Matplotlib

1 Matplotlib简介

Matplotlib 是用于数据可视化的 Python 包，它是一个跨平台库，用于根据数组中的数据制作 2D 图。

Matplotlib 是用 Python 编写的，并使用了 NumPy。

Matplotlib 提供了一个面向对象的 API，有助于使用 Python GUI 工具包（如 PyQt）在应用程序中嵌入绘图。Matplotlib 也可以用于 Python、IPython shell、Jupyter Notebook 和 Web 应用程序服务器。

Matplotlib + NumPy 可以视作 MATLAB 的开源等价物。

2 PyLab模块

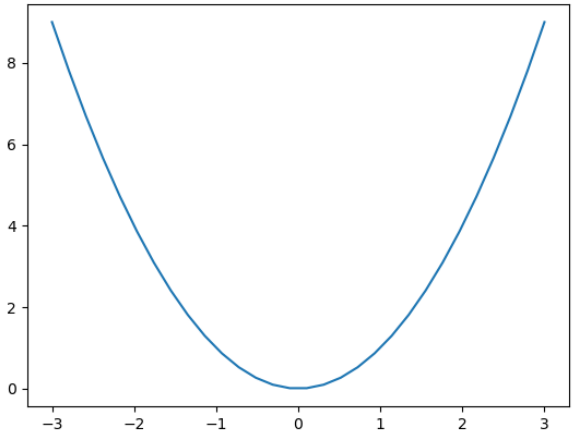
PyLab 是一个面向 Matplotlib 的绘图库接口，其语法和 MATLAB 十分接近。它和`matplotlib.pyplot`模块都能实现 Matplotlib 的绘图功能。

PyLab 是一个单独的模块，随 Matplotlib 软件包一起安装。

不建议使用 PyLab 模块。

基本绘图：提供两个长度相同的数组或序列，用`pylab.plot()`函数绘制曲线，然后用`pylab.show()`函数显示图像。例如：

```
import numpy as np
import pylab as plb

x = np.linspace(-3, 3, 30)
y = x ** 2

plb.plot(x, y)
plb.show()
```



`pylab.plot()`函数的第三个参数是一个字符串，用于设置曲线的样式。可选的设置有：

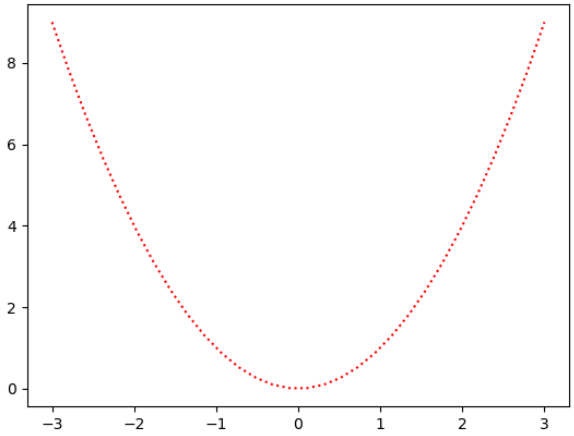
颜色	描述
`b`	蓝色
`g`	绿色
`r`	红色
`c`	青色
`m`	品红
`y`	黄色
`k`	黑色
`w`	白色

点标记	描述
`.`	点状
`,`	像素
`o`	圆形
`v`	朝下的三角形
`^`	朝上的三角形
`<`	朝左的三角形
`>`	朝右的三角形
`s`	正方形

点标记	描述
`p`	五角星
`*`	星形
`h`	1 号六角形
`H`	2 号六角形
`+`	加号
`D`	钻石形
`d`	小版钻石形
`|`	竖直线形
`_`	水平线形
`1`	下箭头
`2`	上箭头
`3`	左箭头
`4`	右箭头
`x`	X 形

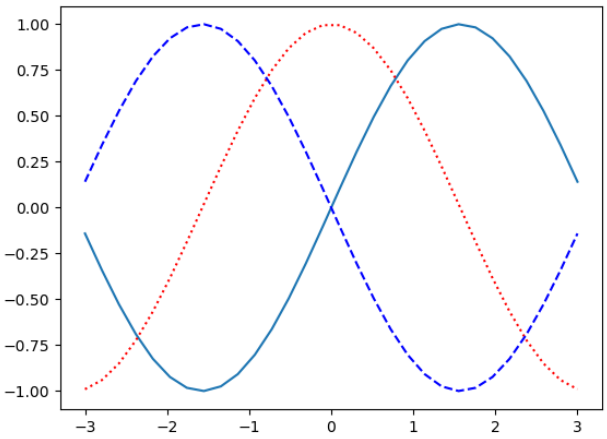
线条样式	描述
`-`	实线
`--`	虚线
`-.`	点划线
`:`	点虚线

```
import numpy as np
import pylab as plb

x = np.linspace(-3, 3, 30)
y = x ** 2

plb.plot(x, y, "r:")  # 红色，点虚线
plb.show()
```



如果需要在同一绘图区域内绘制多个图形，只需要使用多个绘图命令。例如：

```
import numpy as np
import pylab as plb

x = np.linspace(-3, 3, 30)

plb.plot(x, np.sin(x))
plb.plot(x, np.cos(x), 'r:')
plb.plot(x, -np.sin(x), 'b--')
plb.show()
```



`pylab.clf()`函数用于清空图像。

3 `matplotlib.pyplot`模块

3.1 `matplotlib.pyplot` API

`matplotlib.pyplot`是 Matplotlib 中的一个模块，是命令样式函数的集合，使 Matplotlib 像 MATLAB 一样工作。

绘制函数：

函数	描述
`bar`	绘制条形图（柱状图）
`barh`	绘制水平条形图
`boxplot`	绘制箱型图
`hist`	绘制直方图
`hist2d`	绘制 2D 直方图
`pie`	绘制饼图
`plot`	绘制平面曲线和/或标记
`polar`	绘制极坐标图
`scatter`	绘制 x 和 y 的散点图
`stackplot`	绘制堆叠图
`stem`	绘制杆图
`step`	绘制阶梯图
`quiver`	绘制二维矢量场

图像函数：

函数	描述
`imread`	将文件中的图像读入数组
`imsave`	将数组保存为图像文件
`imshow`	将数据显示为图像

轴函数：

函数	描述
`axes`	向图像添加轴
`text`	向轴添加文本
`title`	设置当前图像的标题

函数	描述
`xlabel`	设置 x 轴的标签
`xlim`	获取或设置当前轴的 x 限制
`xscale`	设置 x 轴的缩放比例
`xticks`	获取或设置 x 轴的刻度位置和标签
`ylabel`	设置 y 轴的标签
`ylim`	获取或设置当前轴的 y 限制
`yscale`	设置 y 轴的缩放比例
`yticks`	获取或设置 y 轴的刻度位置和标签

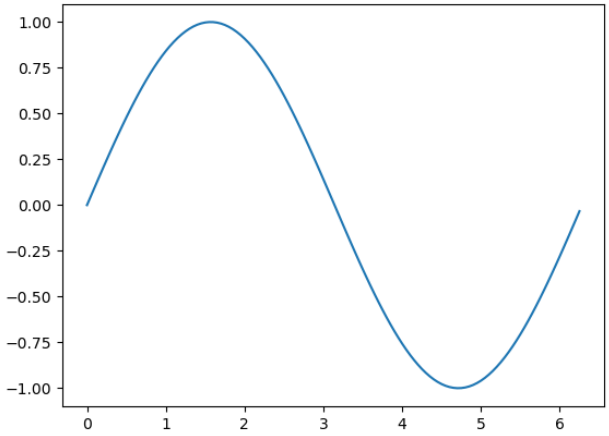
图形函数：

函数	描述
`figtext`	将文字添加到图形
`figure`	创建一个新的图形
`show`	显示所有打开的图形
`savefig`	保存当前图形
`close`	关闭一个图窗口

3.2 简单绘图

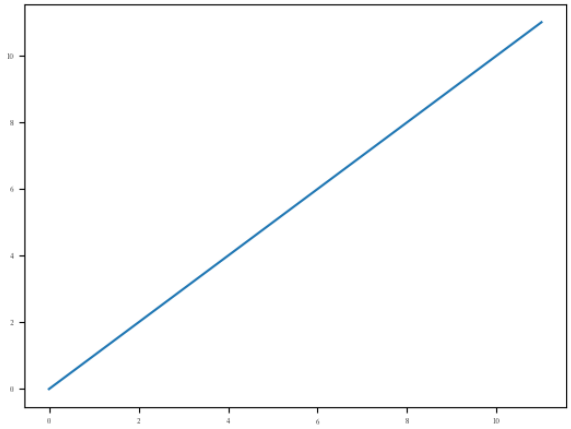
与 PyLab 模块类似，`matplotlib.pyplot` 模块中也有 `plot()` 和 `show()` 函数。例如：

```
import numpy as np
import matplotlib.pyplot as plt
import math

x = np.arange(0, math.pi * 2, 0.05)
y = np.sin(x)

plt.plot(x, y)
plt.show()
```



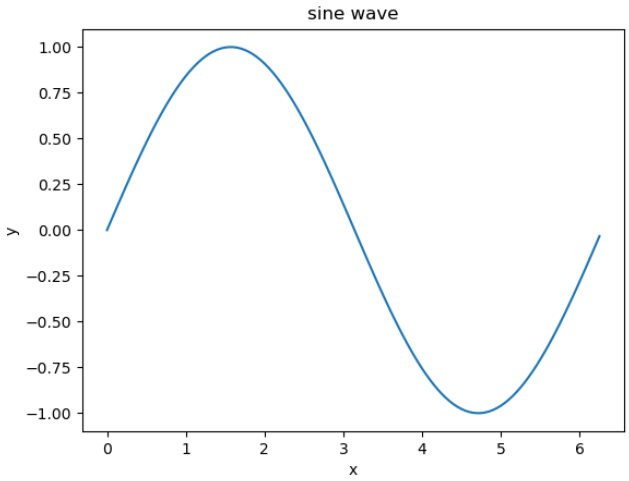
`x`参数是可选的，`y`参数是必需的。当省略`x`时，默认值为从 0 开始的整数索引。例如：

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

plt.plot(range(12))
plt.show()
```



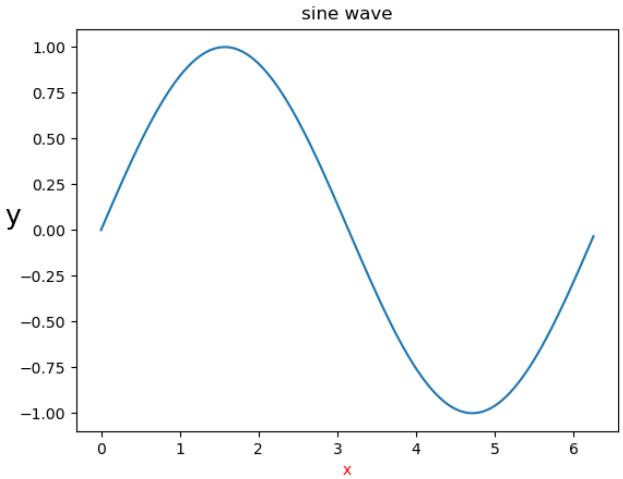
`title()`函数设置图像的标题，`xlabel()`函数设置 x 轴的标签，`ylabel()`函数设置 y 轴的标签。例如：

```
import numpy as np
import matplotlib.pyplot as plt
import math

x = np.arange(0, math.pi * 2, 0.05)
y = np.sin(x)

plt.plot(x, y)
plt.title('sine wave')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```



`xlabel()`函数和`ylabel()`函数可以通过`color`参数设置标签颜色，用`fontsize`参数设置字体大小，用`rotation`设置角度。例如：

```
import numpy as np
import matplotlib.pyplot as plt
import math

x = np.arange(0, math.pi * 2, 0.05)
y = np.sin(x)

plt.plot(x, y)
plt.title('sine wave')
plt.xlabel('x', color='r')
plt.ylabel('y', fontsize=18, rotation=0)
plt.show()
```



在图像中使用中文时，默认情况下无法正常显示，需要配置中文字体。步骤为：

1. 下载 ttf 字体文件（如 Micr.ttf）。
2. 调用`matplotlib.matplotlib_fname()`函数，获得配置文件的路径，例如：D:\Anaconda\Anaconda\envs\exercise\lib\site-packages\matplotlib\mpl-data\matplotlibrc

3. matplotlibrc 是配置文件的名字，进入 mpl-data 文件夹，可以看到 fonts 文件夹。进入 mpl-data\fonts\ttf 文件夹，将下载的 ttf 文件复制到此文件夹下。
4. 用记事本打开 matplotlibrc 配置文件，用 ctrl+F 查找 font.family，找到 `#font.family: sans-serif` 这一行，将前面的 `#` 去掉。
5. 在 matplotlibrc 文件中，用 ctrl+F 查找 font.sans-serif，找到 `#font.sans-serif: DejaVu Sans,...` 这一行，将之前下载的 ttf 文件的名字（Micr）添加进来。
6. 在 matplotlibrc 文件中，用 ctrl+F 查找 axes.unicode_minus，找到 `#axes.unicode_minus: True` ，删除前面的 `#` ，将 `True` 改为 `False` 。保存 matplotlibrc 文件。
7. 删除 Matplotlib 缓冲。
   - Windows：删除 C:\Users\用户名.matplotlib 文件夹
   - Mac 与 Linux：执行命令 `rm -rf ~/.cache/matplotlib`
8. 在 Python 代码中添加设置代码： `plt.rcParams['font.sans-serif'] = ['Micr']`

经过如上配置后，就可以在图像中正常显示中文。

4 使用面向对象思想画图

虽然使用 `matplotlib.pyplot` 模块很容易快速生成绘图，但建议使用面向对象的方法，因为它可以更好地控制和自定义绘图。 `matplotlib.axes.Axes` 类中提供了大多数函数。

面向对象方法的主要思想是创建图形对象，然后只调用该对象的方法或属性，这种方式有助于更好地处理其上有多个绘图的画布。

4.1 `Figure` 类

`matplotlib.figure.Figure` 类是所有 plot 元素的顶级容器，从 `pyplot` 模块调用 `figure()` 函数来实例化 `Figure` 对象。参数为：

1. `figsize` ： `(width, height)` ，以英寸为单位的元组。
2. `dpi` ：每英寸点数
3. `facecolor` ：图的背景颜色
4. `edgecolor` ：图的边缘颜色
5. `linewidth` ：边线宽度

```
import matplotlib.pyplot as plt

fig = plt.figure()
print(fig)  # Figure(640x480)
```

4.2 `Axes` 类

`Axes` 对象是具有数据空间的图像区域。给定的图形可以包含许多 `Axes` 对象，但给定的 `Axes` 对象只能在一个图中。

`Axes` 类及其成员函数是使用面向对象接口的主要入口点。

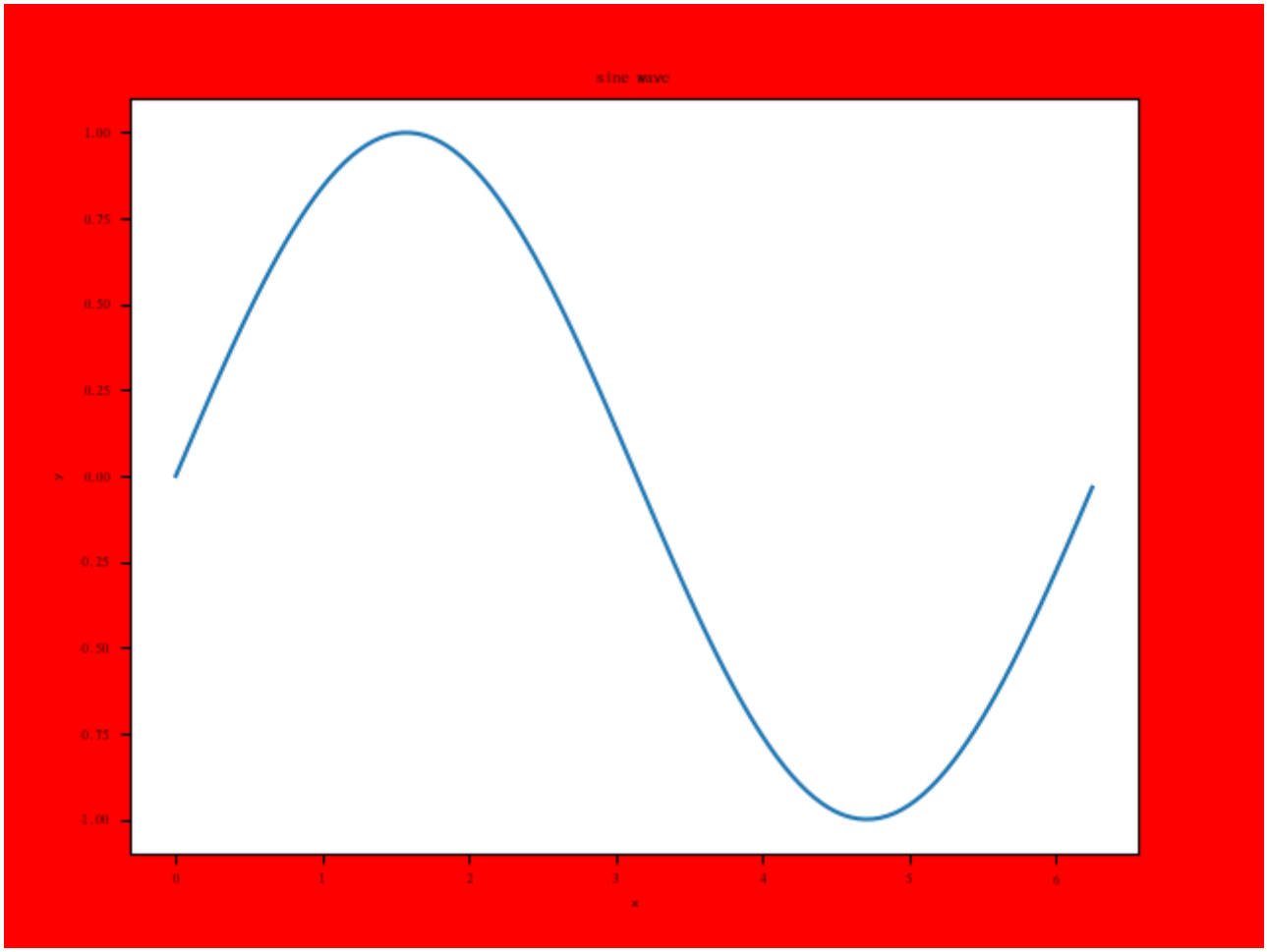
`Figure` 对象通过调用 `add_axes()` 方法将 `Axes` 对象添加到图中，返回 `Axes` 对象。参数为：

1. `rect`：长度为 4 的元组 `(left, bottom, width, height)`，其中 `left` 和 `bottom` 分别指定坐标轴与图像左侧和底部的距离，`width` 和 `height` 分别指定坐标轴的宽度和高度。
2. `projection`：坐标轴的投影类型。可选，默认值为 `None`。
3. `polar`：布尔值，可选，默认为 `False`。取值为 `True` 时，相当于 `projection` 参数取值为 `'polar'`。

添加 `Axes` 对象后，所有的绘图操作都通过 `Axes` 对象来进行。

`Axes.plot()` 方法是 `Axes` 类的基本方法，用于绘制曲线。`Axes.set_title()` 方法用于设置标题，`Axes.set_xlabel()` 和 `Axes.set_ylabel()` 方法分别用于设置 x 轴和 y 轴的标签。例如：

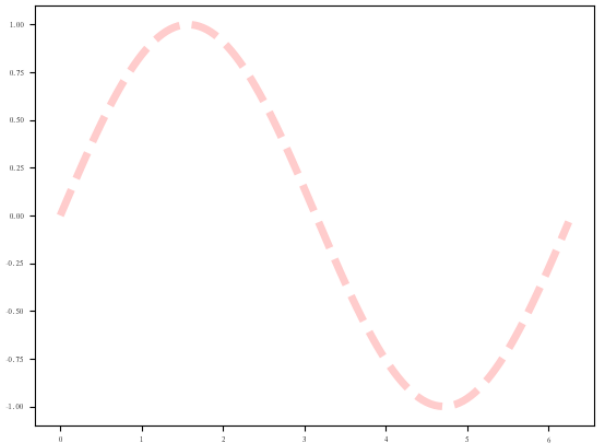
```
import numpy as np
import matplotlib.pyplot as plt
import math

x = np.arange(0, math.pi * 2, 0.05)
y = np.sin(x)

fig = plt.figure(facecolor='r')  # 背景颜色为红色
axes = fig.add_axes([0.1, 0.1, 0.8, 0.8])

axes.plot(x, y)
axes.set_title('sine wave')
axes.set_xlabel('x')
axes.set_ylabel('y')

fig.show()
```



`Axes.plot()`方法的其他参数：

参数	说明
`color`	颜色
`alpha`	透明度
`linestyle`或`ls`	线型
`linewidth`或`lw`	线宽
`marker`	点类型
`markersize`	点大小
`markeredgewidth`	点边缘的宽度
`markeredgecolor`	点边缘的颜色
`markerfacecolor`	点内部的颜色

```
import numpy as np
import matplotlib.pyplot as plt
import math

x = np.arange(0, math.pi * 2, 0.05)
y = np.sin(x)

fig = plt.figure()
axes = fig.add_axes([0.1, 0.1, 0.8, 0.8])

axes.plot(x, y, color='r', alpha=0.2, ls='--', lw=5)

fig.show()
```



`Axes.legend()`方法在坐标轴上添加图例。它有多种原型：

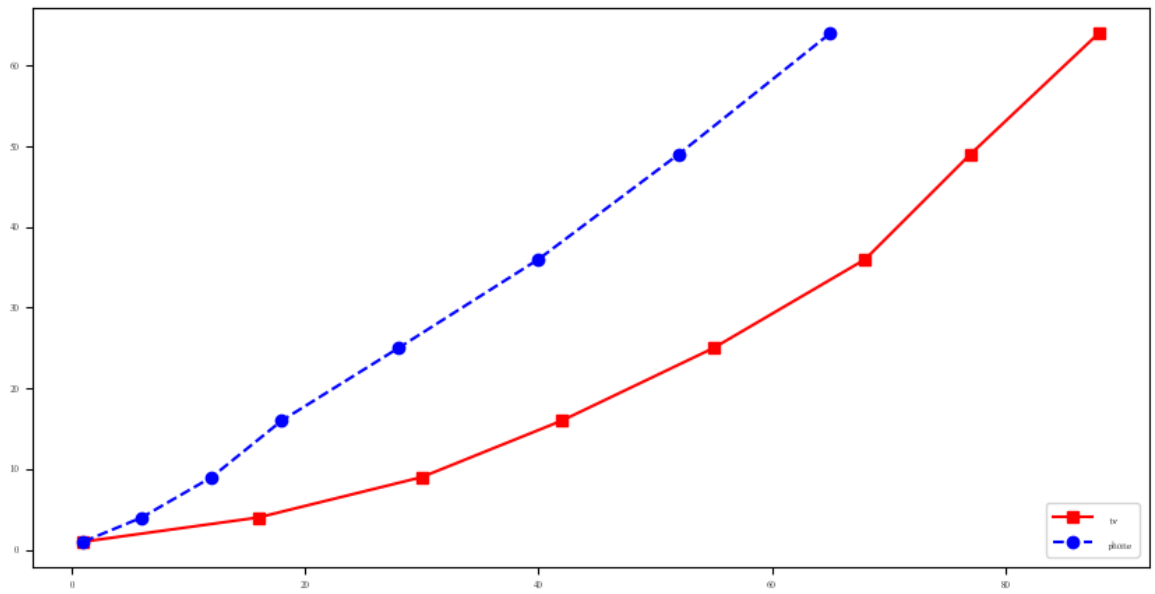
1. `legend()`：自动检测图例中显示的绘图元素。只有设置过标签的绘图元素才会显示。
2. `legend(handles)`：明确列出图例中的绘图元素，每个绘图元素的标签在绘制曲线时指定。
3. `legend(handles, labels)`：明确列出图例中的绘图元素和标签。
4. `legend(labels)`：将所有绘图元素添加到图例中，指定每个绘图元素的标签。

关键字参数`loc`指定图例的位置。取值可以为字符串，或者为字符串对应的数字代码，或者用一个长度为 2 的元组指定图例左下角的坐标。位置字符串及代码如下表所示。

位置字符串	位置代码	说明
`'best'`	0	将图例放置在其他 9 个位置中与绘图元素重叠最小的位置
`'upper right'`	1	右上角
`'upper left'`	2	左上角
`'lower left'`	3	左下角
`'lower right'`	4	右下角

位置字符串	位置代码	说明
`'right'`	5	等价于`'center right'`
`center left`	6	水平居左，垂直居中
`center right`	7	水平居右，垂直居中
`lower center`	8	水平居中，垂直居下
`upper center`	9	水平居中，垂直居上
`center`	10	居中

```
import matplotlib.pyplot as plt

x1 = [1, 16, 30, 42, 55, 68, 77, 88]
x2 = [1, 6, 12, 18, 28, 40, 52, 65]
y = [1, 4, 9, 16, 25, 36, 49, 64]

fig = plt.figure(figsize=(10, 5))
axes = fig.add_axes([0.1, 0.1, 0.8, 0.8])

axes.plot(x1, y, 'rs-')  # 红色，正方形点，实线
axes.plot(x2, y, 'bo--')  # 蓝色，圆形点，虚线
axes.legend(labels=('tv', 'phone'), loc='lower right')  # 图例，位于右下角

fig.show()
```



4.3 `Figure` 与 `Axes` 的关系

类比：在纸上画图，可以选定纸上的多个区域，在不同的区域画不同的图，而这些画图区域必须在纸上才有意义。`Figure` 对象就像一张纸，`Axes` 对象是纸上的画图区域。

`Figure` 对象是一个画布，`Axes` 对象用于在画布上确定画图区域和作图方式。所谓画图，就是在当前的活动 `Figure` 对象中的一个 `Axes` 对象上作图。

一个 `Figure` 对象可以有多个 `Axes` 对象，`Axes` 对象必须在 `Figure` 对象上，要画图必须要有 `Axes` 对象。

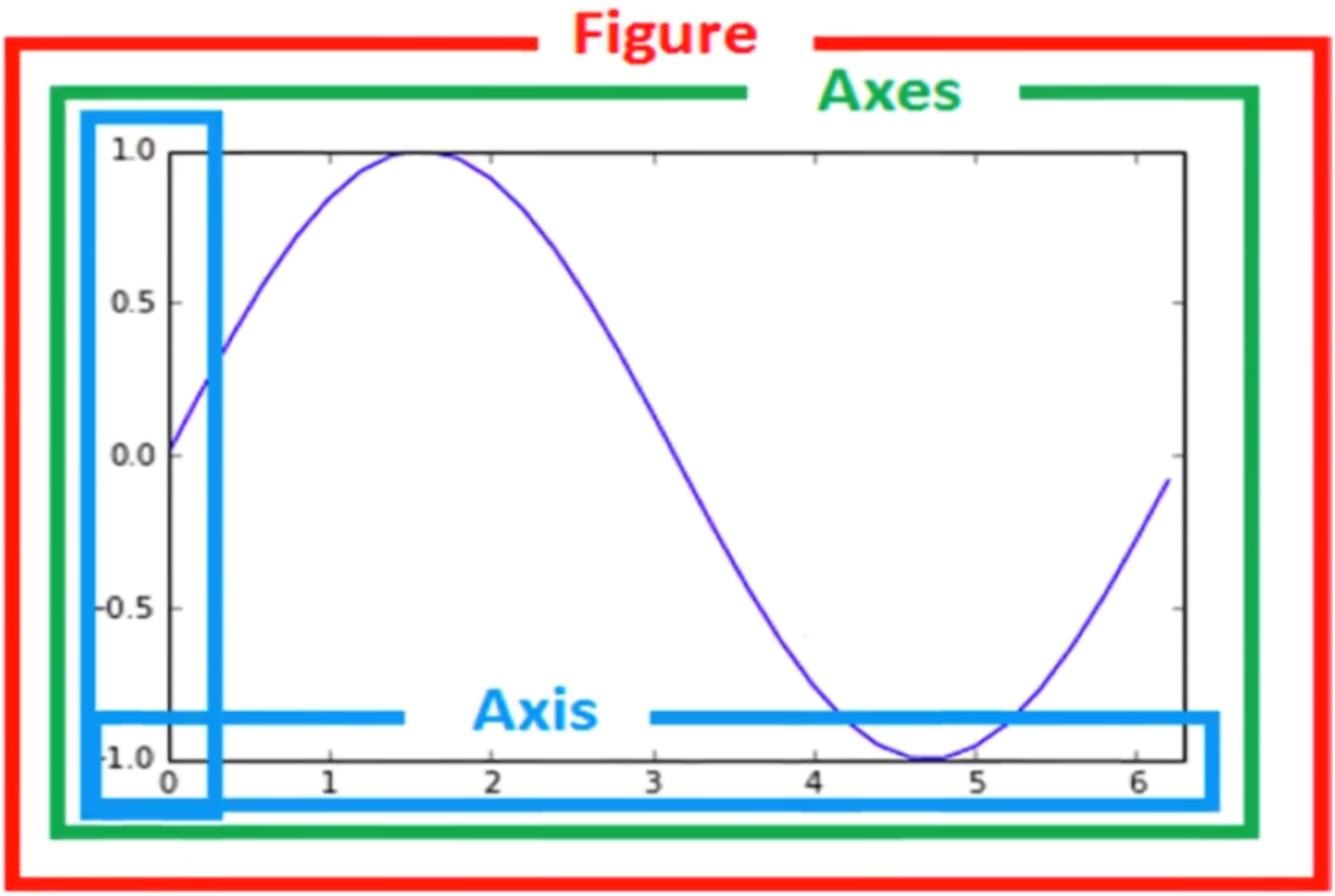


图 4.1　`Figure` 与 `Axes` 的关系

`Figure` 是级别最高的对象，它对应于整个图形表示，通常可以包含多个 `Axes` 对象。

`Axes` 称为坐标轴，每个 `Axes` 对象只属于一个 `Figure` 对象。二维坐标轴由 2 个 `Axis` 对象表示，三维情况下为 3 个。此外，标题、x 轴标签和 y 轴标签也属于 `Axes` 对象。

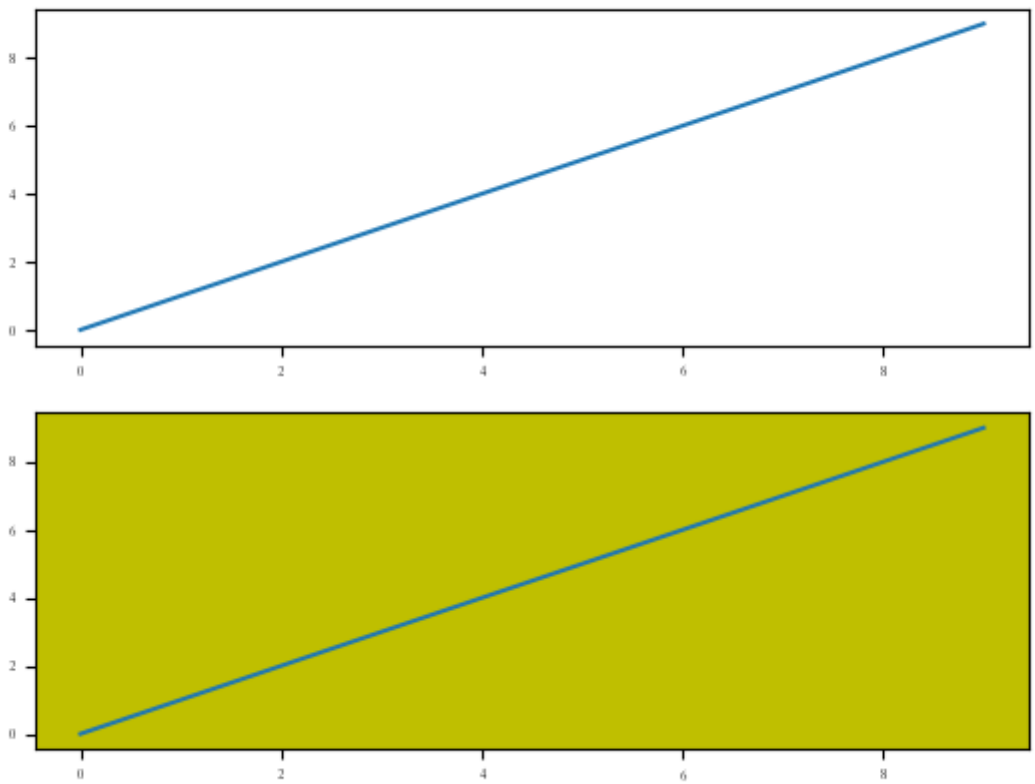
`Axis` 对象管理要在轴上表示的数值，定义坐标范围，并管理刻度和刻度标签。标签的位置由 `Locator` 对象调整，刻度标签的格式由 `Formatter` 对象调整。

4.4 在画布上创建多个子图

`pyplot.subplot(nrows, ncols, index)` 函数返回给定网格位置的 `Axes` 对象。具体说明如下：

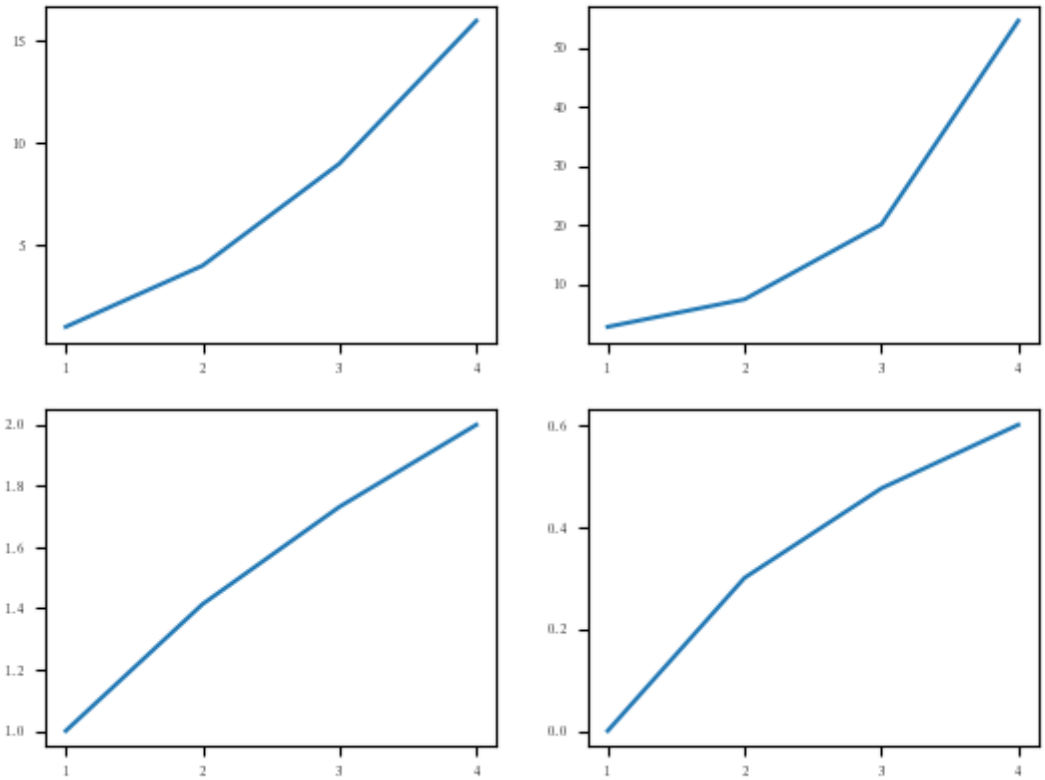
1. 将画布划分成 `nrows` 行、 `ncols` 列的网格，索引从 1 到 `nrows * ncols` ，按行优先顺序递增。在索引为 `index` 的网格中创建 `Axes` 对象。
2. 如果 `nrows` 、 `ncols` 和 `index` 都小于 10，可以将这 3 个参数连着写，中间不加逗号。例如， `subplot(2, 3, 3)` 可以写成 `subplot(233)` 。
3. 创建子图将删除任何与其重叠的预先存在的子图，而不是共享边界。

```
import matplotlib.pyplot as plt

fig = plt.figure()

# 子图 1，位于上方
ax1 = plt.subplot(2, 1, 1)
ax1.plot(range(10))

# 子图 2，位于下方，背景颜色为黄色
ax2 = plt.subplot(212, facecolor='y')
ax2.plot(range(10))

plt.show()
```



`pyplot.subplots(nrows, ncols)` 函数返回一个 `Figure` 对象和一个 `nrows` 行、 `ncols` 列的 `Axes` 对象元组，通过索引访问 `Axes` 对象。例如：

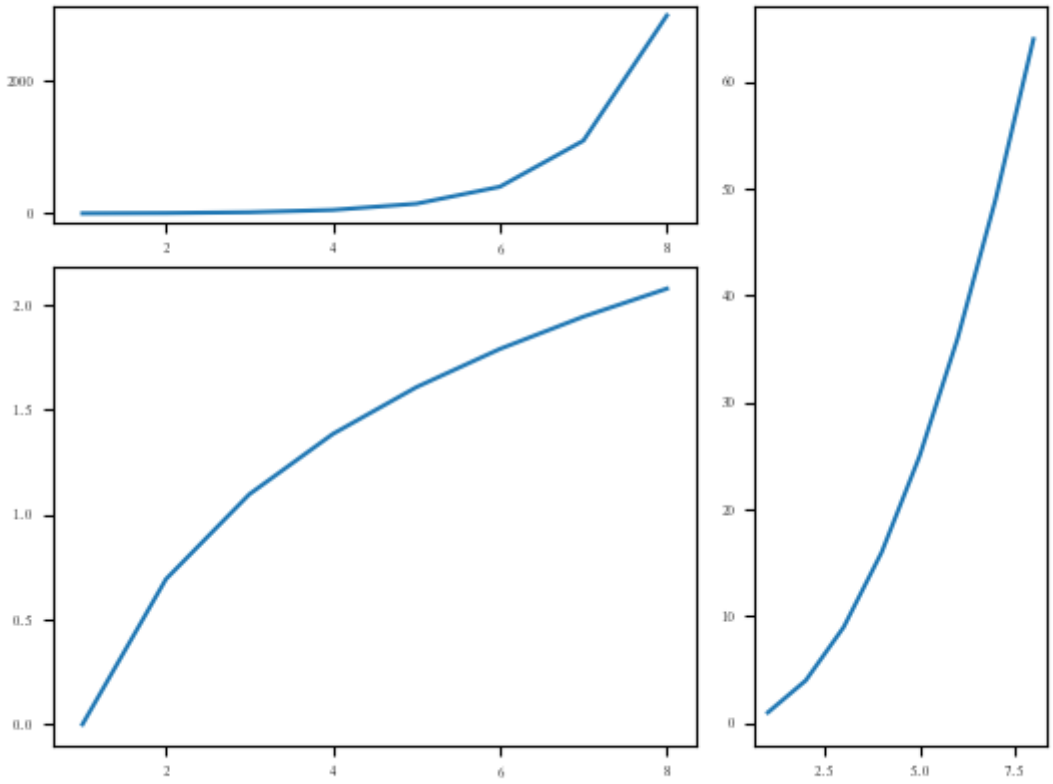
```
import matplotlib.pyplot as plt
import numpy as np

fig, axList = plt.subplots(2, 2)
x = np.arange(1, 5)

axList[0][0].plot(x, x * x)
axList[0][1].plot(x, np.exp(x))
axList[1][0].plot(x, np.sqrt(x))
axList[1][1].plot(x, np.log10(x))

plt.show()
```



`pyplot.subplot2grid()` 函数在网格的特定位置创建 `Axes` 对象，并且允许 `Axes` 对象跨越多个行或列。参数为：

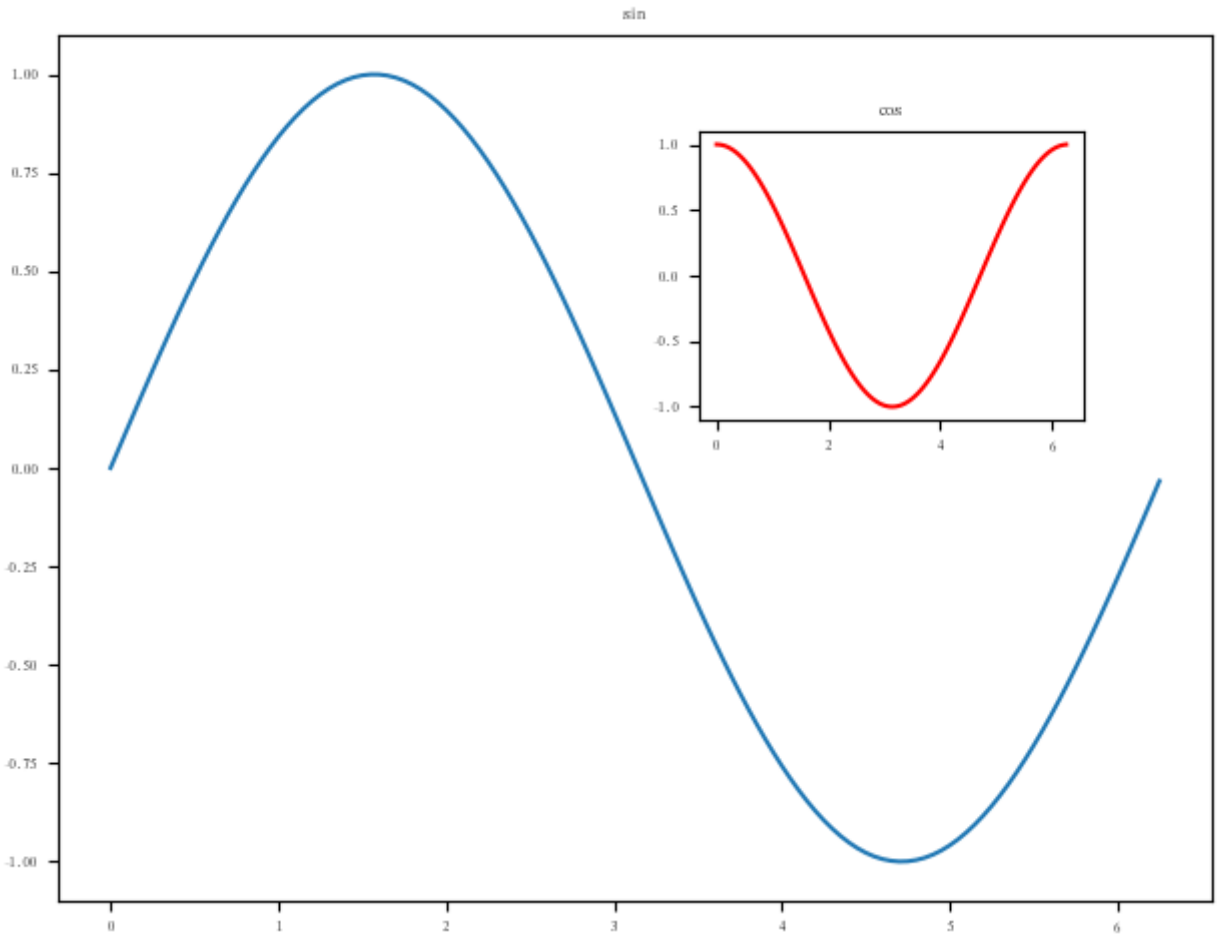
1. `shape`：整数元组，指定网格的行数和列数。
2. `loc`：整数元组，指定 `Axes` 对象的位置。
3. `rowspan`：跨越的行数。可选，默认值为 1。
4. `colspan`：跨越的列数。可选，默认值为 1。

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

x = np.arange(1, 9)

ax1 = plt.subplot2grid((3, 3), (0, 0), colspan=2)
ax1.plot(x, np.exp(x))

ax2 = plt.subplot2grid((3, 3), (0, 2), rowspan=3)
ax2.plot(x, x * x)

ax3 = plt.subplot2grid((3, 3), (1, 0), rowspan=2, colspan=2)
ax3.plot(x, np.log(x))

plt.show()
```



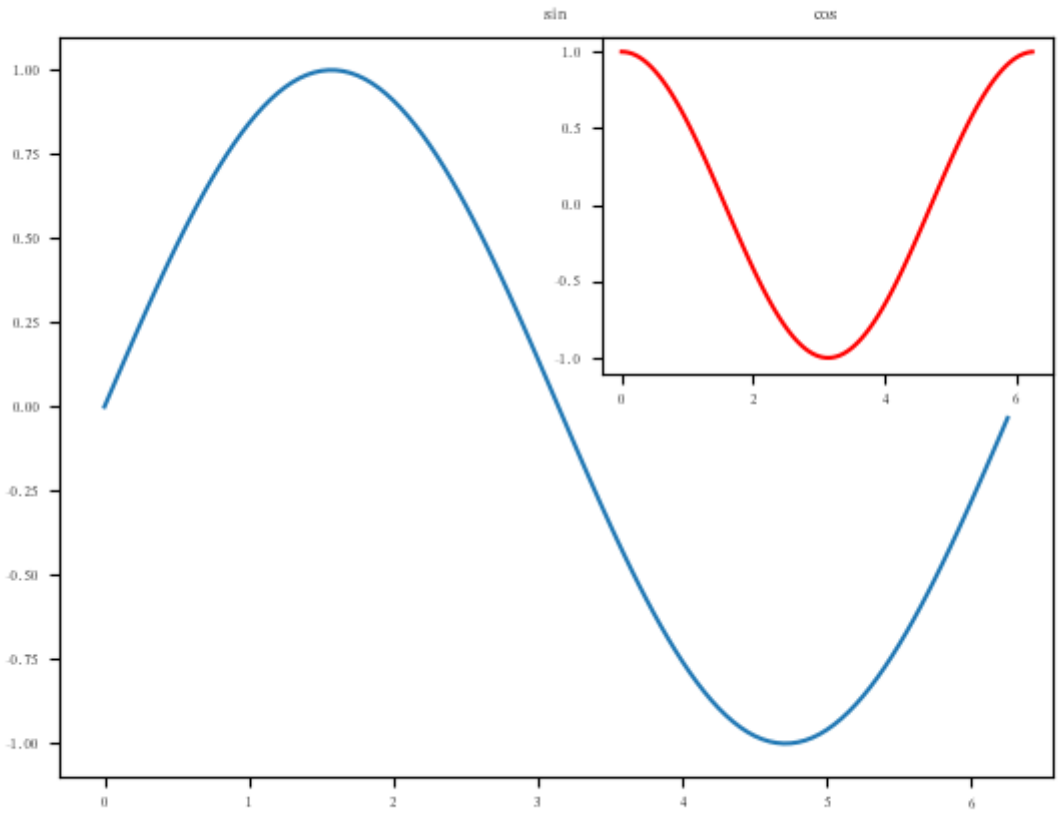
`Figure.add_axes()` 方法也可以在图中插入多个子图。例如：

```
import matplotlib.pyplot as plt
import numpy as np
import math

fig = plt.figure()
x = np.arange(0, math.pi * 2, 0.05)

ax1 = fig.add_axes([0.05, 0.05, 0.9, 0.9])
ax1.plot(x, np.sin(x))
ax1.set_title("sin")

ax2 = fig.add_axes([0.55, 0.55, 0.3, 0.3])
ax2.plot(x, np.cos(x), 'r')
ax2.set_title("cos")

plt.show()
```



`Figure.add_subplot()`方法也可以向图中添加子图，用法与`pyplot.subplot()`函数相同。例如：

```
import matplotlib.pyplot as plt
import numpy as np
import math

fig = plt.figure()
x = np.arange(0, math.pi * 2, 0.05)

ax1 = fig.add_subplot(111)
ax1.plot(x, np.sin(x))
ax1.set_title("sin")

ax2 = fig.add_subplot(222)
ax2.plot(x, np.cos(x), 'r')
ax2.set_title("cos")

plt.show()
```



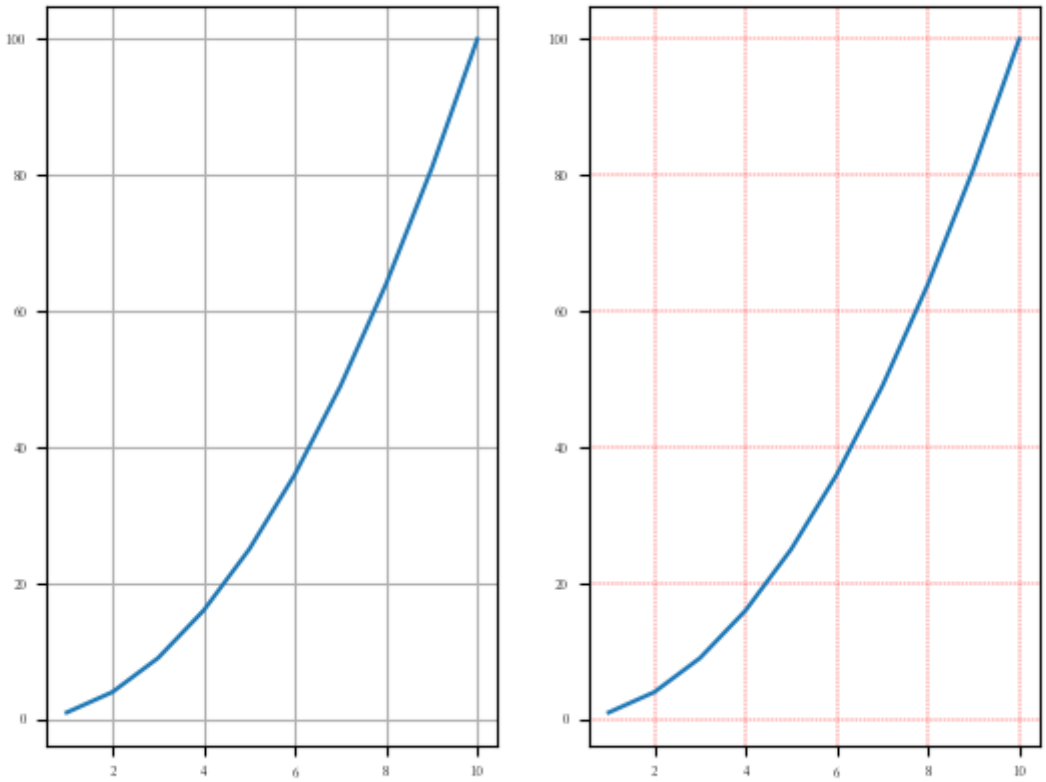
4.5 网格线

`Axes.grid()` 方法用于设置网格线的样式。参数为：

1. `b`：可选，默认值为 `None`。可以设置为布尔值，`True` 为显示网格线，`False` 为不显示。如果设置了 `**kwargs` 参数，则值为 `True`。

2. `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。
3. `axis`：可选，设置显示哪个方向的网格线，可选值有`'both'`、`'x'`或`'y'`，分别表示两个方向、x 轴方向或 y 轴方向。默认值为`'both'`。
4. `**kwargs`：可选，设置网格线样式。可选的关键字有：
   （1）`color`：线条颜色
   （2）`linestyle`或`ls`：线条样式
   （3）`linewidth`或`lw`：线条宽度

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

fig, axList = plt.subplots(1, 2)
x = np.arange(1, 11)

axList[0].plot(x, x * x)
axList[0].grid(True)

axList[1].plot(x, x * x)
axList[1].grid(color='r', ls='--', lw='0.3')

plt.show()
```



4.6 设置轴线

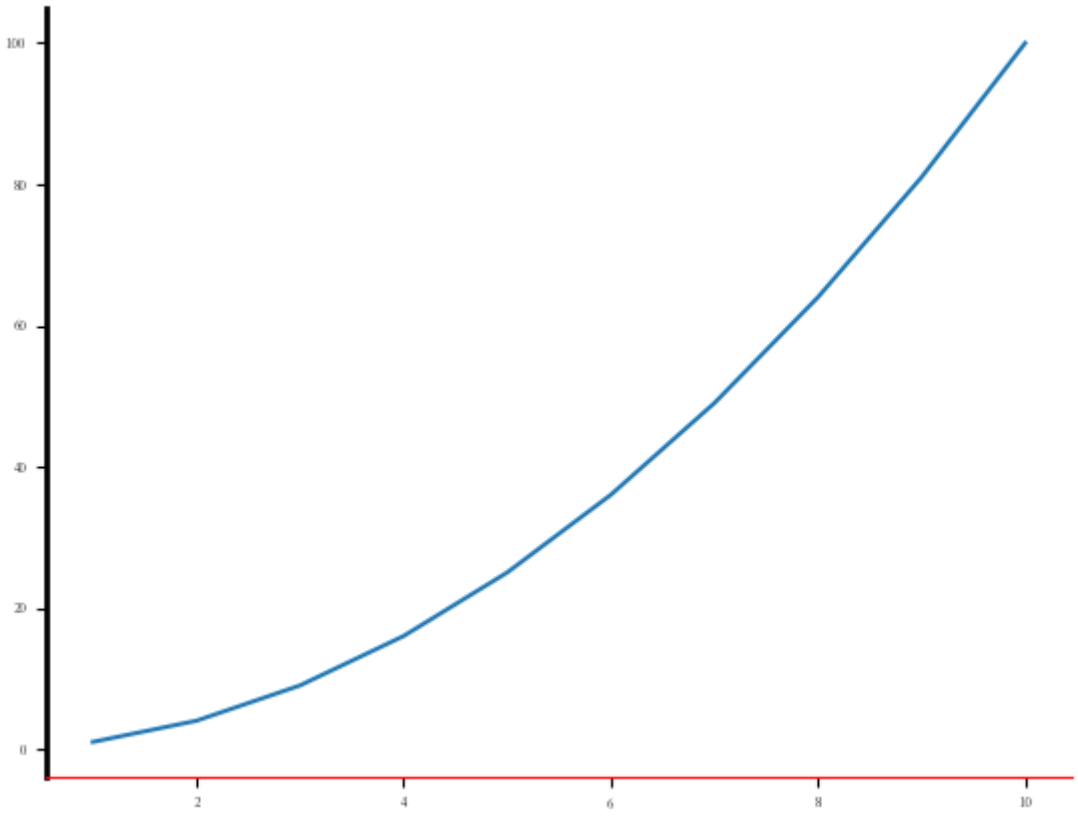
4.6.1 设置基本样式

轴脊（spine）是连接轴刻度的线，划分绘图区域的边界。`Axes`对象的轴脊位于顶部、底部、左侧和右侧。每个轴脊可以通过指定颜色和宽度进行格式化，将轴脊的颜色设置为“无”可以使其不可见。

`Axes`对象的`spines`属性是一个`Spines`对象，保存了`Axes`对象的所有轴脊。它具有类似于字典的接口，可以通过`[]`访问各个轴脊，`'left'`对应于左轴脊，`'right'`对应于右轴脊，`'top'`对应于上轴脊，`'bottom'`对应于下轴脊。也可以使用属性`left`、`right`、`top`、`bottom`来访问。

每个轴脊是一个`Spine`对象，`set_color()`方法用于设置轴脊的颜色，`set_linewidth()`方法设置轴脊的宽度。例如：

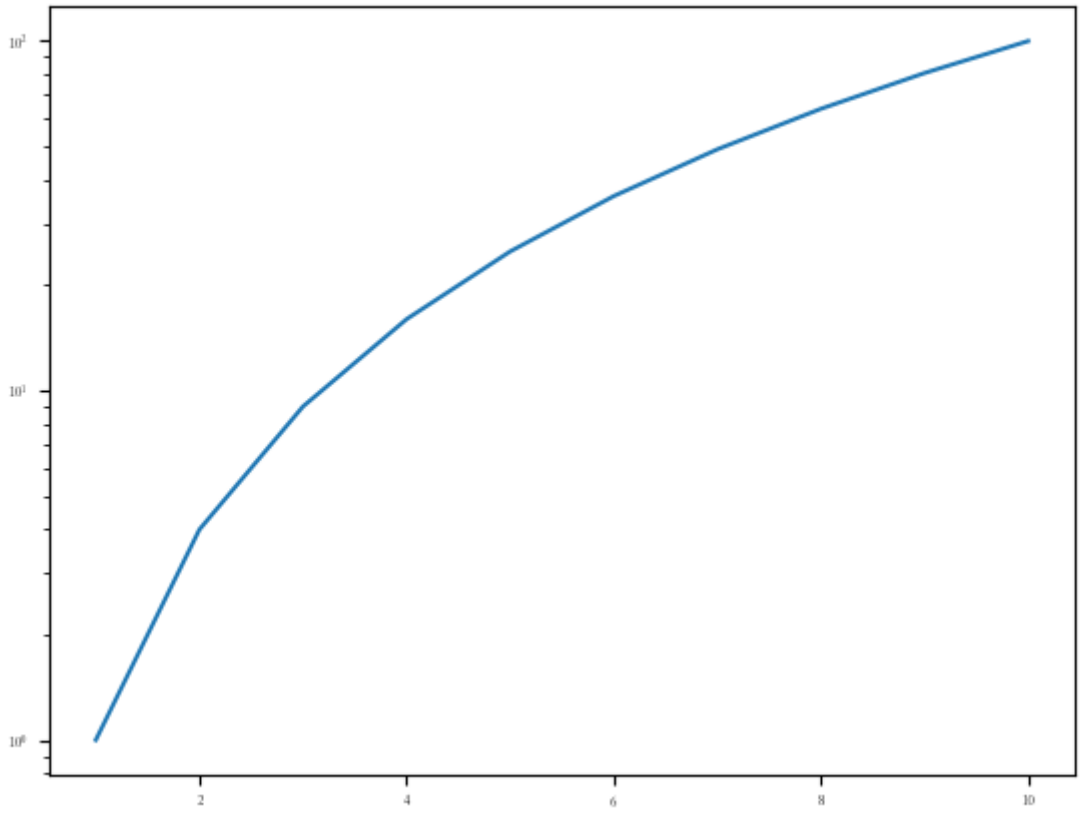
```
import numpy as np
import matplotlib.pyplot as plt

fig = plt.figure()
ax = fig.add_axes([0.1, 0.1, 0.8, 0.8])

x = np.arange(1, 11)
ax.plot(x, x * x)

ax.spines['top'].set_color(None)  # 隐藏上轴脊
ax.spines.right.set_color(None)  # 隐藏右轴脊
ax.spines.left.set_linewidth(2)  # 左轴脊宽度为 2
ax.spines['bottom'].set_color('red')  # 下轴脊颜色为红色

plt.show()
```



4.6.2 格式化轴

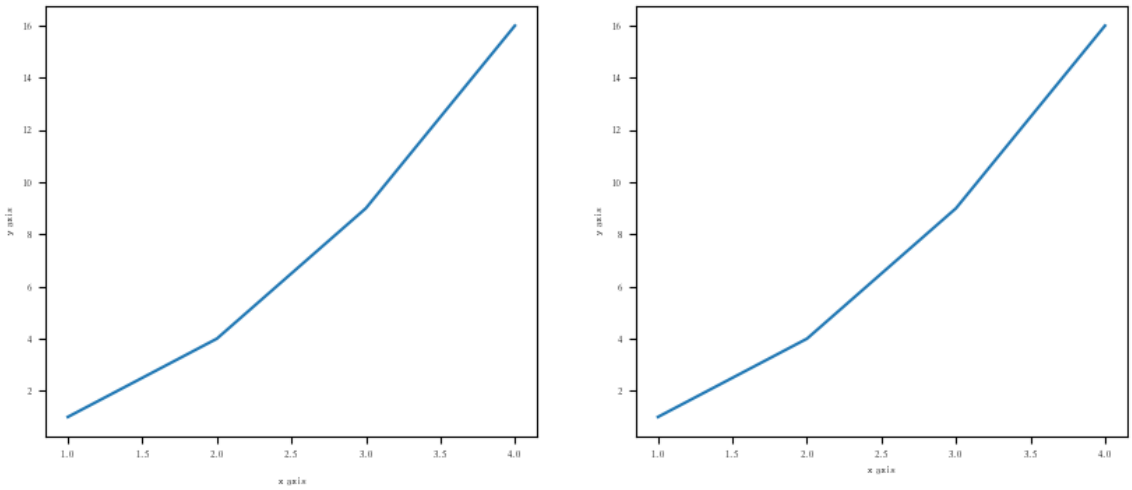
`Axes` 对象的 `set_xscale()` 和 `set_yscale()` 方法用于设置轴的比例，取值为 `'log'` 可以将轴设置为对数比例。例如：

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

fig = plt.figure()
ax = fig.add_axes([0.1, 0.1, 0.8, 0.8])

x = np.arange(1, 11)
ax.plot(x, x * x)

ax.set_yscale("log")  # 将 y 轴设置为对数比例

plt.show()
```



轴的`labelpad`属性用于设置轴标签和轴刻度的距离。例如：

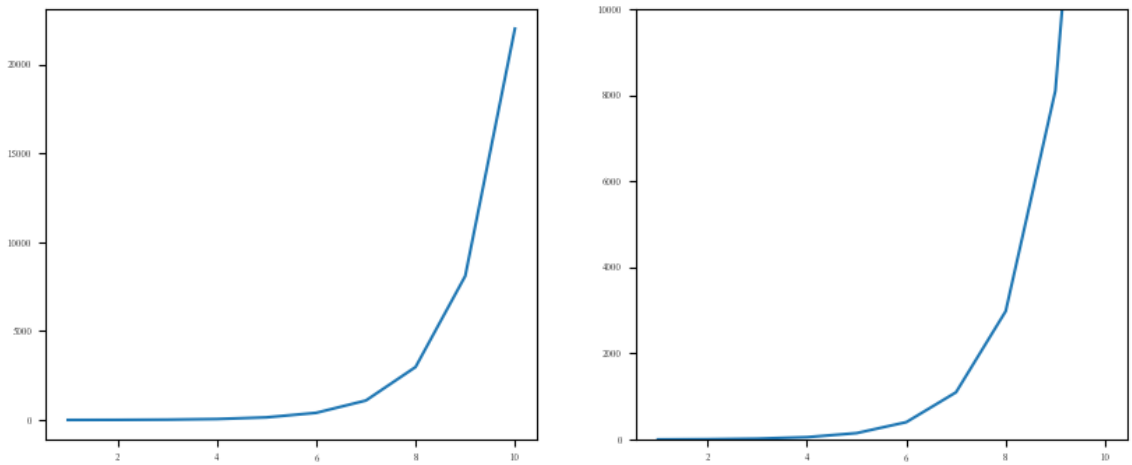
```
import numpy as np
import matplotlib.pyplot as plt

fig, axes = plt.subplots(1, 2, figsize=(10, 4))
x = np.arange(1, 5)

# 设置轴标签距离
axes[0].plot(x, x**2)
axes[0].set_xlabel("x axis")
axes[0].set_ylabel("y axis")
axes[0].xaxis.labelpad = 10  # 为 x 轴标签设置距离

# 不设置轴标签距离，作为对照
axes[1].plot(x, x**2)
axes[1].set_xlabel("x axis")
axes[1].set_ylabel("y axis")

plt.show()
```



4.6.3 设置取值范围

`Axes.set_xlim()` 方法用于设置 x 轴的取值范围，`Axes.set_ylim()` 方法用于设置 y 轴的取值范围。例如：

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

fig, axes = plt.subplots(1, 2, figsize=(10, 4))
x = np.arange(1, 11)
y = np.exp(x)

axes[0].plot(x, y)

axes[1].plot(x, y)
axes[1].set_ylim(0, 10000)  # 设置 y 轴的取值范围为 0-10000

plt.show()
```



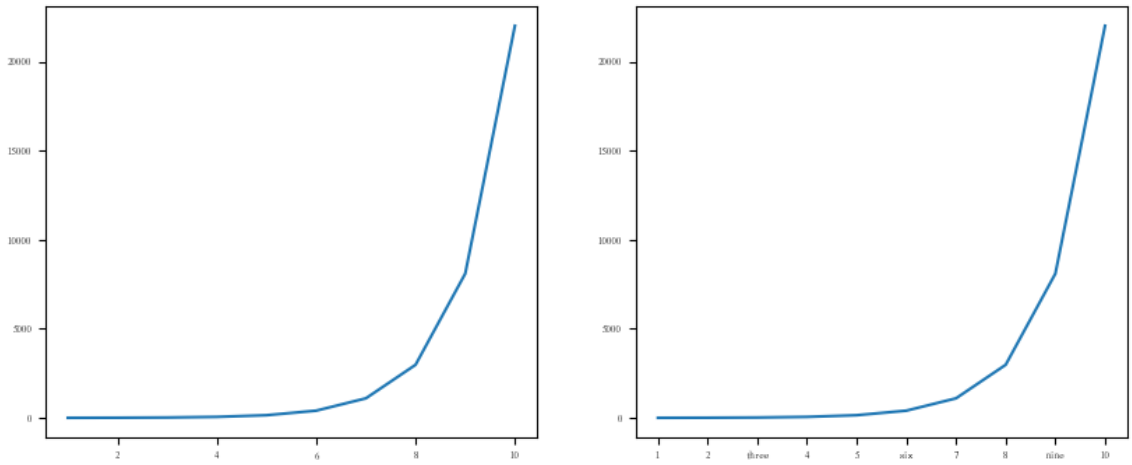
4.6.4 刻度和刻度标签

`Axes`对象的`xticks()`和`yticks()`方法用于设置刻度。参数为列表，列表元素为显示刻度线的位置。

`set_xticklabels()`和`set_yticklabels()`方法用于设置刻度标签。参数为列表，列表元素为标签值。

刻度和刻度标签必须配合使用，并且要对应起来。

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

fig, axes = plt.subplots(1, 2, figsize=(10, 4))
x = np.arange(1, 11)
y = np.exp(x)

axes[0].plot(x, y)

axes[1].plot(x, y)
axes[1].set_xticks(range(1, 11))
axes[1].set_xticklabels(['1', '2', 'three', '4', '5', 'six', '7', '8', 'nine', '10'])

plt.show()
```



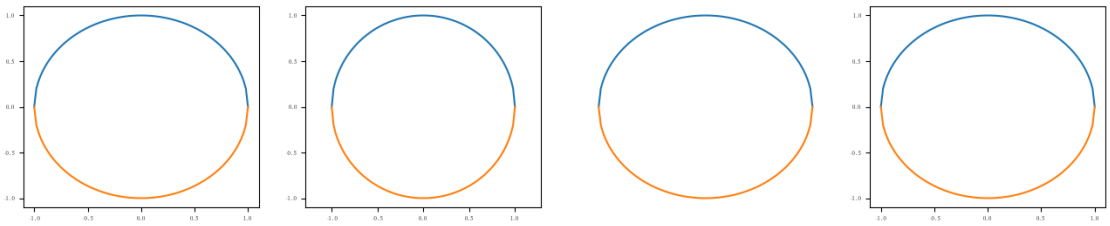
4.6.5 设置轴线类型

`Axes.axis()`方法用于设置轴线类型。常用类型有：

- `'tight'`：仅修改坐标轴极值以显示全部数据，不再进一步自动缩放坐标轴。
- `'on'`：显示坐标轴。
- `'off'`：不显示坐标轴。
- `'equal'`：通过改变坐标轴极值等比例缩放。

- `'scaled'`：通过改变绘图区维度等比例缩放。
- `'auto'`：自动缩放坐标轴。
- `'image'`：使用`'scaled'`模式，但是坐标轴极值等于数据极值。
- `'square'`：设置绘图区为正方形。

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

fig, axes = plt.subplots(1, 4, figsize=(16, 3))
x = np.linspace(-1, 1, 100)
f = lambda x: (1 - x ** 2) ** 0.5
y = np.exp(x)

axes[0].plot(x, f(x))
axes[0].plot(x, -f(x))

axes[1].plot(x, f(x))
axes[1].plot(x, -f(x))
axes[1].axis("equal")

axes[2].plot(x, f(x))
axes[2].plot(x, -f(x))
axes[2].axis("off")

axes[3].plot(x, f(x))
axes[3].plot(x, -f(x))
axes[3].axis("tight")

plt.show()
```



4.7 保存图片

`Figure.savefig()`方法用于将绘图保存到图片。参数为：

1. `fname`：含有文件路径的字符串或 Python 文件对象。图片格式由文件扩展名推断得出。
2. `dpi`：图像分辨率（每英寸点数）。可选，默认为`None`。
3. `facecolor`：图像的背景色。可选，默认为`'w'`（白色）。
4. `edgecolor`：图像的边缘颜色。可选，默认为`'w'`。

5 绘图

5.1 条形图

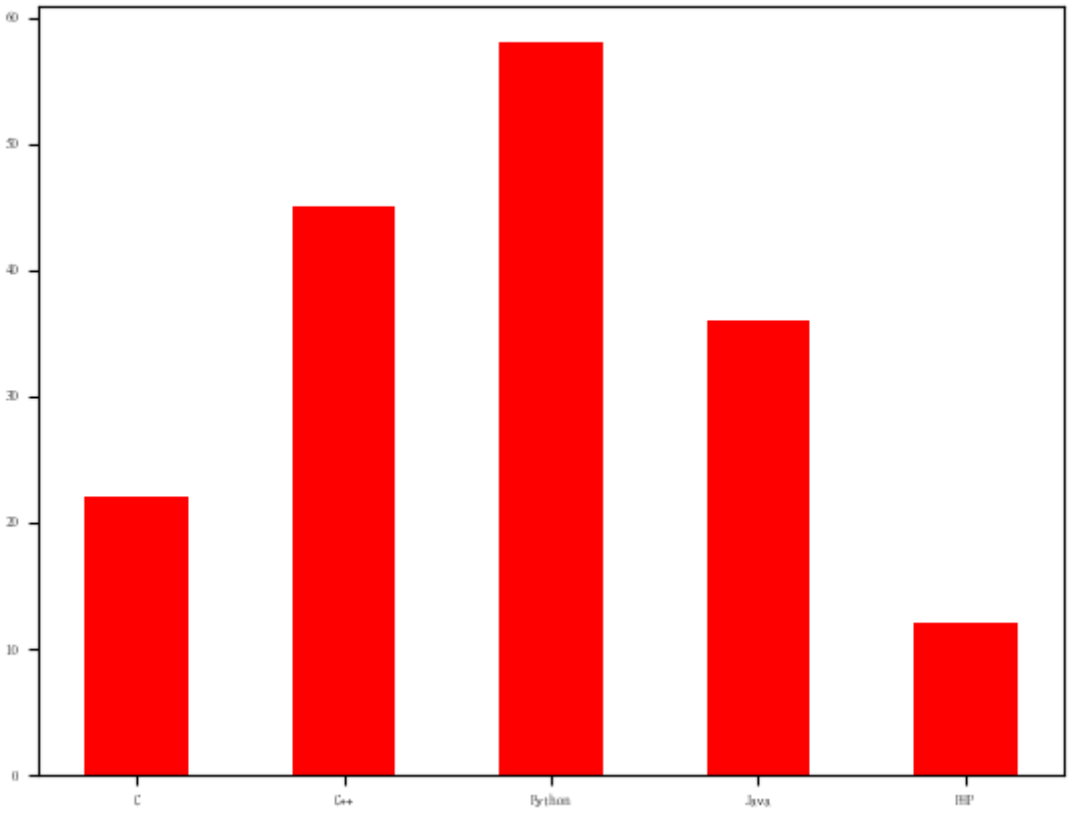
条形图显示了离散类别之间的比较，图标的一个轴显示要比较的特定类别，另一个轴表示测量值。

`Axes.bar()`方法用于绘制垂直条形图。参数为：

1. `x`：表示条形的 x 坐标的标量序列。如果 x 是条形中心或左边缘，则对齐控件。
2. `height`：标量或标量序列，表示条的高度。
3. `width`：标量或标量序列，表示条的宽度。可选，默认为 0.8。
4. `bottom`：标量或标量序列，表示条形底部的 y 坐标。可选，默认为`None`。
5. `align`：条形与刻度的位置关系。可选，`'center'`表示条形中心与刻度对齐，`'edge'`表示条形边缘与刻度对齐，默认为`'center'`。
6. `color`：条形颜色，可以是字符串，也可以是列表。

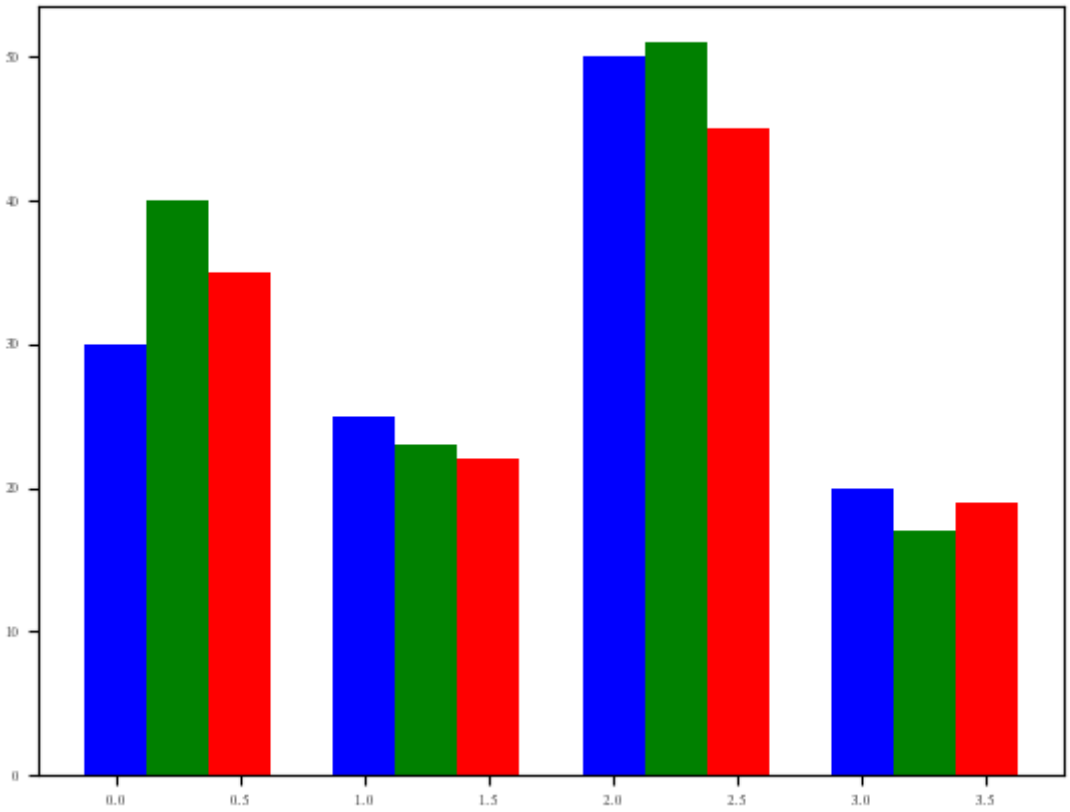
垂直基本条形图：

```
import matplotlib.pyplot as plt

fig = plt.figure()
ax = fig.add_axes([0.1, 0.1, 0.8, 0.8])

langs = ['C', 'C++', 'Python', 'Java', 'PHP']
students = [22, 45, 58, 36, 12]

ax.bar(langs, students, 0.5, color='r')

plt.show()
```



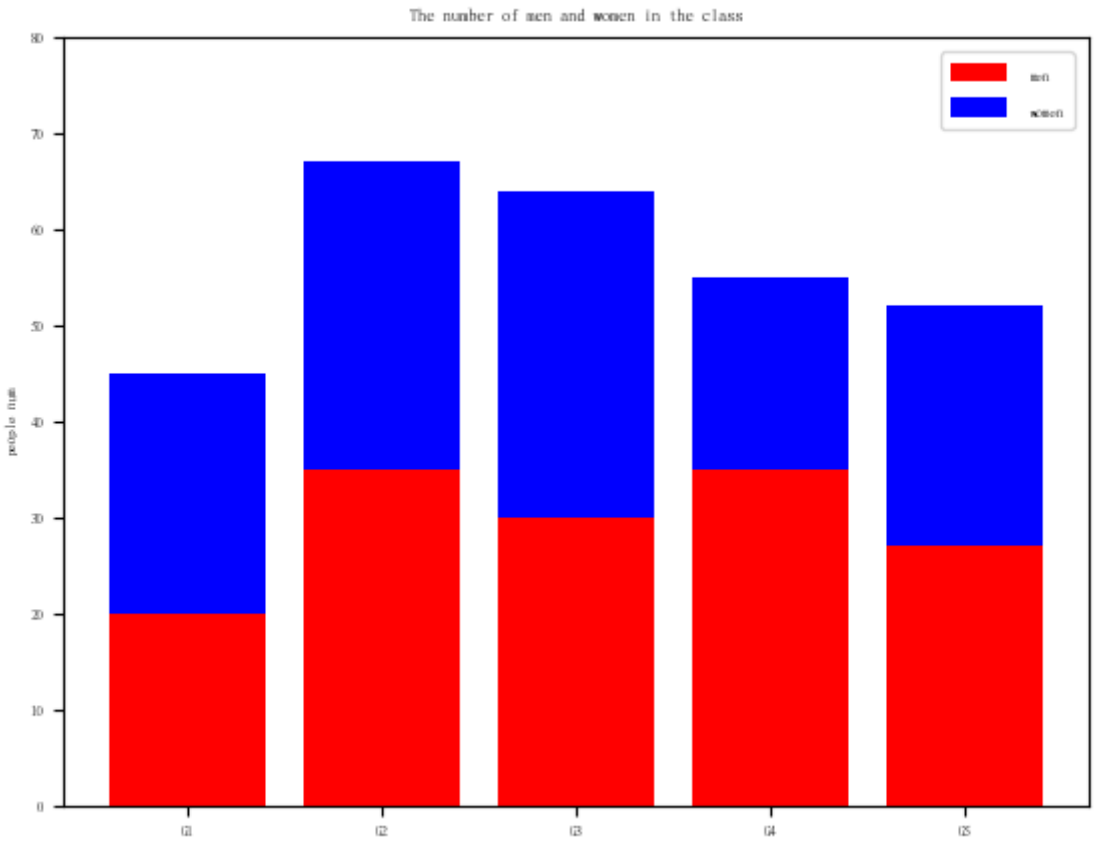
垂直组合条形图：

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

fig = plt.figure()
ax = fig.add_axes([0.1, 0.1, 0.8, 0.8])

X = np.arange(4)
data = [[30, 25, 50, 20],
        [40, 23, 51, 17],
        [35, 22, 45, 19]]

ax.bar(X, data[0], color='b', width=0.25)
ax.bar(X + 0.25, data[1], color='g', width=0.25)
ax.bar(X + 0.5, data[2], color='r', width=0.25)

plt.show()
```



垂直堆叠条形图：

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

fig = plt.figure()
ax = fig.add_axes([0.1, 0.1, 0.8, 0.8])

n = 5  # 类别数
index = np.arange(1, n + 1)
menNum = [20, 35, 30, 35, 27]  # 男性人数
womenNum = [25, 32, 34, 20, 25]  # 女性人数

ax.bar(index, menNum, color='r')  # 男性条
ax.bar(index, womenNum, color='b', bottom=menNum)  # 将女性条的底部设置为男性条的顶部

ax.set_title("The number of men and women in the class")
ax.set_ylabel("people num")
ax.set_xticks(index)
ax.set_xticklabels(['G1', 'G2', 'G3', 'G4', 'G5'])
ax.set_yticks(np.arange(0, 81, 10))
ax.legend(labels=['men', 'women'])

plt.show()
```

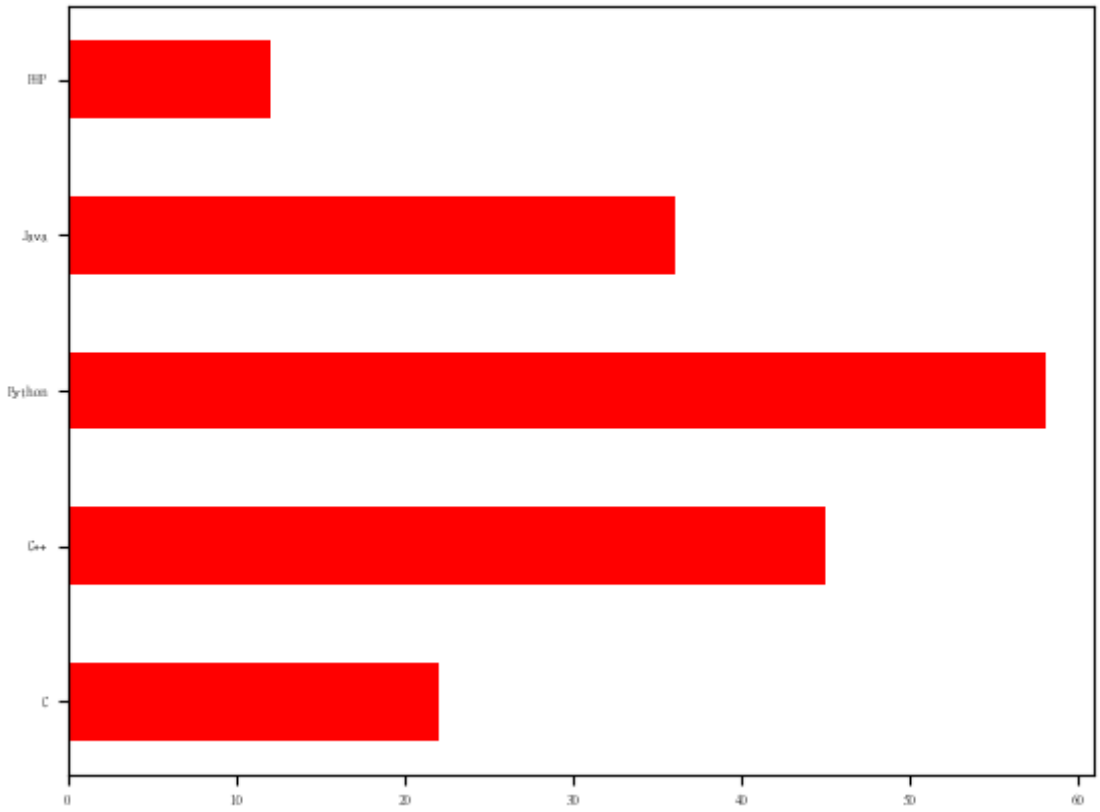


`Axes.barh()` 方法用于绘制水平条形图。参数为：

1. `y`：表示条形的 y 坐标的标量序列。如果 y 是条形中心或左边缘，则对齐控件。
2. `width`：标量或标量序列，表示条的宽度。
3. `height`：标量或标量序列，表示条的高度。可选，默认为 0.8。
4. `left`：标量或标量序列，表示条形左侧的 x 坐标。可选，默认为 `None`。
5. `align`：条形与刻度的位置关系。可选，可选值为 `'center'`、`'edge'`，默认为 `'center'`。
6. `color`：条形颜色，可以是字符串，也可以是列表。

水平条形图的绘制方法与垂直条形图一致。例如：

```
import matplotlib.pyplot as plt

fig = plt.figure()
ax = fig.add_axes([0.1, 0.1, 0.8, 0.8])

langs = ['C', 'C++', 'Python', 'Java', 'PHP']
students = [22, 45, 58, 36, 12]

ax.barh(langs, students, 0.5, color='r')

plt.show()
```



5.2 直方图

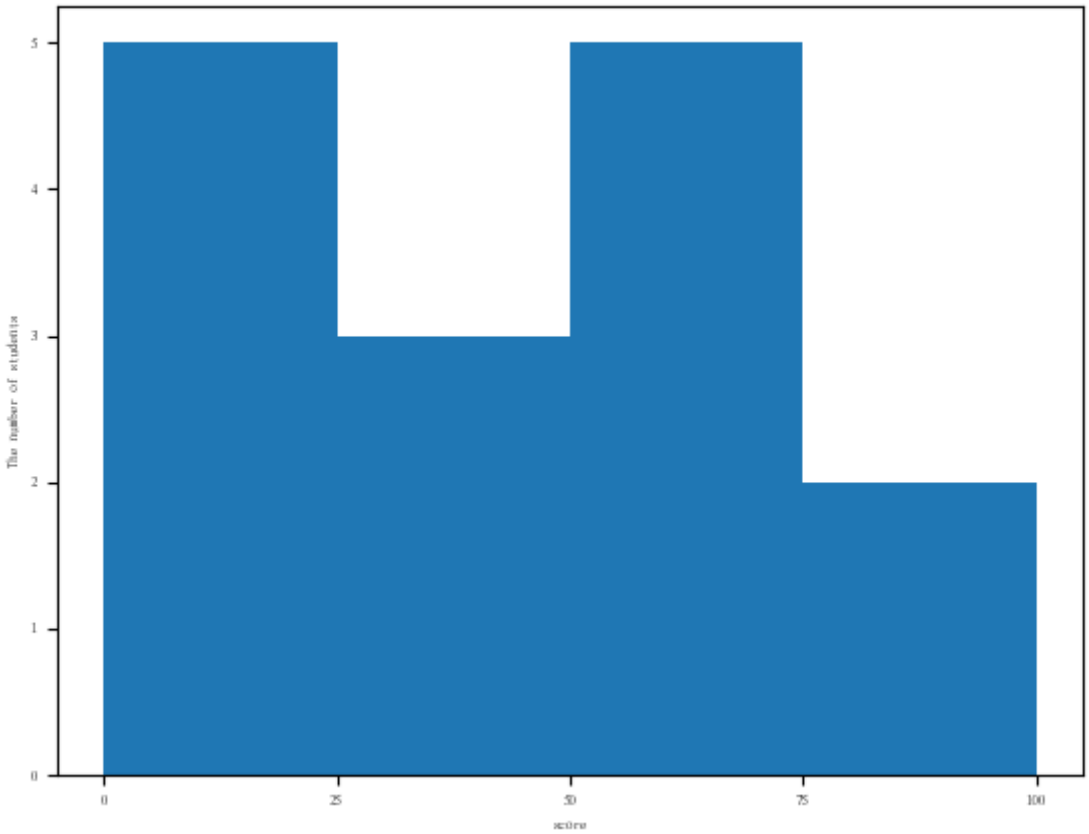
`Axes.hist()` 方法用于绘制直方图。参数为：

1. `x`：数组或数组序列。
2. `bins`：指定一系列连续、非重叠区间。当取值为整数时，将 `range` 参数等分成指定的份数；当取值为序列时，指定所有区间端点，最后一个区间为闭区间，其他区间都是左闭右开区间；当取值为字符串时，可以取指定的值，如 `'auto'`、`'fd'` 等。可选，默认为 `None`。
3. `range`：元组，指定区间的下界和上界。如果不指定，则默认范围为 `(x.min(), x.max())`。如果 `bins` 参数为序列，则 `range` 参数无效。可选，默认为 `None`。
4. `density`：布尔值，如果为 `True`，则 y 轴表示概率密度，并返回概率密度。可选，默认为 `False`。
5. `weights`：一个与 `x` 形状相同的权重数组，`x` 中的每个值只将其相关权重贡献给 bin 参与计数（默认为 1）。如果 `density` 参数为 `True`，权重将被归一化，使得密度在 `range` 范围内的积分保持为 1。可选，默认为 `None`。
6. `cumulative`：如果为 `True`，则直方图中每个 bin 的值为该 bin 的计数加上比它更小的所有 bin 的值。最后一个 bin 的值为数据点的总数。可选，默认为 `False`。
7. `bottom`：数组或常数，指定每个 bin 图形底部的坐标。如果为常数，则将所有 bin 移动相同的距离；如果为数组，则每个 bin 都会独立移动，并且 `bottom` 数组的长度必须和 `bins` 的区间

数相同。可选，默认值为 `None`，相当于取 0。

8. `histtype`：直方图的类型。可选，可能的取值为 `'bar'`、`'barstacked'`、`'step'`、`'stepfilled'`，默认值为 `'bar'`。
9. `align`：条形的水平对齐方式。可选，可能的取值为 `'left'`、`'mid'`、`'right'`，默认值为 `'mid'`。
10. `orientation`：`'horizontal'` 表示水平直方图，`'vertical'` 表示垂直直方图。可选，默认为 `'vertical'`。
11. `rwidth`：将条形的宽度设置为 bin 宽度的一定比例。可选，默认值为 `None`，自动计算宽度。如果 `histtype` 参数为 `'step'` 或 `'stepfilled'`，则忽略此参数。
12. `log`：为 `True` 则将轴设置为对数比例。可选，默认为 `False`。
13. `color`：条形的颜色。可选，默认为 `None`。
14. `label=None`：字符串或字符串序列，为条形设置标签。可选，默认为 `None`。
15. `stacked=False`：如果为 `True`，则会将多个数据堆叠在一起。如果为 `False`，则当 `histtype` 参数为 `'bar'` 时将多个数据并排排列，当 `histtype` 参数为 `'step'` 时将多个数据堆叠排列。可选，默认为 `False`。
16. `normed`：如果为 `True`，则对直方图的值进行归一化处理，形成概率密度。可选，默认为 `False`。

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

fig = plt.figure()
ax = fig.add_axes([0.1, 0.1, 0.8, 0.8])

data = np.array([22, 87, 5, 43, 56, 73, 55, 54, 11, 20, 51, 5, 79, 31, 27])

ax.hist(data, bins=[0, 25, 50, 75, 100])

ax.set_xticks([0, 25, 50, 75, 100])
ax.set_xlabel("score")
ax.set_ylabel("The number of students")

plt.show()
```



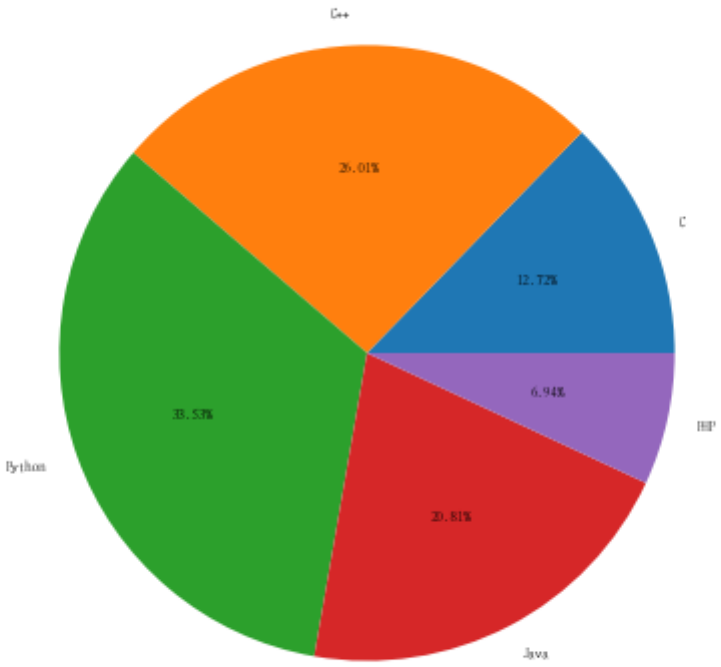
5.3 饼图

`Axes.pie()`方法用于绘制饼图。参数为：

1. `x`：数组，表示每个扇形的大小。
2. `explode`：数组，设置每个扇形的顶点到圆心的距离（比例值）。可选，默认为`None`。
3. `labels`：字符串列表，为每个扇形提供标签。可选，默认为`None`。
4. `colors`：数组，从中循环选取颜色为扇形上色。可选，默认为`None`，此时将使用当前当前活动周期中的颜色。
5. `autopct`：字符串，用于设置扇形内部显示的数值。可选，默认为`None`
6. `pctdistance`：浮点数，设置比例值文字到圆心的距离。可选，默认值为 0.6。
7. `shadow`：布尔值，设置是否绘制阴影。可选，默认值为`False`。
8. `normalize`：布尔值，如果为`True`，则将`x`标准化，使得`sum(x) == 1`，从而生成完整的饼图；如果为`False`，则当`sum(x) <= 1`时生成部分饼图，当`sum(x) > 1`时抛出`ValueError`异常。可选，默认为`True`。
9. `labeldistance`：浮点数，设置标签到圆心的距离。如果取值为`None`，则不显示标签，但会将标签保存起来以供`legend()`函数使用。可选，默认值为 1.1。
10. `startangle`：设置饼图起点的旋转角度，角度值以 x 轴为起点，以逆时针方向为正方向。可选，默认值为 0。

11. `radius`：饼图的半径。可选，默认值为 1。
12. `counterclock`：布尔值，指定绘制方向，`True`表示逆时针，`False`表示顺时针。可选，默认值为`True`。
13. `wedgeprops`：字典，设置扇形的属性。可选，默认为`None`。
14. `textprops`：字典，设置文字的属性。可选，默认为`None`。
15. `center`：圆心坐标。可选，默认值为`(0, 0)`。
16. `frame=False`：布尔值，如果为`True`则绘制边框。可选，默认为`False`。
17. `rotatelabels=False`：布尔值，如果为`True`则将标签旋转到相应扇形的角度。可选，默认为`False`。

```
import matplotlib.pyplot as plt

fig = plt.figure()
ax = fig.add_axes([0.1, 0.1, 0.8, 0.8])
ax.axis('equal')

langs = ['C', 'C++', 'Python', 'Java', 'PHP']
students = [22, 45, 58, 36, 12]

ax.pie(students, labels=langs, autopct='%1.2f%%')

plt.show()
```

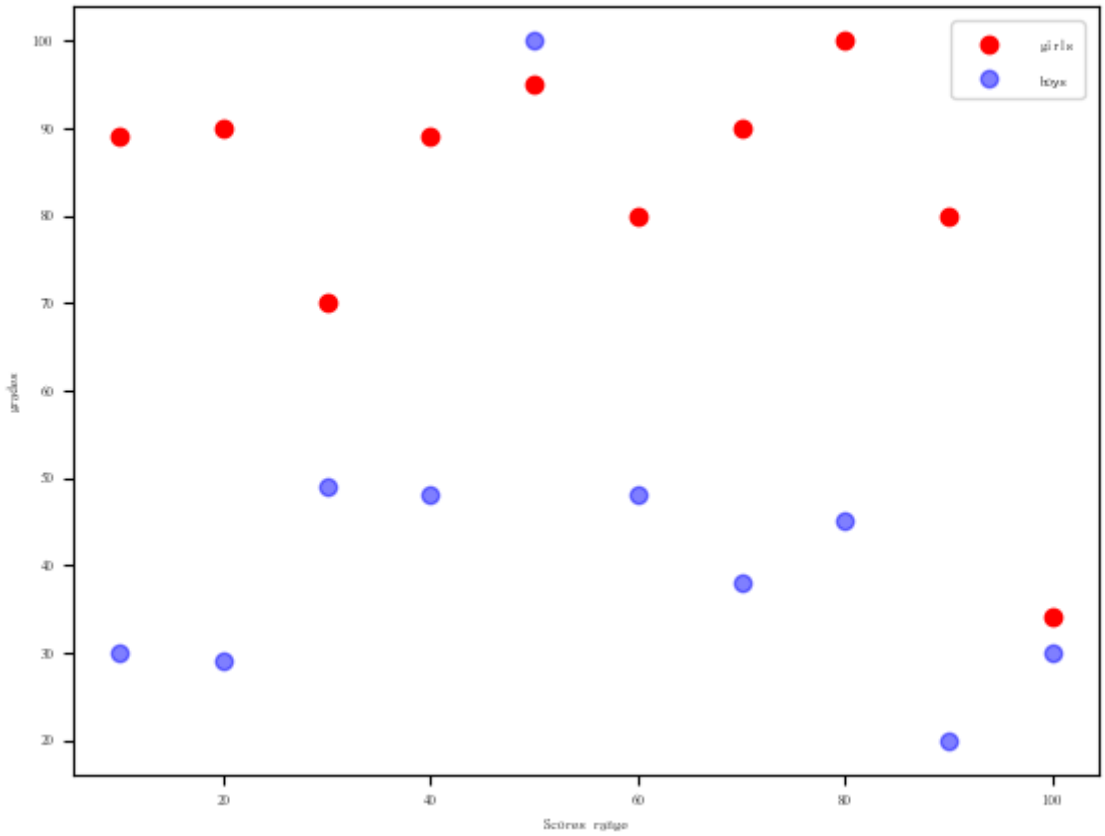


5.4 散点图

`Axes.scatter()` 方法用于绘制散点图。参数为：

1. `x`：浮点数或数组，表示数据的 x 坐标。
2. `y`：浮点数或数组，表示数据的 y 坐标。
3. `s`：浮点数或数组，设置标记点的大小。可选，默认为 `None`。
4. `c`：标记点的颜色。可选，默认为 `None`。
5. `marker`：标记点的样式。可选，默认为 `None`。
6. `cmap`：一个 `Colormap` 对象，或已注册的颜色映射名称，用于将常数映射为颜色。如果 `c` 参数的值为 RGB(A)，则忽略此参数。可选，默认为 `None`。
7. `norm`：在使用 `cmap` 映射到颜色之前，用于将常数缩放到 $[0, 1]$ 范围的归一化方法。可选，默认为 `None`，使用线性缩放，将最小值映射到 0，将最大值映射到 1。
8. `vmin`：当使用常数映射颜色而不使用 `norm` 参数时，定义颜色映射的下界。可选，默认为 `None`。
9. `vmax`：当使用常数映射颜色而不使用 `norm` 参数时，定义颜色映射的上界。可选，默认为 `None`。
10. `alpha`：透明度，取值范围为 $[0, 1]$，0 表示完全透明，1 表示不透明。可选，默认为 `None`。
11. `linewidths`：标记点边缘线的宽度。可选，默认值为 1.5。
12. `edgecolors`：标记点边缘线的颜色。可取的值包括 `'face'`、`'none'`、颜色值或颜色值的序列。可选，默认为 `'face'`，表示边缘线的颜色与点的颜色相同。
13. `plotnonfinite=False`：是否使用非限定的 `c` 参数值（如 `inf`、`-inf`、`nan`）绘制标记点。如果为 `True`，则使用内置的错误颜色来绘制标记点。

```
import matplotlib.pyplot as plt

fig = plt.figure()
ax = fig.add_axes([0.1, 0.1, 0.8, 0.8])

girls_grades = [89, 90, 70, 89, 95, 80, 90, 100, 80, 34]
boys_grades = [30, 29, 49, 48, 100, 48, 38, 45, 20, 30]
grades_range = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

ax.scatter(grades_range, girls_grades, color='r')
ax.scatter(grades_range, boys_grades, color='b', alpha=0.5)

ax.set_xlabel('Scores range')
ax.set_ylabel('grades')
ax.legend(labels=('girls', 'boys'), loc='upper right')

plt.show()
```

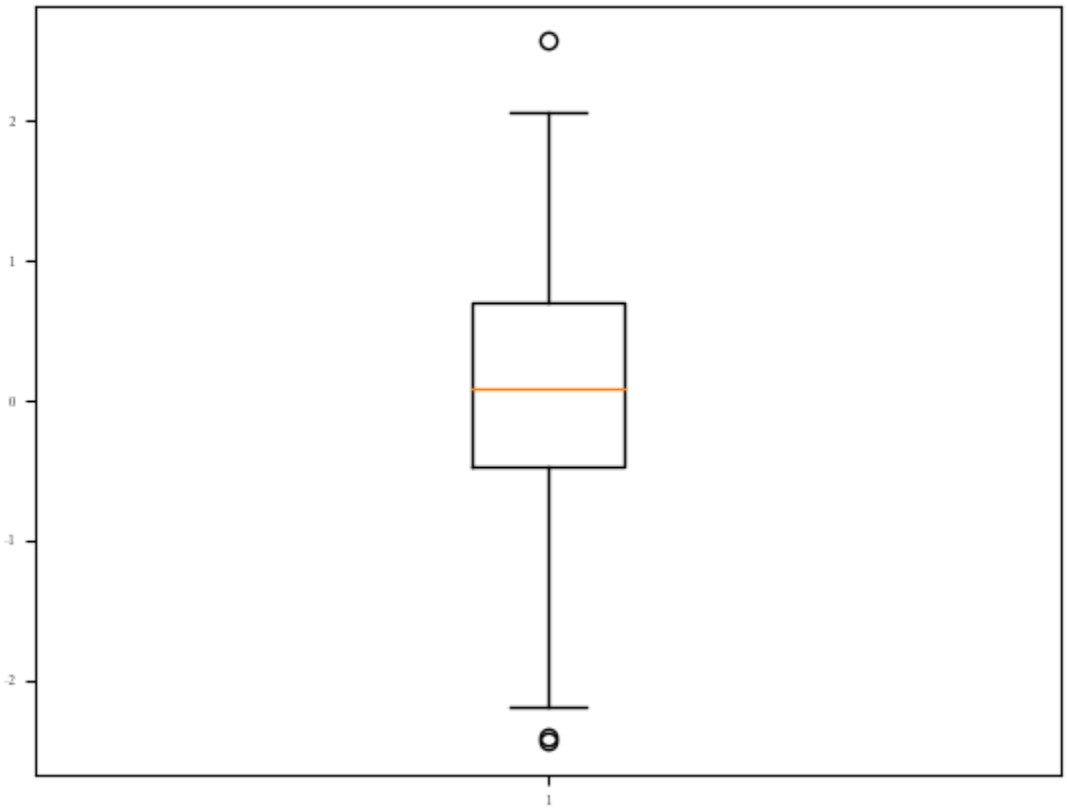


5.5 箱型图

箱型图也称为须状图，显示一组数据的最小值、第一四分位数、中位数、第三四分位数和最大值，并在第一四分位数和第三四分位数之间绘制方框。

`Axes.boxplot()` 方法用于绘制箱型图。例如：

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

fig = plt.figure()
ax = fig.add_axes([0.1, 0.1, 0.8, 0.8])

data = np.random.normal(0, 1, 200)
ax.boxplot(data)

plt.show()
```



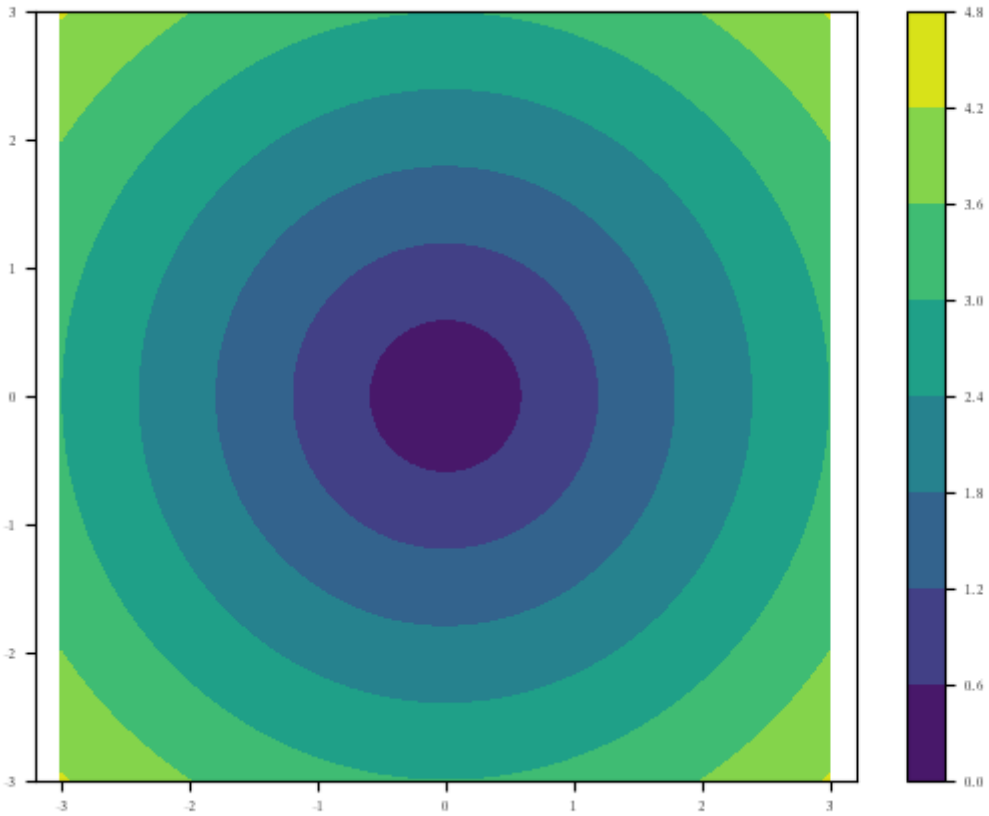
5.6 轮廓图

轮廓图是一种在二维平面上显示三维表面的方法，它绘制了 y 轴上的两个预测变量 X、Y 和轮廓的响应变量 Z，这些轮廓也称为 z 切片或等响应值。如果要查看 Z 如何随输入 X 和 Y 的变化而变化，就可以使用轮廓图。

自变量 x 和 y 通常限于称为 meshgrid 的规则网格，`numpy.meshgrid(x, y)`函数用于创建矩形网格。

`Axes.contourf()`方法用于绘制轮廓图。例如：

```python
import numpy as np
import matplotlib.pyplot as plt

fig = plt.figure()
ax = fig.add_axes([0.1, 0.1, 0.8, 0.8])
ax.axis('equal')

x = np.linspace(-3, 3, 100)
y = np.linspace(-3, 3, 100)

X, Y = np.meshgrid(x, y)
Z = np.sqrt(X ** 2 + Y ** 2)

ct = ax.contourf(X, Y, Z)  # 绘制轮廓图
fig.colorbar(ct)  # 颜色条

plt.show()
```



5.7 图像中的文字、注释、箭头

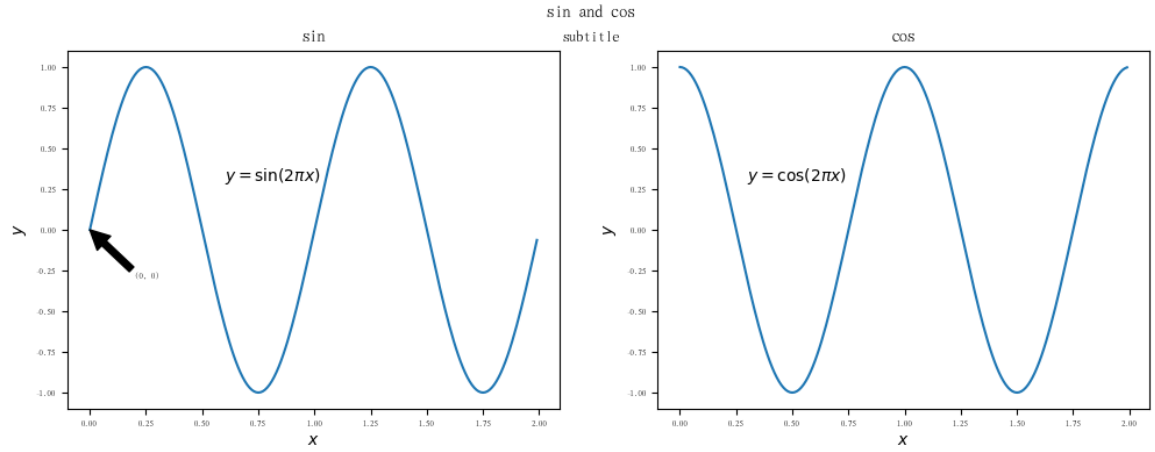
1. `Axes.text()`：在`Axes`对象的任意位置添加文字。
2. `Axes.set_xlabel()`：为 x 轴添加标签。
3. `Axes.set_ylabel()`：为 y 轴添加标签。
4. `Axes.set_title()`：为`Axes`对象添加标题。

5. `Axes.legend()`：为`Axes`对象添加图例。
6. `pyplot.suptitle()`：为`Figure`对象添加中心化的标题。
7. `pyplot.figtext()`：在`Figure`对象的任意位置添加文字。
8. `Axes.annotate()`：为`Axes`对象添加注释，箭头可选。
    - `xy`：设置箭头指示的位置
    - `xytext`：设置注释文字的位置
    - `arrowprops`：字典，设置箭头的样式
        - `width`：设置连接线的宽度
        - `headlength`：设置箭头尖端的长度
        - `headwidth`：设置箭头尖端底部的宽度
        - `facecolor`：设置箭头颜色
        - `shrink`：设置箭头顶点、尾部与指示点、注释文字的距离（比例值）
        - `arrowstyle`：箭头样式
        - `connectionstyle`：连接线的样式

上述所有函数和方法会返回一个`matplotlib.text.Text`对象。

注：可以将 TeXmarkup 文本放在一对`$`符号中，以插入数学公式。

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

fig = plt.figure(figsize=(12, 4))

x = np.arange(0, 2, 0.01)
y1 = np.sin(2 * np.pi * x)
y2 = np.cos(2 * np.pi * x)

ax1 = plt.subplot(121)
ax1.plot(x, y1)
ax1.set_title('sin', fontsize=20)  # 设置标题
ax1.set_xlabel(r'$x$')  # 设置 x 轴标签
ax1.set_ylabel(r'$y$')  # 设置 y 轴标签
ax1.text(0.6, 0.3, r'$y=\sin(2 \pi x)$')  # 添加文字

ax2 = plt.subplot(122)
ax2.plot(x, y2)
ax2.set_title('cos', fontsize=20)  # 设置标题
ax2.set_xlabel(r'$x$')  # 设置 x 轴标签
ax2.set_ylabel(r'$y$')  # 设置 y 轴标签
ax2.text(0.3, 0.3, r'$y=\cos(2 \pi x)$')  # 添加文字

plt.suptitle('sin and cos', fontsize=20)  # 设置中心化标题
plt.show()
```



6 图像处理

6.1 Pillow模块

```
from PIL import Image  # 引入 Pillow 模块
import numpy as np

# 读取图片
path = "cat.jpg"  # 图片文件的路径
cat = Image.open(path)  # 打开图片
cat_data = np.array(cat)  # 转换成 numpy 数组，结果为三维数组，保存每个像素的 RGB 值

# 对 numpy 数组做操作
cat_data = cat_data[:, :, ::-1]  # 将“红绿蓝”变成“蓝绿红”

# 获得操作后的图片
cat2 = Image.fromarray(cat_data)  # 将 numpy 数组转化成图片
```

6.2 Matplotlib中的图像模块

Matplotlib 提供了加载、缩放和显示图像所需的功能，但是 Matplotlib 仅支持 png 图像。

对于 RGB 和 RGBA 图像，Matplotlib 支持`float32`和`uint8`数据类型；对于灰度图，Matplotlib 仅支持`float32`。

`matplotlib.image.imread()`函数用于读取 png 图像数据，返回一个`dtype`为`float32`的 Ndarray 数组，其中每个数组元素都是 0 到 1 之间的浮点数。如果图片不是 png 类型，则会调用 Pillow 模块的相关函数，返回的数组中元素类型为`uint8`。

`matplotlib.pyplot.imshow()`函数用于显示图像，参数为三维数组。

`matplotlib.pyplot.imsave()`函数用于保存图片。第一个参数为文件路径，第二个参数为三维数组。

6.3 Ndarray图像操作

水平翻转：将列倒序（`data = data[:, ::-1, :]`）
上下翻转：将行倒序（`data = data[::-1, :, :]`）
截取图片：数组的切片操作
拼接图片：`numpy.concatenate()`、`numpy.stack()`
切割图片：`numpy.split()`

6.4 图像灰度化

在 RGB 模型中，当 R=G=B 时表示一种灰度颜色，其中 R=G=B 的值称为灰度值，又称强度值、亮度值。灰度图像中的每个像素只需一个字节存放灰度值。

将彩色图像转化成灰度图像的过程称为图像的灰度化处理。

极值法：

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

data = mpimg.imread("cat.png")

# 最小值法，得到的灰度图较暗
data_gray = data.min(axis=-1)  # 取出 RGB 中的最小值
plt.imshow(data_gray, cmap="gray")

# 最大值法，得到的灰度图较亮
data_gray2 = data.max(axis=-1)  # 取出 RGB 中的最大值
plt.imshow(data_gray2, cmap="gray")
```

平均值法：

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

data = mpimg.imread("cat.png")

data_gray = data.mean(axis=-1)  # 求每个像素 RGB 值的均值
plt.imshow(data_gray, cmap="gray")
```

加权平均值法：

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

data = mpimg.imread("cat.png")

weight = [0.299, 0.587, 0.114]
data_gray = np.dot(data, weight)
plt.imshow(data_gray, cmap="gray")
```