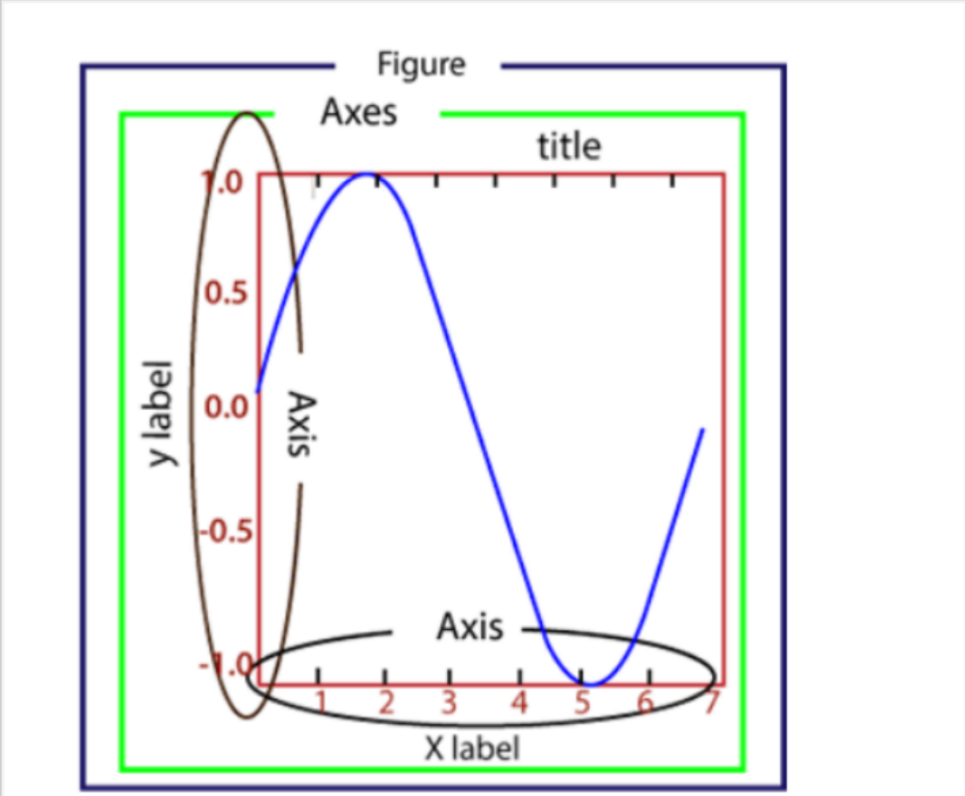
1.概念

Matplotlib 库：是一款用于数据可视化的 Python 软件包，支持跨平台运行，它能够根据 NumPy ndarray 数组来绘制 2D 图像，它使用简单、代码清晰易懂

Matplotlib 图形组成：



- Figure：指整个图形，您可以把它理解成一张画布，它包括了所有的元素，比如标题、轴线等
- Axes：绘制 2D 图像的实际区域，也称为轴域区，或者绘图区
- Axis：指坐标系中的垂直轴与水平轴，包含轴的长度大小（图中轴长为 7）、轴标签（指 x 轴，y 轴）和刻度标签
- Artist：您在画布上看到的所有元素都属于 Artist 对象，比如文本对象（title、xlabel、ylabel）、Line2D 对象（用于绘制2D图像）等
- Matplotlib 功能扩展包：许多第三方工具包都对 Matplotlib 进行了功能扩展，其中有些安装包需要单独安装，也有一些允许与 Matplotlib 一起安装。常见的工具包如下：
    - Basemap：这是一个地图绘制工具包，其中包含多个地图投影，海岸线和国界线
    - Cartopy：这是一个映射库，包含面向对象的映射投影定义，以及任意点、线、面的图像转换能力
    - Excel tools： 这是 Matplotlib 为了实现与 Microsoft Excel 交换数据而提供的工具
    - Mplot3d：它用于 3D 绘图
    - Natgrid：这是 Natgrid 库的接口，用于对间隔数据进行不规则的网格化处理

2.安装

```
pip install matplotlib==3.3.4 -i https://pypi.tuna.tsinghua.edu.cn/simple/
```

3.应用场景

数据可视化主要有以下应用场景：

- 企业领域：利用直观多样的图表展示数据，从而为企业决策提供支持
- 股票走势预测：通过对股票涨跌数据的分析，给股民提供更合理化的建议
- 商超产品销售：对客户群体和所购买产品进行数据分析，促使商超制定更好的销售策略
- 预测销量：对产品销量的影响因素进行分析，可以预测出产品的销量走势

4.常用API

4.1 绘图类型

函数名称	描述
Bar	绘制条形图
Barh	绘制水平条形图
Boxplot	绘制箱型图
Hist	绘制直方图
his2d	绘制2D直方图
Pie	绘制饼状图
Plot	在坐标轴上画线或者标记
Polar	绘制极坐标图
Scatter	绘制x与y的散点图
Stackplot	绘制堆叠图
Stem	用来绘制二维离散数据绘制（又称为火柴图）
Step	绘制阶梯图
Quiver	绘制一个二维按箭头

4.2 Image 函数

函数名称	描述
Imread	从文件中读取图像的数据并形成数组
Imsave	将数组另存为图像文件
Imshow	在数轴区域内显示图像

4.3 Axis 函数

函数名称	描述
Axes	在画布(Figure)中添加轴
Text	向轴添加文本
Title	设置当前轴的标题
Xlabel	设置x轴标签
Xlim	获取或者设置x轴区间大小
Xscale	设置x轴缩放比例
Xticks	获取或设置x轴刻标和相应标签
Ylabel	设置y轴的标签
Ylim	获取或设置y轴的区间大小
Yscale	设置y轴的缩放比例
Yticks	获取或设置y轴的刻标和相应标签

4.4 Figure 函数

函数名称	描述
Figtext	在画布上添加文本
Figure	创建一个新画布
Show	显示数字
Savefig	保存当前画布
Close	关闭画布窗口

5.pylab 模块

PyLab 是一个面向 Matplotlib 的绘图库接口，其语法和 MATLAB 十分相近。

pylab 是 matplotlib 中的一个模块，它将 matplotlib.pyplot 和 numpy 的功能组合在一起，使得你可以直接使用 numpy 的函数和 matplotlib.pyplot 的绘图功能，而不需要显式地导入 numpy 和 matplotlib.pyplot。

优点

- 方便快捷：`pylab` 的设计初衷是为了方便快速绘图和数值计算，使得你可以直接使用 `numpy` 的函数和 `matplotlib.pyplot` 的绘图功能，而不需要显式地导入 `numpy` 和 `matplotlib.pyplot`。
- 简化代码：使用 `pylab` 可以减少导入语句的数量，使代码更简洁。

缺点

- 命名空间污染：pylab 将 numpy 和 matplotlib.pyplot 的功能组合在一起，可能会导致命名空间污染，使得代码的可读性和可维护性降低。
- 不适合大型项目：对于大型项目或需要精细控制的项目，pylab 可能不够灵活。

pyplot 是 matplotlib 中的一个模块，提供了类似于 MATLAB 的绘图接口。它是一个更底层的接口，提供了更多的控制和灵活性。

使用 pyplot 需要显式地导入 numpy 和 matplotlib.pyplot，代码量相对较多。例如：

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

6.常用函数

6.1 plot 函数

pylab.plot 是一个用于绘制二维图形的函数。它可以根据提供的 x 和 y 数据点绘制线条和/或标记。

语法：

```
pylab.plot(x, y, format_string=None, **kwargs)
```

参数：

- x: x 轴数据，可以是一个数组或列表。
- y: y 轴数据，可以是一个数组或列表。
- format_string: 格式字符串，用于指定线条样式、颜色等。
- **kwargs: 其他关键字参数，用于指定线条的属性。

plot 函数可以接受一个或两个数组作为参数，分别代表 x 和 y 坐标。如果你只提供一个数组，它将默认用作 y 坐标，而 x 坐标将默认为数组的索引。

格式字符串：

格式字符串由颜色、标记和线条样式组成。例如：

颜色：

'b'：蓝色 `'g'`：绿色 `'r'`：红色 `'c'`：青色 `'m'`：洋红色 `'y'`：黄色 `'k'`：黑色 `'w'`：白色

标记：

`'.'`：点标记 `','`：像素标记 `'o'`：圆圈标记 `'v'`：向下三角标记 `'^'`：向上三角标记 `'<'`：向左三角标记 `'>'`：向右三角标记 `'s'`：方形标记 `'p'`：五边形标记 `'*'`：星形标记 `'h'`：六边形标记 1 `'H'`：六边形标记 2 `'+'`：加号标记 `'x'`：叉号标记 `'D'`：菱形标记 `'d'`：细菱形标记 `'|'`：竖线标记 `'_'`：横线标记

线条样式：

`'-'`：实线 `'--'`：虚线 `'-.'`：点划线 `':'`：点线

案例：

```
# 导入 pylab 库
import pylab

# 创建数据，使用 linspace 函数
# pylab.linspace 函数生成一个等差数列。这个函数返回一个数组，数组中的数值在指定的区间内均匀分布。
x = pylab.linspace(-6, 6, 40)
# 基于 x 构建 y 的数据
y = x**2
# 绘制图形
pylab.plot(x,y,'r:')
# 展示图形
pylab.show()
```

警告日志关闭

```
import logging
logging.captureWarnings(True)
```

6.2 figure 函数

figure() 函数来实例化 figure 对象，即绘制图形的对象，可以通过这个对象，来设置图形的样式等

参数：

- figsize：指定画布的大小，(宽度,高度)，单位为英寸
- dpi：指定绘图对象的分辨率，即每英寸多少个像素，默认值为80
- facecolor：背景颜色
- dgecolor：边框颜色
- frameon：是否显示边框

6.2.1 figure.add_axes()

Matplotlib 定义了一个 axes 类（轴域类），该类的对象被称为 axes 对象（即轴域对象），它指定了一个有数值范围限制的绘图区域。在一个给定的画布（figure）中可以包含多个 axes 对象，但是同一个 axes 对象只能在一个画布中使用。

参数：

是一个包含四个元素的列表或元组，格式为 [left, bottom, width, height]，其中：

left 和 bottom 是轴域左下角的坐标，范围从 0 到 1。

width 和 height 是轴域的宽度和高度，范围从 0 到 1。

案例：

```
# 创建一个新的图形
fig = pl.figure()

# 添加第一个轴域
ax = fig.add_axes([0.1, 0.1, 0.8, 0.8])
x = pl.linspace(0, 10, 100)
y = pl.sin(x)
ax.plot(x, y)

# 显示图形
pl.show()
```

6.2.2 axes.legend()

legend 函数用于添加图例，以便识别图中的不同数据系列。图例会自动显示每条线或数据集的标签。

参数:

- labels 是一个字符串序列，用来指定标签的名称
- loc 是指定图例位置的参数，其参数值可以用字符串或整数来表示
- handles 参数，它也是一个序列，它包含了所有线型的实例

案例:

```
x = pl.linspace(0, 10, 100)
y1 = pl.sin(x)
y2 = pl.cos(x)

# 创建图形和轴域
fig, ax = pl.subplots()

# 绘制数据
line1 = ax.plot(x, y1)
line2 = ax.plot(x, y2)

# 添加图例，手动指定标签
ax.legend(handles=[line1, line2], labels=['Sine Function', 'Cosine
Function'],loc='center')
# 显示图形
pl.show()
```

也可以将label定义在plot方法中，调用legend方法时不用再定义labels，会自动添加label

```
x = pl.linspace(0, 10, 100)
y1 = pl.sin(x)
y2 = pl.cos(x)

# 创建图形和轴域
fig, ax = pl.subplots()

# 绘制数据
line1, = ax.plot(x, y1, label='Sine Function')
line2, = ax.plot(x, y2, label='Cosine Function')

# 添加图例，手动指定标签
ax.legend(handles=[line1, line2], loc='center')
```

```
# 显示图形
pl.show()
```

legend() 函数 loc 参数：

位置	字符串表示	整数数字表示
自适应	Best	0
右上方	upper right	1
左上方	upper left	2
左下	lower left	3
右下	lower right	4
右侧	right	5
居中靠左	center left	6
居中靠右	center right	7
底部居中	lower center	8
上部居中	upper center	9
中部	center	10

6.3 标题中文乱码

如果标题设置的是中文，会出现乱码

局部处理：

```
plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus']=False
```

全局处理：

首先，找到 `matplotlibrc` 文件的位置，可以使用以下代码：

```
import matplotlib
print(matplotlib.matplotlib_fname())
```

然后，修改 `matplotlibrc` 文件，找到 `font.family` 和 `font.sans-serif` 项，设置为支持中文的字体，如 SimHei。

同时，设置 `axes.unicode_minus` 为 `False` 以正常显示负号。

修改完成后，重启pyCharm。如果不能解决，尝试运行以下代码来实现：

```
from matplotlib.font_manager import _rebuild
_rebuild()
```

6.4 subplot 函数

subplot 是一个较早的函数，用于创建并返回一个子图对象。它的使用比较简单，通常用于创建网格状的子图布局。subplot 的参数通常是一个三位数的整数，其中每个数字代表子图的行数、列数和子图的索引。

add_subplot 是一个更灵活的函数，它是 Figure类的一个方法，用于向图形容器中添加子图。推荐使用 add_subplot，因为它提供了更好的灵活性和控制。

语法：

```
fig.add_subplot(nrows, ncols, index)
```

案例：

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

# 创建数据
x = np.linspace(0, 10, 100)
y1 = np.sin(x)
y2 = np.cos(x)
y3 = np.tan(x)

# 创建图形
fig = plt.figure(figsize=(12, 4))

# 第一个子图
ax1 = fig.add_subplot(1, 3, 1)
ax1.plot(x, y1, label='sin(x)')
ax1.set_title('Sine Wave')
ax1.set_xlabel('X-axis')
ax1.set_ylabel('Y-axis')
ax1.legend()

# 第二个子图
ax2 = fig.add_subplot(1, 3, 2)
ax2.plot(x, y2, label='cos(x)')
ax2.set_title('Cosine Wave')
ax2.set_xlabel('X-axis')
ax2.set_ylabel('Y-axis')
ax2.legend()

# 第三个子图
ax3 = fig.add_subplot(1, 3, 3)
ax3.plot(x, y3, label='tan(x)')
ax3.set_title('Tangent Wave')
ax3.set_xlabel('X-axis')
ax3.set_ylabel('Y-axis')
ax3.legend()

# 显示图形
plt.tight_layout()
plt.show()
```

6.5 subplots 函数

subplots 是 matplotlib.pyplot 模块中的一个函数，用于创建一个包含多个子图（subplots）的图形窗口。subplots 函数返回一个包含所有子图的数组，这使得你可以更方便地对每个子图进行操作。

语法：

```
fig, axs = plt.subplots(nrows, ncols, figsize=(width, height))
```

参数：

- nrows: 子图的行数。
- ncols: 子图的列数。
- figsize: 图形的尺寸，以英寸为单位。

案例：

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

# 创建数据
x = np.linspace(0, 10, 100)
y1 = np.sin(x)
y2 = np.cos(x)
y3 = np.tan(x)

# 创建图形和子图
fig, axs = plt.subplots(1, 3, figsize=(12, 4))

# 第一个子图
axs[0].plot(x, y1, label='sin(x)')
axs[0].set_title('Sine Wave')
axs[0].set_xlabel('X-axis')
axs[0].set_ylabel('Y-axis')
axs[0].legend()

# 第二个子图
axs[1].plot(x, y2, label='cos(x)')
axs[1].set_title('Cosine Wave')
axs[1].set_xlabel('X-axis')
axs[1].set_ylabel('Y-axis')
axs[1].legend()

# 第三个子图
axs[2].plot(x, y3, label='tan(x)')
axs[2].set_title('Tangent Wave')
axs[2].set_xlabel('X-axis')
axs[2].set_ylabel('Y-axis')
axs[2].legend()

# 显示图形
plt.tight_layout()
plt.show()
```

6.6 subplot2gird 函数

subplot2grid 是 matplotlib.pyplot 模块中的一个函数，用于在网格中创建子图。subplot2grid 允许你更灵活地指定子图的位置和大小，以非等分的形式对画布进行切分，使得你可以创建复杂的布局。

语法:

```
ax = plt.subplot2grid(shape, loc, rowspan=1, colspan=1)
```

参数:

- shape: 网格的形状，格式为 (rows, cols)，表示网格的行数和列数，在figure中式全局设置。
- loc: 子图的起始位置，格式为 (row, col)，表示子图在网格中的起始行和列。
- rowspan: 子图占据的行数，默认为 1。
- colspan: 子图占据的列数，默认为 1。

案例:

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

# 创建数据
x = np.linspace(0, 10, 100)
y1 = np.sin(x)
y2 = np.cos(x)
y3 = np.tan(x)
y4 = np.exp(x)

fig = plt.figure(figsize=(12, 8))

# 第一个子图
ax1 = plt.subplot2grid((3, 3), (0, 0), colspan=2)
ax1.plot(x, y1, label='sin(x)')
ax1.set_title('Sine Wave')
ax1.set_xlabel('X-axis')
ax1.set_ylabel('Y-axis')
ax1.legend()

# 第二个子图
ax2 = plt.subplot2grid((3, 3), (0, 2))
ax2.plot(x, y2, label='cos(x)')
ax2.set_title('Cosine Wave')
ax2.set_xlabel('X-axis')
ax2.set_ylabel('Y-axis')
ax2.legend()

# 第三个子图
ax3 = plt.subplot2grid((2, 2), (1, 0))
ax3.plot(x, y3, label='tan(x)')
ax3.set_title('Tangent Wave')
ax3.set_xlabel('X-axis')
ax3.set_ylabel('Y-axis')
ax3.legend()

# 第四个子图
ax4 = plt.subplot2grid((2, 2), (1, 1))
ax4.plot(x, y4, label='exp(x)')
```

```
ax4.set_title('Exponential Wave')
ax4.set_xlabel('X-axis')
ax4.set_ylabel('Y-axis')
ax4.legend()

# 显示图形
plt.tight_layout()
plt.show()
```

6.7 grid 函数

grid 是用于在图形中添加网格线的函数。网格线可以帮助读者更清晰地理解数据的分布和趋势。grid 函数可以应用于 Axes 对象，用于在子图中添加网格线。

语法:

```
ax.grid(b=None, which='major', axis='both', **kwargs)
```

参数:

- b: 是否显示网格线，默认为 None，表示根据当前设置显示或隐藏网格线。
- which: 指定要显示的网格线类型，可以是 'major'（主刻度）、'minor'（次刻度）或 'both'（主刻度和次刻度）。
- axis: 指定要显示网格线的轴，可以是 'both'（两个轴）、'x'（X 轴）或 'y'（Y 轴）。
- **kwargs: 其他可选参数，用于定制网格线的外观，如 color、linestyle、linewidth 等。

案例:

```
# 引入 pyplot
from matplotlib import pyplot as plt
# 引入 numpy
import numpy as np

x = np.linspace(0, 10, 100)
y = np.sin(x)

fig = plt.figure()

axs = fig.add_axes([0.1, 0.1, 0.8, 0.8])
axs.plot(x, y, label='sin(x)')
axs.set_xlabel('X-axis')
axs.set_ylabel('Y-axis')
axs.set_title('Sin Function')
axs.legend(loc='upper center')
axs.grid(True, axis='x', color='red', ls='--', lw=0.5)

plt.show()
```

6.8 xscale 和 yscale 函数

xscale 和 yscale 函数用于设置坐标轴的刻度类型。默认情况下，坐标轴的刻度类型是线性的，但你可以使用 xscale 和 yscale 函数将其更改为对数刻度或其他类型的刻度。

语法:

```
ax.set_xscale(value)
ax.set_yscale(value)
```

参数:

value: 刻度类型，可以是 'linear'（线性刻度）、'log'（对数刻度）、'symlog'（对称对数刻度）、'logit'（对数几率刻度）等。

案例:

```
# 引入 pyplot
from matplotlib import pyplot as plt
# 引入 numpy
import numpy as np

# 创建数据
x = np.linspace(0, 10, 100)
y = np.exp(x)

# 创建图形和子图
fig, ax = plt.subplots()

# 绘制数据
ax.plot(x, y, label='exp(x)')

# 设置标题和标签
ax.set_title('Exponential Function')
ax.set_xlabel('X-axis')
ax.set_ylabel('Y-axis')

# 添加图例
ax.legend()

ax.set_yscale('log')

# 显示图形
plt.show()
```

6.9 set_xlim 和 set_ylim 函数

set_xlim 和 set_ylim 函数用于设置坐标轴的范围。

语法:

```
ax.set_xlim(left, right)
ax.set_ylim(bottom, top)
```

参数:

- left 和 right: X 轴的范围，left 是 X 轴的最小值，right 是 X 轴的最大值。
- bottom 和 top: Y 轴的范围，bottom 是 Y 轴的最小值，top 是 Y 轴的最大值。

案例:

```
# 引入 pyplot
from matplotlib import pyplot as plt
```

```
# 引入 numpy
import numpy as np

import math

# 创建数据
x = np.linspace(0, 10, 100)
y = np.sin(x)

# 创建图形和子图
fig, ax = plt.subplots()

# 绘制数据
ax.plot(x, y, label='sin(x)')

# 设置标题和标签
ax.set_title('Sine Wave')
ax.set_xlabel('X-axis')
ax.set_ylabel('Y-axis')

# 设置 X 轴和 Y 轴的范围
ax.set_xlim(0, 2*math.pi)
ax.set_ylim(-1.5, 1.5)

# 添加图例
ax.legend()
# 显示图形
plt.show()
```

6.10 set_xticks 和 set_yticks 函数

Matplotlib 可以自动根据因变量和自变量设置坐标轴范围，也可以通过 set_xticks() 和 set_yticks() 函数手动指定刻度，接收一个列表对象作为参数，列表中的元素表示对应数轴上要显示的刻度。

语法:

```
ax.set_xticks(ticks)
ax.set_yticks(ticks)
```

参数:

ticks: 一个包含刻度位置的列表或数组。

案例:

```
# 引入 pyplot
from matplotlib import pyplot as plt
# 引入 numpy
import numpy as np
# 数学库
import math

# 创建数据
x = np.linspace(0, 10, 100)
y = np.sin(x)
```

```
# 创建图形和子图
fig, ax = plt.subplots()

# 绘制数据
ax.plot(x, y, label='sin(x)')

# 设置标题和标签
ax.set_title('Sine Wave')
ax.set_xlabel('X-axis')
ax.set_ylabel('Y-axis')

# 设置 X 轴和 Y 轴的刻度位置
ax.set_xticks([0, 2, 4, 6, 8, 10])
ax.set_yticks([-1, -0.5, 0, 0.5, 1])

# 添加图例
ax.legend()

# 显示图形
plt.show()
```

6.11 twinx 和 twiny 函数

twinx 和 twiny 函数用于在同一个图形中创建共享 X 轴或 Y 轴的多个子图。twinx 函数用于创建共享 X 轴的子图，twiny 函数用于创建共享 Y 轴的子图。

语法:

```
ax2 = ax.twinx()
ax2 = ax.twiny()
```

说明:

- ax: 原始的 Axes 对象。
- ax2: 新的 Axes 对象，共享原始 Axes 对象的 X 轴或 Y 轴。

案例:

```
# 引入 pyplot
from matplotlib import pyplot as plt
# 引入 numpy
import numpy as np


# 创建数据
x = np.linspace(0, 10, 100)
y1 = np.sin(x)
y2 = np.exp(x)

# 创建图形和子图
fig, ax1 = plt.subplots()

# 绘制第一个数据集
ax1.plot(x, y1, 'b-', label='sin(x)')
ax1.set_xlabel('X-axis')
ax1.set_ylabel('sin(x)', color='b')
```

```
# 创建共享 X 轴的子图
ax2 = ax1.twinx()

# 绘制第二个数据集
ax2.plot(x, y2, 'r-', label='exp(x)')
ax2.set_ylabel('exp(x)', color='r')

ax1.legend()
ax2.legend()

# 显示图形
plt.show()
```

6.12 柱状图

柱状图（Bar Chart）是一种常用的数据可视化工具，用于展示分类数据的分布情况。

语法:

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

参数:

- x: 柱状图的 X 轴位置。
- height: 柱状图的高度。
- width: 柱状图的宽度，默认为 0.8。
- bottom: 柱状图的底部位置，默认为 0。
- align: 柱状图的对齐方式，可以是 'center'（居中对齐）或 'edge'（边缘对齐）。
- **kwargs: 其他可选参数，用于定制柱状图的外观，如 color、edgecolor、linewidth 等。

案例1:

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


# 数据
categories = ['A', 'B', 'C', 'D']
values = [20, 35, 30, 25]

# 创建图形和子图
fig, ax = plt.subplots()

# 绘制柱状图
ax.bar(categories, values, color='skyblue', linewidth=1.5, width=0.6)

# 设置标题和标签
ax.set_title('Customized Bar Chart')
ax.set_xlabel('Categories')
ax.set_ylabel('Values')

# 显示图形
plt.show()
```

案例2：堆叠柱状图

```
# 数据
categories = ['A', 'B', 'C', 'D']
values1 = [20, 35, 30, 25]
values2 = [15, 25, 20, 10]

# 创建图形和子图
fig, ax = plt.subplots()

# 绘制第一个数据集的柱状图
ax.bar(categories, values1, color='skyblue', label='Values 1')

# 绘制第二个数据集的柱状图，堆叠在第一个数据集上
ax.bar(categories, values2, bottom=values1, color='lightgreen', label='Values
2')

# 设置标题和标签
ax.set_title('Stacked Bar Chart')
ax.set_xlabel('Categories')
ax.set_ylabel('Values')

# 添加图例
ax.legend()

# 显示图形
plt.show()
```

说明：

bottom=values1：绘制第二个数据集的柱状图，堆叠在第一个数据集上

案例3：分组柱状图

```
# 数据
categories = ['A', 'B', 'C', 'D']
values1 = [20, 35, 30, 25]
values2 = [15, 25, 20, 10]

# 创建图形和子图
fig, ax = plt.subplots()

# 计算柱状图的位置
x = np.arange(len(categories))
width = 0.35

# 绘制第一个数据集的柱状图
ax.bar(x - width/2, values1, width, color='skyblue', label='Values 1')

# 绘制第二个数据集的柱状图
ax.bar(x + width/2, values2, width, color='lightgreen', label='Values 2')

# 设置 X 轴标签
ax.set_xticks(x)
ax.set_xticklabels(categories)

# 设置标题和标签
ax.set_title('Grouped Bar Chart')
ax.set_xlabel('Categories')
```

```
ax.set_ylabel('Values')

# 添加图例
ax.legend()

# 显示图形
plt.show()
```

6.13 直方图

直方图（Histogram）是一种常用的数据可视化工具，用于展示数值数据的分布情况。

语法:

```
ax.hist(x, bins=None, range=None, density=False, weights=None, cumulative=False, **kwargs)
```

参数:

- x: 数据数组。
- bins: 直方图的柱数，可以是整数或序列。
- range: 直方图的范围，格式为 `(min, max)`。
- density: 是否将直方图归一化，默认为 `False`。
- weights: 每个数据点的权重。
- cumulative: 是否绘制累积直方图，默认为 `False`。
- **kwargs: 其他可选参数，用于定制直方图的外观，如 `color`、`edgecolor`、`linewidth` 等。

案例:

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

# 生成随机数据，生成均值为 0，标准差为 1 的标准正态分布的随机样本
data = np.random.randn(1000)

# 创建图形和子图
fig, ax = plt.subplots()

# 绘制直方图
ax.hist(data, bins=30, color='skyblue', edgecolor='black')

# 设置标题和标签
ax.set_title('Simple Histogram')
ax.set_xlabel('Value')
ax.set_ylabel('Frequency')

# 显示图形
plt.show()
```

6.14 饼图

饼图（Pie Chart）是一种常用的数据可视化工具，用于展示分类数据的占比情况。

语法:

```
ax.pie(x, explode=None, labels=None, colors=None, autopct=None, shadow=False, startangle=0, **kwargs)
```

参数:

- x: 数据数组，表示每个扇区的占比。
- explode: 一个数组，表示每个扇区偏离圆心的距离，默认为 `None`。
- labels: 每个扇区的标签，默认为 `None`。
- colors: 每个扇区的颜色，默认为 `None`。
- autopct: 控制显示每个扇区的占比，可以是格式化字符串或函数，默认为 `None`。
- shadow: 是否显示阴影，默认为 `False`。
- startangle: 饼图的起始角度，默认为 0。
- **kwargs: 其他可选参数，用于定制饼图的外观。

案例:

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

# 数据
labels = ['A', 'B', 'C', 'D']
sizes = [15, 30, 45, 10]

# 创建图形和子图
fig, ax = plt.subplots()

# 绘制饼图
ax.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90)

# 设置标题
ax.set_title('Simple Pie Chart')

# 显示图形
plt.show()
```

6.15 折线图

使用 plot 函数

案例:

```
from matplotlib import pyplot as plt

# 创建数据
x = np.linspace(0, 10, 100)
y1 = np.sin(x)
y2 = np.cos(x)

# 创建图形和子图
fig, ax = plt.subplots()

# 绘制多条折线图
ax.plot(x, y1, label='sin(x)', color='blue')
ax.plot(x, y2, label='cos(x)', color='red')
```

```
# 设置标题和标签
ax.set_title('Multiple Line Charts')
ax.set_xlabel('X-axis')
ax.set_ylabel('Y-axis')

# 添加图例
ax.legend()

# 显示图形
plt.show()
```

6.16 散点图

散点图（Scatter Plot）是一种常用的数据可视化工具，用于展示两个变量之间的关系。

语法:

```
ax.scatter(x, y, s=None, c=None, marker=None, cmap=None, norm=None, vmin=None,
vmax=None, alpha=None, linewidths=None, edgecolors=None, **kwargs)
```

参数:

- x: X 轴数据。
- y: Y 轴数据。
- s: 点的大小，可以是标量或数组。
- c: 点的颜色，可以是标量、数组或颜色列表。
- marker: 点的形状，默认为 'o'（圆圈）。
- cmap: 颜色映射，用于将颜色映射到数据。
- norm: 归一化对象，用于将数据映射到颜色映射。
- vmin, vmax: 颜色映射的最小值和最大值。
- alpha: 点的透明度，取值范围为 0 到 1。
- linewidths: 点的边框宽度。
- edgecolors: 点的边框颜色。
- **kwargs: 其他可选参数，用于定制散点图的外观。

案例:

```
from matplotlib import pyplot as plt

plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False
fig = plt.figure()
axes = fig.add_axes([.1,.1,.8,.8])
x = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun']
data = [
    [120, 132, 101, 134, 90, 230, 210],
    [220, 182, 191, 234, 290, 330, 310],
]
y0 = data[0]
y1 = data[1]
axes.scatter(x,y0,color='red')
axes.scatter(x,y1,color='blue')
axes.set_title('散点图')
axes.set_xlabel('日期')
axes.set_ylabel('数量')
```

```
plt.legend(labels=['Email', 'Union Ads'],)
plt.show()
```

marker常用的参数值：

- 'o': 圆圈
- 's': 正方形
- 'D': 菱形
- '^': 上三角形
- 'v': 下三角形
- '>': 右三角形
- '<': 左三角形
- 'p': 五边形
- '*': 星形
- '+': 加号
- 'x': 叉号
- '.': 点
- ',': 像素
- '1': 三叉戟下
- '2': 三叉戟上
- '3': 三叉戟左
- '4': 三叉戟右
- 'h': 六边形1
- 'H': 六边形2
- 'd': 小菱形
- '|': 竖线
- '_': 横线