

**二、判断题**

1. 数据可视化是一个抽象的过程。( )
2. 散点图可以清晰地展示数据增减的趋势、速率、规律、峰值等特征。( )
3. 柱形图与直方图展示的效果完全相同。( )
4. matplotlib 只能采用面向对象的方式开发程序。( )

**三、选择题**

1. 下列选项中，关于数据可视化描述错误的是( )。
  - A. 数据可视化可以简单地理解为将不易描述的事物形成可感知画面的过程
  - B. 数据可视化的目的是准确、高效、全面地传递信息
  - C. 数据表格是数据可视化最基础的应用
  - D. 数据可视化对后期数据挖掘具有深远的影响
2. 关于常见的图表，下列描述正确的是( )。
  - A. 柱形图可以反映数据增减的趋势
  - B. 条形图是横置的直方图
  - C. 饼图用于显示数据中各项大小与各项总和的比例
  - D. 雷达图是一种可以展示多变量关系的图表
3. 下列图表中，可以反映 3 个变量之间关系的是( )。
  - A. 折线图
  - B. 柱形图
  - C. 散点图
  - D. 气泡图
4. 下列哪个可视化库可以生成 ECharts 图表？( )
  - A. matplotlib
  - B. seaborn
  - C. bokeh
  - D. pyecharts
5. 下列选项中，属于数据之间逻辑关系的是( )。(多选)
  - A. 比较
  - B. 分布
  - C. 构成
  - D. 联系

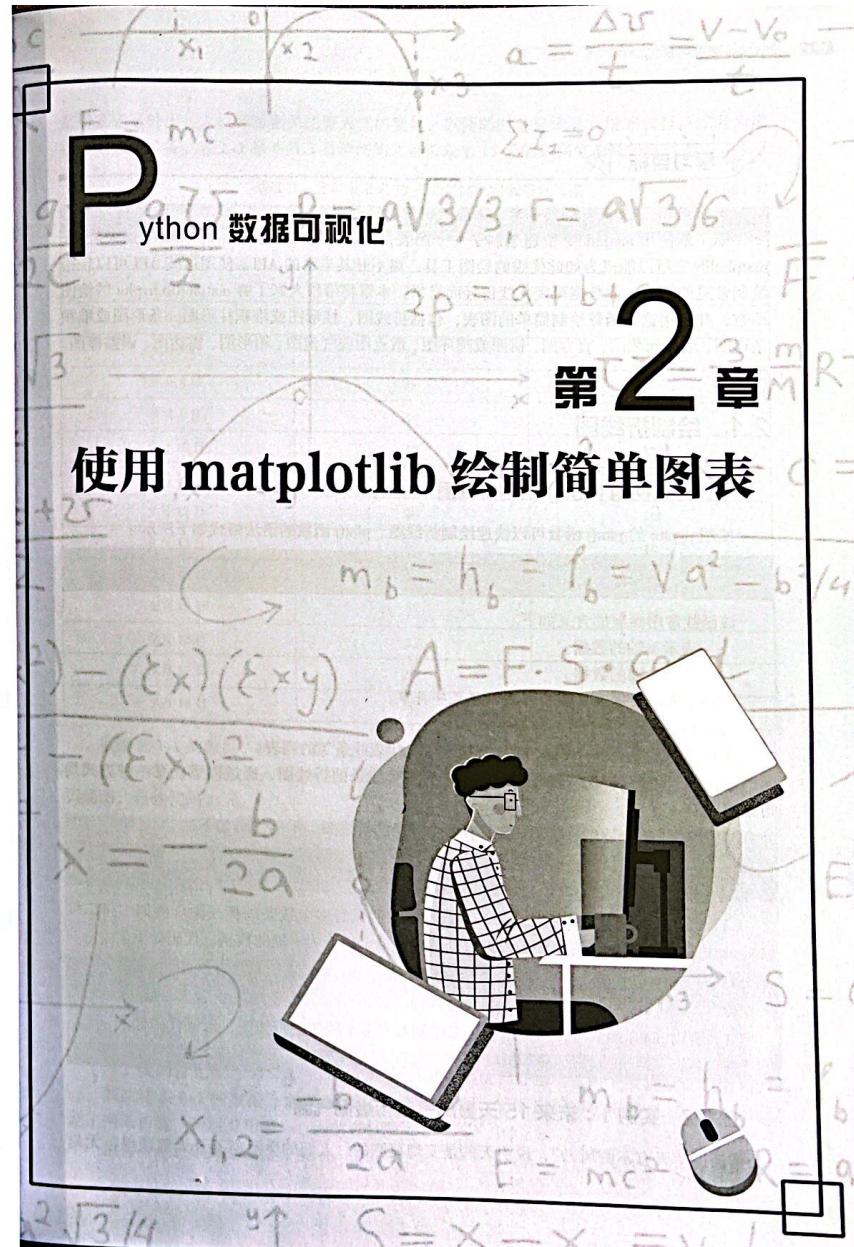
**四、简答题**

1. 请简述数据可视化的概念。
2. 请列举 3 个常见的数据可视化图表及其特点。
3. 请简述 pyplot API 和 object-oriented API 的基本用法。

**五、编程题**

编写程序，分别采用面向对象和面向函数两种方式绘制正弦曲线和余弦曲线。

提示：利用 numpy 的 linspace()、sin() 或 cos() 函数生成样本数据、正弦值或余弦值。



### 学习目标

★掌握 matplotlib 的绘图函数，并能绘制简单的图表

第 1 章使用 matplotlib 快速绘制了一个图表，让读者真切体会到 matplotlib 的强大之处。matplotlib之所以能成为如此优秀的绘图工具，离不开其丰富的 API，使用这些 API 可以轻松绘制常见的图表，使数据可视化变得轻而易举。本章将带领大家了解 matplotlib.pyplot 的绘图函数，并使用这些函数绘制简单的图表，包括折线图、柱形图或堆积柱形图、条形图或堆积条形图、堆积面积图、直方图、饼图或圆环图、散点图或气泡图、箱形图、雷达图、误差棒图。

## 2.1 绘制折线图

### 2.1.1 使用 plot() 绘制折线图

使用 pyplot 的 plot() 函数可以快速绘制折线图。plot() 函数的语法格式如下所示：

```
plot(x, y, fmt, scalex=True, scaley=True, data=None, label=None,
     *args, **kwargs)
```

该函数常用参数的含义如下。

- x：表示 x 轴的数据。
- y：表示 y 轴的数据。
- fmt：表示快速设置线条样式的格式字符串。
- label：表示应用于图例的标签文本。

plot() 函数会返回一个包含 Line2D 类对象（代表线条）的列表。

使用 pyplot 的 plot() 函数还可以绘制具有多个线条的折线图，通过以下任意一种方式均可以完成。

(1) 多次调用 plot() 函数来绘制具有多个线条的折线图，示例代码如下：

```
plt.plot(x1, y1)
plt.plot(x2, y2)
```

(2) 调用 plot() 函数时传入一个二维数组来绘制具有多个线条的折线图。例如，将二维数组 arr 的第一行数据作为 x 轴的数据、其他行数据全部作为 y 轴的数据，代码如下。

```
arr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12]])
plt.plot(arr[0], arr[1:])
```

(3) 调用 plot() 函数时传入多组数据来绘制具有多个线条的折线图，示例代码如下：

```
plt.plot(x1, y1, x2, y2)
```

### 2.1.2 实例 1：未来 15 天最高气温和最低气温

俗话说“天有不测风云”，说明天气是变幻莫测的。人们的生活离不开天气预报，无论

是居家还是外出，人们都时刻关注着天气的变化，以便随时备好伞具、增减衣服，或者为出行计划做好准备。表 2-1 是 9 月 3 日预测的北京市未来 15 天的最高气温和最低气温。

表 2-1 北京市未来 15 天的最高气温和最低气温

单位：℃

日期	最高气温	最低气温
9月4日	32	19
9月5日	33	19
9月6日	34	20
9月7日	34	22
9月8日	33	22
9月9日	31	21
9月10日	30	22
9月11日	29	16
9月12日	30	18
9月13日	29	18
9月14日	26	17
9月15日	23	14
9月16日	21	15
9月17日	25	16
9月18日	31	16

根据表 2-1 的数据，将“日期”这一列的数据作为 x 轴的数据，将“最高气温”和“最低气温”两列的数据作为 y 轴的数据，使用 plot() 函数绘制反映最高气温和最低气温趋势的折线图，具体代码如下。

```
In [1]:
# 01_maximum_minimum_temperatures
import matplotlib.pyplot as plt
import numpy as np
x = np.arange(4, 19)
y_max = np.array([32, 33, 34, 34, 33, 31, 30, 29, 30, 29, 26, 23, 21, 25, 31])
y_min = np.array([19, 19, 20, 22, 22, 21, 22, 16, 18, 18, 17, 14, 15, 16, 16])
# 绘制折线图
plt.plot(x, y_max)
plt.plot(x, y_min)
plt.show()
```

以上代码首先导入了 matplotlib.pyplot 和 numpy 模块，分别将这两个模块重命名为 plt 和 np，其次将表 2-1 的数据分别作为 x 轴和 y 轴的数据，然后连续两次调用 plot() 函数分别绘制了两条折线，最后调用 show() 函数进行展示。

运行程序，效果如图 2-1 所示。

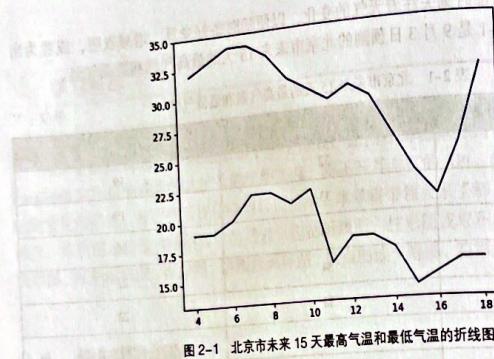


图 2-1 北京市未来 15 天最高气温和最低气温的折线图

图 2-1 中， $x$  轴代表日期， $y$  轴代表温度，位于上方的蓝色折线和下方的橙色折线分别代表最高温度和最低温度。由图 2-1 可知，北京市未来 15 天的最高气温和最低气温都呈现逐步下降后反弹的趋势。

## 2.2 绘制柱形图或堆积柱形图

### 2.2.1 使用 bar() 绘制柱形图或堆积柱形图

使用 pyplot 的 `bar()` 函数可以快速绘制柱形图或堆积柱形图。`bar()` 函数的语法格式如下所示：

```
bar(x, height, width=0.8, bottom=None, align='center',
     data=None, tick_label=None, xerr=None, yerr=None,
     error_kw=None, **kwargs)
```

该函数常用参数的含义如下。

- `x`：表示柱形的  $x$  坐标值。
- `height`：表示柱形的高度。
- `width`：表示柱形的宽度，默认为 0.8。
- `bottom`：表示柱形底部的  $y$  坐标值，默认为 0。
- `align`：表示柱形的对齐方式，有 '`center`' 和 '`edge`' 两个取值，其中 '`center`' 表示将柱形与刻度线居中对齐；'`edge`' 表示将柱形的左边与刻度线对齐。
- `tick_label`：表示柱形对应的刻度标签。
- `xerr, yerr`：若未设为 `None`，则需要为柱形图添加水平 / 垂直误差棒。
- `error_kw`：表示误差棒的属性字典，字典的键对应 `errorbar()` 函数（2.10 节会介绍）的关键字参数。

`bar()` 函数会返回一个 `BarContainer` 类的对象。`BarContainer` 类的对象是一个包含矩形或误

差棒的容器，它亦可以视为一个元组，可以遍历获取每个矩形条或误差棒。此外，`BarContainer` 类的对象也可以访问 `patches` 或 `errorbar` 属性，从而获取图表中所有的矩形条或误差棒。

例如，使用 `bar()` 函数绘制柱形图，代码如下。

```
In [2]:
import numpy as np
import matplotlib.pyplot as plt
x = np.arange(5)
y1 = np.array([10, 8, 7, 11, 13])
# 柱形的宽度
bar_width = 0.3
# 绘制柱形图
plt.bar(x, y1, tick_label=['a', 'b', 'c', 'd', 'e'], width=bar_width)
plt.show()
```

运行程序，效果如图 2-2 所示。

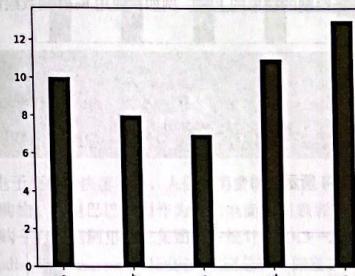


图 2-2 柱形图示例

使用 pyplot 的 `bar()` 函数还可以绘制具有多组柱形的柱形图。例如，使用 `bar()` 函数绘制一个具有两组柱形的柱形图，代码如下。

```
In [3]:
x = np.arange(5)
y1 = np.array([10, 8, 7, 11, 13])
y2 = np.array([9, 6, 5, 10, 12])
# 柱形的宽度
bar_width = 0.3
# 根据多组数据绘制柱形图
plt.bar(x, y1, tick_label=['a', 'b', 'c', 'd', 'e'], width=bar_width)
plt.bar(x+bar_width, y2, width=bar_width)
plt.show()
```

运行程序，效果如图 2-3 所示。

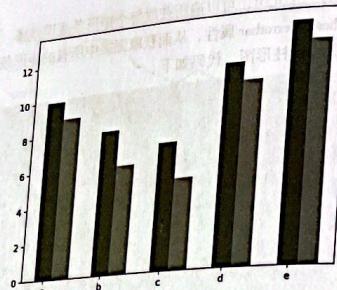


图 2-3 多组柱形的柱形图示例

在使用 pyplot 的 bar() 函数绘制图表时，可以通过给 bottom 参数传值的方式控制柱形的 y 值，使后绘制的柱形位于先绘制的柱形的上方。例如，使用 bar() 函数绘制由两组柱形堆叠而成的堆积柱形图，代码如下。

```
In [4]:
# 绘制堆积柱形图
plt.bar(x, y1, tick_label=['a', 'b', 'c', 'd', 'e'], width=bar_width)
plt.bar(x, y2, bottom=y1, width=bar_width)
plt.show()
```

运行程序，效果如图 2-4 所示。

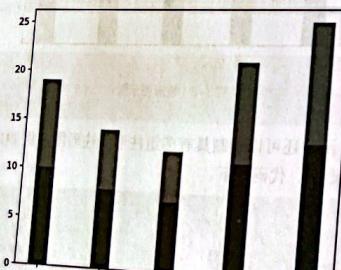


图 2-4 堆积柱形图示例

此外，在使用 pyplot 的 bar() 函数绘制图表时可以通过给 xerr、yerr 参数传值的方式为柱形添加误差棒，示例代码如下：

```
In [5]:
# 偏差数据
```

```
error = [2, 1, 2.5, 2, 1.5]
# 绘制带有误差棒的柱形图
plt.bar(x, y1, tick_label=['a', 'b', 'c', 'd', 'e'], width=bar_width)
plt.bar(x, y1, bottom=y1, width=bar_width, yerr=error)
plt.show()
```

运行程序，效果如图 2-5 所示。

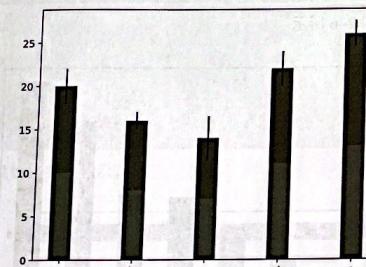


图 2-5 带有误差棒的柱形图示例

## 2.2.2 实例 2：2013—2019 财年阿里巴巴淘宝和天猫平台的 GMV

随着互联网与电子商务的快速发展，人们的消费模式发生了翻天覆地的变化，越来越多的消费者选择网络购物。阿里巴巴集团作为中国电商的引领者，其旗下的淘宝和天猫是深受消费者欢迎的网购平台。据阿里巴巴集团财务统计，2013—2019 财年淘宝和天猫平台的 GMV(Gross Merchandise Volume，一定时间内的成交总额) 如表 2-2 所示。

表 2-2 2013—2019 财年淘宝和天猫平台的 GMV 单位：亿元

财年	GMV
FY2013	10770
FY 2014	16780
FY 2015	24440
FY 2016	30920
FY 2017	37670
FY 2018	48200
FY 2019	57270

根据表 2-2 的数据，将“财年”这一列的数据作为 x 轴的刻度标签，将 GMV 这一列的数据作为 y 轴的数据，使用 bar() 函数绘制各年份对应的 GMV 的柱形图，具体代码如下。

```
In [6]:
# 02_taobao_and_tianmao_GMV
```

```

import matplotlib.pyplot as plt
import numpy as np
x = np.arange(1, 8)
y = np.array([10770, 16780, 24440, 30920, 37670, 48200, 57270])
# 绘制柱形图
plt.bar(x, y, tick_label=["FY2013", "FY2014", "FY2015",
                           "FY2016", "FY2017", "FY2018", "FY2019"], width=0.5)
plt.show()

```

运行程序，效果如图 2-6 所示。

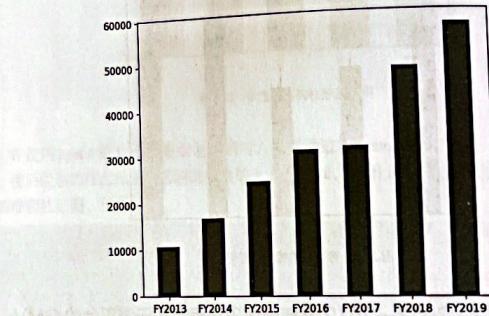


图 2-6 2013—2019 财年淘宝和天猫平台的 GMV 的柱形图

图 2-6 中，*x* 轴代表财年，*y* 轴代表成交总额。由图 2-6 可知，2013—2019 财年的成交总额逐年增加。

## 2.3 绘制条形图或堆积条形图

### 2.3.1 使用 barh() 绘制条形图或堆积条形图

使用 pyplot 的 barh() 函数可以快速绘制条形图或堆积条形图，barh() 函数的语法格式如下所示：

```
barh(y, width=0.8, left=None, align='center', *,  
     **kwargs)
```

该函数常用参数的含义如下。

- *y*：表示条形的 *y* 坐标值。
- *width*：表示条形的宽度，默认值为 0.8。
- *height*：表示条形的高度。
- *left*：条形左侧的 *x* 坐标，默认为 0。

· align：表示条形的对齐方式，有 'center' 和 'edge' 两个取值，其中 'center' 表示将条形与刻度线居中对齐；'edge' 表示将条形的底边与刻度线对齐。

barh() 函数会返回一个 BarContainer 类的对象。

例如，使用 barh() 函数绘制条形图，代码如下。

```

In [7]:  
import numpy as np  
import matplotlib.pyplot as plt  
y = np.arange(5)  
x1 = np.array([10, 8, 7, 11, 13])  
# 条形的高度  
bar_height = 0.3  
# 绘制条形图  
plt.barh(y, x1, tick_label=['a', 'b', 'c', 'd', 'e'], height=bar_height)  
plt.show()

```

运行程序，效果如图 2-7 所示。

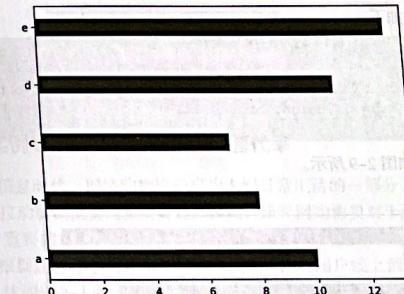


图 2-7 条形图示例

使用 pyplot 的 barh() 函数还可以绘制具有多组条形的条形图。例如，使用 barh() 函数绘制具有两组条形的条形图，代码如下。

```

In [8]:  
y = np.arange(5)  
x1 = np.array([10, 8, 7, 11, 13])  
x2 = np.array([9, 6, 5, 10, 12])  
# 条形的高度  
bar_height = 0.3  
# 根据多组数据绘制条形图  
plt.barh(y, x1, tick_label=['a', 'b', 'c', 'd', 'e'], height=bar_height)  
plt.barh(y+bar_height, x2, height=bar_height)  
plt.show()

```

运行程序，效果如图 2-8 所示。

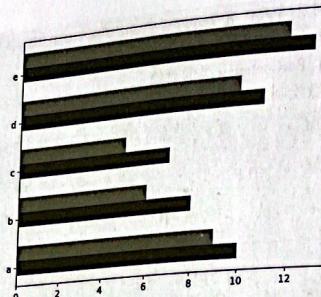


图 2-8 多组条形的条形图示例

使用 pyplot 的 barh() 函数绘制图表时，可以通过给 left 参数传值的方式控制条形的 x 值，使后绘制的条形位于先绘制的条形的右方。例如，使用 barh() 函数绘制由两组条形堆叠而成的堆积条形图，代码如下。

```
In [9]:
# 绘制堆积条形图
plt.barh(y, xl, tick_label=['a', 'b', 'c', 'd', 'e'], height=bar_height)
plt.barh(y, x2, left=xl, height=bar_height)
plt.show()
```

运行程序，效果如图 2-9 所示。

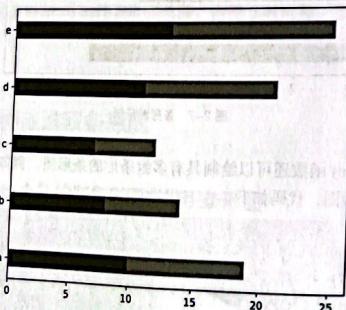


图 2-9 堆积条形图示例

另外，在使用 pyplot 的 barh() 函数绘制图表时，可以通过给 xerr、yerr 参数传值的方式为条形添加误差棒，示例代码如下。

```
In [10]:
# 偏差数据
error = [2, 1, 2.5, 2, 1.5]
```

```
# 绘制带有误差棒的条形图
plt.barh(y, xl, tick_label=['a', 'b', 'c', 'd', 'e'], height=bar_height)
plt.barh(y, x2, left=xl, height=bar_height, xerr=error)
plt.show()
```

运行程序，效果如图 2-10 所示。

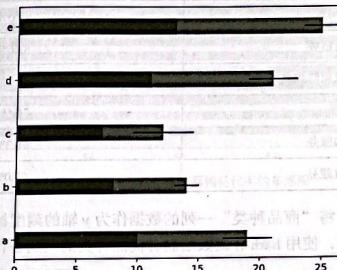


图 2-10 带有误差棒的条形图示例

### 2.3.2 实例 3：各商品种类的网购替代率

如今已进入信息时代，网络购物已经成为人们日常生活的一部分，改变着人们的消费模式和习惯，成为拉动居民消费的重要渠道。因此，研究网购消费对于研判经济形势、促进经济转型升级有着重要的意义。2018 年国家统计局北京调查总队从网购活跃的人群中抽取了 771 个样本，并根据这些样本测算用户网购替代率（网购用户线上消费对线下消费的替代比率）的情况，具体如表 2-3 所示。

表 2-3 各商品种类的网购替代率

商品种类	替代率
家政、家教、保姆等生活服务	95.9%
飞机票、火车票	95.1%
家具	93.5%
手机、手机配件	92.4%
计算机及其配套产品	89.3%
汽车用品	89.2%
通信充值、游戏充值	86.5%
个人护理用品	86.3%
书报杂志及音像制品	86.0%

商品种类	替代率
餐饮、旅游、住宿	85.6%
家用电器	85.4%
食品、饮料、烟酒、保健品	83.5%
家庭日杂用品	82.6%
保险、演出票务	81.6%
服装、鞋帽、家用纺织品	79.8%
数码产品	76.5%
其他商品和服务	76.3%
工艺品、收藏品	67.0%

根据表 2-3 的数据，将“商品种类”一列的数据作为 y 轴的刻度标签，将“替代率”一列的数据作为 x 轴的数据，使用 barh() 函数绘制各商品种类的网购替代率的条形图，具体代码如下。

```
In [11]:
# 03_substitution_rate_online
import matplotlib.pyplot as plt
import numpy as np
# 显示中文
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False
x = np.array([0.959, 0.951, 0.935, 0.924, 0.893,
             0.892, 0.865, 0.863, 0.860, 0.856,
             0.854, 0.835, 0.826, 0.816, 0.798,
             0.765, 0.763, 0.67])
y = np.arange(1, 19)
labels = ['家政、家教、保姆等生活服务', '飞机票、火车票', '家具', '手机、手机配件',
          '计算机及其配套产品', '汽车用品', '通信充值、游戏充值', '个人护理用品',
          '书报杂志及音像制品', '餐饮、旅游、住宿', '家用电器',
          '食品、饮料、烟酒、保健品', '家庭日杂用品', '保险、演出票务',
          '服装、鞋帽、家用纺织品', '数码产品', '其他商品和服务', '工艺品、收藏品']
# 绘制条形图
plt.barh(y, x, tick_label=labels, align="center", height=0.6)
plt.show()
```

需要说明的是，matplotlib 默认不支持显示中文，由于条形图的刻度标签是中文文本，因此需要将系统的字体修改为 SimHei。关于字体的设置会在第 4 章进行详细介绍。  
运行程序，效果如图 2-11 所示。

图 2-11 中，x 轴代表网购替代率，y 轴代表商品种类。由图 2-11 可知，工艺品、收藏品的网购替代率最低，家政、家教、保姆等生活服务的网购替代率最高。

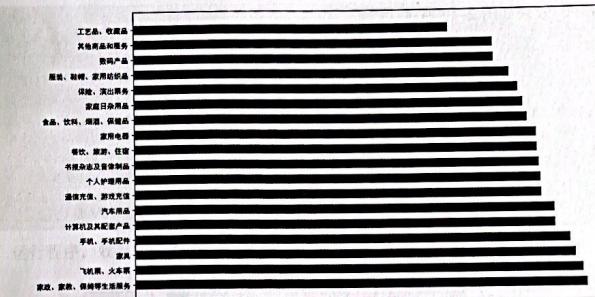


图 2-11 各商品种类的网购替代率的条形图

## 2.4 绘制堆积面积图

### 2.4.1 使用 stackplot() 绘制堆积面积图

使用 pyplot 的 stackplot() 函数可以快速绘制堆积面积图，stackplot() 函数的语法格式如下所示：

```
stackplot(x, y, labels=(), baseline='zero', data=None, *args, **kwargs)
```

该函数常用参数的含义如下。

- x：表示 x 轴的数据，可以是一维数组。
- y：表示 y 轴的数据，可以是二维数组或一维数组序列。
- labels：表示每组折线及填充区域的标签。

• baseline：表示计算基线的方法，包括‘zero’、‘sym’、‘wiggle’ 和 ‘weighted\_wiggle’。其中，‘zero’ 表示恒定零基线，即简单的堆积图；‘sym’ 表示对称于零基线；‘wiggle’ 表示最小化平方斜率的总和；‘weighted\_wiggle’ 表示执行相同的操作，但权重用于说明每层的大小。

例如，使用 stackplot() 函数绘制由 3 条折线及下方填充区域堆叠的堆积面积图，代码如下。

```
In [12]:
import numpy as np
import matplotlib.pyplot as plt
x = np.arange(6)
y1 = np.array([1, 4, 3, 5, 6, 7])
y2 = np.array([1, 3, 4, 2, 7, 6])
y3 = np.array([3, 4, 3, 6, 5, 5])
# 绘制堆积面积图
plt.stackplot(x, y1, y2, y3)
plt.show()
```

运行程序，效果如图 2-12 所示。

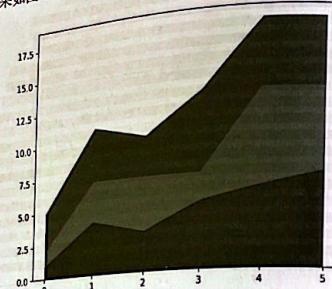


图 2-12 堆积面积图示例

需要说明的是，堆积面积图按照自下而上的顺序逐个堆叠填充区域，因此先绘制的图形位于底部，后绘制的图形位于上方。

#### 2.4.2 实例 4：物流公司物流费用统计

近些年我国物流行业蓬勃发展，目前已经有几千家物流公司。部分物流公司大打价格战，以更低的价格吸引更多的客户，从而抢占市场份额。现在有 A、B、C 三家物流公司，分别对公司去年的总物流费用进行了统计，具体如表 2-4 所示。

表 2-4 A、B、C 物流公司物流费用统计

月份	A 公司	B 公司	C 公司
1	198	203	185
2	215	236	205
3	245	200	226
4	222	236	199
5	200	269	238
6	236	216	200
7	201	298	250
8	253	333	209
9	236	301	246
10	200	349	219
11	266	360	253
12	290	368	288

根据表 2-4 的数据，将“月份”一列的数据作为 x 轴的刻度标签，将“A 公司”“B 公司”“C 公司”这三列的数据分别作为 y 轴的数据，使用 stackplot() 函数绘制 A、B、C 物流公司费用的堆积面积图，具体代码如下。

```
In [13]:
# 04_logistics_cost_statistics
import numpy as np
import matplotlib.pyplot as plt
x = np.arange(1, 13)
y_a = np.array([198, 215, 245, 222, 200, 236, 201, 253, 236, 200, 266, 290])
y_b = np.array([203, 236, 200, 236, 269, 216, 298, 333, 301, 349, 360, 368])
y_c = np.array([185, 205, 226, 199, 238, 200, 250, 209, 246, 219, 253, 288])
# 绘制堆积面积图
plt.stackplot(x, y_a, y_b, y_c)
plt.show()
```

运行程序，效果如图 2-13 所示。

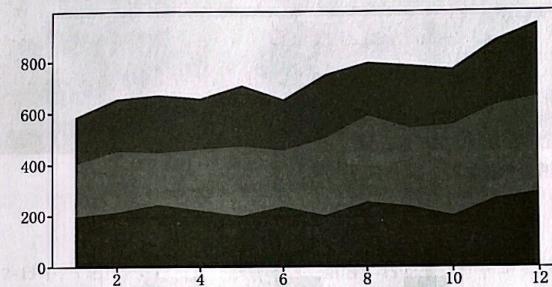


图 2-13 A、B、C 物流公司物流费用的堆积面积图

图 2-13 中，x 轴代表月份，y 轴代表物流费用，底部的蓝色区域代表 A 公司的物流费用，中间的橙色区域代表 B 公司的物流费用，顶部的绿色区域代表 C 公司的物流费用。

## 2.5 绘制直方图

### 2.5.1 使用 hist() 绘制直方图

使用 pyplot 的 hist() 函数可以快速绘制直方图，hist() 函数的语法格式如下所示：

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

该函数常用参数的含义如下。

- x：表示 x 轴的数据，可以为单个数组或多个数组的序列。
- bins：表示矩形条的个数，默认为 10。
- range：表示数据的范围。若没有提供 range 参数的值，则数据范围为 (x.min(), x.max())。

- cumulative：表示是否计算累计频数或频率。
- histtype：表示直方图的类型，支持‘bar’、‘barstacked’、‘step’、‘stepfilled’四种取值，其中‘bar’为默认值，代表传统的直方图；‘barstacked’代表堆积直方图；‘step’代表未填充的线条直方图；‘stepfilled’代表填充的线条直方图。
- align：表示矩形条边界的对齐方式，可设置为‘left’、‘mid’或‘right’，默认为‘mid’。
- orientation：表示矩形条的摆放方式，默认为‘vertical’，即垂直方向。
- rwidth：表示矩形条宽度的百分比，默认为0。若 histtype 的值为‘step’或‘stepfilled’，则直接忽略 rwidth 参数的值。
- stacked：表示是否将多个矩形条以堆积形式摆放。

例如，绘制一个具有8个矩形条填充的线条直方图，代码如下。

```
In [14]:
import numpy as np
import matplotlib.pyplot as plt
# 准备 50 个随机测试数据
scores = np.random.randint(0, 100, 50)
# 绘制直方图
plt.hist(scores, bins=8, histtype='stepfilled')
plt.show()
```

运行程序，效果如图 2-14 所示。

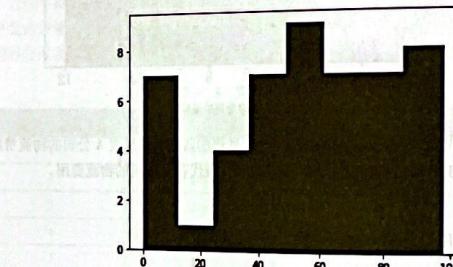


图 2-14 填充的线条直方图示例

### 2.5.2 实例 5：人脸识别的灰度直方图

随着计算机技术的不断发展，人工智能的应用已渗透到人们日常生活的方方面面，其中人脸识别技术是近两年较为热门的话题之一。人脸识别技术是一种生物特征识别技术，它通过从装有摄像头的终端设备拍摄的人脸图像中抽取人的个性化特征，以此来识别别人的身份。灰度直方图便是实现人脸识别的方法之一，它将数字图像的所有像素，按照灰度值的大小，统计其出现的频率。

下面使用一组 10000 个随机数作为人脸识别的灰度值，使用 hist() 函数绘制一个灰度直方图，具体代码如下。

```
In [15]:
# 05_face_recognition
import matplotlib.pyplot as plt
import numpy as np
# 10000 个随机数
random_state = np.random.RandomState(19680801)
random_x = random_state.randn(10000)
# 绘制包含 25 个矩形条的直方图
plt.hist(random_x, bins=25)
plt.show()
```

运行程序，效果如图 2-15 所示。

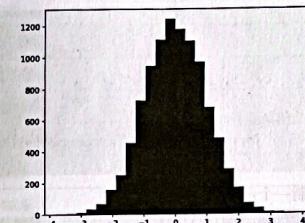


图 2-15 人脸识别的灰度值的直方图

图 2-15 中，x 轴代表灰度值，y 轴代表频率。由图 2-15 可知，位于 -0.5 ~ 0 之间的灰度值最多，位于 -4 ~ -3 或 3 ~ 4 之间的灰度值最少。

## 2.6 绘制饼图或圆环图

### 2.6.1 使用 pie() 绘制饼图或圆环图

使用 pyplot 的 pie() 函数可以快速地绘制饼图或圆环图，pie() 函数的语法格式如下所示：

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

该函数常用参数的含义如下。

- x：表示扇形或楔形的数据。
- explode：表示扇形或楔形离开圆心的距离。
- labels：表示扇形或楔形对应的标签文本。
- autopct：表示控制扇形或楔形的数值显示的字符串，可通过格式字符串指定小数点后的位数。
- pctdistance：表示扇形或楔形对应的数值标签距离圆心的比例，默认为 0.6。

- shadow : 表示是否显示阴影。
- labeldistance : 表示标签文本的绘制位置 (相对于半径的比例), 默认为 1.1。
- startangle : 表示起始绘制角度, 默认从 x 轴的正方向逆时针绘制。
- radius : 表示扇形或楔形的半径。
- wedgeprops : 表示控制扇形或楔形属性的字典。例如, 通过 wedgeprops = {'width': 0.7} 将楔形的宽度设为 0.7。
- textprops : 表示控制图表中文本属性的字典。
- center : 表示图表的中心点位置, 默认为 (0,0)。
- frame : 表示是否显示图框。

例如, 使用 pie() 函数绘制一个饼图, 代码如下。

```
In [16]:
import numpy as np
import matplotlib.pyplot as plt
data = np.array([20, 50, 10, 15, 30, 55])
pie_labels = np.array(['A', 'B', 'C', 'D', 'E', 'F'])
# 绘制饼图: 半径为 0.5, 数值保留 1 位小数
plt.pie(data, radius=1.5, labels=pie_labels, autopct='%.1f%%')
plt.show()
```

例如, 使用 pie() 函数绘制一个圆环图, 代码如下。

```
In [17]:
import numpy as np
import matplotlib.pyplot as plt
data = np.array([20, 50, 10, 15, 30, 55])
pie_labels = np.array(['A', 'B', 'C', 'D', 'E', 'F'])
# 绘制圆环图: 外圆半径为 1.5, 楔形宽度为 0.7
plt.pie(data, radius=1.5, wedgeprops={'width': 0.7}, labels=pie_labels,
        autopct='%.1f%%', pctdistance=0.75)
plt.show()
```

两个示例运行的效果如图 2-16 所示。

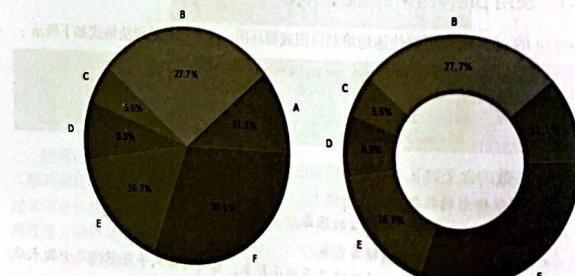


图 2-16 饼图与圆环图示例

## 2.6.2 实例 6：支付宝月账单报告

近年来随着移动支付 App 的出现, 人们的生活发生了翻天覆地的变化, 无论是到超市选购商品, 还是跟朋友聚餐, 或是来一场说走就走的旅行, 都可以使用移动支付 App 轻松完成支付, 非常便捷。支付宝是人们使用较多的移动支付方式, 它拥有自动记录每月账单的功能, 可以方便用户了解每月资金的流动情况。例如, 用户 A 某月使用支付宝的消费明细如表 2-5 所示。

表 2-5 用户 A 某月使用支付宝的消费明细

单位: 元

分类	金额
购物	800
人情往来	100
餐饮美食	1000
通信物流	200
生活日用	300
交通出行	200
休闲娱乐	200
其他	200
总支出	3000

根据表 2-5 的数据, 将“分类”一列的数据作为饼图的标签, 将各分类对应的金额与总支出金额的比例作为饼图的数据, 使用 pie() 函数绘制用户 A 某月支付宝消费情况的饼图, 具体代码如下。

```
In [18]:
# 06_monthly_bills_of_alipay
import matplotlib.pyplot as plt
import matplotlib as mpl
mpl.rcParams['font.sans-serif'] = ['SimHei']
mpl.rcParams['axes.unicode_minus'] = False
# 饼图外侧的说明文字
kinds = ['购物', '人情往来', '餐饮美食', '通信物流', '生活日用',
         '交通出行', '休闲娱乐', '其他']
# 饼图的数据
money_scale = [800 / 3000, 100 / 3000, 1000 / 3000, 200 / 3000,
               300 / 3000, 200 / 3000, 200 / 3000]
dev_position = [0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1]
# 绘制饼图
plt.pie(money_scale, labels=kinds, autopct='%.1f%%', shadow=True,
        explode=dev_position, startangle=90)
plt.show()
```

运行程序, 效果如图 2-17 所示。

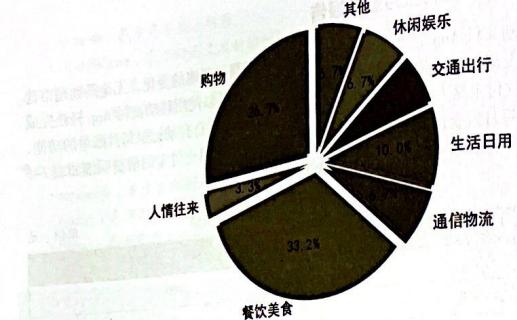


图 2-17 用户 A 某月支付宝账单报告的饼图

由图 2-17 可知，绿色扇形的面积最大，说明餐饮美食方面的支出在当月总支出中占比最大；橙色扇形的面积最小，说明人情往来的支出在当月总支出中占比最小。

## 2.7 绘制散点图或气泡图

### 2.7.1 使用 scatter() 绘制散点图或气泡图

使用 pyplot 的 scatter() 函数可以快速绘制散点图或气泡图，scatter() 函数的语法格式如下所示：

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

该函数常用参数的含义如下。

- x, y：表示数据点的位置。
  - s：表示数据点的大小。
  - c：表示数据点的颜色。
  - marker：表示数据点的样式，默认为圆形。
  - cmap：表示数据点的颜色映射表，仅当参数 c 为浮点数组时才使用。
  - norm：表示数据亮度，可以取值为 0 ~ 1。
  - vmin, vmax：表示亮度的最小值和最大值。若传入了 norm 参数，则忽略 vmin 和 vmax 参数。
  - alpha：表示透明度，可以取值为 0 ~ 1。
  - linewidths：表示数据点边缘的宽度。
  - edgecolors：表示数据点边缘的颜色。
- 使用 scatter() 函数绘制一个散点图，代码如下。

```
In [19]:
num = 50
x = np.random.rand(num)
y = np.random.rand(num)
plt.scatter(x, y)
```

使用 scatter() 函数绘制一个气泡图，代码如下。

```
In [20]:
num = 50
x = np.random.rand(num)
y = np.random.rand(num)
area = (30 * np.random.rand(num))**2
plt.scatter(x, y, s=area)
```

两个示例运行的效果如图 2-18 所示。

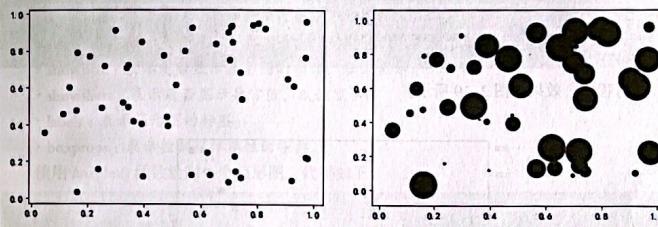


图 2-18 散点图与气泡图示例

### 2.7.2 实例 7：汽车速度与制动距离的关系

汽车的制动距离主要取决于车速。若车速增加 1 倍，则汽车的制动距离将增大至近 4 倍。某汽车生产公司对一批丰田汽车进行抽样测试，并分别记录了不同的车速对应的制动距离，具体如表 2-6 所示。

表 2-6 车速与制动距离的关系

车速 (km/h)	制动距离 (m)	车速 (km/h)	制动距离 (m)
10	0.5	110	59.5
20	2.0	120	70.8
30	4.4	130	83.1
40	7.9	140	96.4
50	12.3	150	110.7
60	17.7	160	126.0
70	24.1	170	142.2
80	31.5	180	159.4
90	39.9	190	177.6
100	49.2	200	196.8

根据表 2-6 的数据, 将“车速 (km/h)”一列的数据作为 x 轴的数据, 将“制动距离 (m)”一列的数据作为 y 轴的数据, 使用 scatter() 函数绘制汽车速度与制动距离关系的散点图, 具体代码如下。

```
In [21]:
# 07_vehicle_speed_and_braking_distance
import numpy as np
import matplotlib.pyplot as plt
plt.rcParams['font.sans-serif'] = 'SimHei'
plt.rcParams['axes.unicode_minus'] = False
# 准备 x 轴和 y 轴的数据
x_speed = np.arange(10, 210, 10)
y_distance = np.array([0.5, 2.0, 4.4, 7.9, 12.3,
                      17.7, 24.1, 31.5, 39.9, 49.2,
                      59.5, 70.8, 83.1, 96.4, 110.7,
                      126.0, 142.2, 159.4, 177.6, 196.8])
# 绘制散点图
plt.scatter(x_speed, y_distance, s=50, alpha=0.9)
plt.show()
```

运行程序, 效果如图 2-19 所示。

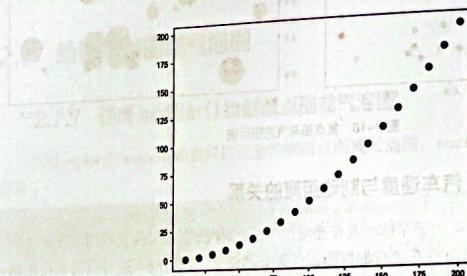


图 2-19 汽车速度与制动距离的散点图

图 2-19 中, x 轴代表车速, y 轴代表制动距离。由图 2-19 可知, 恒定条件下, 制动距离随着车速的增大而增加。

## 2.8 绘制箱形图

### 2.8.1 使用 boxplot() 绘制箱形图

使用 pyplot 的 boxplot() 函数可以快速绘制箱形图, boxplot() 函数的语法格式如下所示:

```
boxplot(x, notch=None, sym=None, vert=None, whis=None, positions=None,
        widths=None, patch_artist=None, bootstrap=None, usermedians=None,
```

```
conf_intervals=None, meanline=None, showmeans=None, showcaps=None,
showbox=None, showfliers=None, boxprops=None, labels=None,
flierprops=None, medianprops=None, meanprops=None, capprops=None,
whiskerprops=None, manage_ticks=True, autorange=False,
zorder=None, *, data=None)
```

该函数常用参数的含义如下:

- x : 绘制箱形图的数据。
- sym : 表示异常值对应的符号, 默认为空心圆圈。
- vert : 表示是否将箱形图垂直摆放, 默认为垂直摆放。
- whis : 表示箱形图上下须与上下四分位的距离, 默认为 1.5 倍的四分位差。
- positions : 表示箱体的位置。
- widths : 表示箱体的宽度, 默认为 0.5。
- patch\_artist : 表示是否填充箱体的颜色, 默认不填充。
- meanline : 表示是否横跨箱体的线条标出中位数, 默认不使用。
- showcaps : 表示是否显示箱体顶部和底部的横线, 默认显示。
- showbox : 表示是否显示箱形图的箱体, 默认显示。
- showfliers : 表示是否显示异常值, 默认显示。
- labels : 表示箱形图的标签。
- boxprops : 表示控制箱体属性的字典。

使用 boxplot() 函数绘制一个箱形图, 代码如下。

```
In [22]:
import numpy as np
import matplotlib.pyplot as plt
data = np.random.randn(100)
# 绘制箱形图: 显示中位数的线条, 箱体宽度为 0.3, 填充箱体颜色, 不显示异常值
plt.boxplot(data, meanline=True, widths=0.3, patch_artist=True,
            showfliers=False)
plt.show()
```

运行程序, 效果如图 2-20 所示。

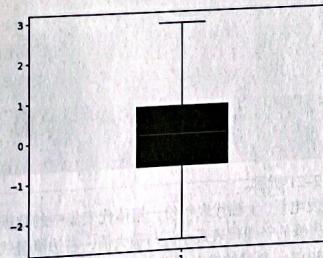


图 2-20 箱形图示例

## 2.8.2 实例 8：2017 年和 2018 年全国发电量统计

“中国报告大厅”网站对 2017 年和 2018 年的发电量分别进行了监测统计，结果如表 2-7 所示。

表 2-7 2017 年和 2018 年全国发电量统计

2018 年		月份	发电量 (亿千瓦·时)	2017 年
月份	发电量 (亿千瓦·时)			
1月	5200	1月	4605.2	
2月	5254.5	2月	4710.3	
3月	5283.4	3月	5168.9	
4月	5107.8	4月	4767.2	
5月	5443.3	5月	4947	
6月	5550.6	6月	5203	
7月	6400.2	7月	6047.4	
8月	6404.9	8月	5945.5	
9月	5483.1	9月	5219.6	
10月	5330.2	10月	5038.1	
11月	5543	11月	5196.3	
12月	6199.9	12月	5698.6	

根据表 2-7 的数据，将“发电量(亿千瓦·时)”两列的数据作为 x 轴的数据，将“2017 年”和“2018 年”作为 y 轴的刻度标签，使用 boxplot() 函数绘制 2017 年和 2018 年全国发电量的箱形图，具体代码如下。

```
In [23]:
# 08_generation_capacity
import numpy as np
import matplotlib.pyplot as plt
plt.rcParams['font.family'] = 'SimHei'
plt.rcParams['axes.unicode_minus'] = False
data_2018 = np.array([5200, 5254.5, 5283.4, 5107.8, 5443.3, 5550.6,
                     6400.2, 6404.9, 5483.1, 5330.2, 5543, 6199.9])
data_2017 = np.array([4605.2, 4710.3, 5168.9, 4767.2, 4947, 5203,
                     6047.4, 5945.5, 5219.6, 5038.1, 5196.3, 5698.6])
# 绘制箱形图
plt.boxplot([data_2018, data_2017], labels=('2018年', '2017年'),
            meanline=True, widths=0.5, vert=False, patch_artist=True)
plt.show()
```

运行程序，效果如图 2-21 所示。

图 2-21 中，x 轴代表发电量，y 轴代表年份，箱形内部的中位数，箱形左右边缘的竖线代表最小值与最大值，右边缘竖线右侧的空心圆圈代表异常值。由图 2-21 可知，2017 年每月的发电量大多分布于 4800 亿~5300 亿千瓦·时范围内，2018 年每月的发电量大多分布于 5250 亿~5700 亿千瓦·时范围内。

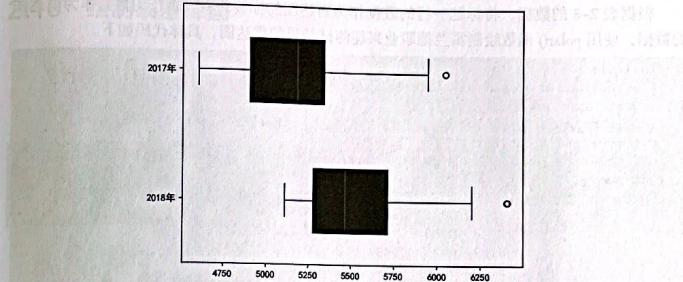


图 2-21 2017 年和 2018 年全国发电量统计的箱形图

## 2.9 绘制雷达图

### 2.9.1 使用 polar() 绘制雷达图

使用 pyplot 的 polar() 函数可以快速绘制雷达图，polar() 函数的语法格式如下所示：

```
polar(theta, r, **kwargs)
```

该函数常用参数的含义如下。

- theta：表示每个数据点所在射线与极径的夹角。
- r：表示每个数据点到原点的距离。

### 2.9.2 实例 9：霍兰德职业兴趣测试

霍兰德职业兴趣测试是美国职业指导专家霍兰德根据他本人大量的职业咨询经验及其职业类型理论编制的测评工具。根据个人兴趣的不同，霍兰德将人格分为研究型(I)、艺术型(A)、社会型(S)、企业型(E)、传统型(C)和现实型(R)6个维度，每个人的性格都是这6个维度不同程度的组合。假设现在 6 名用户分别进行了测试，得出的测试结果如表 2-8 所示。

表 2-8 6 名用户霍兰德职业兴趣测试的结果

	研究型	艺术型	社会型	企业型	传统型	现实型
用户 1	0.40	0.32	0.35	0.30	0.30	0.88
用户 2	0.85	0.35	0.30	0.40	0.40	0.30
用户 3	0.43	0.89	0.30	0.28	0.22	0.30
用户 4	0.30	0.25	0.48	0.85	0.45	0.40
用户 5	0.20	0.38	0.87	0.45	0.32	0.28
用户 6	0.34	0.31	0.38	0.40	0.92	0.28

根据表 2-8 的数据, 将标题一行的数据作为雷达图的标签, 将其余行的数据作为雷达图的数据, 使用 polar() 函数绘制霍兰德职业兴趣测试结果的雷达图, 具体代码如下。

```
In [24]:
# 09_holland_professional_interest_test
import numpy as np
import matplotlib.pyplot as plt
plt.rcParams['font.family'] = 'SimHei'
plt.rcParams['axes.unicode_minus'] = False
dim_num = 6
data = np.array([[0.40, 0.32, 0.35, 0.30, 0.30, 0.88],
                [0.85, 0.35, 0.30, 0.40, 0.40, 0.30],
                [0.43, 0.89, 0.30, 0.28, 0.22, 0.30],
                [0.30, 0.25, 0.48, 0.85, 0.45, 0.40],
                [0.20, 0.38, 0.87, 0.45, 0.32, 0.28],
                [0.34, 0.31, 0.38, 0.40, 0.92, 0.28]])
angles = np.linspace(0, 2 * np.pi, dim_num, endpoint=False)
angles = np.concatenate([angles, [angles[0]]])
data = np.concatenate([data, [data[0]]])
# 维度标签
rradar_labels = ['研究型(I)', '艺术型(A)', '社会型(S)', '企业型(E)', '传统型(C)', '现实型(R)']
radar_labels = np.concatenate([rradar_labels, [rradar_labels[0]]])
# 绘制雷达图
plt.polar(angles, data)
# 设置极坐标的标签
plt.thetagrids(angles * 180 / np.pi, labels=radar_labels)
# 填充多边形
plt.fill(angles, data, alpha=0.25)
plt.show()
```

运行程序, 效果如图 2-22 所示。

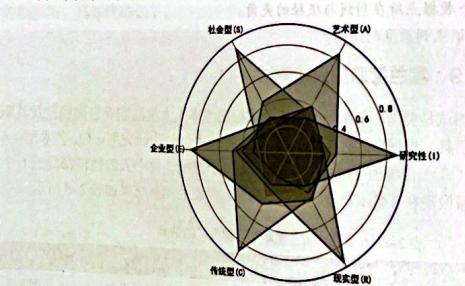


图 2-22 霍兰德职业测试的雷达图

图 2-22 中, 紫色的多边形代表用户 1 的测试结果; 棕色的多边形代表用户 2 的测试结果; 蓝色的多边形代表用户 3 的测试结果; 红色的多边形代表用户 4 的测试结果; 橙色的多边形代表用户 5 的测试结果; 绿色的多边形代表用户 6 的测试结果。由图 2-22 可知, 用户 1 倾向于现实型人格; 用户 2 倾向于研究型人格; 用户 3 倾向于艺术型人格; 用户 4 倾向于企业型人格; 用户 5 倾向于社会型人格; 用户 6 倾向于传统型人格。

## 2.10 绘制误差棒图

### 2.10.1 使用 errorbar() 绘制误差棒图

使用 pyplot 的 errorbar() 函数可以快速绘制误差棒图, errorbar() 函数的语法格式如下所示:

```
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)
```

该函数常用参数的含义如下。

- x, y: 表示数据点的位置。
- xerr, yerr: 表示数据的误差范围。
- fmt: 表示数据点的标记样式和数据点之间连接线的样式。
- ecolor: 表示误差棒的线条颜色。
- elinewidth: 表示误差棒的线条宽度。
- capszie: 表示误差棒边界横杆的大小。
- capthick: 表示误差棒边界横杆的厚度。

使用 errorbar() 函数绘制一个误差棒图, 代码如下。

```
In [25]:
import numpy as np
import matplotlib.pyplot as plt
x = np.arange(5)
y = (25, 32, 34, 20, 25)
y_offset = (3, 5, 2, 3, 3)
plt.errorbar(x, y, yerr=y_offset, capsize=3, capthick=2)
plt.show()
```

运行程序, 效果如图 2-23 所示。

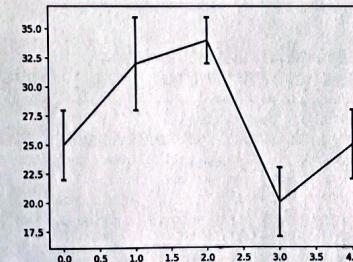


图 2-23 误差棒图示例

运行程序，效果如图 2-24 所示。

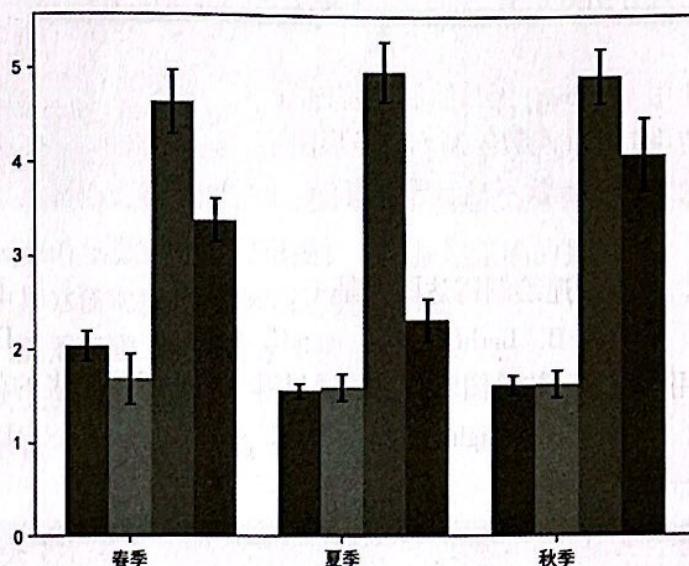


图 2-24 4 个树种不同季节的细根生物量的误差棒图

图 2-24 中， $x$  轴代表季节， $y$  轴代表细根生物量，蓝色、橙色、绿色、红色的柱形依次代表马尾松、樟树、杉木、桂花，柱形上方的黑色短线代表误差棒。由图 2-24 可知，杉木的细根生物量最多，说明杉木吸收水分和养分的能力最强；樟树的细根生物量最少，说明樟树吸收水分和养分的能力最弱。

### 注意：

本章所介绍的简单图表（雷达图除外）除了可以使用 pyplot 模块的绘图函数绘制，还可以通过 Axes 类中与绘图函数同名的方法进行绘制。例如，pyplot 模块的 bar() 函数与 Axes 类的 bar() 方法都可以绘制柱形图，它们的参数几乎相同（self 除外）。由于本章设计的实例相对比较简单，因此所有的实例都采用 pyplot 模块的绘制函数实现。

## 2.11 本章小结

本章主要介绍了如何使用 matplotlib 的绘图函数绘制简单的图表，包括折线图、柱形图或堆积柱形图、条形图或堆积条形图、堆积面积图、直方图、饼图或圆环图、散点图或气泡图、箱形图、雷达图、误差棒图。希望大家通过学习本章的内容，能够掌握绘图函数的用法，并可以使用这些函数绘制简单的图表，从而为后续的学习打好扎实的基础。

## 2.12 习题

### 一、填空题

1. plot() 函数会返回一个包含多个 \_\_\_\_\_ 类对象的列表。

2. 常见的\_\_\_\_\_包括堆积面积图、堆积柱形图和堆积条形图。  
 3. pyplot 绘制的直方图默认有\_\_\_\_\_个矩形条。

## 二、判断题

1. pyplot 只能使用 errorbar() 函数绘制误差棒图。( )  
 2. pyplot 可以使用 barh() 函数绘制堆积条形图。( )  
 3. pyplot 绘制的箱形图默认不显示异常值。( )

## 三、选择题

1. 下列函数中，可以快速绘制雷达图的是( )。  
 A. bar() B. barh() C. hist() D. polar()  
 2. 当 pyplot 调用 barh() 函数绘图时，可以通过哪个参数设置图表的刻度标签？( )  
 A. width B. height C. tick\_label D. align  
 3. 请阅读下面一段代码：

```
plt.bar(x, y1, tick_label=["A", "B", "C", "D"])
plt.bar(x, y2, bottom=y1, tick_label=["A", "B", "C", "D"])
```

以上代码中 bar() 函数的 bottom 参数的作用是( )。

- A. 将后绘制的柱形置于先绘制的柱形下方  
 B. 将后绘制的柱形置于先绘制的柱形上方  
 C. 将后绘制的柱形置于先绘制的柱形左方  
 D. 将后绘制的柱形置于先绘制的柱形右方  
 4. 下列选项中，程序运行的效果为圆环图的是( )。

A.

```
import numpy as np
import matplotlib.pyplot as plt
data = np.array([20, 50, 10, 15, 30, 55])
pie_labels = np.array(['A', 'B', 'C', 'D', 'E', 'F'])
plt.pie(data, labels=pie_labels)
plt.show()
```

B.

```
import numpy as np
import matplotlib.pyplot as plt
data = np.array([20, 50, 10, 15, 30, 55])
pie_labels = np.array(['A', 'B', 'C', 'D', 'E', 'F'])
plt.pie(data, radius=1.5, labels=pie_labels)
plt.show()
```

C.

```
import numpy as np
import matplotlib.pyplot as plt
data = np.array([20, 50, 10, 15, 30, 55])
pie_labels = np.array(['A', 'B', 'C', 'D', 'E', 'F'])
plt.pie(data, radius=1.5, explode=[0, 0.2, 0, 0, 0], labels=pie_labels)
plt.show()
```

D.

```
import numpy as np
import matplotlib.pyplot as plt
data = np.array([20, 50, 10, 15, 30, 55])
pie_labels = np.array(['A', 'B', 'C', 'D', 'E', 'F'])
plt.pie(data, radius=1.5, wedgeprops={'width': 0.6}, labels=pie_labels)
plt.show()
```

5. 关于使用 boxplot() 函数绘制的箱形图，下列描述正确的是( )。

- A. 箱形图中异常值对应的符号默认为星号  
 B. 箱形图只能垂直摆放，无法水平摆放  
 C. 箱形图默认显示箱体  
 D. 箱形图默认不会显示异常值

## 四、编程题

1. 已知实验中学举行了高二期中模拟考试，考试后分别计算了全体男生、女生各科的平均成绩，结果如表 2-10 所示。

表 2-10 高二男生、女生各科的平均成绩

学科	平均成绩（男）	平均成绩（女）
语文	85.5	94
数学	91	82
英语	72	89.5
物理	59	62
化学	66	49
生物	55	53

按照以下要求绘制图表：

- (1) 绘制柱形图。柱形图的 x 轴为学科，y 轴为平均成绩。  
 (2) 绘制堆积柱形图。堆积柱形图的 x 轴为学科，y 轴为平均成绩。

2. 拼多多作为互联网电商的一匹黑马，短短几年内用户规模已经超过 3 亿。2019 年 9 月拼多多平台对所有子类目的销售额进行了统计，结果如表 2-11 所示。

表 2-11 拼多多平台子类目的销售额 单位：亿元

子类目	销售额
童装	29665
奶粉辅食	3135.4
孕妈专区	4292.4
洗护喂养	5240.9
宝宝尿裤	5543.4
春夏新品	5633.8
童车童床	6414.5
玩具文娱	9308.1
童鞋	10353

根据表 2-11 的数据绘制一个反映拼多多平台子类目销售额占比情况的饼图。