第 3 章

Matplotlib数据可视化实战

本章学习目标

（1）熟练掌握扩展库 Matplotlib 及其依赖库的安装方法。
（2）了解 Matplotlib 的绘图一般过程。
（3）熟练掌握折线图的绘制与属性设置。
（4）熟练掌握散点图的绘制与属性设置。
（5）熟练掌握柱状图的绘制与属性设置。
（6）熟练掌握饼状图的绘制与属性设置。
（7）熟练掌握雷达图的绘制与属性设置。
（8）了解三维曲线、曲面的绘制与属性设置。
（9）熟练掌握绘图区域的切分与属性设置。
（10）熟练掌握图例属性的设置。
（11）熟练掌握坐标轴属性的设置。
（12）了解事件响应与处理机制的工作原理。
（13）了解图形填充的方法。
（14）了解保存绘图结果的方法。

3.1 数据可视化库 Matplotlib 基础

视频二维码：3.1

在扩展库 Matplotlib 中有 2 个最常用的绘图模块：pylab 和 pyplot。其中，pylab 中除了包含 pyplot 模块中的函数，还包含了扩展库 NumPy 中的常用函数，可以直接通过 pylab 进行调用，不需要再额外导入 NumPy。

使用 pylab 或 pyplot 绘图的一般过程为：首先生成、读入或计算得到数据，然后根据实际需要绘制折线图、散点图、柱状图、饼状图、雷达图、箱线图、三维曲线 / 曲面以及极坐标系图形，接下来设置坐标轴标签（使用 matplotlib.pyplot 模块的 xlabel()、ylabel() 函数或轴域的 set_xlabel()、set_ylabel() 方法）、坐标轴刻度（使用 matplotlib.pyplot 模块的 xticks()、yticks() 函数或轴域的 set_xticks()、set_yticks() 方法）、图例（可以使用 matplotlib.pyplot 模块的 legend() 函数或轴域的同名方法）、标题（可以使用 matplotlib.pyplot 模块的 title()、suptitle() 函数或轴域的 set_title() 方法）等图形属性，最后显示或保存绘图结果。

每一种图形都有特定的应用场景，对于不同类型的数据和可视化要求，要选择最合适类型的图形进行展示，不能生硬地套用某种图形。

Matplotlib 默认情况下无法直接显示中文字符。如果图形中需要显示中文字符，可以使用 import matplotlib.pyplot as plt 导入模块 pyplot，然后查看 plt.rcParams 字典的当前值并进行必要的修改，也可以通过 pyplot 模块的 xlabel()、ylabel()、xticks()、yticks()、title() 等函数或轴域（也称子图）对象对应的方法的 fontproperties 参数对坐标轴标签、坐标轴刻度、标题单独进行设置；如果需要设置图例中的中文字符字体可以通过 legend() 函数的 prop 参数进行设置。

使用下面的代码可以查看所有的可用字体。

```
from matplotlib.font_manager import fontManager

names = sorted([f.name for f in fontManager.ttflist])
for name in names:
    print(name)
```

如果安装了新字体之后在自己的程序中仍无法使用，可以删除文件 C:/Users/.../.matplotlib/fontlist-v330.json，然后重新运行程序。

在进行可视化时，应尽量避免仅仅依赖于颜色不同来区分同一个图形中的多个线条、柱或面片，还应借助于线型、线宽、端点符号、填充符号等属性来提高区分度。因为有时不仅要在计算机上查看图形，可能还需要打印，但是并不能保证总是有彩色打印机，灰度打印时颜色信息丢失后就很难区分不同颜色的线条、柱或面片了。

同一组数据可以使用不同形式的图形进行可视化，既可以绘制折线图，也可以绘制柱状图、散点图、饼状图等其他图形，具体采用哪种图形最终取决于客户的要求和应用场景，确定之后调用相应的函数即可。

以二维直角坐标系为例，使用 plot() 函数绘制折线图时，数据用来确定折线图上若干采样点的 x、y 坐标，使用直线段依次连接这些顶点，如果顶点足够密集则可以形成光滑曲线。如果使用 scatter() 函数绘制散点图，数据用来确定若干顶点的 x、y 坐标，然后在这些位置上绘制指定大小和颜色的散点符号。如果使用 bar() 函数绘制柱状图，数据用来确定若干柱的位置（x 坐标）和高度（y 坐标）。

绘制图形并设置外围属性之后可以调用 pyplot 模块的 show() 函数直接显示图形，也可以使用 savefig() 函数或图形对象的同名方法保存为图片文件。savefig() 函数完整用法如下。

```
savefig(fname, *, dpi='figure', format=None, metadata=None,
        bbox_inches=None, pad_inches=0.1, facecolor='auto',
        edgecolor='auto', backend=None, **kwargs)
```

savefig() 函数中参数的含义如表 3-1 所示。

表 3-1　savefig() 函数中参数的含义

参数名称	含　义
fname	要保存的文件名
dpi	图形的分辨率（dots per inch，每英寸多少像素），例如 96、300、600，如果不指定则使用 Python 安装目录下配置文件 Lib\site-packages\matplotlib\mpl-data\matplotlibrc 中 savefig.dpi 的值
facecolor、edgecolor	设置图形的背景色和边框颜色，默认均为白色
format	用来指定保存文件的类型和扩展名，可以设置为 'png'、'pdf'、'ps'、'eps'、'svg' 以及 'jpeg'、'jpg'、'tif'、'tiff' 等其他后端所支持的类型。如果不指定该参数，则根据参数 fname 字符串指定的文件扩展名来确定类型
transparent	如果设置为 True 则子图透明，如果此时没有设置 facecolor 和 edgecolor 则整个图形也透明
bbox_inches	用来指定保存图形的哪一部分，如果设置为 'tight' 则使用能够包围图形的最小边框
pad_inches	用来设置当 bbox_inches='tight' 时图形的内边距
bbox_extra_artists	用来指定当 bbox_inches='tight' 时应考虑保存的额外图形元素

Matplotlib 绘制图形有很多种样式和风格，下面代码列出了所有可用的样式。

```
>>> import matplotlib.pyplot as plt
>>> plt.style.available                    # 查看所有可用的图形样式
['bmh', 'classic', 'dark_background', 'fivethirtyeight', 'ggplot', 'grayscale',
 'seaborn-bright', 'seaborn-colorblind', 'seaborn-dark-palette', 'seaborn-dark',
```

```
 'seaborn-darkgrid', 'seaborn-deep', 'seaborn-muted', 'seaborn-notebook',
 'seaborn-paper', 'seaborn-pastel', 'seaborn-poster', 'seaborn-talk',
 'seaborn-ticks', 'seaborn-white', 'seaborn-whitegrid', 'seaborn']
```

下面的代码演示了如何指定图形样式，图 3-1 和图 3-2 分别演示了默认样式和 fivethirtyeight 两种样式的效果，其他样式可以自行测试。

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

# 指定图形样式
plt.style.use('fivethirtyeight')
x = np.arange(0, 7, 0.01)
y = np.sin(x)
plt.plot(x, y)
plt.show()
```

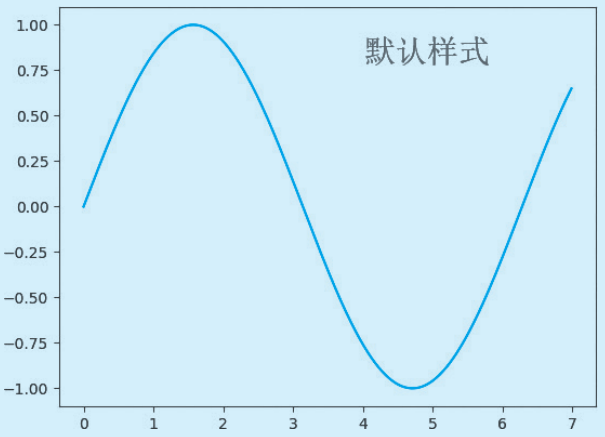

图 3-1　默认样式显示效果

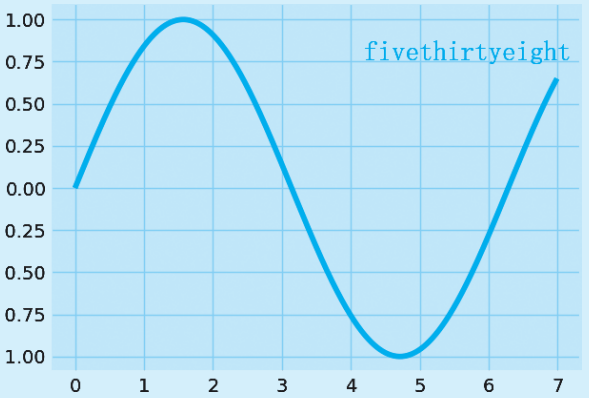

图 3-2　fivethirtyeight 样式显示效果

3.2　绘制折线图

pyplot 模块中的函数 plot() 或者子图对象的同名方法用来绘制折线图，也可以同时或单独绘制采样点，返回包含折线图的列表。完整语法如下。

```
plot(*args, scalex=True, scaley=True, data=None, **kwargs)
```

可能的调用形式如下。

```
plot([x], y, [fmt], *, data=None, **kwargs)
plot([x], y, [fmt], [x2], y2, [fmt2], ..., **kwargs)
```

其中，参数 x、y 用来设置采样点坐标；参数 fmt 用来设置颜色、线型、端点符号，格式为 '[marker][line][color]' 或 '[color][marker][line]'，例如 'ro'、'go-'、'rs'。其他常用的参数还有 color/c、alpha、label、linestyle/ls、linewidth/lw、marker、markeredgecolor/mec、markeredgewidth/mew、markerfacecolor/mfc、markersize、pickradius、snap 等。可使用 help(plt.plot) 查看完整用法和参数含义，其中 marker 和 ls 参数使用较多，ls 参数的值可以为 '-'（表示实心线）、'--'（表示短画线）、'-.'（表示点画线）、':'（表示点线），marker 参数可能的值与含义如表 3-2 所示。

表 3-2　marker 参数取值范围与含义

字　符	含　义	字　符	含　义
.	点	,	像素
o	圆	v	向下的三角形
^	向上的三角形	<	向左的三角形
>	向右的三角形	*	星形
1	向下的三尖形	2	向上的三尖形
3	向左的三尖形	4	向右的三尖形
8	八边形	s	正方形
p	五边形	P	粗加号
h	1 号六边形	H	2 号六边形
+	加号	x	叉号
X	填充的叉号	d	细金刚石
D	金刚石	_	横线
|	竖线		

使用下面的代码可以绘制不同 marker 参数的散点图，显示效果如图 3-3 所示。

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

markers = '.,ov^<>*12348spPhH+xXdD_|'
x, y = np.mgrid[1:10:5j, 1:10:5j]
for x_pos, y_pos, marker in zip(x.flatten(), y.flatten(), markers):
    plt.scatter(x_pos, y_pos, marker=marker, s=100)
    plt.text(x_pos+0.2, y_pos-0.2, s=repr(marker))
plt.axis('off')
plt.show()
```

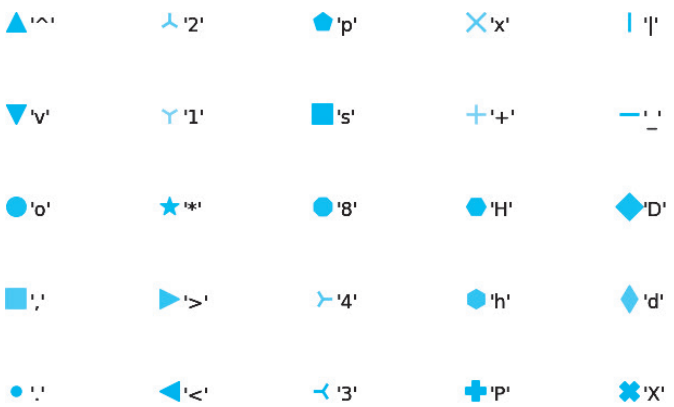

图 3-3　marker 参数不同取值的显示效果

例 3-1　绘制带有中文标题、坐标轴标签和图例的正弦、余弦图像。运行结果如图 3-4 所示。

```
import matplotlib.pylab as pl

t = pl.arange(0.0, 2.0*pl.pi, 0.01)          # 自变量取值范围
s = pl.sin(t)                                # 计算正弦函数值
z = pl.cos(t)                                # 计算余弦函数值
pl.plot(t,                                   # 采样点 x 轴坐标
        s,                                   # 采样点 y 轴坐标
        label='正弦',                        # 标签
        color='red')                         # 红色
pl.plot(t, z, label='余弦',
        lw=3, ls='--', color='blue')         # 3 像素宽，虚线，蓝色
pl.xlabel('x- 变量',                         # 标签文本
          fontproperties='STKAITI',          # 字体
          fontsize=18)                       # 字号
pl.ylabel('y- 正弦余弦函数值', fontproperties='simhei', fontsize=18)
```

视频二维码：例 3-1

```
pl.title('sin-cos 函数图像',                    # 标题文本
         fontproperties='STLITI',               # 字体
         fontsize=24)                           # 字号
pl.legend(prop='STKAITI')          # 创建图例，自动提取图形的线性、颜色、标签等属性
pl.grid(alpha=0.7, ls='-.')                     # 半透明网格线，点画线
pl.show()                                       # 显示绘制的结果图像
```

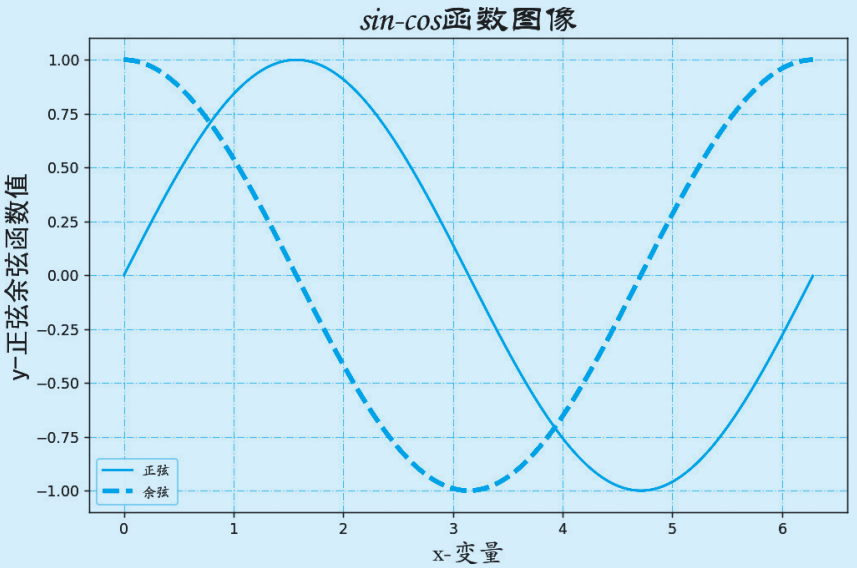

图 3-4　例 3-1 程序运行结果

例 3-2　在绘图结果中添加水平线和垂直线。运行结果如图 3-5 所示。

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

x = np.linspace(0, 2*np.pi, 100)
y = np.sin(x)
plt.plot(x, y, 'r-', lw=2, label='sin')       # 绘制正弦曲线
# 在纵坐标 -0.5 和 0.5 处绘制两条水平直线，蓝色虚线
plt.axhline(-0.5, color='blue', ls='--', label='axhline')
plt.axhline(0.5, color='blue', ls='--')
# 在横坐标绘制垂直直线，绿色点画线
plt.axvline(np.pi, color='green', ls='-.', label='axvline')
# 设置 y 轴刻度位置和文本
plt.yticks([-1, -0.5, 0, 0.5, 1], ['-1', 'axhline', '0', 'axhline', '1'])
plt.legend()
plt.show()
```

视频二维码：例 3-2

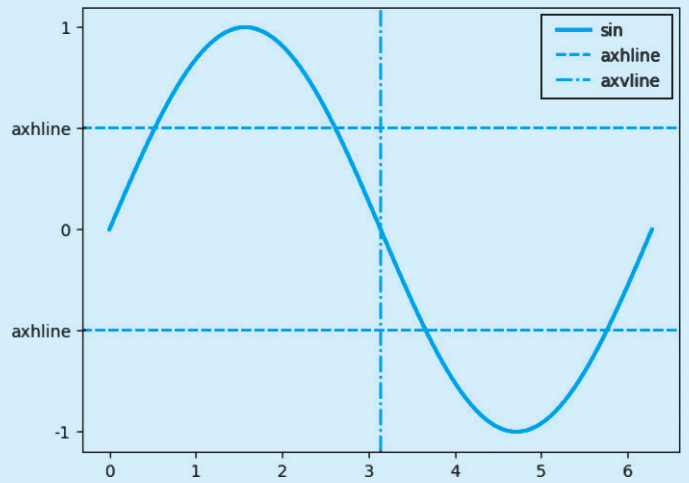

图 3-5　例 3-2 程序运行结果

例 3-3　绘制螺旋线，只绘制采样点。运行结果如图 3-6 所示。

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

theta = np.arange(0, 8*np.pi, 0.1)  # 4 个圆周的角度，单位为弧度
r = np.arange(20, 20+len(theta))
plt.plot(r*np.cos(theta), r*np.sin(theta), 'ro')   # 在采样点位置处绘制红色圆圈
plt.show()
```

视频二维码：例 3-3

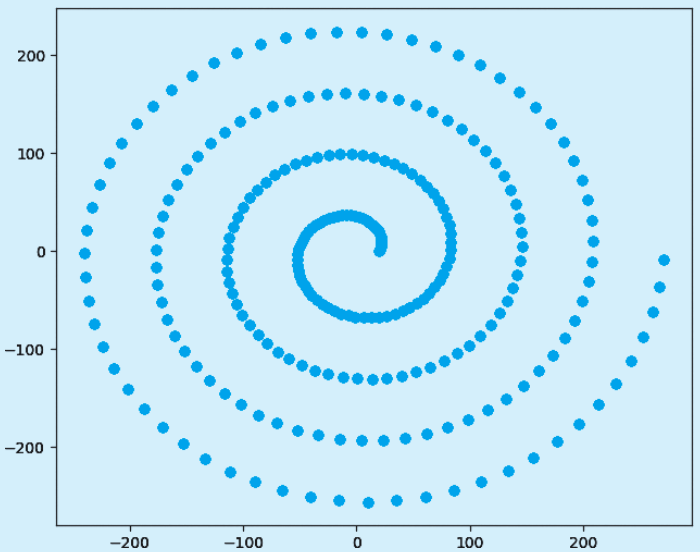

图 3-6　例 3-3 程序运行结果

例 3-4　同时绘制多条折线。运行结果如图 3-7 所示。

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

x = range(10)
y = np.random.randint(20, 50, (10,3))    # 10 行 3 列位于 [20,50) 区间的随机数
plt.plot(x, y, label=['a','b','c'])      # 绘制 3 条折线图，每列数据对应一条折线图
plt.legend()
plt.show()
```

视频二维码：例 3-4

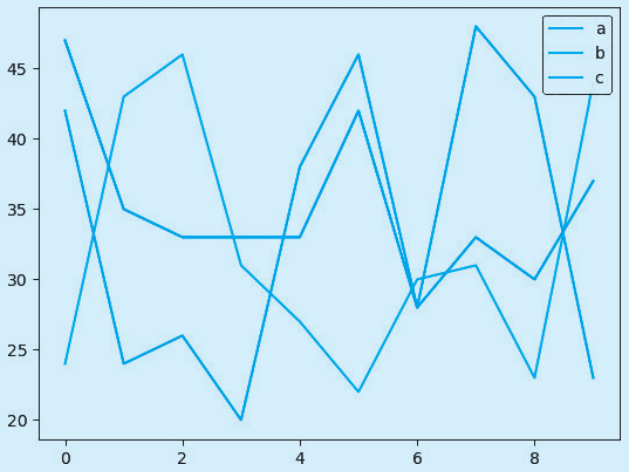

图 3-7　例 3-4 程序运行结果

例 3-5　同时绘制多条曲线。运行结果如图 3-8 所示。

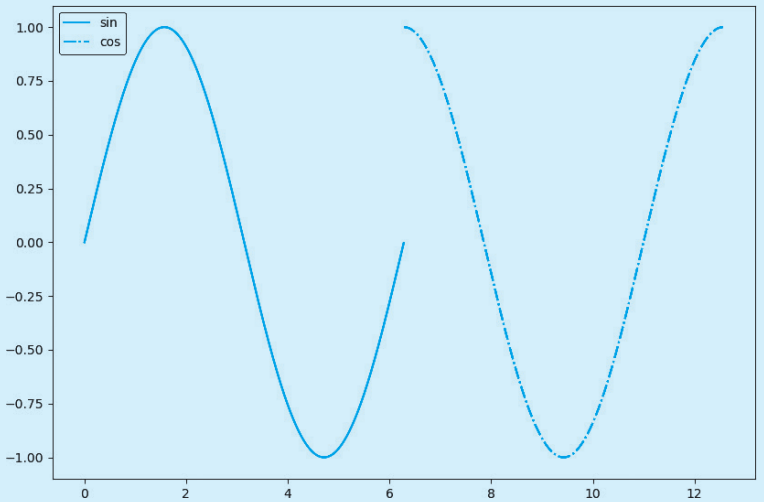

图 3-8　例 3-5 程序运行结果

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

x1 = np.arange(0, 2*np.pi, 0.01)
y1 = np.sin(x1)
x2 = np.arange(2*np.pi, 4*np.pi, 0.01)
y2 = np.cos(x2)
# 绘制折线图，指定每条曲线的采样点位置和线条属性
lines = plt.plot(x1, y1, 'r-', x2, y2, 'b-.')
plt.legend(lines, ['sin','cos'])      # 为两条曲线创建图例
plt.show()
```

视频二维码：例 3-5

例 3-6　绘制龟兔赛跑中兔子和乌龟的行走轨迹。运行结果如图 3-9 所示。

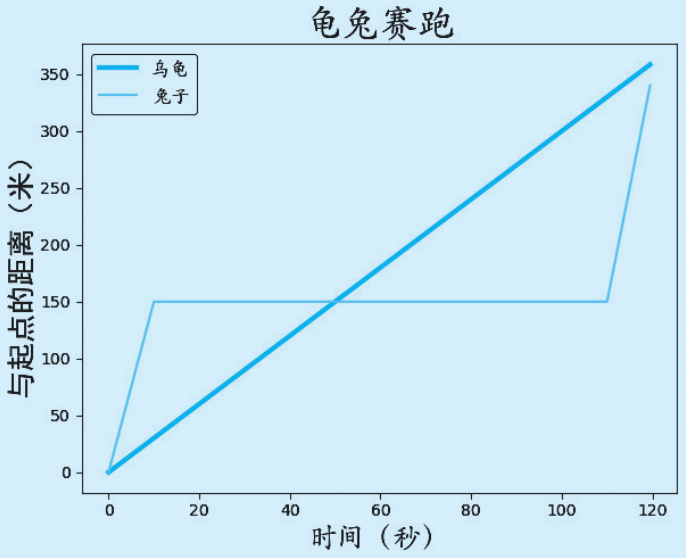

图 3-9　例 3-6 程序运行结果

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

t = np.arange(0, 120, 0.5)                          # 时间轴
rabbit = np.piecewise(t,                            # 兔子的运行轨迹，分段函数
                      [t<10, t>110],                # 兔子跑步的两个时间段
                      [lambda x:15*x,               # 兔子第一段时间的路程
                       lambda x:20*(x-110)+150,     # 第二个时间段的路程
                       lambda x:150])               # 兔子中间睡觉时的路程
tortoise = 3 * t                                    # 小乌龟一直在匀速前进
```

视频二维码：例 3-6

```
plt.plot(t, tortoise, label='乌龟', lw=3)
plt.plot(t, rabbit, label='兔子')
plt.title('龟兔赛跑', fontproperties='STKAITI', fontsize=24)
plt.xlabel('时间（秒）', fontproperties='STKAITI', fontsize=18)
plt.ylabel('与起点的距离（米）', fontproperties='simhei', fontsize=18)
myfont = fm.FontProperties(fname=r'C:\Windows\Fonts\STKAITI.ttf', size=12)
plt.legend(prop=myfont)                                    # 设置图例中的中文字体和字号
plt.show()
```

例 3-7　一笔绘制红色五角星。运行结果如图 3-10 所示。

代码一：

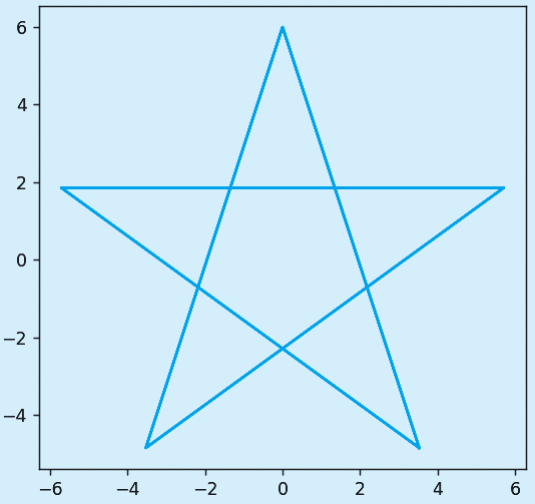

图 3-10　例 3-7 程序运行结果

视频二维码：例 3-7

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

r = 6                                          # 外接圆半径
angles = np.linspace(0, 2*np.pi, 5, endpoint=False)
x = r * np.sin(angles)
y = r * np.cos(angles)
plt.plot([x[2],x[0],x[3],x[1],x[4],x[2]],
         [y[2],y[0],y[3],y[1],y[4],y[2]], 'r')
plt.gca().set_aspect('equal')          # 设置坐标轴纵横比相等
plt.show()
```

代码二：

```
import numpy as np
```

```
import matplotlib.pyplot as plt

r = 6
angles = np.linspace(0, 4*np.pi, 6)
x = r * np.sin(angles)
y = r * np.cos(angles)
plt.plot(x, y, 'r')
plt.gca().set_aspect('equal')
plt.show()
```

例 3-8　使用三角函数绘制花瓣图案。运行结果如图 3-11 所示。

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

r = 6
angles = np.arange(0, np.pi*2, 0.01)
x = r * np.cos(4*angles) * np.cos(angles)      # 把 4 改成其他数字可以得到不同图案
y = r * np.cos(4*angles) * np.sin(angles)
plt.plot(x, y, 'r')
plt.gca().set_aspect('equal')
plt.show()
```

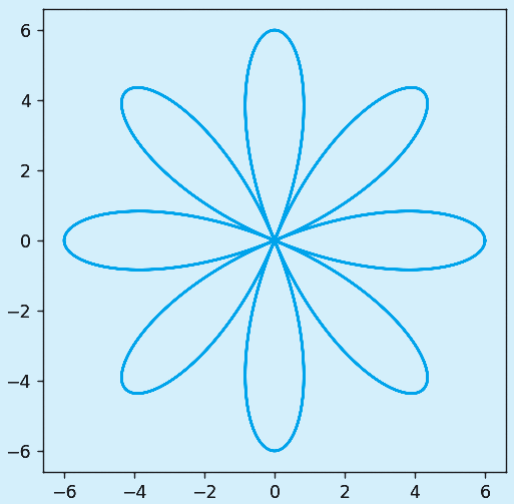

图 3-11　例 3-8 程序运行结果

例 3-9　某质点的初始速度和加速度已知，绘制该质点第 5~20s 速度和位移的曲线。运行结果如图 3-12 所示。

```
import numpy as np
```

```
import matplotlib.pyplot as plt

v0, a = 3, 1.8                                  # 初始速度和加速度
t = np.arange(5, 21)                            # 时间轴，第 5~20s
v = v0 + a*t                                    # 速度
x = v0*t + 0.5*a*t*t                            # 位移
fig, (ax1, ax2) = plt.subplots(1, 2)            # 创建左、右两个子图
# 设置子图之间的水平间距，wspace 值为子图平均宽度的比例
plt.subplots_adjust(wspace=0.5)
plt.sca(ax1)                                    # 选择左边子图为当前子图
plt.plot(t, v, c='red')                         # 在当前子图中绘制折线图
plt.title('时间 - 速度', fontproperties='STKAITI', fontsize=24)
plt.xlabel('时间（s）', fontproperties='STKAITI', fontsize=18)
plt.ylabel('速度（m/s）', fontproperties='STKAITI', fontsize=18)
plt.xlim(5, 21)                                 # 设置坐标轴刻度范围
plt.ylim(0, 40)

plt.sca(ax2)                                    # 选择右边子图为当前子图
plt.plot(t, x, c='blue')                        # 在当前子图中绘制折线图
plt.title('时间 - 位移', fontproperties='STKAITI', fontsize=24)
plt.xlabel('时间（s）', fontproperties='STKAITI', fontsize=18)
plt.ylabel('位移（m）', fontproperties='STKAITI', fontsize=18)
plt.xlim(5, 21)
plt.ylim(0, 450)
plt.show()
```

视频二维码：例 3-9

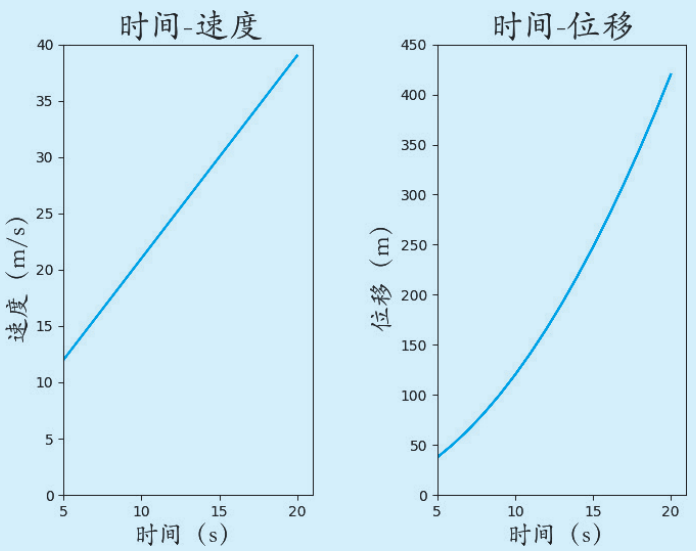

图 3-12　例 3-9 程序运行结果

例 3-10　绘制误差线图。运行结果如图 3-13 所示。

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

x, y = [1, 3, 5, 8, 9], [5, 9, 3, 5, 10]
plt.errorbar(x, y,                                  # 数据点位置
             xerr=1, yerr=[1,1,1,0.5,0.5],          # 两个方向的误差范围
                      # 设置线条和端点符号
                      # fmt='none' 时表示不绘制数据点及连线，只绘制误差标记
             fmt='-.*',
             ecolor='orange',                       # 误差线颜色
             errorevery=2,                          # 每 2 个数据点绘制一个误差线
             lolims=True,                           # 只绘制上侧的误差线
             xlolims=True)                          # 只绘制右侧的误差线
plt.show()
```

视频二维码：例 3-10

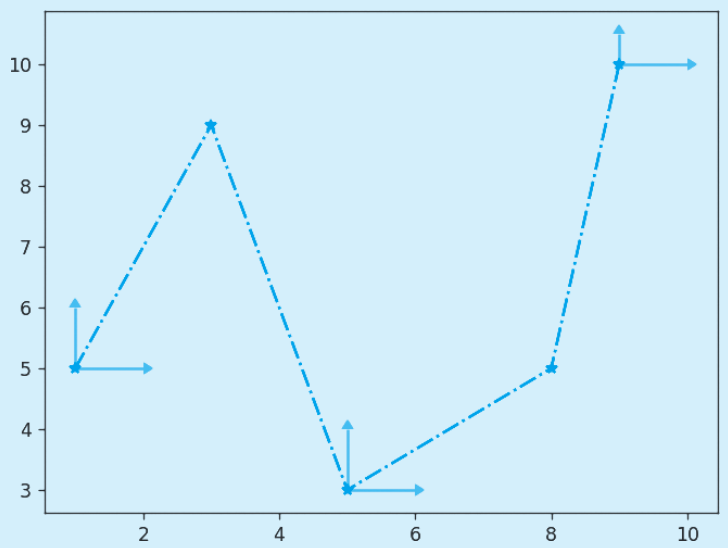

图 3-13　例 3-10 程序运行结果

例 3-11　已知某学校附近一个烧烤店 2022 年每个月的营业额如表 3-3 所示。编写程序绘制折线图对该烧烤店全年营业额进行可视化，使用红色点画线连接每个月的数据，并在每个月的数据处使用三角形进行标记。运行结果如图 3-14 所示。

视频二维码：例 3-11

表 3-3　烧烤店营业额

月份	1	2	3	4	5	6	7	8	9	10	11	12
营业额 / 万元	5.2	2.7	5.8	5.7	7.3	9.2	18.7	15.6	20.5	18.0	7.8	6.9

```
import matplotlib.pyplot as plt

month = range(1, 13)
money = [5.2, 2.7, 5.8, 5.7, 7.3, 9.2, 18.7, 15.6, 20.5, 18.0, 7.8, 6.9]
# 参数 mfc（marker face color）设置散点符号内部颜色
# 参数 mec（marker edge color）设置散点符号边线颜色
plt.plot(month, money, 'r-.v', mfc='b', mec='y')
plt.xlabel('月份', fontproperties='simhei', fontsize=14)
plt.ylabel('营业额（万元）', fontproperties='simhei', fontsize=14)
plt.title('烧烤店 2022 年营业额变化趋势图', fontproperties='simhei',
          fontsize=18)
plt.tight_layout()                          # 紧缩四周空白，扩大绘图区域可用面积
plt.show()
```

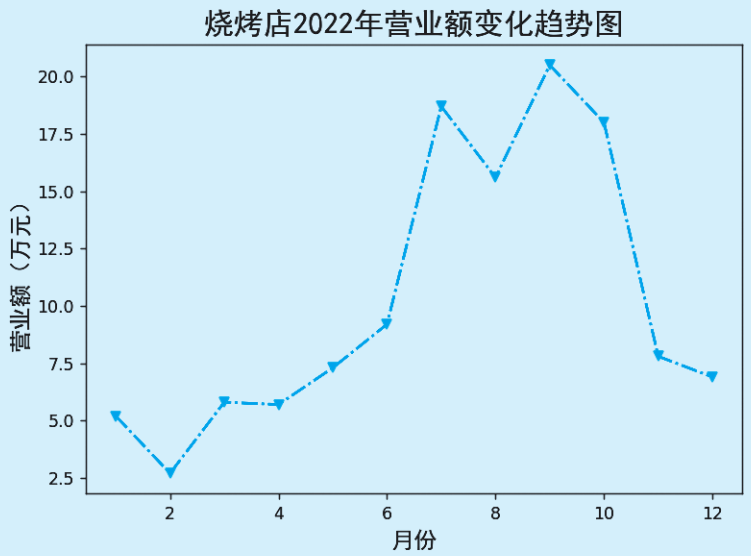

图 3-14　例 3-11 程序运行结果

例 3-12　绘制折线图模拟连续信号与数字信号。运行结果如图 3-15 所示。

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

t = np.arange(0, 6*np.pi, 0.05)
# 连续信号与数字信号的函数值
t_sin = np.sin(t)
t_digital1 = np.piecewise(t_sin, [t_sin>0, t_sin<0], [1,-1])
t_digital2 = np.round_(t_sin)
# 在标签字符串首尾加 $ 符号可以调用 Latex 引擎渲染为公式
plt.plot(t, t_sin, label='$sin(x)$', color='red', lw=1)
```

视频二维码：例 3-12

```
plt.plot(t, t_digital1, 'b--', label='digital1')
plt.plot(t, t_digital2, 'g-.', label='digital2')
plt.ylim(-2.0, 2.0)
plt.legend()
plt.show()
```

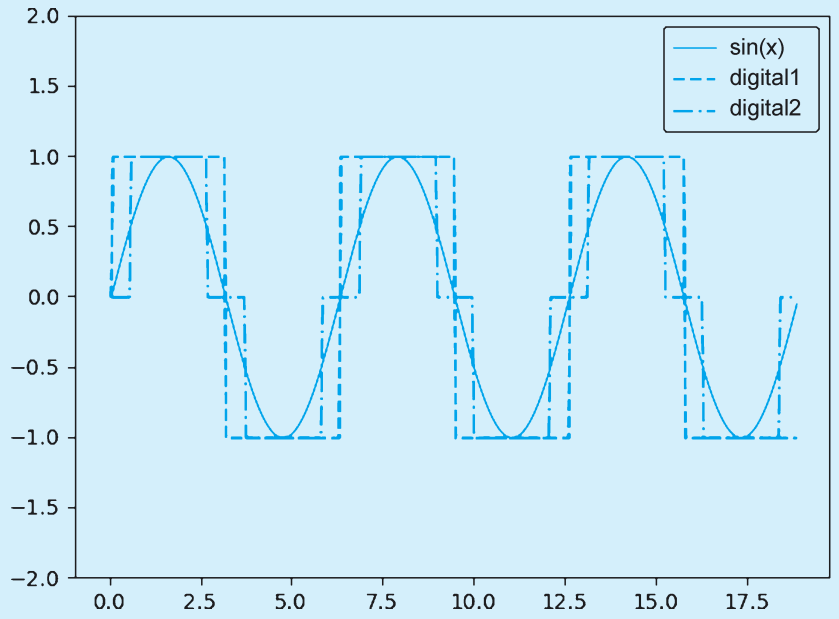

图 3-15　例 3-12 程序运行结果

例 3-13　绘制尼哥米德蚌线。

给定一条定直线 m 和直线外一个定点 O。定点与定直线的距离为 a。过定点 O 作一条直线 n 与定直线 m 交于点 P。在直线 n 上点 P 的两侧分别取到点 P 的距离为 b 的点 Q 和点 Q'。那么，点 P 在直线 m 上运动时，点 Q 和 Q' 的运动轨迹合在一起就叫作尼哥米德蚌线（或尼科梅德斯蚌线）。图 3-16 演示了尼哥米德蚌线的生成原理，分 3 种可能的情况：① a>b 时，蚌线的两支都不经过点 O，如图 3-17 左侧图形所示；② a=b 时，蚌线有一支有一个尖点经过点 O，如图 3-17 中间图形所示；③ a<b 时，蚌线有一支经过点 O 且在 O 处有一个小绕环，如图 3-17 右侧图形所示。

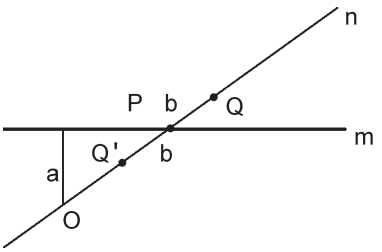

图 3-16　尼哥米德蚌线生成原理

视频二维码：例 3-13

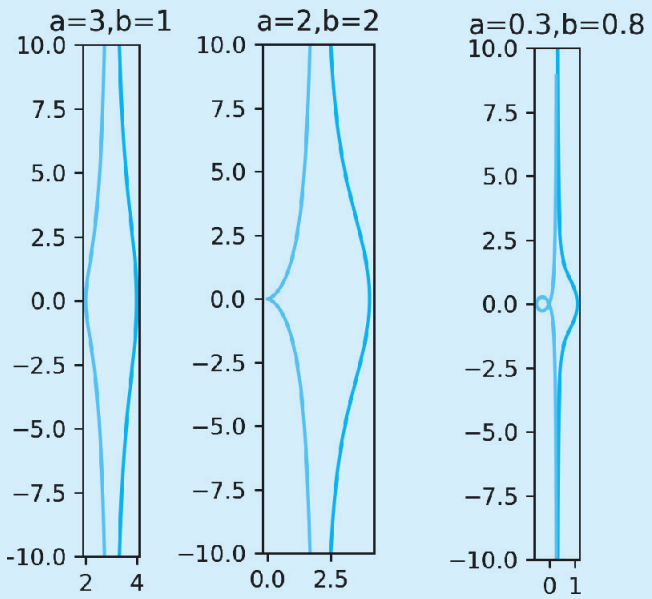

图 3-17　例 3-13 程序运行结果

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

def draw(a, b):
    plt.figure(figsize=(10, 200), dpi=240)
    t = np.arange(-1.55, 1.55, 0.01)
    x1 = a + b*np.cos(t)
    y1 = a*np.tan(t) + b*np.sin(t)
    x2 = a - b*np.cos(t)
    y2 = a*np.tan(t) - b*np.sin(t)
    plt.plot(x1, y1, x2, y2)
    plt.title(f'{a=},{b=}')
    plt.ylim(-10, 10)
    plt.gca().set_aspect('equal')
    plt.savefig('{},{}.jpg'.format(a,b))

draw(3, 1)
draw(2, 2)
draw(0.3, 0.8)
```

例 3-14　在第一象限中，任意反比例函数 xy=k 与任意矩形 OABC 的两个交点的连线始终与矩形对角线平行，编写程序验证这一点。运行结果如图 3-18 所示。

视频二维码：例 3-14

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

k = 1                                   # 反比例函数 xy=k 的常数 k
m, n = 6, 3                             # 矩形右上角坐标 (m,n)
x = np.arange(0.1, m+0.5, 0.02)         # 第一象限中反比例函数曲线上顶点的 x 坐标
y = k / x                               # 根据反比例函数 xy=k 计算顶点 y 坐标
plt.plot(x, y, 'b')                     # 绘制第一象限指定区间内的反比例函数图像
# 绘制矩形，从左下角出发，向右、上、左、下
plt.plot([0,m,m,0,0], [0,0,n,n,0], 'r')
plt.plot([0,m], [n,0], 'g')                             # 矩形对角线
plt.plot([k/n,m], [n,k/m], 'g')                         # 矩形与反比例函数的交点连线

for x, y, ch in zip([0,m,m,0,k/n,m], [0,0,n,n,n,k/m], 'OABCDE'):
    plt.text(x, y+0.02, ch)                             # 绘制顶点与交点的符号
plt.xlim(-0.1, m+1)                                     # 设置坐标轴跨度
plt.ylim(-0.1, n+1)
plt.title(f'k={k},m={m},n={n}', fontsize=20)             # 设置图形标题
plt.gca().set_aspect(True)                              # 设置图形纵横比相等
plt.show()
```

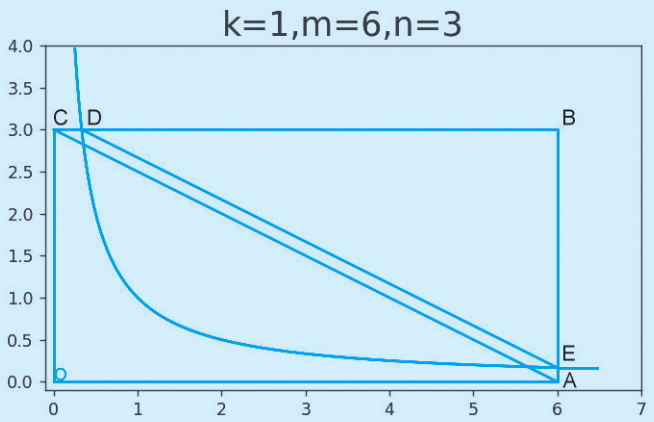

图 3-18　例 3-14 程序运行结果

例 3-15　绘制函数曲线，计算并标记极值。运行结果如图 3-19 所示。

代码一：

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

start, end = 0, 10                      # 函数自变量取值范围
x = np.arange(start, end, 0.01)         # 计算所有采样点的 x 坐标、y 坐标
```

视频二维码：例 3-15

```
y = 3*np.sin(x) + 5*np.cos(3*x)
s, = plt.plot(x, y, 'r-')                              # 绘制折线图，保存绘制结果
# 设置子区间长度，在每个子区间（不包含端点）内寻找极值
# 调整区间大小时会影响极值数量，应使得每个子区间内都包含波峰和波谷
span = 66

for start in range(0, len(y), span):
    sectionY = y[start:start+span]                     # 每个子区间的自变量与函数值
    sectionX = x[start:start+span]
    localMax = sectionY.max()                          # 局部最大值和局部最小值
    localMin = sectionY.min()

    # 方案一：
    argsort_result = sectionY.argsort()                # 按值大小升序排序的索引
    # 区间内所有最大值的索引和所有最小值的索引
    args_max = argsort_result[-len(sectionY[sectionY==localMax]):]
    args_min = argsort_result[:len(sectionY[sectionY==localMin])]
    args_max = list(set(args_max)-{0,span-1})          # 去除子区间端点
    if args_max:
        s1 = plt.scatter(sectionX[args_max], sectionY[args_max],
                         marker='*', c='b')
    args_min = list(set(args_min)-{0,span-1})
    if args_min:
        s2 = plt.scatter(sectionX[args_min], sectionY[args_min],
                         marker='*', c='g')
    # 方案二：
##    for index, yy in enumerate(sectionY):
##        if yy==localMax and index not in (0, span-1):
##            # 在极大值处绘制一个蓝色五角星
##            s1 = plt.scatter(sectionX[index], yy, marker='*', c='b')
##        elif yy==localMin and index not in (0, span-1):
##            # 在极小值处绘制一个绿色五角星
##            s2 = plt.scatter(sectionX[index], yy, marker='*', c='g')
plt.legend([s,s1,s2], ['curve','local max','local min'])
plt.show()
```

代码二：

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

start, end = 0, 10
x = np.arange(start, end, 0.01)
```

```
y = 3*np.sin(x) + 5*np.cos(3*x)
s, = plt.plot(x, y, 'r-')

for index, yy in enumerate(y):
    if index == 0:
        if yy > y[1]:
            s1 = plt.scatter(x[index], yy, marker='*', c='b')
        elif yy < y[1]:
            s2 = plt.scatter(x[index], yy, marker='*', c='g')
    elif index == len(y)-1:
        if yy > y[index-1]:
            s1 = plt.scatter(x[index], yy, marker='*', c='b')
        elif yy < y[index-1]:
            s2 = plt.scatter(x[index], yy, marker='*', c='g')
    elif yy>=y[index-1] and yy>=y[index+1]:
        s1 = plt.scatter(x[index], yy, marker='*', c='b')
    elif yy<=y[index-1] and yy<=y[index+1]:
        s2 = plt.scatter(x[index], yy, marker='*', c='g')

plt.legend([s,s1,s2], ['curve','local max','local min'])
plt.show()
```

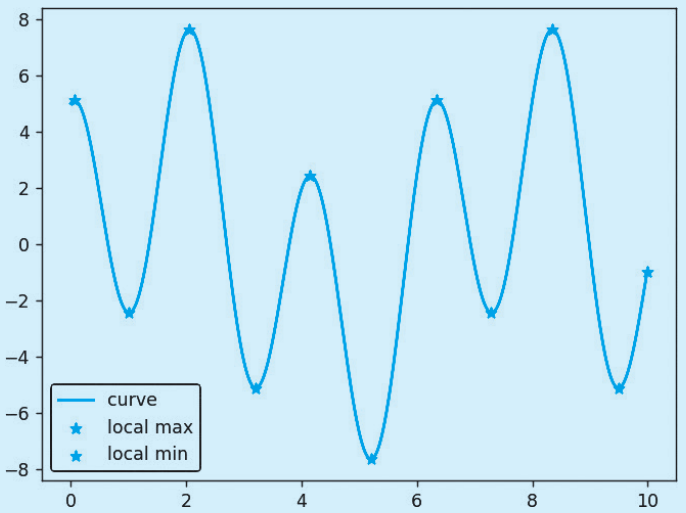

图 3-19　例 3-15 程序运行结果

例 3-16　使用折线图可视化角谷猜想（给定任意正整数，如果是偶数就除以 2，如果是奇数就乘以 3 再加 1，最终总能得到 1）中正整数变为 1 的过程。运行结果如图 3-20 所示。

视频二维码：例 3-16