色，`#008000` RGB 颜色符串。多条曲线不指定颜色时，会自动选择不同颜色。
>
> **线型参数：** `-` 实线，`--` 破折线，`-.` 点划线，`:` 虚线。
>
> **标记字符：** `.` 点标记，`,` 像素标记(极小点)，`o` 实心圈标记，`v` 倒三角标记，`^` 上三角标记，`>` 右三角标记，`<` 左三角标记...等等。

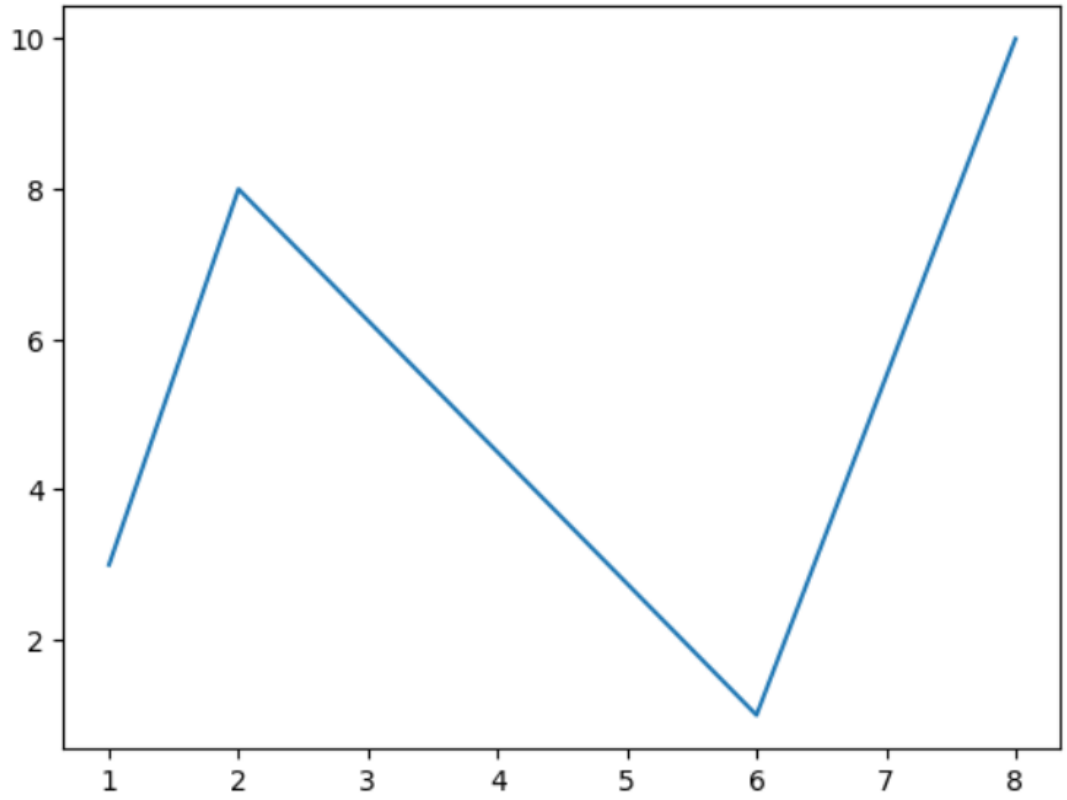
绘制两点坐标

```
xpoints = np.array([1, 8])
ypoints = np.array([3, 10])

plt.plot(xpoints, ypoints, `o`)
plt.show()
```



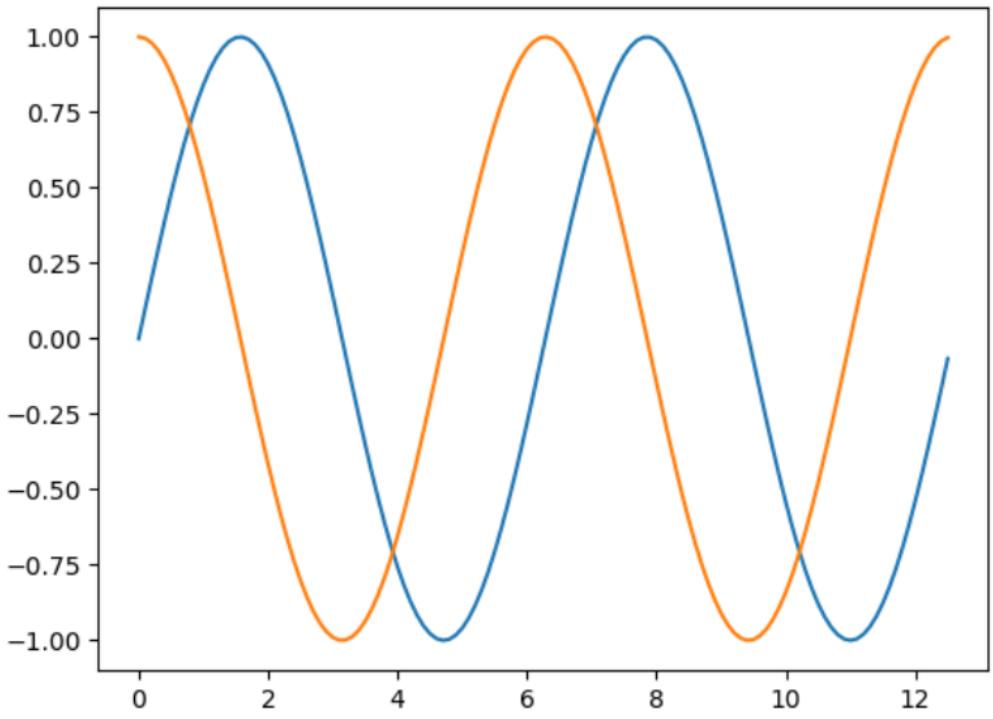
绘制折线图

```
xpoints = np.array([1, 2, 6, 8])
ypoints = np.array([3, 8, 1, 10])

plt.plot(xpoints, ypoints)
plt.show()
```



绘制正弦余弦函数

```
x = np.arange(0,4*np.pi,0.1)   # start,stop,step
y = np.sin(x)
z = np.cos(x)
plt.plot(x,y,x,z)
plt.show()
```



绘图标记

使用maker参数来定义图上的标记

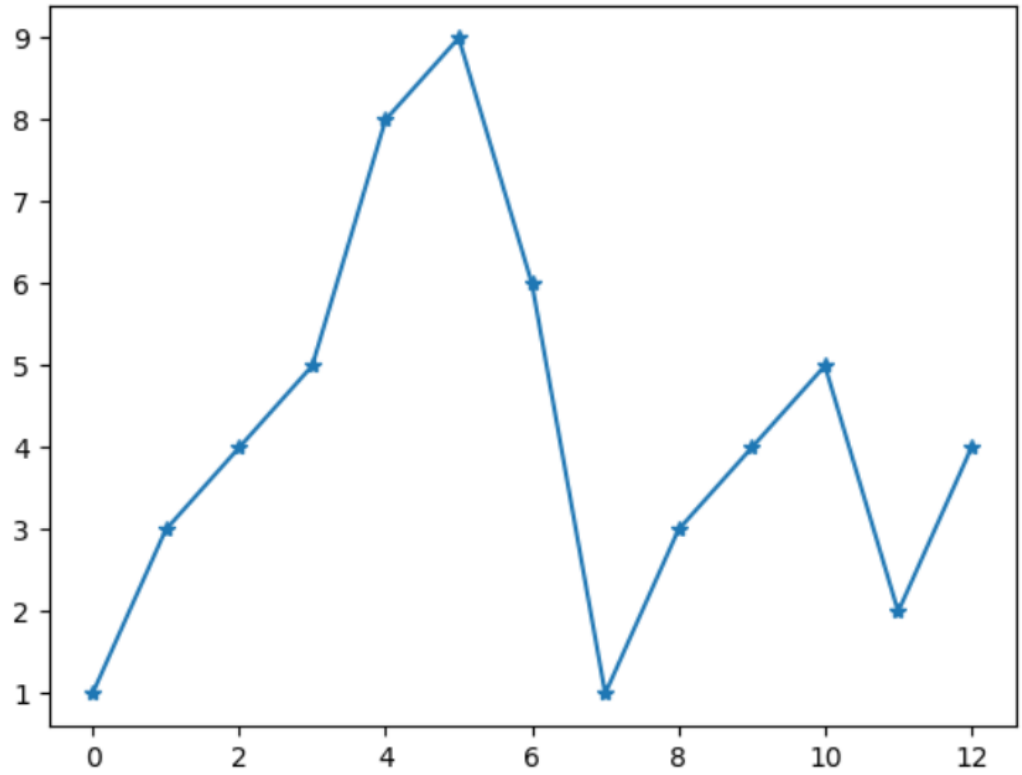
marker 可以定义的符号如下：

标记	符号	描述
`.`		点
`,`		像素点
`o`		实心圆
`v`		下三角
`^`		上三角
`<`		左三角
`>`		右三角
`1`		下三叉
`2`		上三叉
`3`		左三叉
`4`		右三叉
`8`		八角形
`s`		正方形
`p`		五边形
`P`		加号（填充）
`*`		星号
`h`		六边形 1
`H`		六边形 2
`+`		加号
`x`		乘号 `x`
`X`		乘号 `x` (填充)
`D`		菱形

标记	符号	描述
`d`		瘦菱形
`|`		竖线
`_`		横线
`0 (TICKLEFT)`		左横线
`1 (TICKRIGHT)`		右横线
`2 (TICKUP)`		上竖线
`3 (TICKDOWN)`		下竖线
`4 (CARETLEFT)`		左箭头
`5 (CARETRIGHT)`		右箭头
`6 (CARETUP)`		上箭头
`7 (CARETDOWN)`		下箭头
`8 (CARETLEFTBASE)`		左箭头 (中间点为基准)
`9 (CARETRIGHTBASE)`		右箭头 (中间点为基准)
`10 (CARETUPBASE)`		上箭头 (中间点为基准)
`11 (CARETDOWNBASE)`		下箭头 (中间点为基准)
`None` , `or`		没有任何标记
`...`		渲染指定的字符。例如 f 以字母`f`为标记。

使用星花进行标记

```
ypoints = np.array([1,3,4,5,8,9,6,1,3,4,5,2,4])

plt.plot(ypoints, marker = `*`)
plt.show()
```



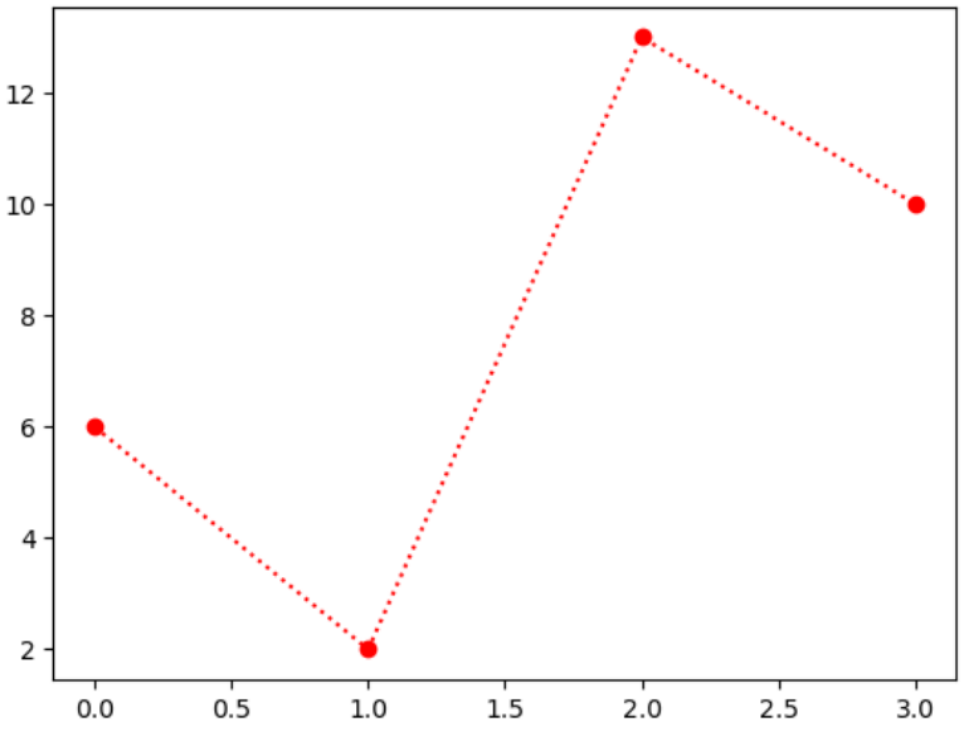
`fmt`参数

`fmt` 参数定义了基本格式，如标记、线条样式和颜色。

> `fmt` = `[marker][line][color]`

例如：`o:r`，`o`表示实心圆标记，`:`表示虚线，`r` 表示颜色为红色。

```
ypoints = np.array([6, 2, 13, 10])

plt.plot(ypoints, `o:r`)
plt.show()
```



线类型：

线类型标记	描述
`-`	实线
`:`	虚线
`--`	破折线
`-.`	点划线

颜色类型：

颜色标记	描述
`r`	红色
`g`	绿色
`b`	蓝色
`c`	青色
`m`	品红
`y`	黄色
`k`	黑色
`w`	白色

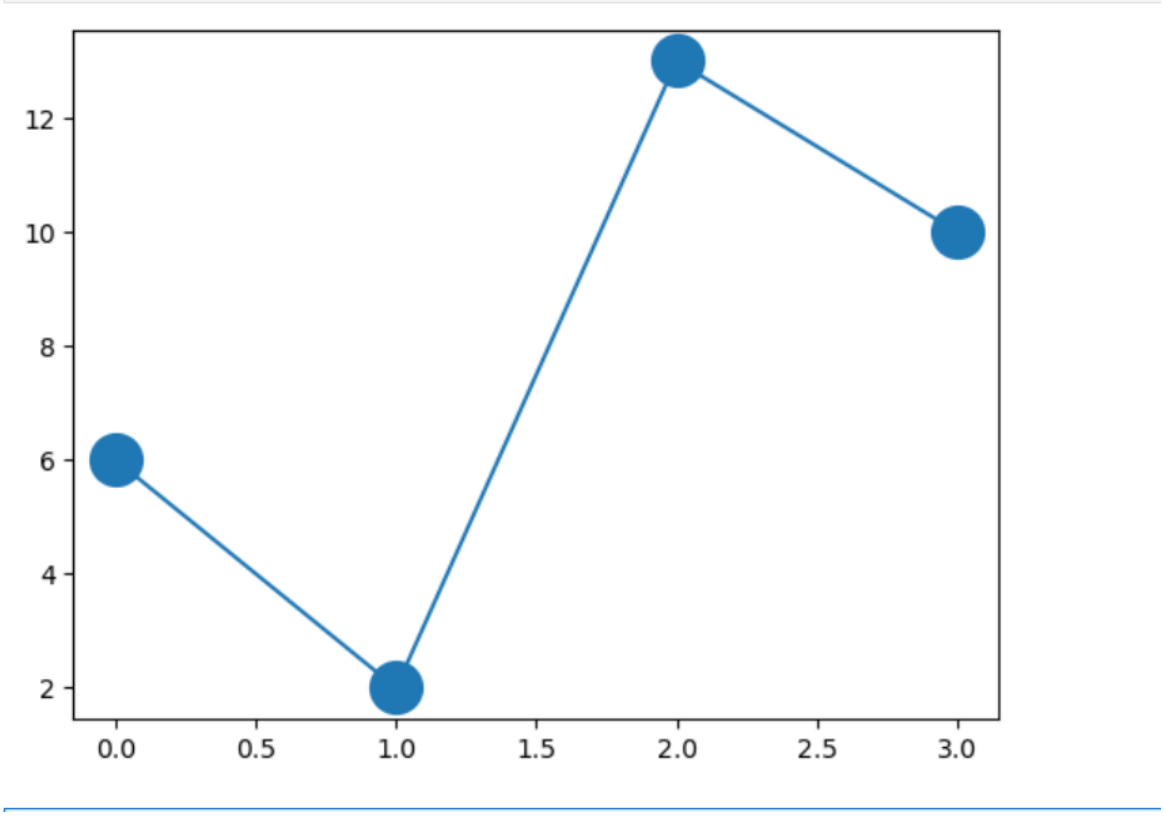
标记大小与颜色

我们可以自定义标记的大小与颜色，使用的参数分别是：

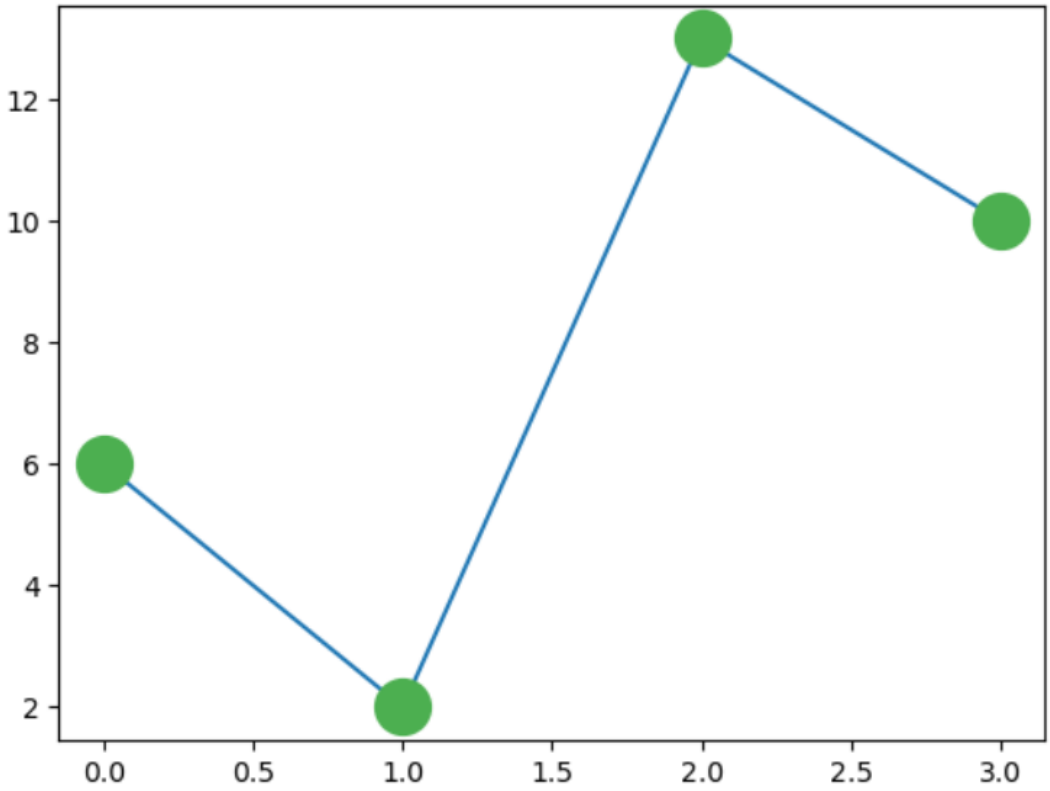
- `markersize`，简写为 `ms`：定义标记的大小。
- `markerfacecolor`，简写为 `mfc`：定义标记内部的颜色。
- `markeredgecolor`，简写为 `mec`：定义标记边框的颜色。

例子：

```
ypoints = np.array([6, 2, 13, 10])

plt.plot(ypoints, marker = `o`, ms = 20)
plt.show()
```




```
# 自定义标记内部与边框的颜色：
ypoints = np.array([6, 2, 13, 10])
plt.plot(ypoints, marker = `o`, ms = 20, mec = `#4CAF50`, mfc =
`#4CAF50`)
plt.show()
```



绘图线

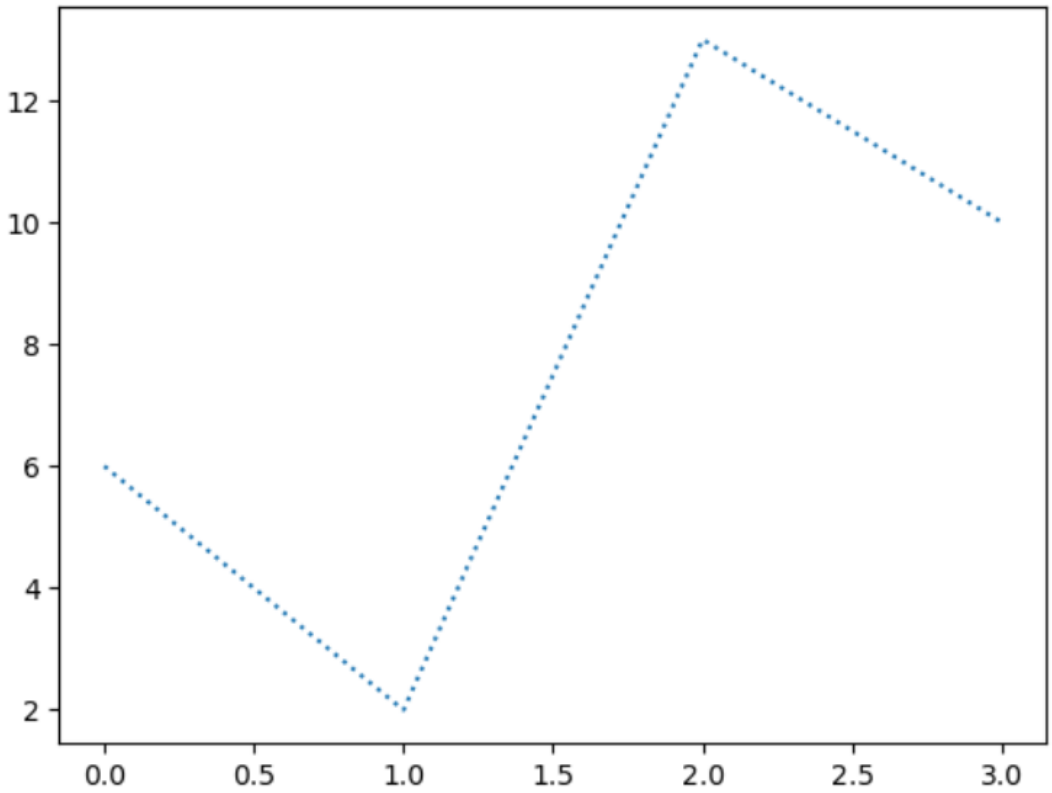
线的类型

线的类型可以使用 `linestyle` 参数来定义，简写为 `ls`。

类型	简写	说明
`solid` (默认)	`-`	实线
`dotted`	`:`	点虚线
`dashed`	`--`	破折线
`dashdot`	`-.`	点划线
`None`	或 `	不画线

虚线

```
ypoints = np.array([6, 2, 13, 10])

plt.plot(ypoints, linestyle = `dotted`)
plt.show()
```



线的颜色

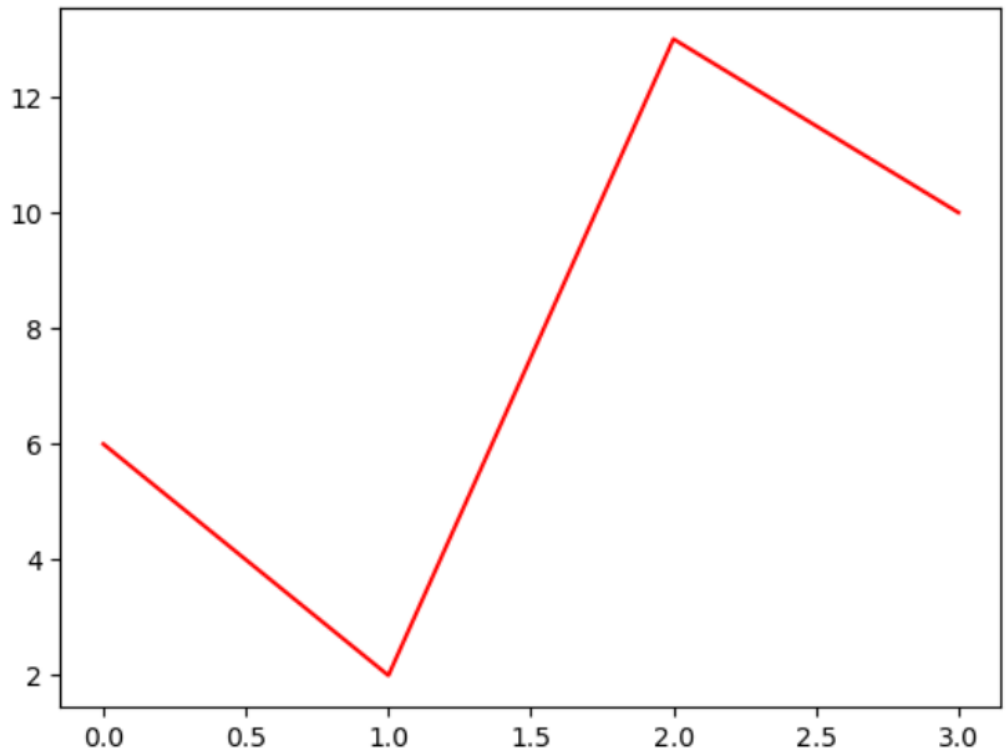
线的颜色可以使用 **color** 参数来定义，简写为 **c**。

颜色类型：

颜色标记	描述
`r`	红色
`g`	绿色
`b`	蓝色
`c`	青色
`m`	品红
`y`	黄色
`k`	黑色
`w`	白色

红色的线

```
ypoints = np.array([6, 2, 13, 10])

plt.plot(ypoints, color = 'r')
plt.show()
```

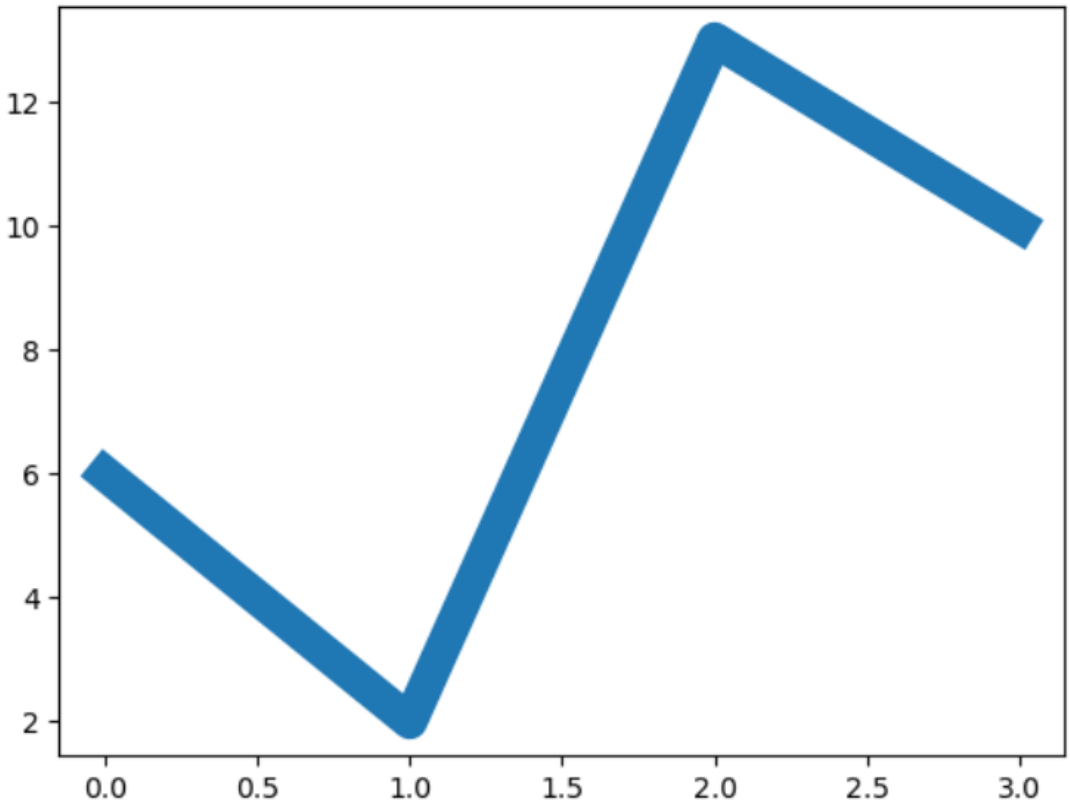


线的宽度

线的宽度可以使用 **linewidth** 参数来定义，简写为 **lw**，值可以是浮点数，如：**1**、**2.0**、**5.67** 等。

例子

```
ypoints = np.array([6, 2, 13, 10])

plt.plot(ypoints, linewidth = '12.5')
plt.show()
```



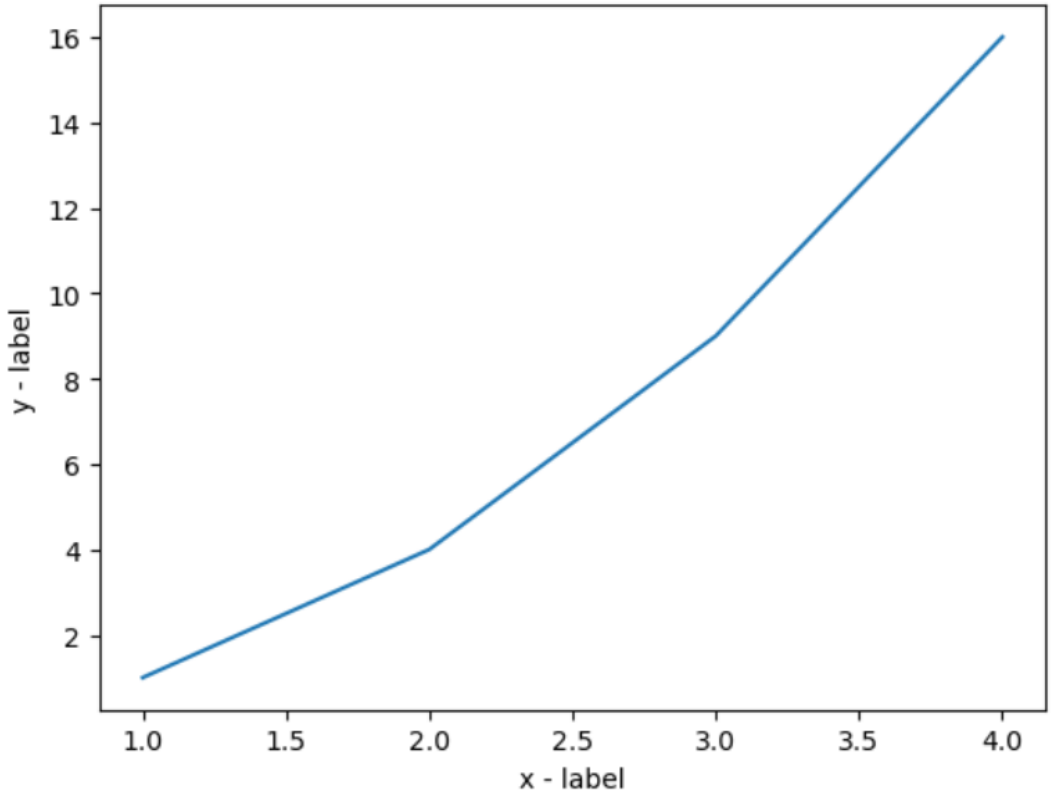
轴标签和标题

轴标签

xlable()、ylable()

使用 **xlabel()** 和 **ylabel()** 方法来设置 `x` 轴和 `y` 轴的标签。

```
x = np.array([1, 2, 3, 4])
y = np.array([1, 4, 9, 16])
plt.plot(x, y)

plt.xlabel("x - label")
plt.ylabel("y - label")

plt.show()
```

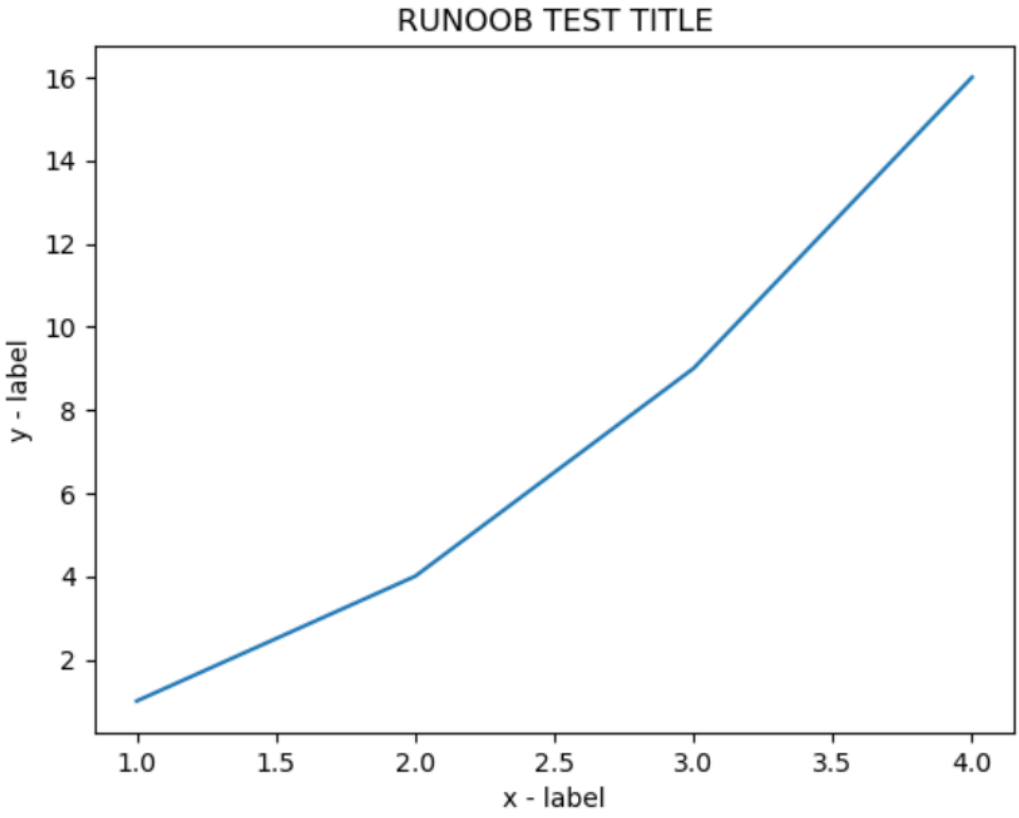


标题

`title()`

使用 **title()** 方法来设置标题。

```
x = np.array([1, 2, 3, 4])
y = np.array([1, 4, 9, 16])
plt.plot(x, y)

plt.title("RUNOOB TEST TITLE")
plt.xlabel("x - label")
plt.ylabel("y - label")

plt.show()
```



网格线

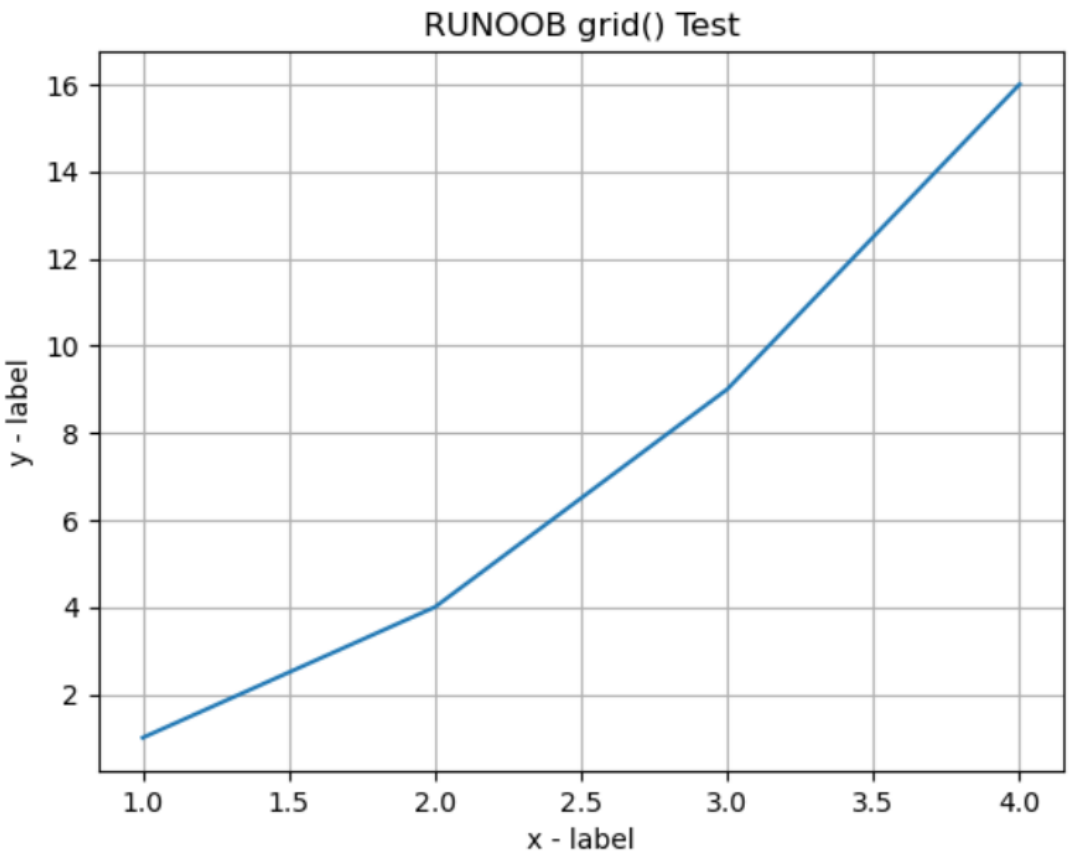
grid()

`grid()`方法语法格式如下：

```
matplotlib.pyplot.grid(b=None, which='major', axis='both', )
```

> **参数说明：**
>
> - **b**：可选，默认为 `None`，可以设置布尔值，`True` 为显示网格线，`false` 为不显示，如果设置 `**kwargs` 参数，则值为 `True`。
> - **which**：可选，可选值有 `major`、`minor` 和 `both`，默认为 `major`，表示应用更改的网格线。
> - **axis**：可选，设置显示哪个方向的网格线，可以是取 `'both'`（默认），`'x'` 或 `'y'`，分别表示两个方向，`x` 轴方向或 `y` 轴方向。
> - ****kwargs**：可选，设置网格样式，可以是 `color='r'`, `linestyle='-'` 和 `linewidth=2`，分别表示网格线的颜色，样式和宽度。

```
x = np.array([1, 2, 3, 4])
y = np.array([1, 4, 9, 16])


plt.title("RUNOOB grid() Test")
plt.xlabel("x - label")
plt.ylabel("y - label")

plt.plot(x, y)

plt.grid()

plt.show()
```

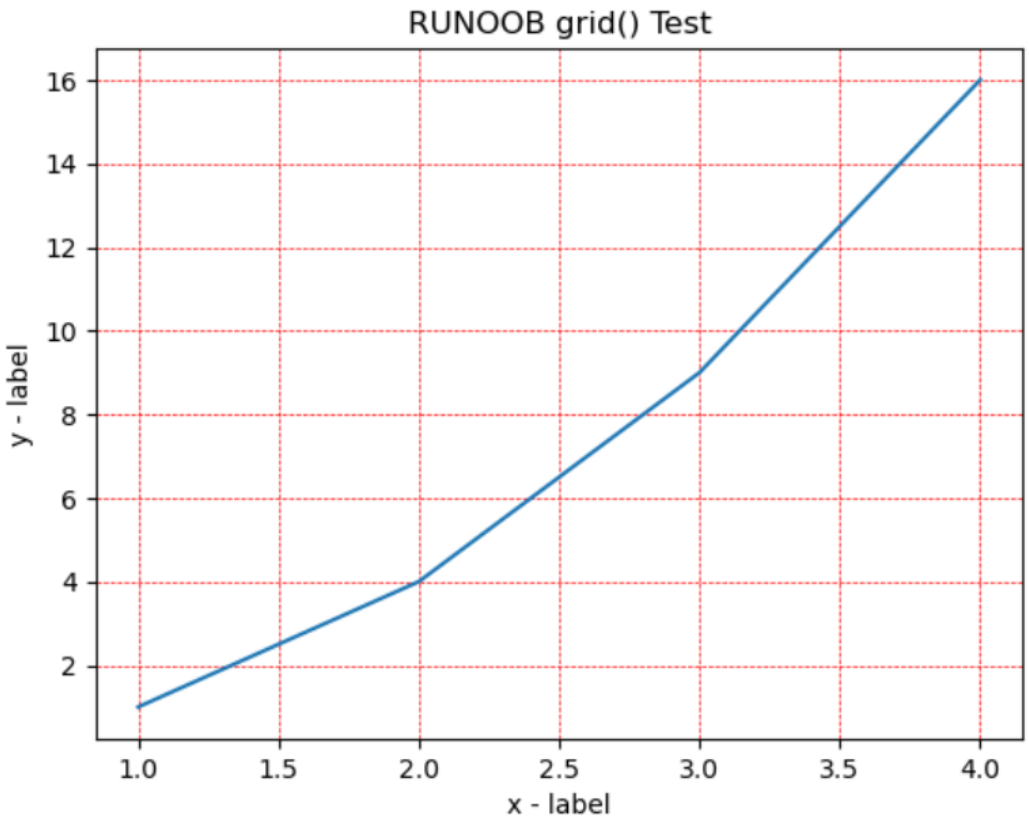


添加一个简单的网格线，并设置网格线的样式，格式如下：

```
grid(color = 'color', linestyle = 'linestyle', linewidth = number)
```

> **参数说明：**
>
> - **color**：'b' 蓝色，'m' 洋红色，'g' 绿色，'y' 黄色，'r' 红色，'k' 黑色，'w' 白色，'c' 青绿色，'#008000' RGB 颜色符串。
> - **linestyle**：'-' 实线，'--' 破折线，'-.' 点划线，':' 虚线。
> - **linewidth**：设置线的宽度，可以设置一个数字。

```
x = np.array([1, 2, 3, 4])
y = np.array([1, 4, 9, 16])


plt.title("RUNOOB grid() Test")
plt.xlabel("x - label")
plt.ylabel("y - label")

plt.plot(x, y)

plt.grid(color = 'r', linestyle = '--', linewidth = 0.5)

plt.show()
```



绘制多图

`subplots()`

`subplots()`方法语法格式如下：

```
matplotlib.pyplot.subplots(nrows=1, ncols=1, *, sharex=False,
sharey=False, squeeze=True, subplot_kw=None, gridspec_kw=None,
**fig_kw)
```

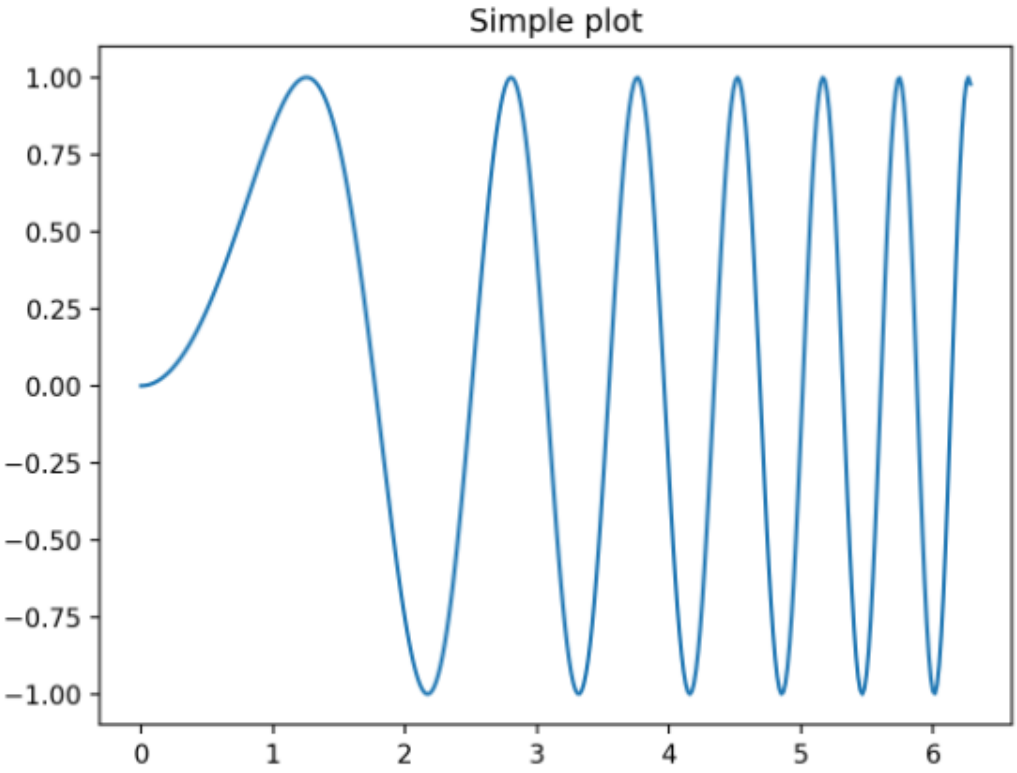
> **参数说明：**
>
> - **nrows**：默认为`1`，设置图表的行数。

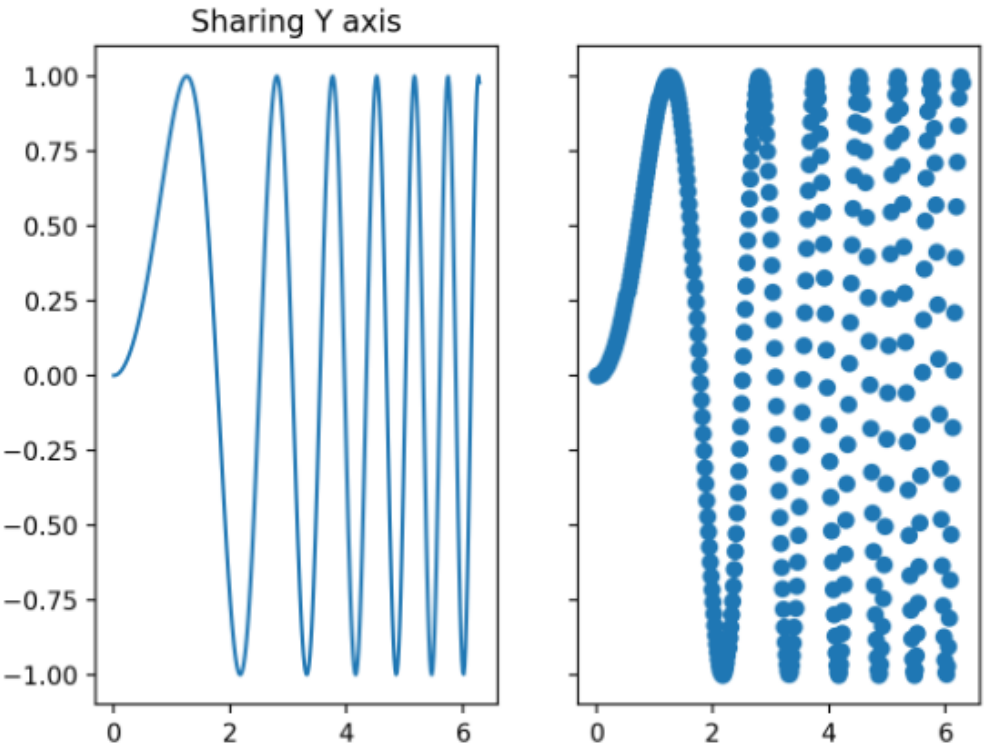
- **ncols**：默认为 `1`，设置图表的列数。
- **sharex、sharey**：设置 `x、y` 轴是否共享属性，默认为 `False`，可设置为 `'none'`、`'all'`、`'row'` 或 `'col'`。 `False` 或 `none` 每个子图的 `x` 轴或 `y` 轴都是独立的，`True` 或 `'all'`：所有子图共享 `x` 轴或 `y` 轴，`'row'` 设置每个子图行共享一个 `x` 轴或 `y` 轴，`'col'`：设置每个子图列共享一个 `x` 轴或 `y` 轴。
- **squeeze**：布尔值，默认为 `True`，表示额外的维度从返回的 `Axes`(轴)对象中挤出，对于 $N \times 1$ 或 $1 \times N$ 个子图，返回一个 `1` 维数组，对于 $N \times M$，`N>1` 和 `M>1` 返回一个 `2` 维数组。如果设置为 `False`，则不进行挤压操作，返回一个元素为 `Axes` 实例的`2`维数组，即使它最终是`1x1`。
- **subplot_kw**：可选，字典类型。把字典的关键字传递给 `add_subplot()` 来创建每个子图。
- **gridspec_kw**：可选，字典类型。把字典的关键字传递给`GridSpec` 构造函数创建子图放在网格里(grid)。
- **fig_kw**：把详细的关键字参数传给`figure()`函数。

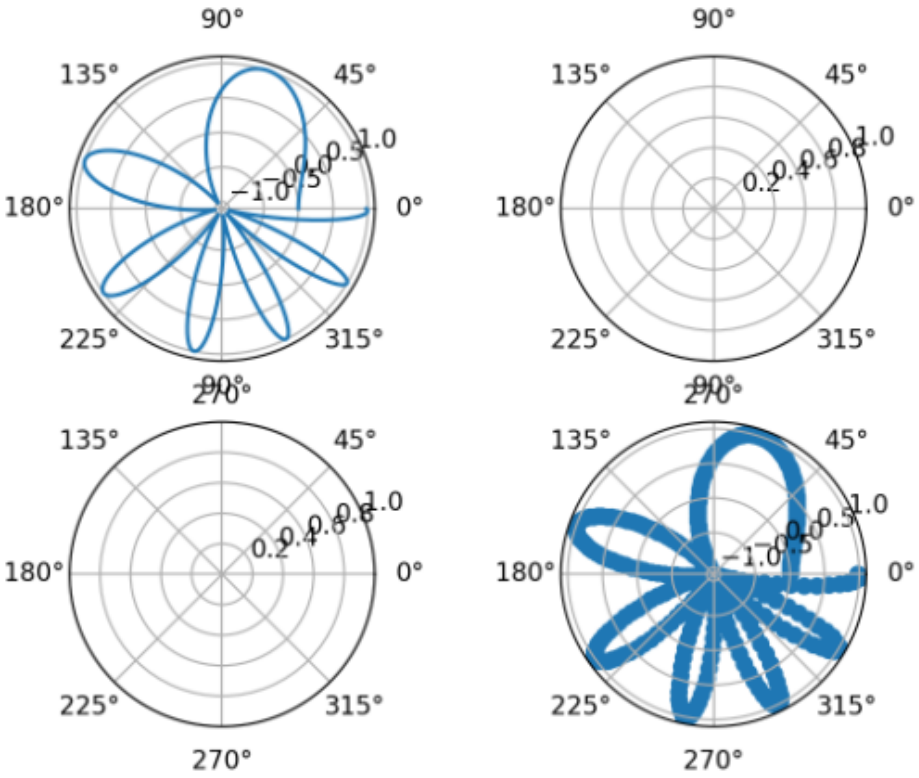
实例：

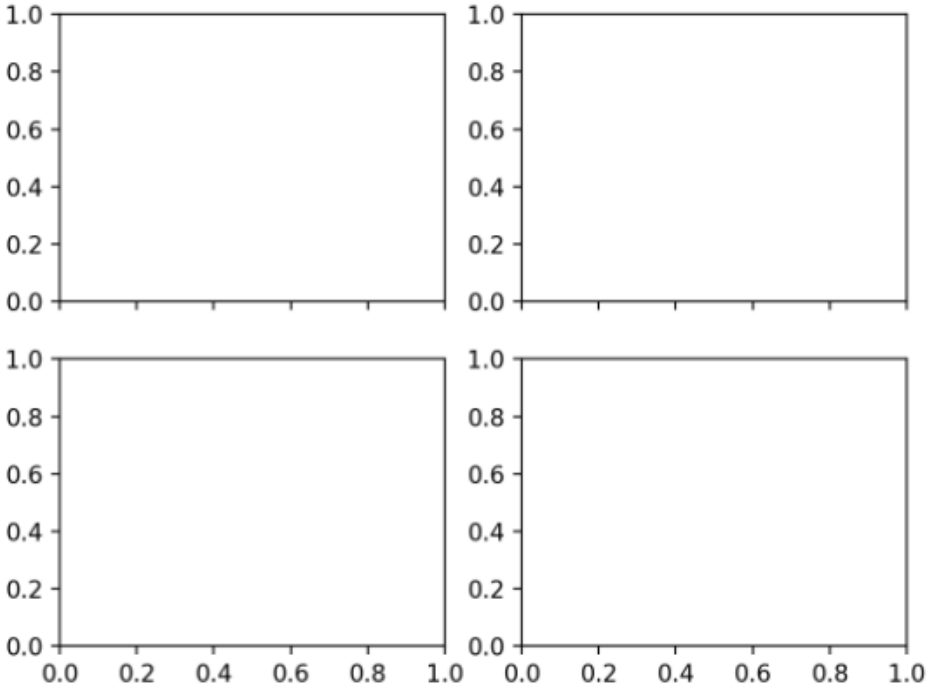
```
# 创建一些测试数据 -- 图1
x = np.linspace(0, 2*np.pi, 400)
y = np.sin(x**2)

# 创建一个画像和子图 -- 图2
fig, ax = plt.subplots()
ax.plot(x, y)
ax.set_title('Simple plot')

# 创建两个子图 -- 图3
f, (ax1, ax2) = plt.subplots(1, 2, sharey=True)
ax1.plot(x, y)
ax1.set_title('Sharing Y axis')
ax2.scatter(x, y)

# 创建四个子图 -- 图4
fig, axs = plt.subplots(2, 2, subplot_kw=dict(projection="polar"))
axs[0, 0].plot(x, y)
axs[1, 1].scatter(x, y)

# 共享 x 轴
plt.subplots(2, 2, sharex='col')

# 共享 y 轴
plt.subplots(2, 2, sharey='row')
```

```
# 共享 x 轴和 y 轴
plt.subplots(2, 2, sharex='all', sharey='all')

# 这个也是共享 x 轴和 y 轴
plt.subplots(2, 2, sharex=True, sharey=True)

# 创建标识为 10 的图，已经存在的则删除
fig, ax = plt.subplots(num=10, clear=True)

plt.show()
```




Sharing Y axis
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
2
4
6


90°
135°
45°
180°
0°
225°
315°
270°
0.2
0.4
0.6
0.8
1.0
−1.0
−0.5
0.0
0.5



散点图

scatter()

`scatter()`方法语法格式如下：

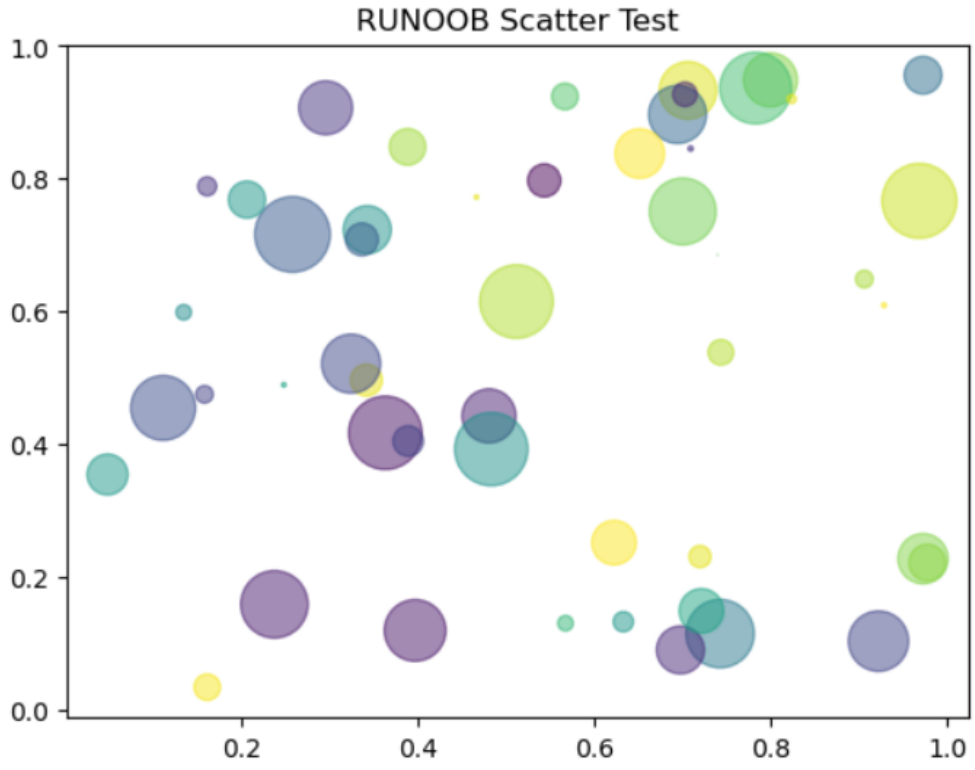
```
matplotlib.pyplot.scatter(x, y, s=None, c=None, marker=None,
cmap=None, norm=None, vmin=None, vmax=None, alpha=None,
linewidths=None, *, edgecolors=None, plotnonfinite=False, data=None,
**kwargs)
```

> **参数说明：**
>
> - **x，y**：长度相同的数组，也就是我们即将绘制散点图的数据点，输入数据。
> - **s**：点的大小，默认`20`，也可以是个数组，数组每个参数为对应点的大小。
> - **c**：点的颜色，默认蓝色`'b'`，也可以是个`RGB`或`RGBA`二维行数组。
> - **marker**：点的样式，默认小圆圈`'o'`。
> - **cmap**：`Colormap`，默认`None`，标量或者是一个`colormap`的名字，只有`c`是一个浮点数数组的时才使用。如果没有申明就是`image.cmap`。
> - **norm**：`Normalize`，默认`None`，数据亮度在`0-1`之间，只有`c`是一个浮点数的数组的时才使用。
> - **vmin，vmax：**：亮度设置，在`norm`参数存在时会忽略。
> - **alpha：**：透明度设置，`0-1`之间，默认`None`，即不透明。
> - **linewidths：**：标记点的长度。

- **edgecolors:** ：颜色或颜色序列，默认为 `'face'`，可选值有 `'face'`，`'none'`，`None`。
- **plotnonfinite:** ：布尔值，设置是否使用非限定的 `c`（`inf`，`-inf`或`nan`）绘制点。
- ****kwargs:** ：其他参数。

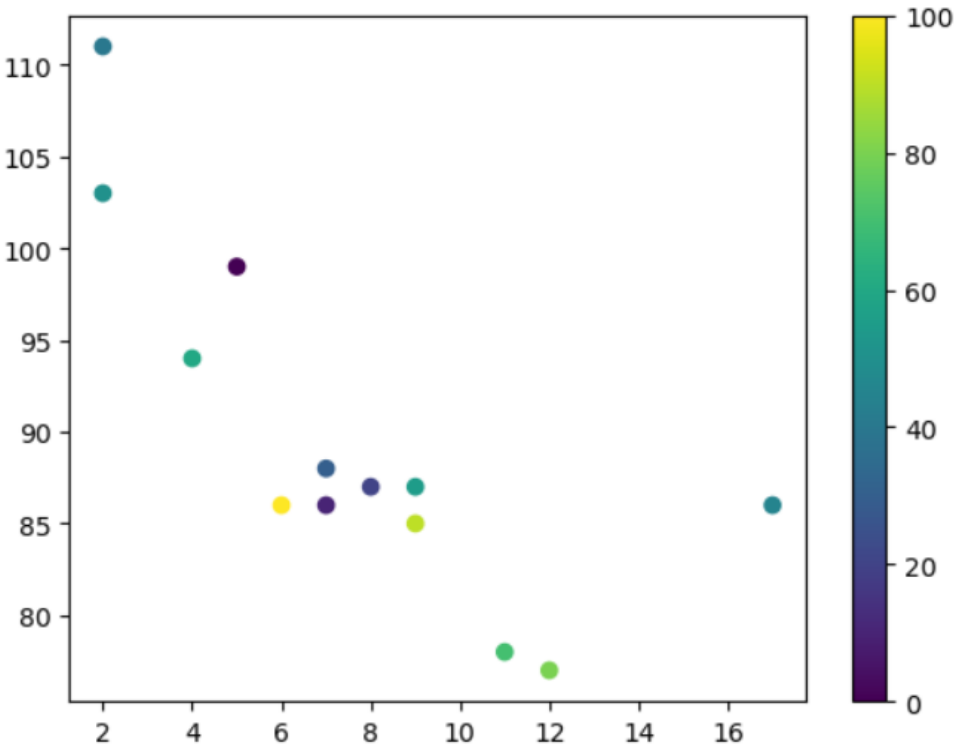
实例：

```
# 随机数生成器的种子
np.random.seed(19680801)


N = 50
x = np.random.rand(N)
y = np.random.rand(N)
colors = np.random.rand(N)
area = (30 * np.random.rand(N))**2  # 0 to 15 point radii

plt.scatter(x, y, s=area, c=colors, alpha=0.5) # 设置颜色及透明度

plt.title("RUNOOB Scatter Test") # 设置标题

plt.show()
```

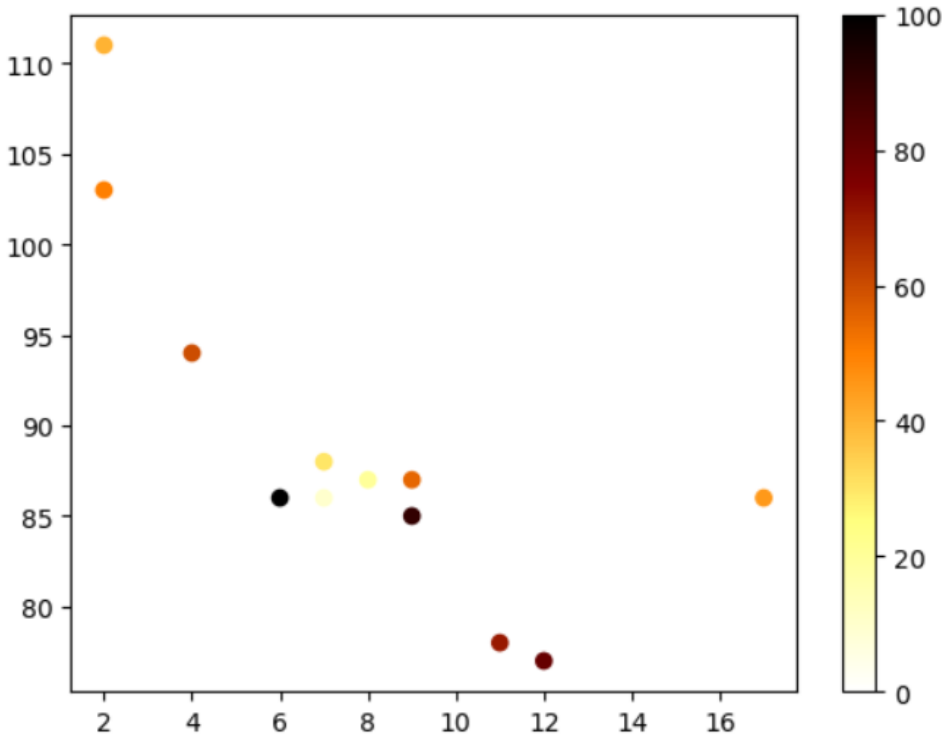


色条参数

```
x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
colors = np.array([0, 10, 20, 30, 40, 45, 50, 55, 60, 70, 80, 90,
100])  # 设定色条中的每个点的取值

plt.scatter(x, y, c=colors, cmap='viridis')

plt.colorbar()

plt.show()
```




```
x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
colors = np.array([0, 10, 20, 30, 40, 45, 50, 55, 60, 70, 80, 90,
100])

plt.scatter(x, y, c=colors, cmap='afmhot_r')
plt.colorbar()
plt.show()
```



柱形图

`bar()`

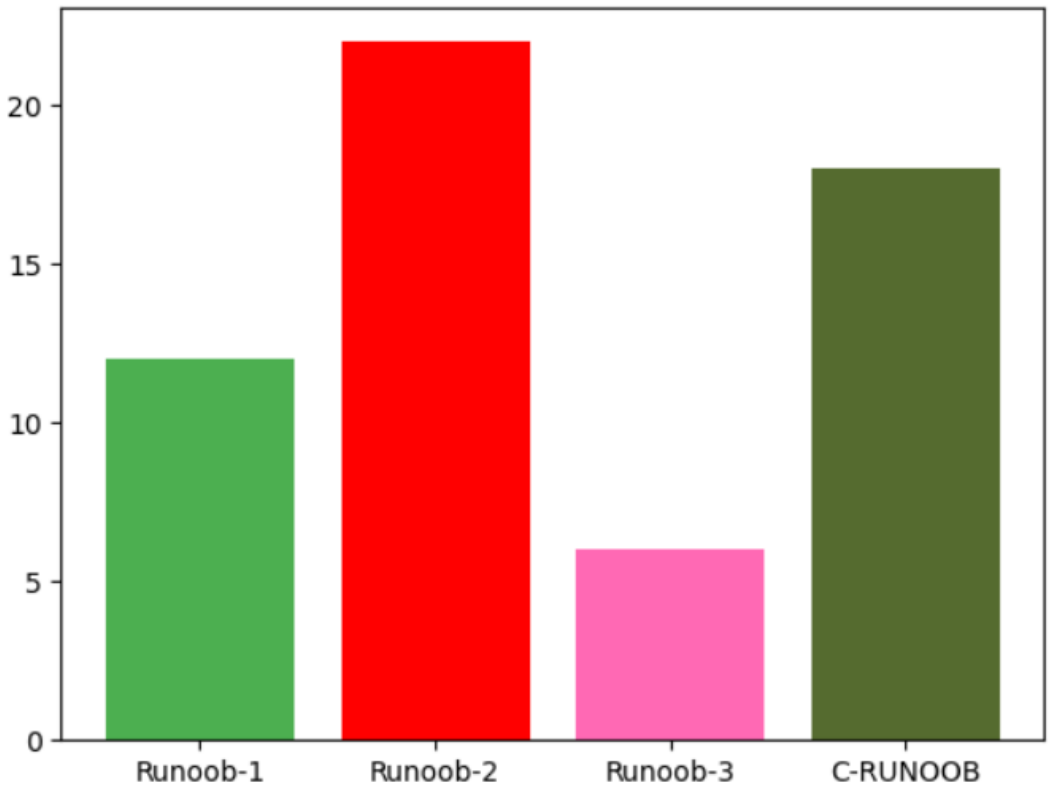
`bar()`方法语法格式如下：

```
matplotlib.pyplot.bar(x, height, width=0.8, bottom=None, *,
align='center', data=None, **kwargs)
```

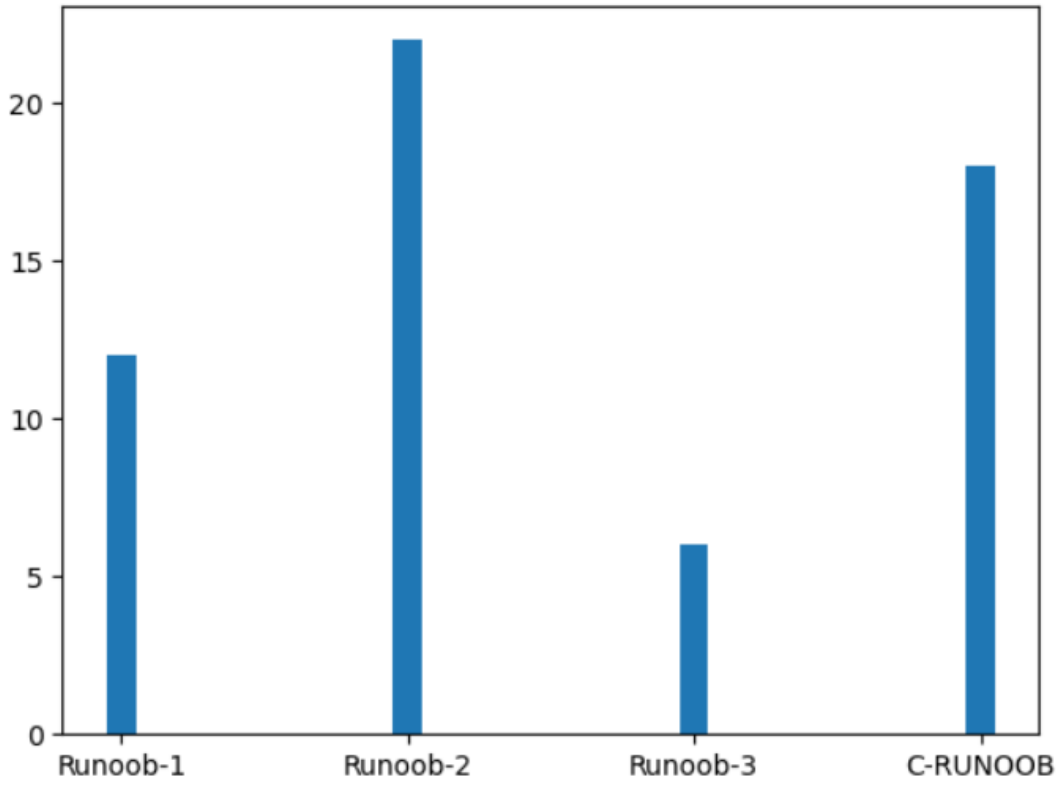
> **参数说明：**
>
> - **x**：浮点型数组，柱形图的 `x` 轴数据。
> - **height**：浮点型数组，柱形图的高度。
> - **width**：浮点型数组，柱形图的宽度。
> - **bottom**：浮点型数组，底座的 `y` 坐标，默认 `0`。
> - **align**：柱形图与 `x` 坐标的对齐方式，`'center'` 以 `x` 位置为中心，这是默认值。`'edge'`：将柱形图的左边缘与 `x` 位置对齐。要对齐右边缘的条形，可以传递负数的宽度值及 `align='edge'`。
> - ****kwargs**：：其他参数。

例子

```
x = np.array(["Runoob-1", "Runoob-2", "Runoob-3", "C-RUNOOB"])
y = np.array([12, 22, 6, 18])

plt.bar(x, y,  color = ["#4CAF50","red","hotpink","#556B2F"])
plt.show()
```




```
x = np.array(["Runoob-1", "Runoob-2", "Runoob-3", "C-RUNOOB"])
y = np.array([12, 22, 6, 18])

plt.bar(x, y, width = 0.1)
plt.show()
```



饼图

pie()

`pie()` 方法语法格式如下：

```
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)[source]
```

> **参数说明：**
>
> - **x**：浮点型数组或列表，用于绘制饼图的数据，表示每个扇形的面积。
> - **explode**：数组，表示各个扇形之间的间隔，默认值为`0`。
> - **labels**：列表，各个扇形的标签，默认值为 `None`。
> - **colors**：数组，表示各个扇形的颜色，默认值为 `None`。
> - **autopct**：设置饼图内各个扇形百分比显示格式，**%d%%** 整数百分比，**%0.1f** 一位小数， **%0.1f%%** 一位小数百分比， **%0.2f%%** 两位小数百分比。
> - **labeldistance**：标签标记的绘制位置，相对于半径的比例，默认值为 `1.1`，如**<1**则绘制在饼图内侧。
> - **pctdistance**：：类似于 `labeldistance`，指定 `autopct` 的位置刻度，默认值为 `0.6`。

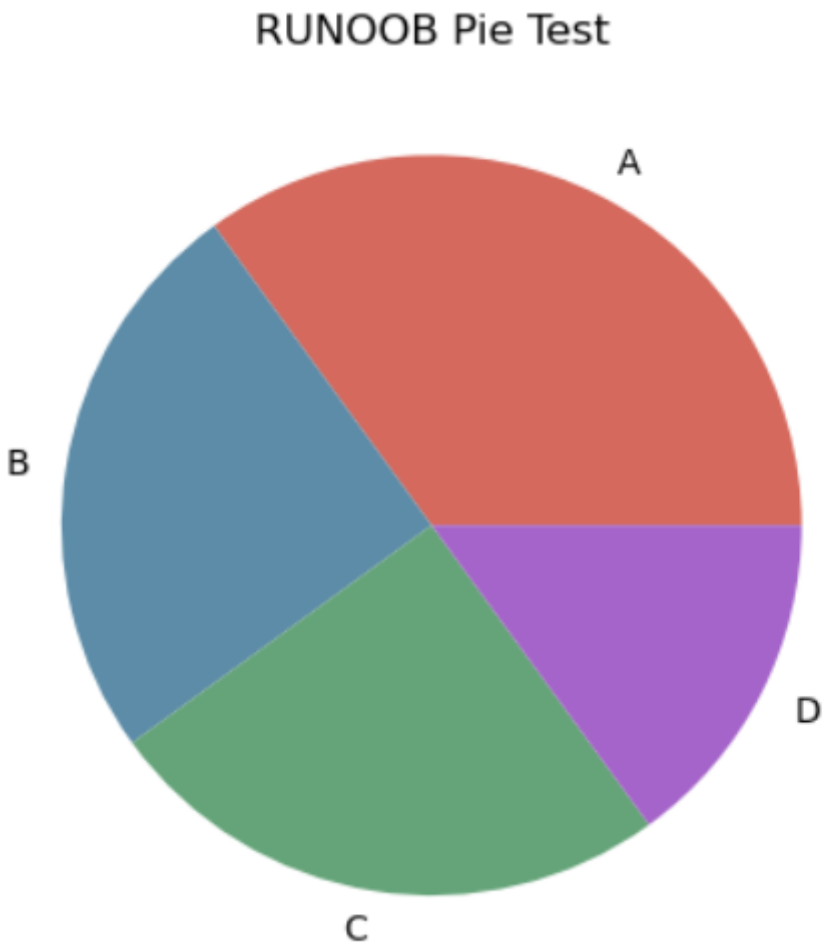
- **shadow:** ：布尔值 `True` 或 `False`，设置饼图的阴影，默认为 `False`，不设置阴影。
- **radius:** ：设置饼图的半径，默认为 `1`。
- **startangle:** ：用于指定饼图的起始角度，默认为从`x` 轴正方向逆时针画起，如设定 `=90` 则从 `y`轴正方向画起。
- **counterclock**：布尔值，用于指定是否逆时针绘制扇形，默认为 `True`，即逆时针绘制，`False` 为顺时针。
- **wedgeprops**：字典类型，默认值 `None`。用于指定扇形的属性，比如边框线颜色、边框线宽度等。例如：`wedgeprops={'linewidth':5}` 设置`wedge` 线宽为 `5`。
- **textprops**：字典类型，用于指定文本标签的属性，比如字体大小、字体颜色等，默认值为 `None`。
- **center**：浮点类型的列表，用于指定饼图的中心位置，默认值：`(0,0)`。
- **frame**：布尔类型，用于指定是否绘制饼图的边框，默认值：`False`。如果是 `True`，绘制带有表的轴框架。
- **rotatelabels**：布尔类型，用于指定是否旋转文本标签，默认为`False`。如果为 `True`，旋转每个 `label`到指定的角度。
- **data**：用于指定数据。如果设置了 `data` 参数，则可以直接使用数据框中的列作为 `x`、`labels` 等参数的值，无需再次传递。

除此之外，`pie()` 函数还可以返回三个参数：

- **wedges**：一个包含扇形对象的列表。
- **texts**：一个包含文本标签对象的列表。
- **autotexts**：一个包含自动生成的文本标签对象的列表。

实例

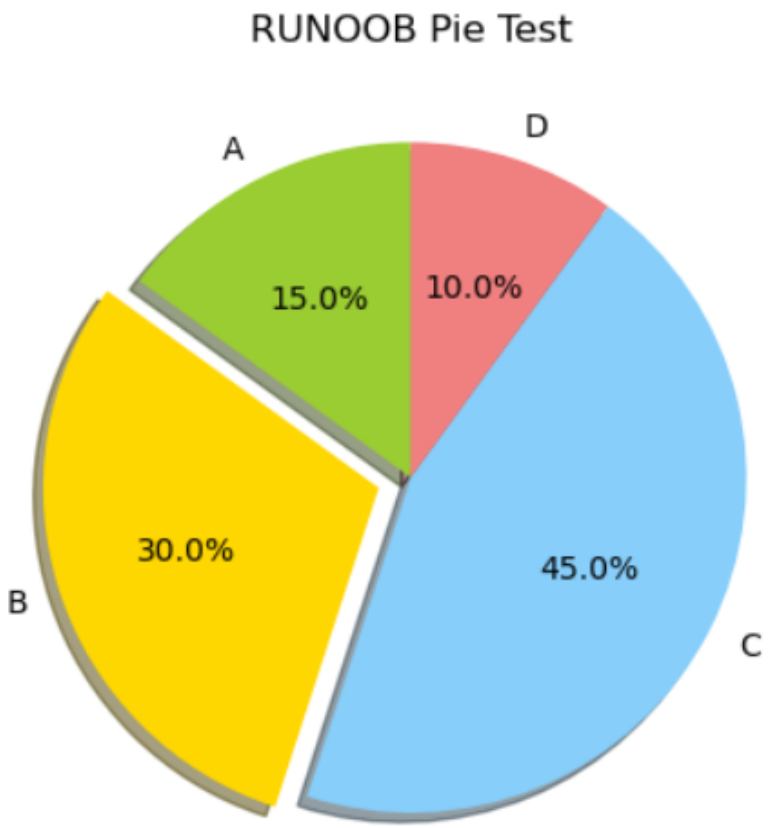
```
y = np.array([35, 25, 25, 15])

plt.pie(y,
        labels=['A','B','C','D'], # 设置饼图标签
        colors=["#d5695d", "#5d8ca8", "#65a479", "#a564c9"], # 设置饼图颜色
       )
plt.title("RUNOOB Pie Test") # 设置标题
plt.show()
```



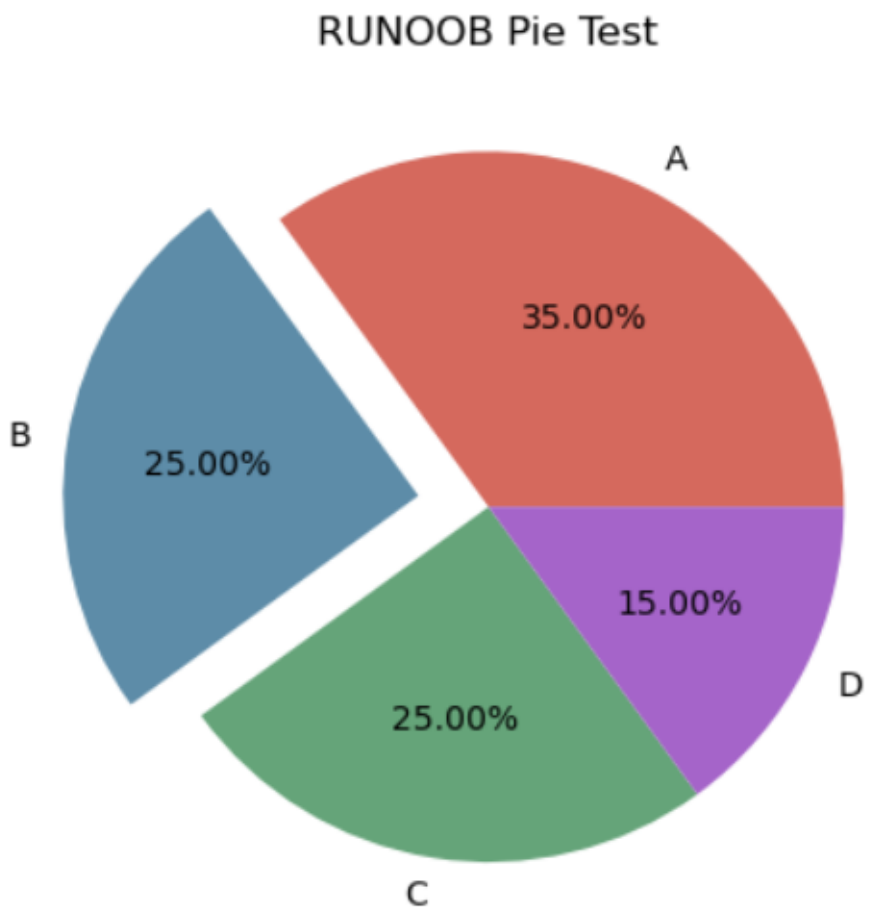
突出显示某个扇形

```
# 数据
sizes = [15, 30, 45, 10]

# 饼图的标签
labels = ['A', 'B', 'C', 'D']

# 饼图的颜色
colors = ['yellowgreen', 'gold', 'lightskyblue', 'lightcoral']

# 突出显示第二个扇形
explode = (0, 0.1, 0, 0)

# 绘制饼图
plt.pie(sizes, explode=explode, labels=labels, colors=colors,
        autopct='%1.1f%%', shadow=True, startangle=90)

# 标题
plt.title("RUNOOB Pie Test")

# 显示图形
```

```
plt.show()
```




```
y = np.array([35, 25, 25, 15])

plt.pie(y,
        labels=['A','B','C','D'], # 设置饼图标签
        colors=["#d5695d", "#5d8ca8", "#65a479", "#a564c9"], # 设置饼图
颜色
        explode=(0, 0.2, 0, 0), # 第二部分突出显示，值越大，距离中心越远
        autopct='%.2f%%', # 格式化输出百分比
       )
plt.title("RUNOOB Pie Test")
plt.show()
```



直方图

hist()

`hist()` 方法语法格式如下：

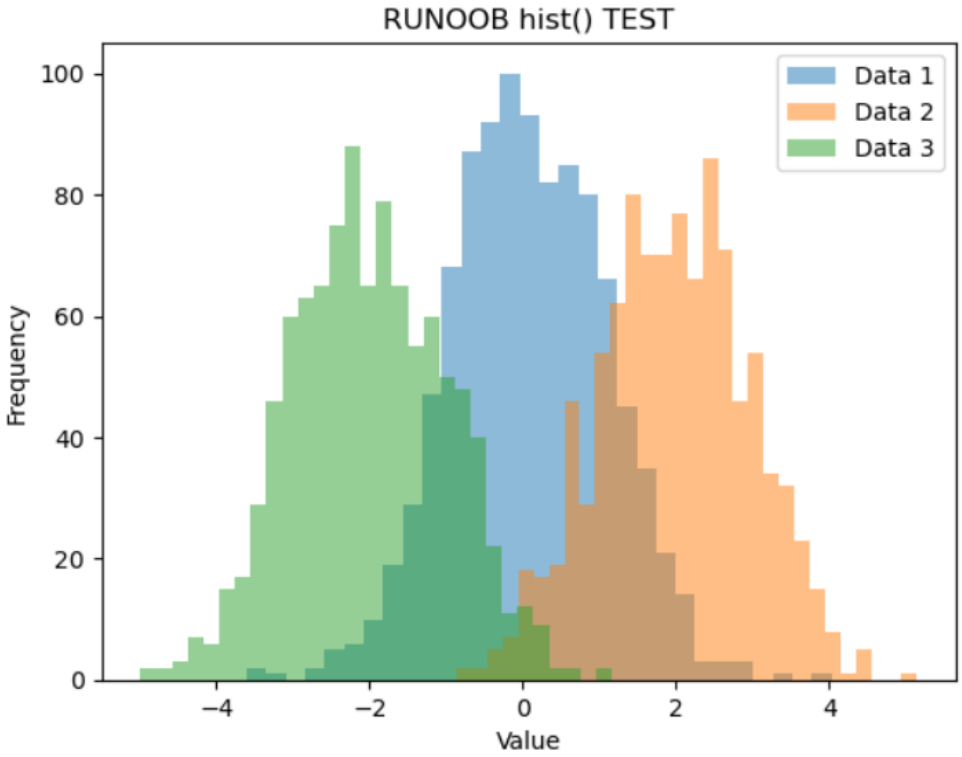
```
matplotlib.pyplot.hist(x, bins=None, range=None, density=False,
weights=None, cumulative=False, bottom=None, histtype='bar',
align='mid', orientation='vertical', rwidth=None, log=False,
color=None, label=None, stacked=False, **kwargs)
```

> **参数说明：**
>
> - **x**：表示要绘制直方图的数据，可以是一个一维数组或列表。
> - **bins**：可选参数，表示直方图的箱数。默认为`10`。
> - **range**：可选参数，表示直方图的值域范围，可以是一个二元组或列表。默认为`None`，即使用数据中的最小值和最大值。
> - **density**：可选参数，表示是否将直方图归一化。默认为`False`，即直方图的高度为每个箱子内的样本数，而不是频率或概率密度。
> - **weights**：可选参数，表示每个数据点的权重。默认为`None`。
> - **cumulative**：可选参数，表示是否绘制累积分布图。默认为`False`。

- **bottom**：可选参数，表示直方图的起始高度。默认为`None`。
- **histtype**：可选参数，表示直方图的类型，可以是`'bar'`、`'barstacked'`、`'step'`、`'stepfilled'`等。默认为`'bar'`。
- **align**：可选参数，表示直方图箱子的对齐方式，可以是`'left'`、`'mid'`、`'right'`。默认为`'mid'`。
- **orientation**：可选参数，表示直方图的方向，可以是`'vertical'`、`'horizontal'`。默认为`'vertical'`。
- **rwidth**：可选参数，表示每个箱子的宽度。默认为`None`。
- **log**：可选参数，表示是否在`y`轴上使用对数刻度。默认为`False`。
- **color**：可选参数，表示直方图的颜色。
- **label**：可选参数，表示直方图的标签。
- **stacked**：可选参数，表示是否堆叠不同的直方图。默认为`False`。
- ****kwargs**：可选参数，表示其他绘图参数。

实例

```
# 生成三组随机数据
data1 = np.random.normal(0, 1, 1000)
data2 = np.random.normal(2, 1, 1000)
data3 = np.random.normal(-2, 1, 1000)

# 绘制直方图
plt.hist(data1, bins=30, alpha=0.5, label='Data 1')
plt.hist(data2, bins=30, alpha=0.5, label='Data 2')
plt.hist(data3, bins=30, alpha=0.5, label='Data 3')

# 设置图表属性
plt.title('RUNOOB hist() TEST')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.legend()

# 显示图表
plt.show()
```



显示图像

`imshow()`

`imshow()` 方法语法格式如下：

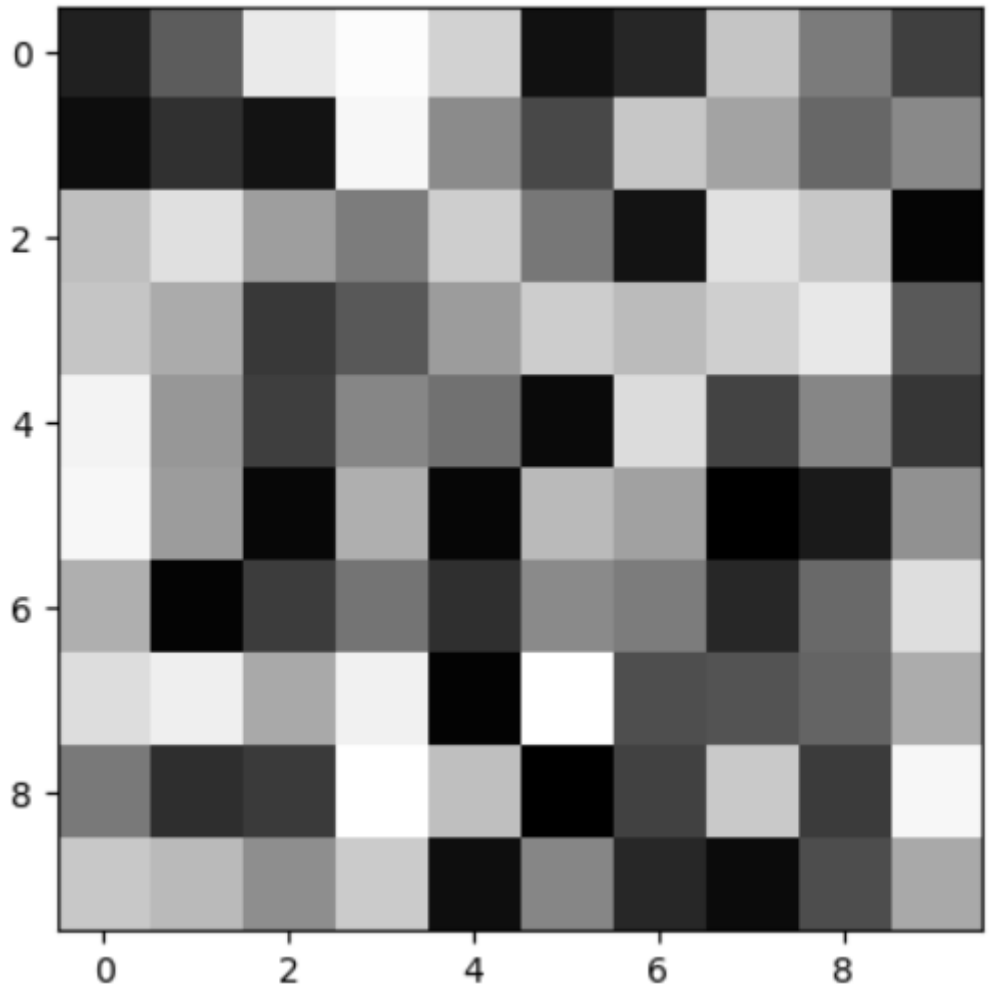
```
imshow(X, cmap=None, norm=None, aspect=None, interpolation=None,
alpha=None, vmin=None, vmax=None, origin=None, extent=None,
shape=None, filternorm=1, filterrad=4.0, imlim=None, resample=None,
url=None, *, data=None, **kwargs)
```

> **参数说明：**
>
> - **X**：输入数据。可以是二维数组、三维数组、PIL图像对象、`matplotlib`路径对象等。
> - **cmap**：颜色映射。用于控制图像中不同数值所对应的颜色。可以选择内置的颜色映射，如`gray`、`hot`、`jet`等，也可以自定义颜色映射。
> - **norm**：用于控制数值的归一化方式。可以选择`Normalize`、`LogNorm`等归一化方法。
> - **aspect**：控制图像纵横比（aspect ratio）。可以设置为`auto`或一个数字。
> - **interpolation**：插值方法。用于控制图像的平滑程度和细节程度。可以选择`nearest`、`bilinear`、`bicubic`等插值方法。
> - **alpha**：图像透明度。取值范围为0~1。
> - **origin**：坐标轴原点的位置。可以设置为`upper`或`lower`。

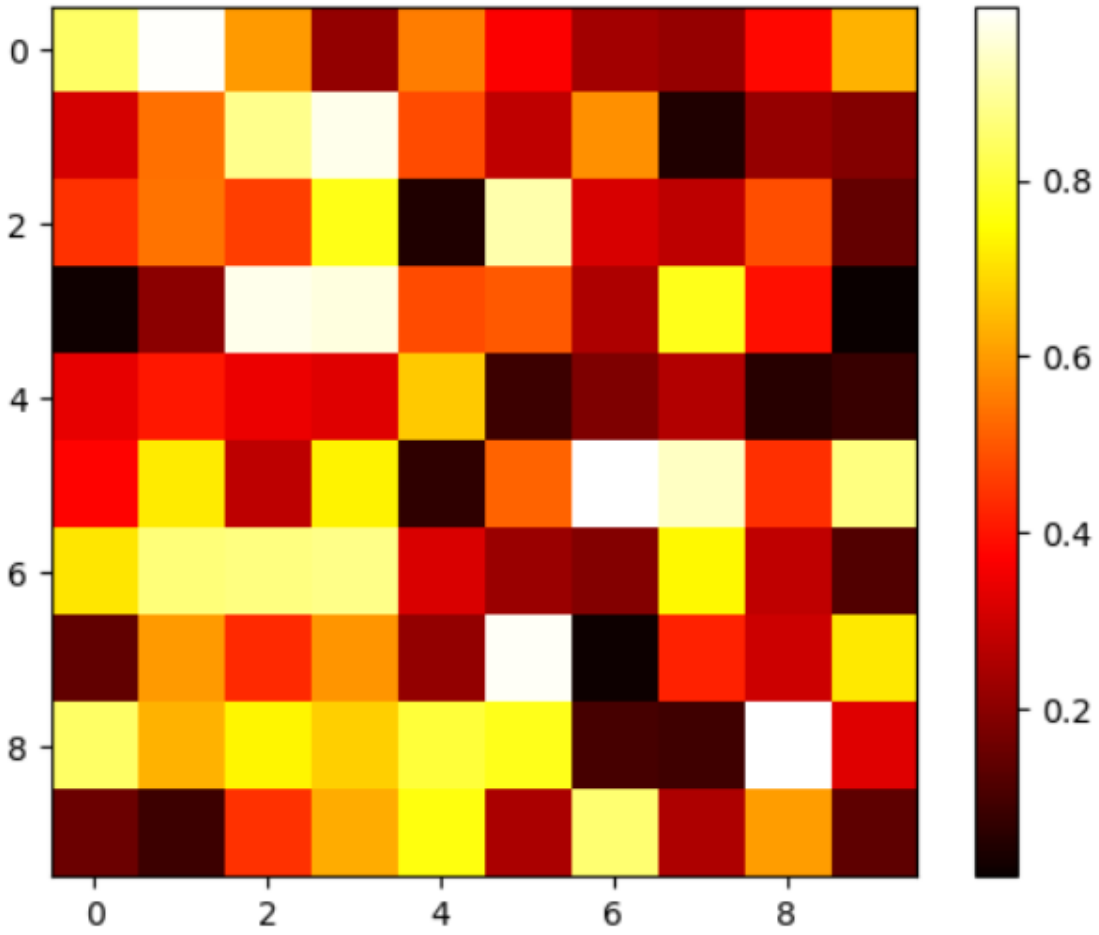
- **extent**：控制显示的数据范围。可以设置为`[xmin, xmax, ymin, ymax]`。
- **vmin**、**vmax**：控制颜色映射的值域范围。
- **filternorm** 和 **filterrad**：用于图像滤波的对象。可以设置为`None`、`antigrain`、`freetype`等。
- **imlim**： 用于指定图像显示范围。
- **resample**：用于指定图像重采样方式。
- **url**：用于指定图像链接。

实例：

```
# 生成一个二维随机数组
img = np.random.rand(10, 10)

# 绘制灰度图像
plt.imshow(img, cmap='gray')

# 显示图像
plt.show()
```



```
# 生成一个二维随机数组
data = np.random.rand(10, 10)

# 绘制热力图
plt.imshow(data, cmap='hot')

# 显示图像
plt.colorbar()
plt.show()
```



保存图像

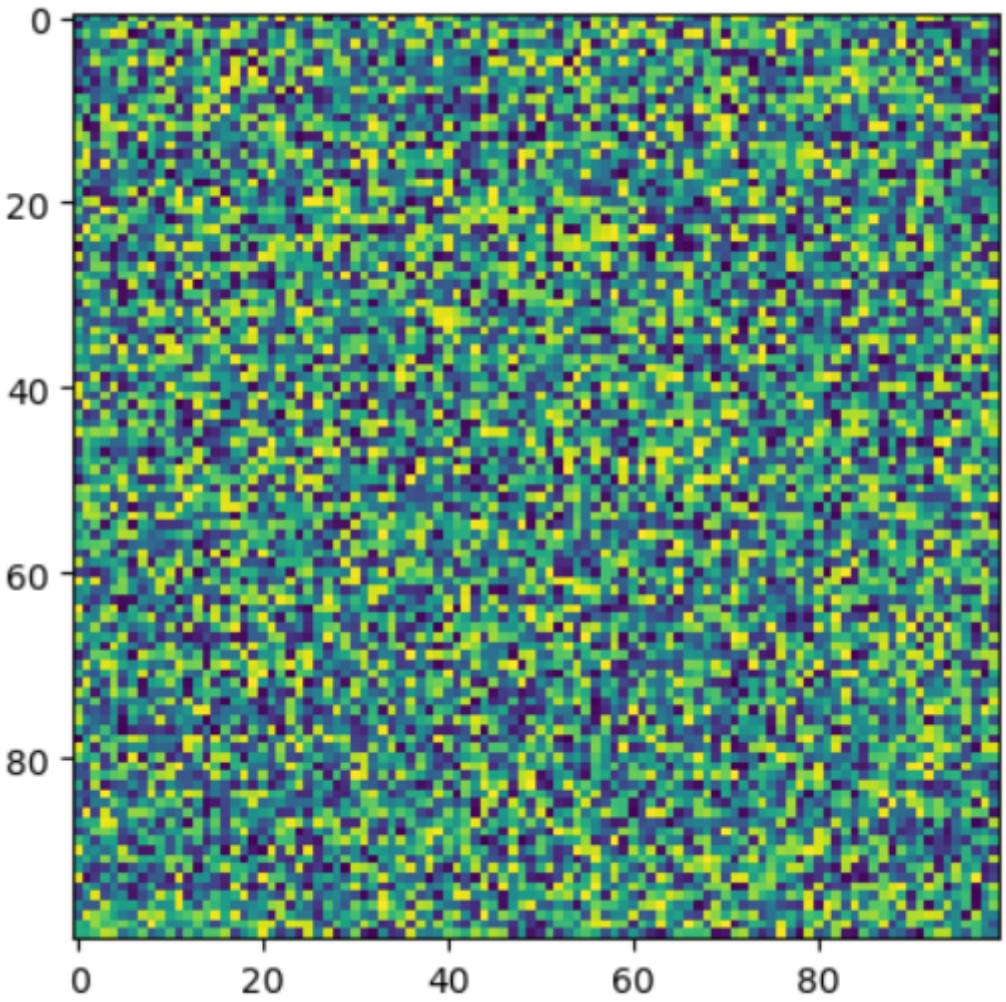
imsave()

`imsave()`方法的语法如下：

```
matplotlib.pyplot.imsave(fname, arr, **kwargs)
```

> **参数说明：**
>
> - **fname**：保存图像的文件名，可以是相对路径或绝对路径。
> - **arr**：表示图像的`NumPy`数组。
> - **kwargs**：可选参数，用于指定保存的图像格式以及图像质量等参数。

```
# 创建一个二维的图像数据
img_data = np.random.random((100, 100))

# 显示图像
plt.imshow(img_data)

# 保存图像到磁盘上
plt.imsave('./access/runoob-test.png', img_data)
```



读取图像

`imread()`

`imread()` 方法的语法如下：

```
matplotlib.pyplot.imread(fname, format=None)
```

> **参数说明：**
>
> - **fname**：指定了要读取的图像文件的文件名或文件路径，可以是相对路径或绝对路径。

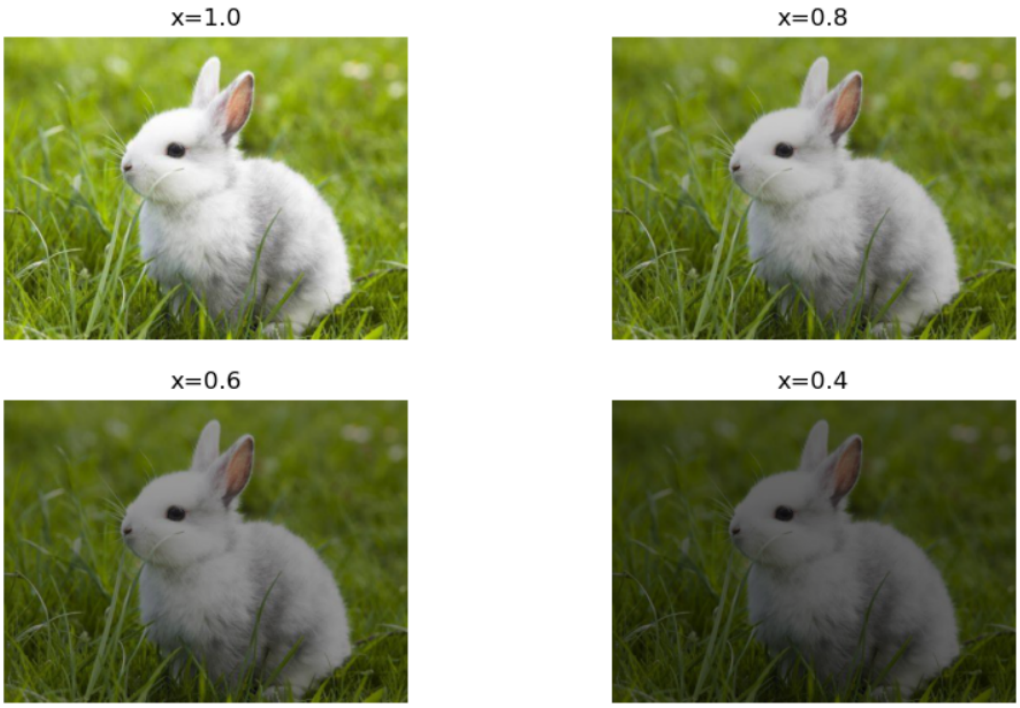
> - **format**： 参数指定了图像文件的格式，如果不指定，则默认根据文件后缀名来自动识别格式。

实例

图片链接：兔子

将图片乘一个$0 <= x <= 1$的值，会将图片变暗。

```
img_array = plt.imread('./access/rabbit.jpg')
rabbit = img_array/255
#print(rabbit)

# 显示图像
plt.figure(figsize=(10,6))

for i in range(1,5):
    plt.subplot(2,2,i)
    x = 1 - 0.2*(i-1)
    plt.axis('off') #hide coordinate axes
    plt.title('x={:.1f}'.format(x))
    plt.imshow(rabbit*x)

plt.show()
```



裁剪图像

```
img_array = plt.imread('./access/rabbit.jpg')
rabbit = img_array/255
#print(rabbit)

# 显示图像
plt.figure(figsize=(6,6))
plt.imshow(rabbit[:300,100:400,:])  # 裁剪
plt.axis('off')
plt.show()
```

RGB值的修改

```
img_array = plt.imread('./access/rabbit.jpg')
rabbit = img_array/255
#print(rabbit)

# 显示图像
red_rabbit = rabbit.copy()

red_rabbit[:, :,[1,2]] = 1

plt.figure(figsize=(10,10))
plt.imshow(red_rabbit)
plt.axis('off')
plt.show()
```