ChatGPT

Matplotlib 面向对象绘图接口教程：Figure 和 Axes 使用指南

plt.subplots() 返回 Figure 和 Axes 的原因与结构关系

使用 Matplotlib 的 `plt.subplots()` 会一次性创建**图形（Figure）**和**轴域（Axes）**对象，并返回它们 [1]。Figure 相当于整张画布或绘图窗口，是所有可视化元素的顶级容器；Axes 则是图中实际绘图的子区（通常也叫“子图”），包含了坐标系（即 x 轴和 y 轴）以及各种绘制的元素 [2]。一个 Figure 可以包含多个 Axes（如多子图布局），每个 Axes 相当于一张独立的子图。

Figure 和 Axes 之间通过**容器-内容**关系协作：Figure 负责整个图形级别的设置（如图像大小、保存输出等），而大部分绘图操作都发生在 Axes 上。例如，创建 Axes 后，可以调用其方法绘制数据（如 `ax.plot()`）、设置坐标标签和标题（如 `ax.set_xlabel()`、`ax.set_title()`）等。Axes 内部还维护着两个 Axis 对象用于处理刻度和坐标范围，但这些通常通过 Axes 的方法间接控制。除了最顶层的 Figure 以外，几乎所有可见的元素都属于某个 Axes——因此 Axes 被称为 Matplotlib 面向对象接口的“网关”，通过它可以访问和控制图中的大部分内容。

简单来说，`plt.subplots()` 返回 Figure 和 Axes，便于我们分别操作图层次属性和轴层次属性。例如：

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()  # 创建 Figure 和 Axes
ax.plot([0, 1, 2], [1, 2, 3])  # 在 Axes 上绘图
ax.set_xlabel('X 轴标签')       # 设置 Axes 的 X 轴标签
fig.suptitle('图形标题')        # 设置整个 Figure 的标题
plt.show()
```

上例中，`fig` 是 Figure 对象，用于设置整张图的标题、大小或保存文件等；`ax` 是 Axes 对象，用于绘制数据和设置坐标系属性。这样的返回设计使代码更加清晰：我们可以直接使用 `fig` 和 `ax` 分别管理不同层次的元素 [1]。

> **提示：**调用 `plt.subplots()` 不传参数时会创建包含单个 Axes 的 Figure [1]；这是一种推荐的用法，可确保我们明确获取 Figure 和 Axes 对象进行后续操作。

Figure 对象及其常用方法

Figure 对象代表整个绘图窗口或画布。创建 Figure 的常见方式是使用 `plt.figure()` 或 `plt.subplots()`（二者都会产生 Figure）。获得 Figure 实例后，我们可以使用其方法来添加子图、调整图形、保存输出等。下面总结 Figure 最常用的几种方法：

- **`Figure.add_subplot()`：添加子图 Axes** – 在当前 Figure 中按照网格布局添加一个子图，并返回一个 Axes 对象 [3]。基本调用形式包括：

  ```
  ax = fig.add_subplot(nrows, ncols, index, **kwargs)
  ```

其中 `nrows`, `ncols` 指定子图网格的行列数，`index` 指定放置位置（从1开始，行优先）[4]。例如，`fig.add_subplot(2, 3, 5)` 会在 Figure 的2行×3列网格中添加位于第5个位置的子图[5]。该方法也支持传入一个三位整数如 `fig.add_subplot(235)`，含义同上[6]。返回的 Axes 实际类型为 `AxesSubplot`（详见下文），你可以直接对其绘图。

```
fig = plt.figure()
ax1 = fig.add_subplot(2, 2, 1)  # 添加2x2网格中的第1个子图
ax2 = fig.add_subplot(2, 2, 2)  # 添加第2个子图
ax1.plot([...])  # 在第一个 Axes 上绘图
```

- **`Figure.add_axes()`：手动添加轴域** – 在 Figure 中按给定的位置和大小添加一个 Axes。调用时需要提供一个 `[left, bottom, width, height]` 列表，定义Axes相对于Figure的坐标（0~1的比例）和尺寸。例如：

  ```
  ax = fig.add_axes([0.1, 0.1, 0.8, 0.8])  # 在Figure中添加占据80%宽高的Axes
  ```

  以上将在 Figure 内部 (10%偏移左和下边界) 添加一个Axes，大小为 Figure 的80%。这种方法适合精确布置Axes的位置，例如在主图中插入小图（嵌入轴）等。

- **`Figure.subplots()`：快捷创建网格子图** – Figure 实例也提供了 `subplots()` 方法，其用法类似于 `plt.subplots`，可以一次性在当前 Figure 中生成一个网格布局的多个 Axes。例如：

  ```
  fig = plt.figure()
  axs = fig.subplots(2, 2)  # 在已有Figure中创建2x2的Axes数组
  ```

  这将返回形状为2x2的 Axes 数组 `axs`。需要注意，通常我们直接使用 `plt.subplots()` 创建 Figure，因此较少单独使用 `Figure.subplots()` 方法。

- **`Figure.suptitle()`：设置整体标题** – 给 Figure 增加一个总体标题，位于所有子图之上[7]。调用例如：

  ```
  fig.suptitle('整体标题', fontsize=16)
  ```

  可选参数如 `fontsize` 控制标题字体大小等。`suptitle` 常用于当 Figure 包含多子图时，为整个图形添加描述标题。

- **`Figure.savefig()`：保存图像文件** – 将当前 Figure 保存为图像或矢量图文件[8]。基本用法：

  ```
  fig.savefig('output.png', dpi=300, transparent=True)
  ```

  参数说明：`fname` 为文件名或路径，后缀决定格式（如 `.png`, `.pdf`, `.svg` 等）；`dpi` 指定分辨率（默认使用 Figure 自身的 dpi 设置）[9]；`transparent=True` 时保存的图片背景透明[10]；`bbox_inches='tight'` 可以剪除图形外多余空白边距。[11] 此外 `facecolor`, `edgecolor` 可设置保存图的背景和边框颜色等。savefig 是最终生成高质量输出的重要方法。

- **`Figure.set_size_inches()`：调整图形尺寸** – 设置 Figure 的宽度和高度（单位为英寸） [12] 。例如：

  ```
  fig.set_size_inches(6, 4)  # 将图形尺寸设为6x4英寸
  ```

  默认 `forward=True` 表示同时更新 Figure 的画布对象以实时生效 [13] 。等价的还有 `fig.set_figwidth()` 和 `fig.set_figheight()` 可分别调整宽或高。使用该方法可以精确控制输出图像大小（例如投稿期刊要求的尺寸），尤其结合保存dpi可以控制像素大小。

- **其他常用方法**：Figure 对象还有许多其他方法，例如：

- `fig.subplots_adjust(left, right, top, bottom, wspace, hspace)` 调整子图间距；
- `fig.tight_layout()` 自动紧凑布局，避免标签重叠；
- `fig.legend()` 在整张图上添加图例（汇总多个子图的元素） [14] ；
- `fig.colorbar(mappable, ax=...)` 为指定 Axes 添加颜色条；
- `fig.text(x, y, s, **kwargs)` 在 Figure 上添加文本（相对整幅图的坐标）；
- `fig.canvas.mpl_connect(event, callback)` 绑定交互事件（详见下文交互部分）。

上述方法使我们能够灵活控制图形级别的布局和输出。例如，在多个子图场景下，常用 `fig.suptitle()` 添加总标题，`fig.legend()` 汇总各子图图例，`fig.savefig()` 导出成品等。

Axes 对象及其常用方法

Axes 对象表示绘图区域（子图），提供了大量方法用于绘制各种数据、设置坐标轴和注解信息 [2] 。拿到 Axes 实例（通常通过 `plt.subplots()` 或 `fig.add_subplot()` 获得）后，我们一般对其调用方法完成绘图任务。下面按用途分类，介绍 Axes 常用的方法及参数。

数据绘制方法（Plotting）

Axes 提供了许多绘图方法来生成不同类型的图表。常用的包括折线图、散点图、条形图、图像等：

- **`Axes.plot()`：绘制折线图或散点线图** – 以连线或标记的形式绘制 y 对 x 的曲线 [15] 。典型调用：

  ```
  ax.plot(x, y, fmt, **kwargs)
  ```

  其中 `x`、`y` 为数据序列（如果只传一个参数则视为 y，x 默认为序号索引），`fmt` 是可选的格式字符串，如 `'ro--'` 表示红色虚线带圆点标记 [16] 。`**kwargs` 接受 Line2D 对象的属性，如 `color='green'`，`linewidth=2`，`marker='o'`，`linestyle='--'` 等，用于定制线条样式 [17] 。**返回值**是包含所绘制线条的列表（通常一条线对应一个 `matplotlib.lines.Line2D` 对象）。如果一次调用传入多组 x、y 数据，则会绘制多条线 [18] [19] 。

示例：

```
x = np.linspace(0, 2*np.pi, 100)
y = np.sin(x)
ax.plot(x, y, 'b--', linewidth=2, label='sin(x)')  # 绘制sin曲线，蓝色虚线，标签用于图例
```

上例使用 `Axes.plot` 绘制了正弦波曲线，并设置了线型、颜色和标签。

- **`Axes.scatter()`：绘制散点图** – 绘制 x-y 平面上的一系列离散点 [20]。典型调用：

  ```
  ax.scatter(x, y, s=size, c=color, marker=marker, alpha=透明度, **kwargs)
  ```

  `x`、`y`为数据点坐标序列，`s`控制点的大小（默认单位是点的面积）[21]；`c`可以是单一颜色或颜色序列/数组，用于给不同点着色 [22]；`marker`参数指定点的形状，比如`'o'`圆点、`'^'`三角等；`alpha`控制透明度。除此之外也能使用`cmap`映射数值到颜色，`edgecolors`设置点边框色，`linewidths`设置边框宽度等 [23] [22]。**返回值**是一个`PathCollection`对象，表示散点集合。

示例：

```
sizes = np.random.rand(len(x)) * 100  # 每个点随机大小
colors = np.random.rand(len(x))       # 每个点对应颜色值
ax.scatter(x, y, s=sizes, c=colors, cmap='viridis', marker='^', alpha=0.7)
# 绘制散点图，使用尺寸数组和以viridis颜色映射着色的不透明三角形点
```

- **`Axes.bar()`：绘制条形图** – 在 x 指定的位置绘制垂直柱形条 [24]。基本调用：

  ```
  ax.bar(x, height, width=0.8, bottom=0, align='center', **kwargs)
  ```

  `x`可以是类目标签序列或数值型位置 [25]；`height`为每个柱子的高度 [26]；`width`指定柱宽（标量或序列） [27]，默认为0.8；`bottom`指定柱子的起始高度（默认为0，即从0开始向上） [28]；`align`决定`x`所给位置是柱子的中心还是边缘 [29]（`'center'`或`'edge'`）。该方法返回一个`BarContainer`对象，其中包含了所有`Rectangle`条形实例。常用参数还包括：`color`/`facecolor`设置柱体颜色，`edgecolor`边框颜色，`linewidth`边框宽度，`tick_label`自定义每个柱的刻度标签，`label`为整个柱集合指定标签以用于图例 [30] [31]。

示例：

```
categories = ['A', 'B', 'C']
values = [5, 8, 3]
ax.bar(categories, values, color='skyblue', edgecolor='gray')
# 绘制类别A/B/C对应的柱状图
```

- **`Axes.imshow()`：显示矩阵图像** – 在 Axes 上显示二维数据（如图像） [32]。调用形式：

  ```
  ax.imshow(X, cmap=None, norm=None, aspect=None, interpolation=None, origin=None, **kwargs)
  ```

  其中`X`为要显示的二维数组或图像数据（形状可为 M×N 的标量数据，或 M×N×3/4 的RGBA图像） [33]。标量数组会根据`cmap`映射为颜色图显示 [34]；可以通过`vmin`/`vmax`设置映射的数值范围 [35]。`aspect`控制像素宽高比（如`'equal'`保持1:1像素， [36]），`interpolation`控制插值方法（如`'nearest'`, `'bilinear'`等， [37]），`origin='lower'`可使数组第[0,0]元素显示在左下角（默认为左上）。**返回值**是一个`AxesImage`对象，表示显示的图像。常用于显示图片或矩阵形式的数据，如热图等。

示例：

```
matrix = np.random.rand(10, 10)
ax.imshow(matrix, cmap='viridis', interpolation='nearest')
ax.set_title('Heatmap')
```

上述方法之外，Axes 还支持绘制更多类型的图，例如 `ax.hist()` 绘制直方图，`ax.boxplot()` 箱线图，`ax.fill_between()` 填充曲线之间区域，`ax.contour()` 等高线图，`ax.pcolormesh()` 伪彩色网格等等。这些都是构建在 Axes 基础上的高级方法，这里不一一展开。

坐标轴设置与注解方法

绘制图形后，我们经常需要添加标题、坐标轴标签、调节刻度范围、添加图例和标注说明等。Axes 提供了一系列方法来设置这些元素：

- **`Axes.set_title()`：设置子图标题** – 在 Axes 顶部添加标题文字 [38]。调用：

  ```
  ax.set_title(label, fontdict=None, loc=None, pad=None, **kwargs)
  ```

  `label` 为标题文本字符串。可选参数：`loc` 控制标题位置（`'center'` 居中 [39]、`'left'` 左对齐、`'right'` 右对齐），默认居中；`pad` 标题与 Axes 上边界的距离（以磅为单位，默认约6磅）[40]；`fontdict` 可提供字典设置文本字体属性（不推荐直接用，官方建议用关键字参数传字体属性）[41]。其他文本属性如 `fontsize`, `color`, `fontweight` 等也可作为 kwargs 传入。此方法返回一个 `matplotlib.text.Text` 对象代表标题文本。

```
ax.set_title('子图标题', fontsize=12, loc='left')
```

（如上将标题设为左对齐，字号12）

- **`Axes.set_xlabel()` / `Axes.set_ylabel()`：设置坐标轴标签** – 为 x轴或y轴添加描述标签 [42]。用法类似：

  ```
  ax.set_xlabel('X轴名称', labelpad=值, loc='center')
  ```

  其中 `labelpad` 控制标签与轴线的间距，默认为4磅左右 [43]；`loc` 可选 `'left'`, `'center'`, `'right'` 指定 x轴标签相对于轴的对齐方式（对于 y轴则有 `'bottom'`, `'center'`, `'top'`）[44]。大多数情况下直接传入文本即可，其他字体属性可用关键字参数调整：

```
ax.set_xlabel('时间 (s)', fontsize=10)
ax.set_ylabel('距离 (km)', fontsize=10)
```

- **`Axes.set_xlim()` / `Axes.set_ylim()`：设置坐标范围** – 调整 x轴或y轴显示的数值范围 [45]。常用形式：

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

其中 `left`/`bottom` 和 `right`/`top` 分别为坐标轴的最小值和最大值 [46]。如果传入 `None` 则表示不改变相应一侧的限制 [46]。也可一次传入元组，如 `ax.set_xlim((xmin, xmax))`。调用此方法会返回实际设置后的 `(min, max)` 元组。举例：

```
ax.set_xlim(0, 10)
ax.set_ylim(-1, 1)
```

这将固定 x 轴范围为 [0,10]，y 轴范围为 [-1,1]。

- **`Axes.legend()`：添加图例** – 在 Axes 上显示一块图例说明，用于标识各种绘制元素 [47]。两种常用方式：

- **自动模式**：直接调用 `ax.legend()`，Matplotlib 会自动搜集该 Axes 上有标签的艺术家（如通过 `label` 参数设置了标签的折线、散点等）来生成图例 [48]。凡是标签以“_”开头的艺术家将被忽略（默认不显示） [49]。通常我们在绘图方法（如 `plot`, `scatter`）里传入 `label='说明文本'`，然后调用一次 `ax.legend()` 即可生成图例 [50]。
- **手动模式**：调用 `ax.legend(handles=[list_of_artists], labels=[list_of_labels], loc=...)` [51]，显式指定哪些元素和对应标签进入图例。这在需要自定义图例内容或从多个Axes合并图例时有用。

`loc` 参数控制图例位置，可以使用字符串如 `'upper right'`, `'lower left'`, `'center'`, `'best'`（自动选择不遮挡数据的位置）等 [52] [53]。默认 loc='best'。 [54] 其他常用参数：`bbox_to_anchor` 可以将图例放置在轴域外部配合 `loc` 调整位置，`ncol` 设置图例列数，`fontsize` 图例字体大小等。legend 方法返回一个 `matplotlib.legend.Legend` 对象。

典型示例：

```
ax.plot(x, y1, label='数据1')
ax.plot(x, y2, label='数据2')
ax.legend(loc='upper left')  # 在左上角显示图例，包含“数据1”和“数据2”
```

- **`Axes.annotate()`：添加注解箭头** – 在图中指定位置添加文本注解，并可以用箭头指向某个数据点 [55]。调用格式较多样，常用形式：

```
ax.annotate(text, xy=(x, y), xytext=(x2, y2), arrowprops={'arrowstyle': '->', ...}, **kwargs)
```

基本参数：`text` 是要显示的字符串；`xy` 是注释指向的位置（数据坐标） [56]；可选的 `xytext` 指定文本放置的位置坐标，如果不提供则文本放在 `xy` 处 [57]。`arrowprops` 是一个字典，用于设置箭头的样式，如箭头类型 `'->'`、颜色 `'color'`、箭头头部尺寸 `'headlength'` 等。如果提供了 `arrowprops`，则在 `xytext` 和 `xy` 之间绘制箭头 [58]。还可以使用 `xycoords` 和 `textcoords` 参数来调整 `xy` 和文本坐标系（如数据坐标、轴坐标等） [59] [60]。高级用法可以参考官方文档关于注解部分 [61]。

示例：

```
ax.scatter(x, y)  # 画一些点
ax.annotate('重要点', xy=(x0, y0), xytext=(x0+0.5, y0+0.5),
            arrowprops=dict(arrowstyle='->', color='red'))
```

上述代码在数据点 `(x0, y0)` 处标注“重要点”文字，并用红色箭头从文本指向该点。

- **`Axes.text()`：添加文本** – 在 Axes 上的任意位置添加文本，不带箭头 [62]。调用：

  ```
  ax.text(x, y, s, fontdict=None, **kwargs)
  ```

  这里 `x, y` 是文本位置的坐标（默认为数据坐标系） [63]，`s` 是文本字符串内容。与 `annotate` 不同的是，`text()` 只是放置文字，没有指向箭头。常用参数和 `set_title` 类似，比如字体大小 `fontsize`，颜色 `color`，透明度 `alpha` 等均可通过kwargs传入。还可以通过参数 `transform=ax.transAxes` 等将坐标设为轴比例坐标（0~1范围）。

示例：

```
ax.text(0.5, 0.5, '中心点', transform=ax.transAxes,
        horizontalalignment='center', verticalalignment='center',
        fontsize=14, color='gray', alpha=0.5)
```

这将在 Axes 的中心位置（50%宽、50%高处）添加半透明的“中心点”文本。

以上方法让我们可以方便地美化和标注图表。例如，使用 `set_title` 和 `set_xlabel/ylabel` 可以为子图添加标题和坐标说明，使用 `legend` 可以区分不同数据系列，使用 `annotate` 和 `text` 可以强调特定数据点或添加额外说明，使图形更具信息性。

AxesSubplot 类型说明

需要注意，当我们通过 `plt.subplots()` 或 `fig.add_subplot()` 创建子图时，返回的 Axes 实例实则为 **`matplotlib.axes._subplots.AxesSubplot`** 类的对象。`AxesSubplot` 继承自 `Axes` [64]（因此具有前述 Axes 的所有方法），只是额外包含了子图在网格中的定位等信息。在使用上，这两者无差别；通常我们直接将其视为 Axes 来使用即可。例如：

```
fig, ax = plt.subplots()
print(type(ax))
# 输出: <class 'matplotlib.axes._subplots.AxesSubplot'>
```

如上，`ax` 的实际类型是 AxesSubplot。这只是 Matplotlib 内部实现的细节，一般并不影响我们调用 Axes 的各种方法。但了解这一点有助于理解文档和调试信息：很多情况下 “Axes” 可以是不同子类的实例（如极坐标投影时返回的是 `PolarAxesSubplot` 等），但它们对用户提供统一的接口 [64]。

Figure 和 Axes 对象的交互功能

Matplotlib 不仅可以创建静态图像，还支持与用户交互，例如响应鼠标点击、移动以及键盘按键等事件。Figure 提供了 `canvas.mpl_connect()` 方法来连接事件回调 [65] [66]。基本用法：

```
cid = fig.canvas.mpl_connect('事件名称', 回调函数)
```

- **事件名称**是字符串，例如：
- `'button_press_event'`：鼠标按下事件 [67]；
- `'button_release_event'`：鼠标释放；
- `'motion_notify_event'`：鼠标移动；
- `'key_press_event'`：键盘按键按下 [68]；
- `'pick_event'`：触发“选取”事件（需艺术家开启picker属性）；
- `'axes_enter_event'` / `'axes_leave_event'`：光标进入或离开某个 Axes [66] [69]；
- `'figure_enter_event'` / `'figure_leave_event'`：光标进入或离开 Figure 窗口 [65]。

- **回调函数**的定义需接受一个事件参数。Matplotlib 会在事件发生时调用此函数，并传入一个事件对象，其属性包含事件相关信息，如鼠标坐标（`event.xdata`, `event.ydata`），按键（`event.key`），发生事件的 Axes（`event.inaxes`）等。

例如，下面代码实现点击折线图时在控制台打印被点击的数据点坐标： [70] [71]

```
fig, ax = plt.subplots()
line, = ax.plot(np.random.rand(10), 'o-', picker=5)  # 开启picker，容差5pt
def on_pick(event):
    this_line = event.artist
    xdata, ydata = this_line.get_xdata(), this_line.get_ydata()
    ind = event.ind  # 被选中的点索引列表
    print("选中了点：", list(zip(xdata[ind], ydata[ind])))
fig.canvas.mpl_connect('pick_event', on_pick)
```

在上例中，`line` 设置了 `picker=5` 激活选取，允许鼠标在距离点5磅以内算作选中 [70]。当用户点击点附近时，会触发 `'pick_event'`，调用 `on_pick` 打印出选中点的数据坐标 [72]。

此外，还有一些Axes级别的交互，如放大缩小、平移等，这些通过 Matplotlib 的导航工具栏提供，不需要手动编码。但如果需要自定义交互行为，`mpl_connect` 是主要途径。例如，可以结合 `'motion_notify_event'` 实现鼠标跟踪显示坐标，结合 `'key_press_event'` 实现按键切换曲线等。

当不再需要监听某事件时，可以使用 `fig.canvas.mpl_disconnect(cid)` 通过连接id取消绑定。

> **提示：**Matplotlib 的交互功能需要在支持交互的后端（如TkAgg、Qt5Agg等）下运行，并通常在 `plt.show()` 后实时生效。在脚本中使用需确保程序不会立即退出（例如放在交互式环境或使用 `plt.pause()` 等）。

常用 Figure 与 Axes 方法速查表

下面的表格汇总了上述提到的 Figure 和 Axes 对象的常用方法、主要参数及简要说明，便于查阅。

Figure 常用方法：

方法名称	参数（主要）	作用说明
Figure.add_subplot	`nrows, ncols, index, **kwargs`	按网格布局添加子图，返回一个 AxesSubplot [3]。`nrows,ncols` 指定网格大小，`index` 从1开始指定位置。kwargs 可包括 `projection='polar'` 等投影类型。
Figure.add_axes	`[left, bottom, width, height]`	在指定位置手动添加 Axes。坐标以Figure的宽高比例表示(0~1)。适合自定义子图大小或嵌套子图。
Figure.subplots	`nrows=1, ncols=1, **kwargs`	创建网格子图并返回 Axes 对象数组。等价于 pyplot.subplots，但作用于已有Figure。常用参数有 `sharex`, `sharey`, `figsize` 等。
Figure.suptitle	`t, fontsize=None, x=0.5, y=0.98, **kwargs`	添加整个 Figure 的标题 [7]。`t` 为标题文本，`x,y` 指定标题相对Figure的位置（默认顶部居中）。
Figure.savefig	`fname, dpi=None, bbox_inches=None, transparent=False, **kwargs`	将当前 Figure 保存到文件 [8]。`fname` 文件名或路径，`dpi` 分辨率，`bbox_inches='tight'` 可自动剪除边界空白，`transparent` 是否背景透明。支持多种格式（根据后缀或 format 指定）。
Figure.set_size_inches	`w, h, forward=True`	设置 Figure 尺寸（宽、高，英寸） [12]。`forward=True` 时立即更新窗口大小。可用来在保存前调整图形尺寸。
Figure.tight_layout	无 or `pad=1.08, w_pad=None, h_pad=None, rect=None`	自动调整子图参数以紧凑排列子图，避免标签重叠。`pad` 为图像边缘留白比例。等效于 plt.tight_layout但针对当前Figure。
Figure.subplots_adjust	`left=None, right=None, top=None, bottom=None, wspace=None, hspace=None`	手动调整子图间隔和边距。参数取0~1的比例，控制Figure边缘到子图的距离（left等）或子图间水平/垂直间距（wspace,hspace）。

Axes 常用方法：

方法名称	参数（主要）	作用说明
Axes.plot	`x, y=None, fmt='', data=None, **kwargs`	绘制折线或标记图 [73]。`x,y` 为数据序列，`fmt` 可选格式字符串（如 `'ro-'`），kwargs 可指定 `color, linewidth, marker` 等 Line2D 属性。返回 Line2D 列表。
Axes.scatter	`x, y, s=None, c=None, marker=None, cmap=None, **kwargs`	绘制散点图 [20]。`s` 控制点大小，`c` 指定颜色序列或单色（可结合 `cmap`），`marker` 点形状。支持许多 `Collection` 属性如透明度alpha、边框edgecolors等。返回 PathCollection。

方法名称	参数（主要）	作用说明
Axes.bar	`x, height, width=0.8, bottom=0, align='center', **kwargs`	绘制垂直条形图[24]。`x` 为条位置（类别或数值），`height` 为高度，`width` 宽度，`bottom` 起始高度（可用于堆叠图），`align` 柱对齐方式。kwargs 常用 `color` / `edgecolor` 等。返回 BarContainer（包含一组 Rectangle）。
Axes.imshow	`X, cmap=None, norm=None, aspect=None, interpolation=None, origin=None, **kwargs`	显示二维数据为图像[32]。`X` 为数组或图像，`cmap` 指定颜色映射（若 X 为标量矩阵），`aspect` 控制宽高比（'equal'等），`interpolation` 插值方式（'nearest' 等），`origin` 指定[0,0]在左上还是左下。返回 AxesImage。
Axes.set_title	`label, fontdict=None, loc=None, pad=None, **kwargs`	设置Axes标题[38]。`label` 标题文本，`loc` 位置（'center'默认，或'left','right'），`pad` 与顶部距离（磅）。kwargs 可传入字体大小颜色等(Text属性)。
Axes.set_xlabel	`xlabel, labelpad=None, loc=None, **kwargs`	设置X轴标签[42]。`labelpad` 标签与轴距离，`loc` 水平对齐方式（'center'等）。kwargs支持Text属性调整字体。`set_ylabel` 用法类似（loc 支持 'bottom','top' 等）。
Axes.set_xlim	`left=None, right=None, **kwargs`	设置X轴显示范围[46]。`left,right` 分别为下限上限，传 None 表示保持当前值。`Axes.set_ylim` 同理。返回设定后的 (xmin, xmax)。
Axes.legend	`handles=None, labels=None, loc='best', **kwargs`	添加图例[47]。不传 handles/labels 则自动使用已有标记艺术家的标签[48]。`loc` 指定位置（字符串如'upper right'或代码0-10）[54]。常用kwargs：`title` 图例标题，`ncol` 列数，`fontsize` 字体大小，`bbox_to_anchor` 放置位置等。返回 Legend 对象。
Axes.annotate	`text, xy, xytext=None, xycoords='data', textcoords=None, arrowprops=None, **kwargs`	添加带箭头的注解[55]。`xy` 为箭头指向点的数据坐标，`xytext` 为文本位置（不指定则文本在 `xy` 处），`arrowprops` 是箭头样式字典（如 `arrowstyle` 等）[58]。`xycoords` / `textcoords` 可更改坐标系。用于突出强调特定点。
Axes.text	`x, y, s, fontdict=None, **kwargs`	添加文本[62]。在数据坐标 `(x,y)` 处添加字符串 `s`。可通过 `transform=ax.transAxes` 将 (x,y) 解释为相对Axes坐标(0-1)以定位在Axes内特定比例位置。kwargs 同样支持所有文本属性设置字体外观。返回 Text 对象。
Axes.twinx	（无参数）	创建共用x轴但y轴独立的双y轴 Axes。常用于不同量纲的数据共用横轴绘图。调用如 `ax2 = ax.twinx()` 会在同一子图位置生成第二个y轴。类似的 `ax.twiny()` 共用y轴生成双x轴。

（注：上表未穷尽所有方法，只列出**最常用**的一部分。更多 Axes 方法请参考 Matplotlib 官方文档。）

综合示例

下面通过多个示例，将 Figure 和 Axes 的上述方法组合起来，展示如何构建专业水准的图形。

1. 多子图布局示例

图1：包含四个子图的 Figure 示例，每个子图展示不同类型的图表。左上：折线图，右上：散点图，左下：柱状图，右下：矩阵图像。使用 `fig.subplots(2,2)` 创建网格布局，并对每个 Axes 调用相应方法绘图和设置标题。

上图代码实现：

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

fig, axs = plt.subplots(2, 2, figsize=(6,5))  # 创建2x2网格布局的子图
# 子图 (0,0)：折线图
x = np.linspace(0, 2*np.pi, 100)
y = np.sin(x)
axs[0, 0].plot(x, y, color='b', linestyle='--', label='sin(x)')
axs[0, 0].set_title("Line Plot")
axs[0, 0].legend()

# 子图 (0,1)：散点图
N = 50
x2 = np.random.rand(N); y2 = np.random.rand(N)
colors = np.random.rand(N); sizes = (np.random.rand(N) * 50) + 10
axs[0, 1].scatter(x2, y2, c=colors, s=sizes, alpha=0.7)
axs[0, 1].set_title("Scatter Plot")

# 子图 (1,0)：柱状图
categories = ['A', 'B', 'C', 'D']; values = [5, 3, 9, 6]
axs[1, 0].bar(categories, values, color='skyblue')
axs[1, 0].set_title("Bar Chart")

# 子图 (1,1)：图像
matrix = np.random.rand(6, 6)
im = axs[1, 1].imshow(matrix, cmap='viridis', aspect='auto')
axs[1, 1].set_title("Image Plot")

fig.suptitle("Multiple Subplots Example")
fig.tight_layout()
plt.show()
```

这段代码演示了如何使用 `fig.subplots(2,2)` 同时创建多个 Axes，并分别绘制不同类型的图。通过 `fig.suptitle` 添加了总标题，通过 `tight_layout` 自动调整了子图布局以避免标题和标签重叠。图例仅在需要的子图上添加，矩阵图像使用了 `aspect='auto'` 来允许各向缩放充满子图区域。

2. 嵌套子图（子 Figure）示例

Matplotlib 3.4+ 引入了 SubFigure 概念，使我们可以在 Figure 内再划分子区域，形成嵌套布局。[74] [75] 下面示例演示左右两块区域，其中左侧再细分为上下两图：

图2：嵌套子图示例。整个 Figure 水平分成左右两大区域 (SubFigure)；左侧 SubFigure 内含两个上下排列的折线图子图；右侧 SubFigure 包含一个图像及其颜色条。

实现上述复杂布局的代码：

```
fig = plt.figure(constrained_layout=True, figsize=(6,3))
# 将Figure水平分为左(2/3宽)和右(1/3宽)两个SubFigure
subfigs = fig.subfigures(1, 2, width_ratios=[2, 1])
left_subfig, right_subfig = subfigs

# 左侧SubFigure再拆成2行1列的Axes
axs_left = left_subfig.subplots(2, 1)
# 上方Axes绘制折线
x = np.linspace(0, 4*np.pi, 100)
axs_left[0].plot(x, np.sin(x), label='sin')
axs_left[0].set_title("Left Top"); axs_left[0].legend()
# 下方Axes绘制另一条折线
axs_left[1].plot(x, np.sin(2*x), color='orange', label='sin(2x)')
axs_left[1].set_title("Left Bottom"); axs_left[1].legend()

# 右侧SubFigure中添加一个Axes并绘制图像
ax_right = right_subfig.subplots()
data = np.random.rand(10, 10)
im = ax_right.imshow(data, cmap='viridis')
ax_right.set_title("Right Image")
# 在右侧SubFigure上添加颜色条（与Axes同高）
right_subfig.colorbar(im, ax=ax_right)

fig.suptitle("Nested Subplots Example")
plt.show()
```

此示例首先用 `fig.subfigures` 将 Figure 按比例分为左右两块 SubFigure[76]。左块通过 `.subplots(2,1)` 产生两个 Axes 上下排列，右块直接 `.subplots()` 得到单个 Axes。我们在左上和左下分别绘制了不同曲线并加上图例，在右侧绘制了一个 `imshow` 图像并使用 SubFigure 的 `colorbar` 方法为其添加颜色条（注意这里 `colorbar` 是调用在 SubFigure 上，作用范围也是该 SubFigure 内）。最终用 `fig.suptitle` 设置整体标题。启用 `constrained_layout=True` 或调用 `fig.tight_layout()` 可避免子Figure之间或颜色条与图像之间的拥挤重叠。

通过 SubFigure，我们实现了更灵活的嵌套布局：不同子区域内可以各自安排自己的子图网格。这对于构造复杂布局（例如左边多个图配合右边一个大图，或包含嵌套小图的情景）非常有用。

3. 双坐标轴示例（双 Y 轴）

在科学绘图中，有时需要在同一子图中展示两个不同量纲但共享横轴的曲线，例如左侧y轴表示温度、右侧y轴表示降水量。Matplotlib 提供了 `Axes.twinx()` 方法来创建共享x轴的第二y轴。下面示例展示如何绘制双y轴：

图3：双 Y 轴示例。同一 Axes 上左侧 Y 轴（蓝色）显示 sin 曲线，右侧 Y 轴（红色）显示二次函数曲线。两个 y 轴分别有不同刻度和标签。

实现代码：

```
fig, ax1 = plt.subplots(figsize=(5,3))
t = np.linspace(0, 10, 100)
y1 = np.sin(t)
y2 = 0.1 * t**2

# 左侧 y 轴曲线
line1, = ax1.plot(t, y1, color='blue', label='sin(t)')
ax1.set_xlabel('t')
ax1.set_ylabel('sin(t)', color='blue')
ax1.tick_params(axis='y', labelcolor='blue')  # 刻度标签颜色同步蓝色

# 右侧 y 轴曲线
ax2 = ax1.twinx()  # 共享x轴
line2, = ax2.plot(t, y2, color='red', label='0.1*t^2')
ax2.set_ylabel('0.1 * t^2', color='red')
ax2.tick_params(axis='y', labelcolor='red')  # 刻度标签颜色同步红色

# 合并图例
lines = [line1, line2]
labels = [line.get_label() for line in lines]
ax1.legend(lines, labels, loc='upper left')

ax1.set_title('Dual Y-Axis Example')
fig.tight_layout()
plt.show()
```

首先通过 `ax1 = fig.subplots()` 获取初始 Axes，然后调用 `ax1.twinx()` 得到共享相同x轴的第二个 Axes（ax2）。在 ax1 上绘制蓝色的 $\sin(t)$ 曲线，并设置左侧y轴标签和刻度颜色为蓝色；在 ax2 上绘制红色的 $0.1t^2$ 曲线，对应右侧y轴，并设其标签和刻度为红色。为了区分两条曲线，我们将它们各自的 Line2D 对象提取出来，手动在 ax1 上调用 `legend`，传入这两个 Line2D 列表和标签，从而组合出一份图例。使用 `tight_layout` 保证左右y轴标签不会彼此重叠。最终得到如图所示具有双y轴的图表。

在此过程中，`twinx()` 创建的 ax2 与 ax1 共享同一个 x 轴，因此鼠标缩放拖动会同时作用于两条曲线，保证对齐。在需要双x轴的情况也可类似使用 `ax.twiny()`。需要注意控制不同轴的颜色和标签对应，确保读者可以区分。

以上完整示例展示了 Matplotlib 面向对象接口的威力：通过合理地组合 Figure 和 Axes 对象的方法，我们可以创造出各种专业且复杂的图形布局。同时，这种接口使代码组织清晰，易于逐步调整图形各部分属性，从而精

确地控制输出结果。读者可结合本教程内容，在实践中灵活运用 Figure 和 Axes 接口，绘制出符合自己需求的高质量图表。 17

---

1 Create multiple subplots using plt.subplots — Matplotlib 3.10.6 documentation
https://matplotlib.org/stable/gallery/subplots_axes_and_figures/subplots_demo.html

2 Introduction to Axes (or Subplots) — Matplotlib 3.10.6 documentation
https://matplotlib.org/stable/users/explain/axes/axes_intro.html

3 4 5 6 7 14 64 74 75 76 matplotlib.figure — Matplotlib 3.10.6 documentation
https://matplotlib.org/stable/api/figure_api.html

8 9 10 11 matplotlib.pyplot.savefig — Matplotlib 3.10.6 documentation
https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.savefig.html

12 13 matplotlib.figure.Figure.set_size_inches — Matplotlib 3.10.6 documentation
https://matplotlib.org/stable/api/_as_gen/matplotlib.figure.Figure.set_size_inches.html

15 16 17 18 19 73 matplotlib.axes.Axes.plot — Matplotlib 3.10.6 documentation
https://matplotlib.org/stable/api/_as_gen/matplotlib.axes.Axes.plot.html

20 21 22 23 matplotlib.axes.Axes.scatter — Matplotlib 3.10.6 documentation
https://matplotlib.org/stable/api/_as_gen/matplotlib.axes.Axes.scatter.html

24 25 26 27 28 29 30 31 matplotlib.axes.Axes.bar — Matplotlib 3.10.6 documentation
https://matplotlib.org/stable/api/_as_gen/matplotlib.axes.Axes.bar.html

32 33 34 35 36 37 matplotlib.axes.Axes.imshow — Matplotlib 3.10.6 documentation
https://matplotlib.org/stable/api/_as_gen/matplotlib.axes.Axes.imshow.html

38 39 40 41 matplotlib.axes.Axes.set_title — Matplotlib 3.10.6 documentation
https://matplotlib.org/stable/api/_as_gen/matplotlib.axes.Axes.set_title.html

42 43 44 matplotlib.axes.Axes.set_xlabel — Matplotlib 3.10.6 documentation
https://matplotlib.org/stable/api/_as_gen/matplotlib.axes.Axes.set_xlabel.html

45 46 matplotlib.axes.Axes.set_xlim — Matplotlib 3.10.6 documentation
https://matplotlib.org/stable/api/_as_gen/matplotlib.axes.Axes.set_xlim.html

47 48 49 50 51 52 53 54 matplotlib.axes.Axes.legend — Matplotlib 3.10.6 documentation
https://matplotlib.org/stable/api/_as_gen/matplotlib.axes.Axes.legend.html

55 56 57 58 59 60 61 matplotlib.axes.Axes.annotate — Matplotlib 3.10.6 documentation
https://matplotlib.org/stable/api/_as_gen/matplotlib.axes.Axes.annotate.html

62 63 matplotlib.axes.Axes.text — Matplotlib 3.10.6 documentation
https://matplotlib.org/stable/api/_as_gen/matplotlib.axes.Axes.text.html

65 66 69 70 71 72 Event handling and picking — Matplotlib 3.10.6 documentation
https://matplotlib.org/stable/users/explain/figure/event_handling.html

67 matplotlib.pyplot.connect
https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.connect.html

68 Understanding matplotlib event handling - Stack Overflow
https://stackoverflow.com/questions/42553539/understanding-matplotlib-event-handling-what-are-event-and-mpl-connect