Matplotlib可视化

Matplotlib是一个Python的2D绘图库，通过Matplotlib，便可生成折线图，直方图，条形图，饼状图，散点图等。

如果是用 Anaconda ，会默认安装matplotlib库，也可通过 `conda install matplotlib` 或者通过 `pip install matplotlib` 进行安装

\u报错

在Python中 \ 是转义符，\u表示其后是UNICODE编码，因此语法冲突，\u会报错。

相关的：如果代码写入路径中有C:\Users这类的，也会出现类似警告

解决方法：

\	方式	具体
1	在路径前面加r	(r'c:\Users...)
2	替换为双反斜杠	('c:\\Users\\...)
3	替换为正斜杠	('c:/Users/...)

折线图 plt.plot()

plt.plot([x],y,[fmt],data=None,kwargs)

In [1]:
```
import matplotlib.pyplot as plt
import numpy as np
```

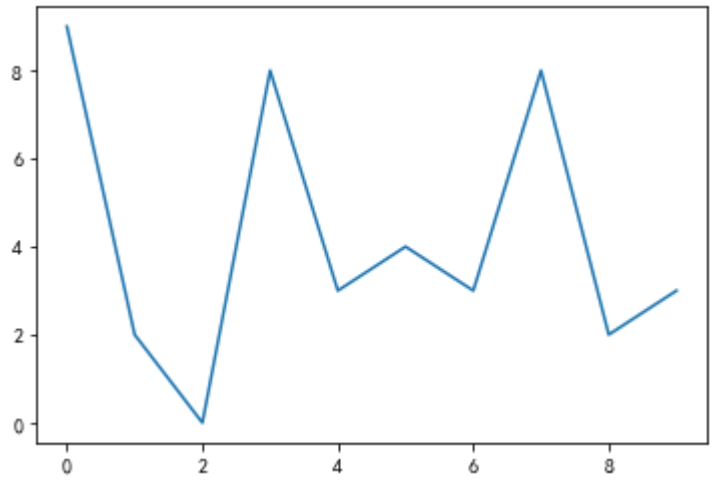
In [2]:
```
plt.rcParams["font.sans-serif"] = ["SimHei"]
plt.rcParams["axes.unicode_minus"] = False  # 显示中文属性设置
```

In [3]:
```
y = np.random.randint(0, 10, size=10)
y
```

Out[3]:
```
array([9, 2, 0, 8, 3, 4, 3, 8, 2, 3])
```

In [4]:
```
plt.plot(y)  # x默认range(len(y))
```

Out[4]:
```
[<matplotlib.lines.Line2D at 0x2a728920580>]
```

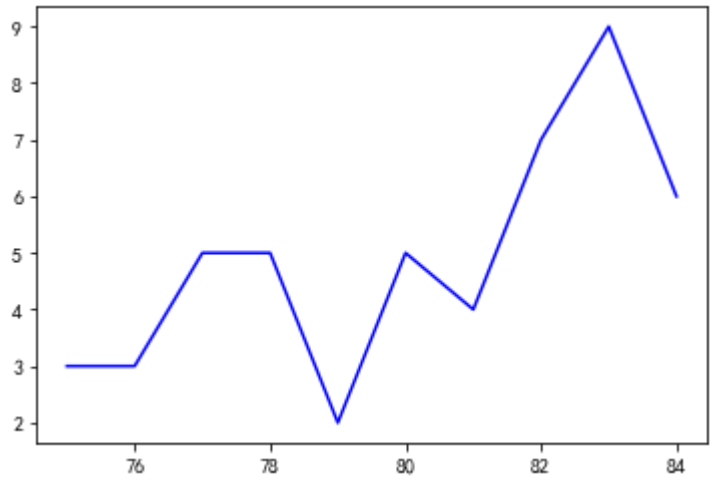


In [5]:
```
y
```

Out[5]:
```
array([9, 2, 0, 8, 3, 4, 3, 8, 2, 3])
```

In [6]:
```
y1 = np.random.randint(0, 10, size=10)
x1 = range(75, 85)
plt.plot(x1, y1,'b-')
```

Out[6]:
```
[<matplotlib.lines.Line2D at 0x2a729086f70>]
```



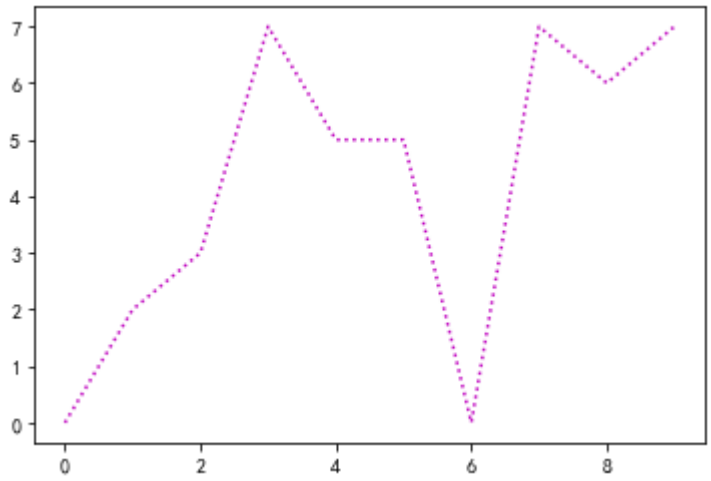
x,y只能是位置参数，不能作为关键字参数传递（不能赋值传递）

设置线条基本样式： fmt 参数

fmt 可以传一个字符串，修改图像样式。默认的绘制样式是 b- ，也就是蓝色实体线条。

In [7]:
```
y = np.random.randint(0, 10, size=10)
x = range(10, 20)
plt.plot(y, "m:")  # fmt也是位置参数
```

Out[7]:
```
[<matplotlib.lines.Line2D at 0x2a7290f1760>]
```



其中使用：代表点线，是matplotlib的一个缩写。这些缩写还有以下的：

字符	类型	字符	类型	字符	类型	字符	类型	字符	类型
'-'	实线	'--'	虚线	'-.'	虚点线	':'	点线	'.'	点
','	像素点	'o'	圆点	'v'	下三角点	'^'	上三角点	'<'	左三角点
'>'	右三角点	'1'	下三叉点	'2'	上三叉点	'3'	左三叉点	'4'	右三叉点
's'	正方点	'p'	五角点	'*'	星形点	'h'	六边形点1	'H'	六边形点2
'+'	加号点	'x'	乘号点	'D'	实心菱形点	'd'	瘦菱形点	'_'	横线点

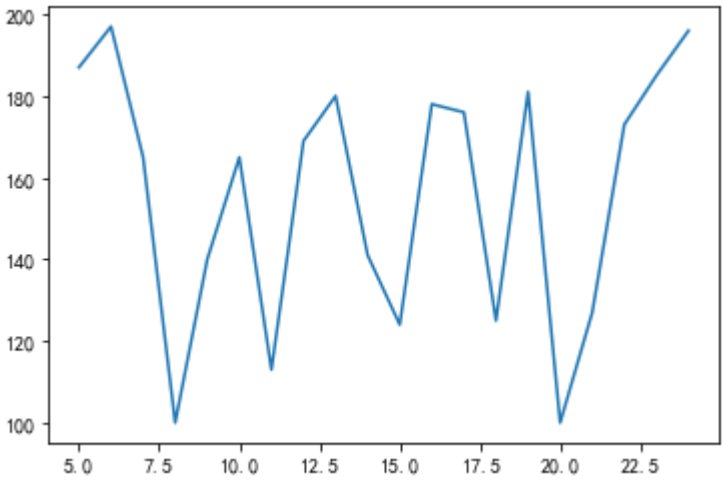
字符	颜色	字符	颜色	字符	颜色	字符	颜色
'b'	蓝色，blue	'g'	绿色，green	'r'	红色，red	'c'	青色，cyan
'm'	品红，magenta	'y'	黄色，yellow	'k'	黑色，black	'w'	白色，white

In [8]:
```
mydata = {"a": range(5, 25)
          , "b": np.random.randint(0, 10, size=20)
          , "c列": np.random.randint(20, 30, size=20)
          , "d列名": np.random.randint(100, 200, size=20)
          }
mydata
```

Out[8]:
```
{'a': range(5, 25),
 'b': array([5, 8, 6, 6, 0, 0, 5, 3, 4, 1, 0, 0, 6, 9, 1, 5, 4, 8, 9, 9]),
 'c列': array([26, 21, 21, 25, 27, 22, 26, 20, 21, 25, 29, 21, 20, 29, 29, 28, 29,
        29, 26, 20]),
 'd列名': array([187, 197, 165, 100, 140, 165, 113, 169, 180, 141, 124, 178, 176,
        125, 181, 100, 127, 173, 185, 196])}
```

In [9]:
```
plt.plot("a", "d列名", data=mydata)
```

Out[9]:
```
[<matplotlib.lines.Line2D at 0x2a729157280>]
```



In [10]:
```
# 数据框格式输入也可以，先导入Pands库
import pandas as pd
```

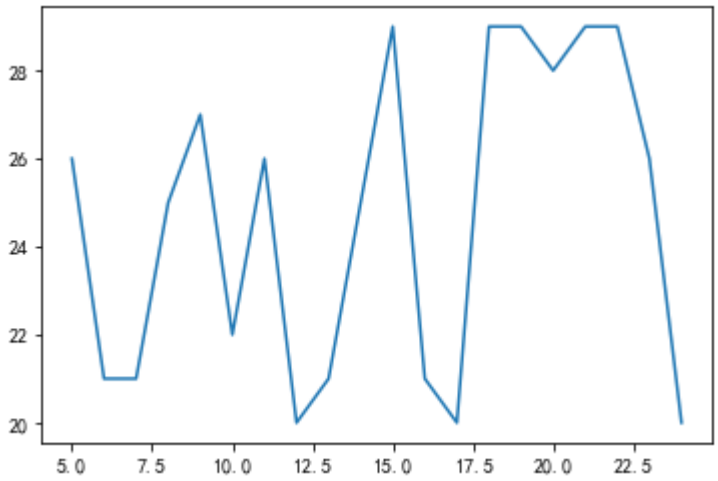
In [11]:
```
mydf = pd.DataFrame(mydata)  # data也可是数据框格式对象，如：mydf
mydf
```

Out[11]:

	a	b	c列	d列名
0	5	5	26	187
1	6	8	21	197
2	7	6	21	165
3	8	6	25	100
4	9	0	27	140
5	10	0	22	165
6	11	5	26	113
7	12	3	20	169
8	13	4	21	180
9	14	1	25	141
10	15	0	29	124
11	16	0	21	178
12	17	6	20	176
13	18	9	29	125
14	19	1	29	181
15	20	5	28	100
16	21	4	29	127
17	22	8	29	173
18	23	9	26	185
19	24	9	20	196

In [12]:
```
plt.plot("a", "c列", data=mydf)  # fmt参数选择线条样式
```

Out[12]:
```
[<matplotlib.lines.Line2D at 0x2a729e962b0>]
```



数据载入（读取）

pd.read_csv("路径")；pd.read_excel("路径")

pd.read_csv (filepath, header='infer', names=None, encoding=None,sheet_name=None)

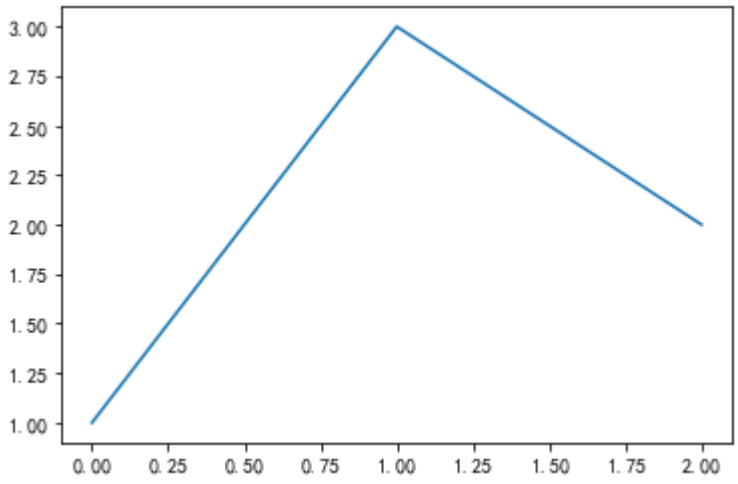
1. csv文件有表头并且是第一行，那么 names 和 header 都无需指定;
2. csv文件有表头、但表头不是第一行，可能从下面几行开始才是真正的表头和数据，这个时候指定 header=行索引 即可;
3. csv文件没有表头，全部是纯数据，设置 header=None ,并可以通过 names=['列名'] 手动生成表头;
4. csv文件有表头、但是这个表头你不想用，这个时候同时指定 names 和 header 。 先用 header 选出表头,然后再用 names 将表头替换掉，其实就等价于将数据读取进来之后再对列名进行rename;

In [13]:
```
s = pd.Series([1, 3, 2])
s
```

Out[13]:
```
0    1
1    3
2    2
dtype: int64
```

In [14]:
```
plt.plot(s)
```

Out[14]:
```
[<matplotlib.lines.Line2D at 0x2a729ef71f0>]
```



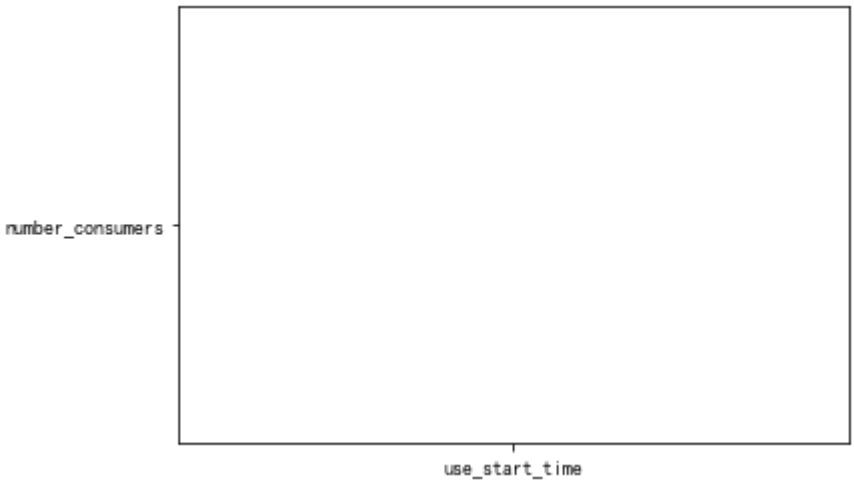
In [15]:
```
data1 = pd.read_csv("../data/meal_order_info.csv", encoding='gbk')
data2 = data1.head(5)
data2
```

Out[15]:

	订单id	客户id	消费人数	消费方式	桌子id	桌号	消费金额	菜品总数	付费金额	开始时间	锁单时间	收银id	终端id	订单号	门店id
0	417	1442	4	NaN	1501	1022	165	5	165	2016/8/1 11:05	2016/8/1 11:11	NaN	NaN	NaN	330
1	301	1095	3	NaN	1430	1031	321	6	321	2016/8/1 11:15	2016/8/1 11:31	NaN	NaN	NaN	328
2	413	1147	6	NaN	1488	1009	854	15	854	2016/8/1 12:42	2016/8/1 12:54	NaN	NaN	NaN	330
3	415	1166	4	NaN	1502	1023	466	10	466	2016/8/1 12:51	2016/8/1 13:08	NaN	NaN	NaN	330
4	392	1094	10	NaN	1499	1020	704	24	704	2016/8/1 12:58	2016/8/1 13:07	NaN	NaN	NaN	330

In [16]:
```
plt.plot("use_start_time", "number_consumers", "-.", data=data2)  # fmt参数选择线条样
```

Out[16]:
```
[<matplotlib.lines.Line2D at 0x2a729f5e8e0>]
```



In [17]:
```
data = pd.read_excel("../data/meal_order_detail.xlsx",sheet_name="meal_order_detail1"
data
```

Out[17]:

	detail_id	order_id	dishes_id	logicprn_name	parent_class_name	dishes_name	itemis_
0	2956	417	610062	NaN	NaN	蒜蓉生蚝	
1	2958	417	609957	NaN	NaN	蒙古烤羊腿	
2	2961	417	609950	NaN	NaN	大蒜苋菜	
3	2966	417	610038	NaN	NaN	芝麻烤紫菜	
4	2968	417	610003	NaN	NaN	蒜香包	
...	...	...	...	...	...	...	
2774	6750	774	610011	NaN	NaN	白饭/大碗	
2775	6742	774	609996	NaN	NaN	牛尾汤	
2776	6756	774	609949	NaN	NaN	意文柠檬汁	
2777	6763	774	610014	NaN	NaN	金玉良缘	
2778	6764	774	610017	NaN	NaN	酸辣藕丁	

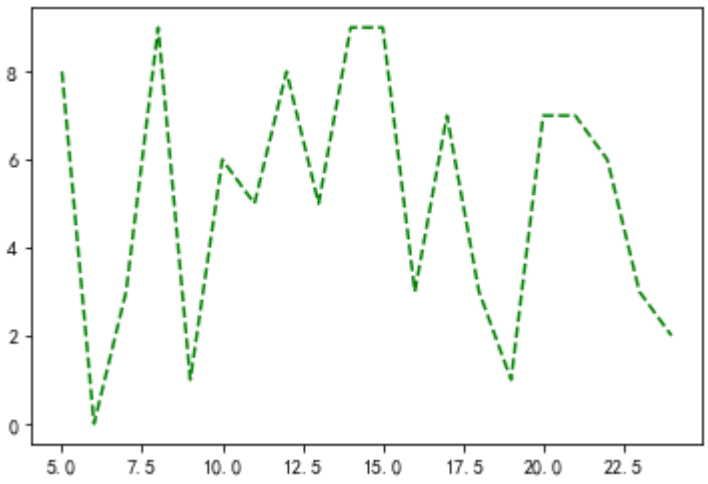
2779 rows × 19 columns

关键字color修改线条颜色

In [18]:
```
mydata = {"a": range(5, 25),
          "b": np.random.randint(0, 10, size=20)
          }
```

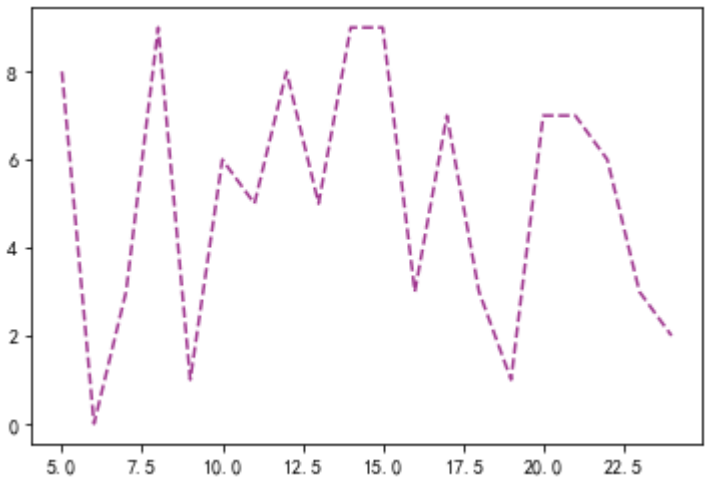
In [19]:
```
# 使用颜色名称
plt.plot("a", "b", "--", data=mydata, color="g")
```

Out[19]: [<matplotlib.lines.Line2D at 0x2a72b711b50>]



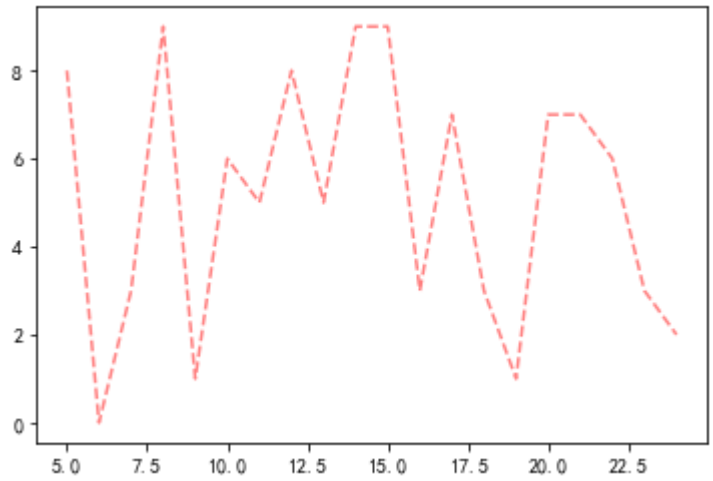
In [20]:
```
# 十六进制表示颜色表示法：6个十六进制数
plt.plot("a", "b", "--", data=mydata, color="#a03790")
```

Out[20]: [<matplotlib.lines.Line2D at 0x2a72b6cea90>]



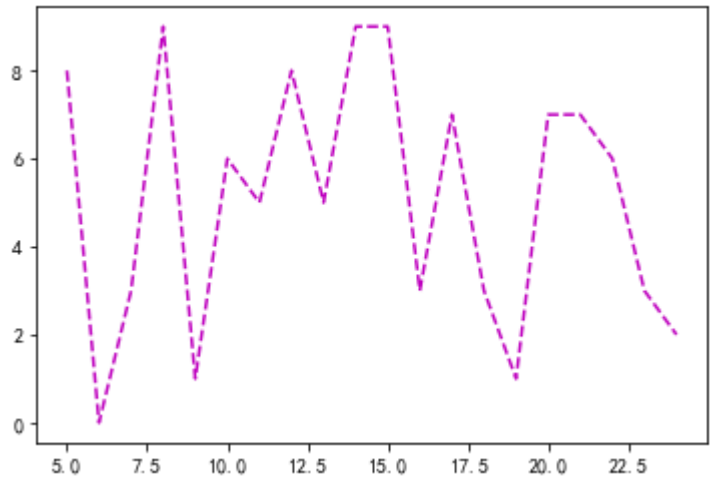
In [21]:
```
# 元组RGB颜色表示法：R、G、B、透明度（取值为0~1之间的值）
plt.plot("a", "b", "--", data=mydata, color=(1, 0, 0, 0.5))
```

Out[21]: [<matplotlib.lines.Line2D at 0x2a72b7688b0>]



In [22]:
```
# 使用的是颜色名称，那么fmt参数可以和线的形状写在同一个字符串中
plt.plot("a", "b", "m--", data=mydata)
```

Out[22]: [<matplotlib.lines.Line2D at 0x2a72b64e730>]



plot([x],y,[fmt],data=None,kwargs)小结：

1. plt.plot可以只传 Y轴 的值，如果只传 Y轴 的值，那么 X轴 就会默认使用 range(0,len(y))
2. plt.plot的 x 、 y 、 fmt 参数不能够作为关键字参数来传递(不能赋值传递)，只能作为位置参数来传。
3. plt.plot中的 data 参数可以为一个字典或者DataFrame对象，然后在 x 和 y 位置上指定这个列的名字，那么plot会自动读取。
4. plt.plot的 fmt 参数可以设置线条的样式以及颜色。
5. plt.plot的 color 参数可以使用**字母、十六进制(#6个16进制的值)**，或者是**RGBA**的方式来设置颜色。
6. 绘制图像时，图像上方一般会出现一个函数的输出,如果不想要输出函数，可以直接在代码末加上一个分号
7. plt.plot函数的官方文档（除了常规参数外，其他参数都是传给Line2D相关属性，它指的是图像对象）：
   https://matplotlib.org/api/_as_gen/matplotlib.pyplot.plot.html#matplotlib.pyplot.plot (https://matplotlib.org/api/_as_gen/matplotlib.pyplot.plot.html#matplotlib.pyplot.plot)

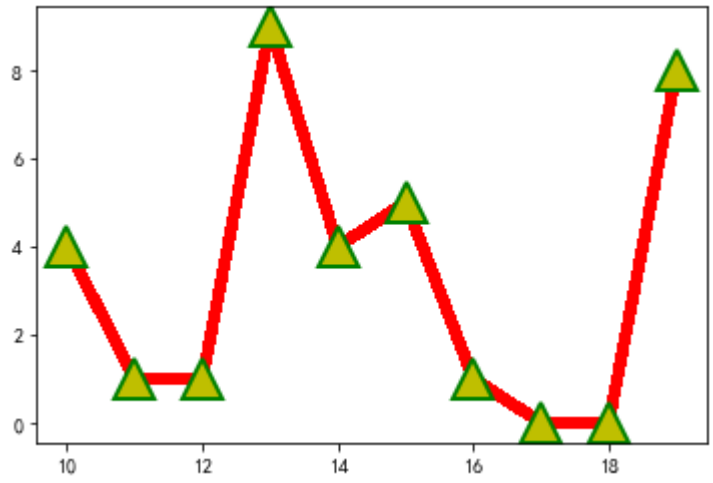
设置线条样式

**kwargs关键字属性设置线条样式

- linewidth：线宽(lw)，ls：线型，c:颜色
- marker：标记点型，ms：标记点大小，mfc：标记点颜色
- mec：标记点边框颜色，mew：标记点边框线条宽度

In [23]:
```
y = np.random.randint(0, 10, size=10)
x = range(10, 20)
y2 = np.sin(x)
plt.plot(x, y
         , linewidth=5, ls="-", c="r"
         ,aa=False
         , marker="^", ms=20
         , mfc="y"
         , mec="g"
         , mew=2
         )
# plt.plot(y2)
```

Out[23]:
```
[<matplotlib.lines.Line2D at 0x2a72b79e820>]
```



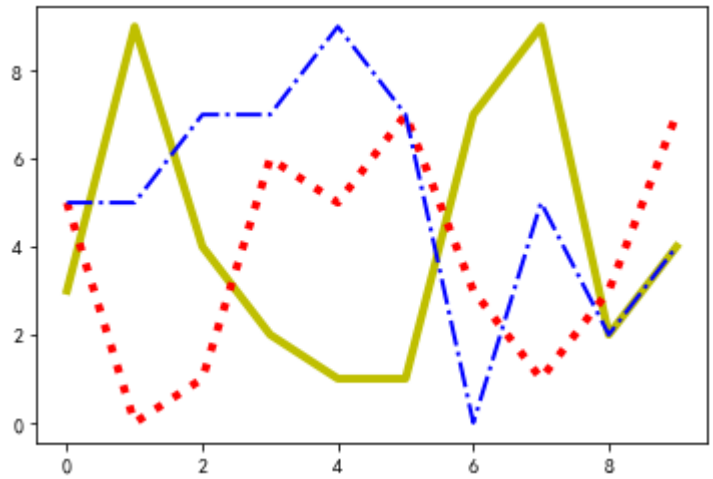
通过plot函数中相关线条参数对线条样式进行设置

同一个坐标系绘制多个图

多次调用plot()方法或在plot(x1,y1,x2,y2,x3,y3)中依次传入多个x,y坐标

In [24]:
```
y1 = np.random.randint(0, 10, size=10)
y2 = np.random.randint(0, 10, size=10)
y3 = np.random.randint(0, 10, size=10)
plt.plot(y3, lw=4, c="y", ls="-")
# plt.figure(figsize=(15, 10))
plt.plot(y1, lw=4, c="r", ls=":")
# plt.figure(figsize=(15,10))
plt.plot(y2, lw=2, c="b", ls="-.")
# plt.plot(range(10),y1,range(10),y2,range(10),y3)
```

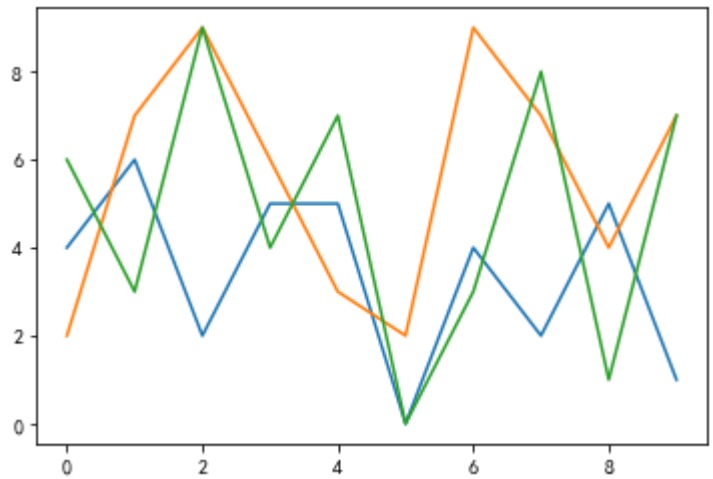
Out[24]:
```
[<matplotlib.lines.Line2D at 0x2a72b825c10>]
```



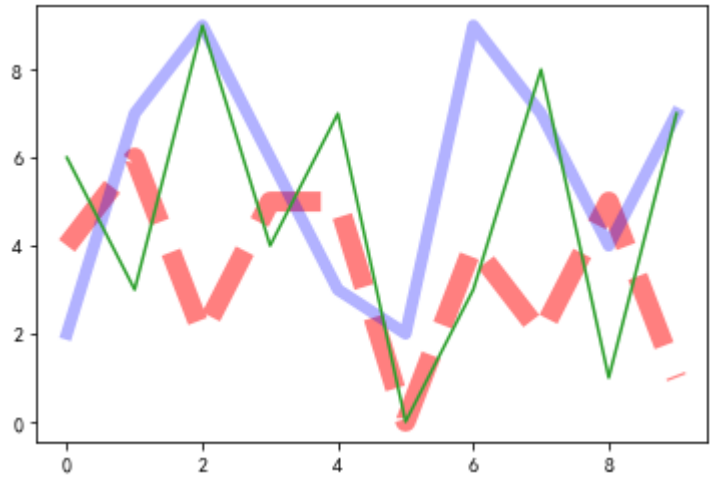
plot()方法返回一个装有Line2D对象的列表,拿到这个Line2D对象后,可通过 set_属性名 设置每一个线条的样式

In [25]:
```
y1 = np.random.randint(0, 10, size=10)
y2 = np.random.randint(0, 10, size=10)
y3 = np.random.randint(0, 10, size=10)
x = range(10)
lines = plt.plot(x, y1, x, y2, x, y3)
lines
```

Out[25]:
```
[<matplotlib.lines.Line2D at 0x2a72b87c790>,
 <matplotlib.lines.Line2D at 0x2a72b87c850>,
 <matplotlib.lines.Line2D at 0x2a72b87c910>]
```



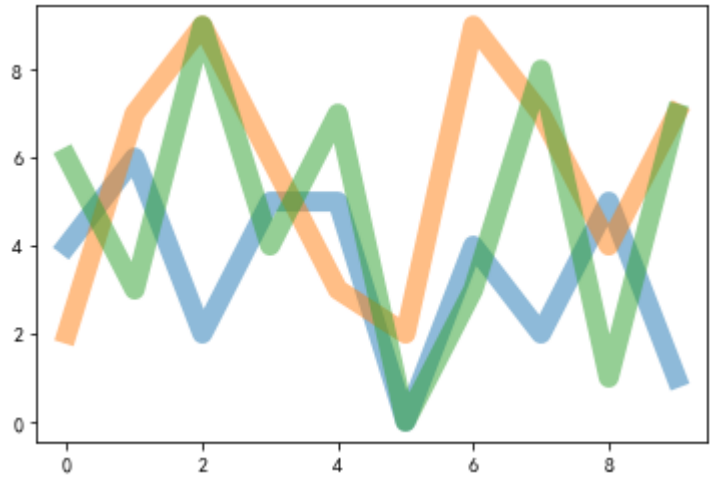
In [26]:
```
lines = plt.plot(x, y1, x, y2, x, y3)
# 设置y1的相关信息
line = lines[0]
line.set_color("r")
line.set_linewidth(10)
line.set_alpha(0.5)
line.set_linestyle('--')
# 设置y2的相关信息
line = lines[1]
line.set_color("b")
line.set_linewidth(6)
line.set_alpha(0.3)
```



使用plt.setp：一次性可以设置多根线条的样式

In [27]:
```
lines = plt.plot(x, y1, x, y2, x, y3)
plt.setp(lines, linewidth=10, alpha=0.5)
```

Out[27]: [None, None, None, None, None, None]



线条Line2D的属性值修改

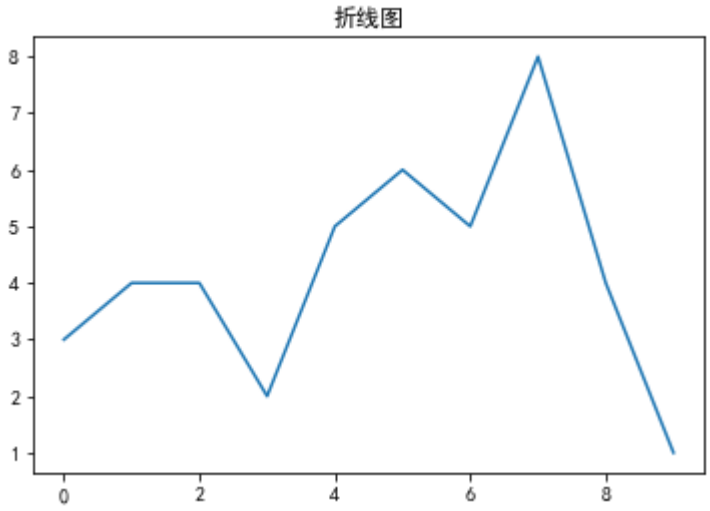
1. plot()绘图时，可传递Line2D的属性值，修改线条样式。
2. 使用plot返回的线条通过 set_属性名() 的方式单独去设置线条的属性。
3. 使用 plt.setp() 来一次性设置多个线条的样式。

图像描述

设置标题： plt.title()

In [28]:
```
y = np.random.randint(0, 10, size=10)
plt.plot(y)
plt.title("折线图")
```

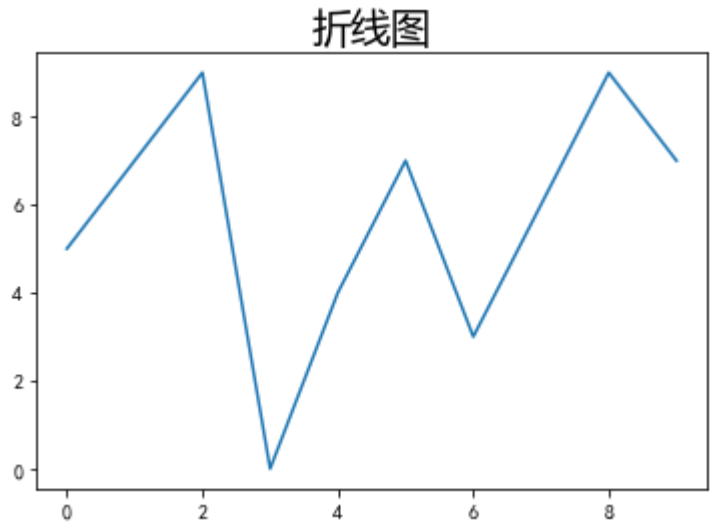
Out[28]: Text(0.5, 1.0, '折线图')



默认情况下是显示不了中文的，需要设置字体。可以通过以下方法来实现

In [29]:
```
# 方法1：通过plt.title()中设置fontproperties参数
# 加载字体的时候，可以到C:\Windows\Fonts中找你喜欢的并且可以显示中文的字体

from matplotlib import font_manager

font = font_manager.FontProperties(
    fname=r"C:\\Windows\\Fonts\\msyh.ttc", size=20)
y = np.random.randint(0, 10, size=10)
plt.plot(y)
plt.title("折线图", fontproperties=font)
```

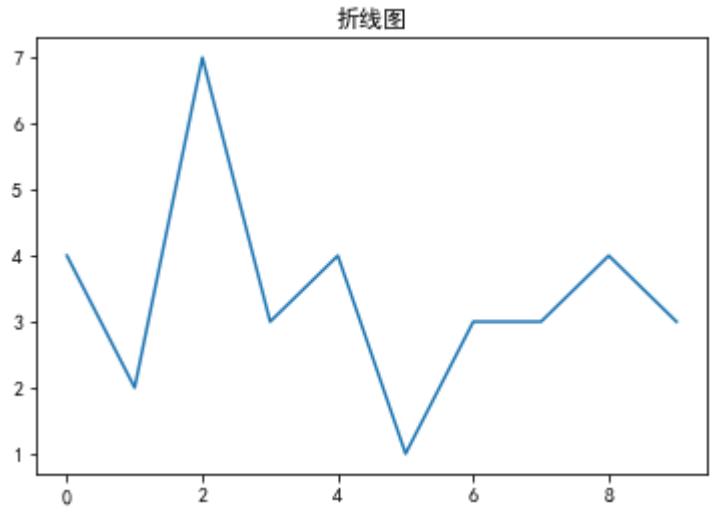
Out[29]: Text(0.5, 1.0, '折线图')



In [30]:
```
# 方法2：统一设置中文属性
plt.rcParams["font.sans-serif"] = ["SimHei"]
plt.rcParams["axes.unicode_minus"] = False  # 显示中文属性设置
```

In [31]:
```
y = np.random.randint(0, 10, size=10)
plt.plot(y)
plt.title("折线图")
```

Out[31]: Text(0.5, 1.0, '折线图')



默认plot是不支持中文的，若要支持中文，在代码前面加上两条属性设置：

- 1) plt.rcParams['font.sans-serif']=['SimHei'];plt.rcParams['axes.unicode_minus'] = False

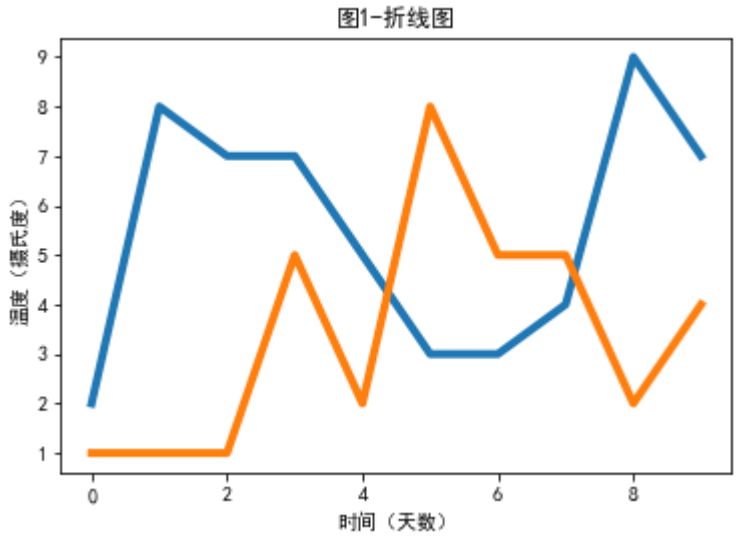
- 2) 创建一个字体对象 matplotlib.font_manager.FontProperties ，然后指定 fname 参数，需要显示中文时调用fontproperties
- 获取字体路径的方式：Windows->Fonts->右键->属性->位置+文件名

设置坐标轴名称:plt.xlabel()和plt.ylabel()

用来设置x轴和y轴的的名称

In [32]:
```
y1 = np.random.randint(0, 10, size=10)
y2 = np.random.randint(0, 10, size=10)
lines = plt.plot(range(10), y1, range(10), y2)
plt.setp(lines, linewidth=4)

# plt.title("折线图",fontproperties=font)
plt.title("图1-折线图")
plt.xlabel("时间（天数）")
plt.ylabel("温度（摄氏度）")
```

Out[32]: Text(0, 0.5, '温度（摄氏度）')



设置坐标轴刻度plt.xticks()

plt.xticks(ticks, labels,rotaion)

- ticks 参数：只要跟数据量保持一致就可以了，比如有20个数据，那么只要产生20个数据的数组或者列表都可以。
- labels 参数：是用来设置每个刻度上的文本显示，如果想要显示字符串类型，可以将字符串列表赋值给lebels参数，同时，保证列表的长度必须和x轴刻度（ticks取值）的长度（元素个数）保持一致。
- rotaion 参数：文本旋转角度

设置坐标轴范围plt.ylim()和plt.xlim()

plt.ylim((start,end)) ：用来修改y轴的取值范围，同理x轴用 **plt.xlim()**

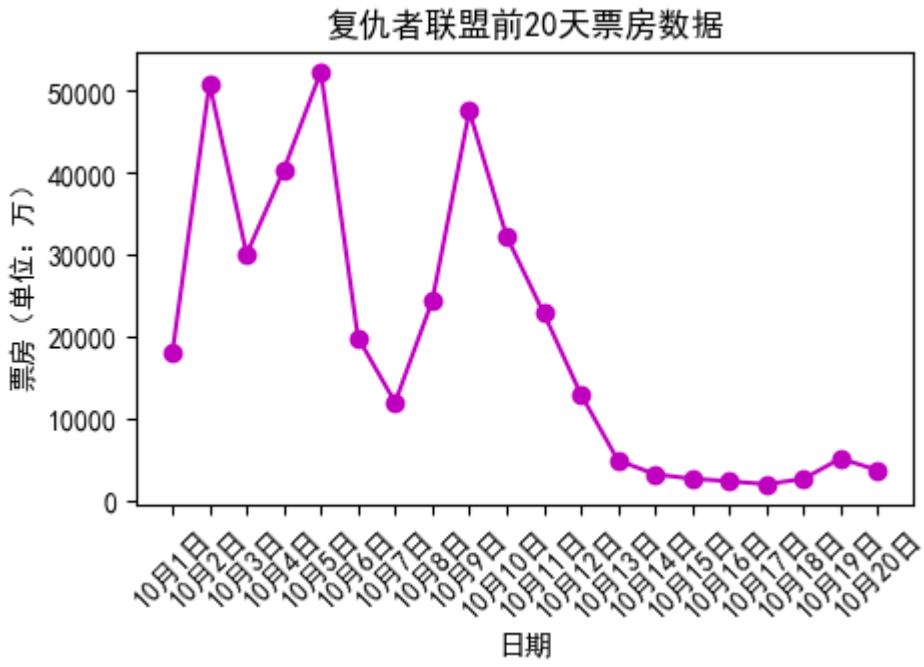
设置图像大小：plt.figure()

如果想要调整图片（画布）的大小和像素，可以通过 `plt.figure()` 来实现:
plt.figure(figsize=(长,宽), dpi=None) ： dpi分辨率的默认取值是100

保存图像：plt.savefig()

plt.savefig("路径\name.png"):name.png表示保存之后的文件名

In [33]:
```
avenger = [17974.4,50918.4,30033.0,40329.1,52330.2,19833.3,
    11902.0,24322.6, 47521.8,32262.0,22841.9,12938.7,4835.1,
    3118.1,2570.9,2267.9,1902.8,2548.9,5046.6,3600.8,]
plt.figure(figsize=(5, 3), dpi=100)  # 画布
plt.plot(avenger, marker = "o", color = 'm')
plt.xticks(range(20)
           , ["10月%d日" % i for i in range(1, 21)]
           , rotation = 45
          )
# plt.ylim((0,60000))
plt.xlabel("日期")
plt.ylabel("票房（单位：万）")
plt.title("复仇者联盟前20天票房数据")
plt.grid(False)  # 设置图像背景网格
#plt.savefig("电影票房折线图.png")  # 保存图像
plt.show()
```



设置文本标注plt.annotate()

plt.annotate(text,xy,xytext,arrowprops={}) 是用来做文本注释的:

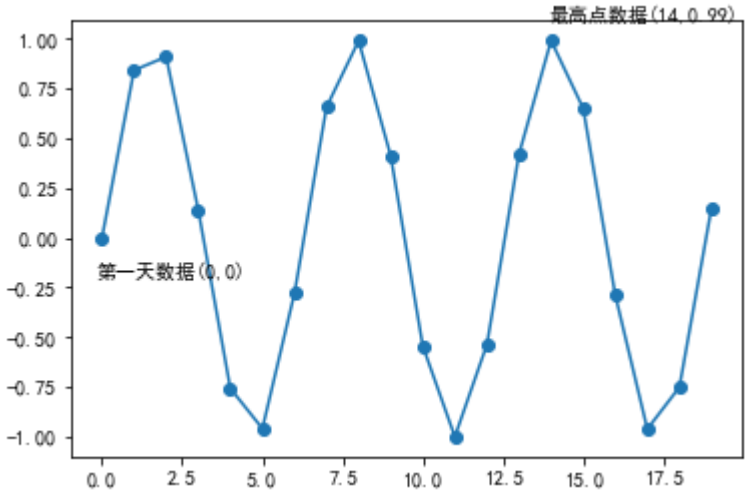
- `text` 参数：需要标注的文本内容。
- `xy` 参数：需要被标注的点的坐标（对哪个点进行标注）。
- `xytext` 参数：标注文本放置的具体位置坐标。（放在哪个位置）

- arrowprops 参数：箭头的相关属性属性。

annotate的官方文档：
https://matplotlib.org/api/_as_gen/matplotlib.pyplot.annotate.html#matplotlib.pyplot.annotate
(https://matplotlib.org/api/_as_gen/matplotlib.pyplot.annotate.html#matplotlib.pyplot.annotate)

In [34]:
```
y = np.sin(np.arange(20))
plt.plot(y, marker="o")
plt.annotate("第一天数据(0,0)",xy=(0,0),xytext=(0-0.2,0-0.2))
# plt.text(0,0,"第一天数据(0,0)",ha='center',va= 'bottom',size=11,color='r')
plt.annotate("最高点数据(14,0.99)",xy=(np.argmax(y),np.max(y)),xytext=(14-0.1,0.99+0
# plt.text(14,0.99,"最高点数据(14,0.99)",ha='center',va= 'bottom',size=11,color='r')
# for index, value in enumerate(y):
#     plt.annotate(
#         "(%d,%.2f)" % (index, value),
#         xy=(index, value),
#         xytext=(index - 0.1, value + 0.1),
#     )
# plt.ylim((-1.2, 1.2))
```

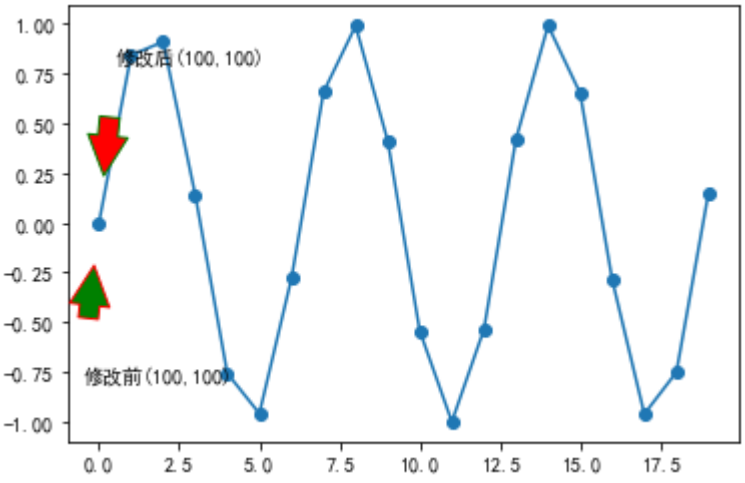
Out[34]: Text(13.9, 1.09, '最高点数据(14,0.99)')



标注箭头参数 arrowprops 设置

In [35]:
```
y = np.sin(np.arange(20))
plt.plot(y, marker="o")
# 箭头标注
plt.annotate(
    "修改前(100,100)",
    xy=(0, 0),
    xytext=(0 - 0.5, 0 - 0.8),
    arrowprops={
        "width": 10,
        "headwidth": 20,
        "headlength": 20,
        "shrink": 0.3,
        "facecolor": "g",
        "edgecolor": "r",
    }
)
plt.annotate(
    "修改后(100,100)",
    xy=(0, 0),
    xytext=(0 + 0.5, 0 + 0.8),
    arrowprops={
        "width": 10,
        "headwidth": 20,
        "headlength": 20,
        "shrink": 0.3,
        "facecolor": "r",
        "edgecolor": "g",
    },
)
```

Out[35]: Text(0.5, 0.8, '修改后(100,100)')



enumerate() 函数用于将一个可遍历的数据对象(如列表、元组或字符串)组合为一个索引序列，同时列出数据下标和数据，一般用在 for 循环当中

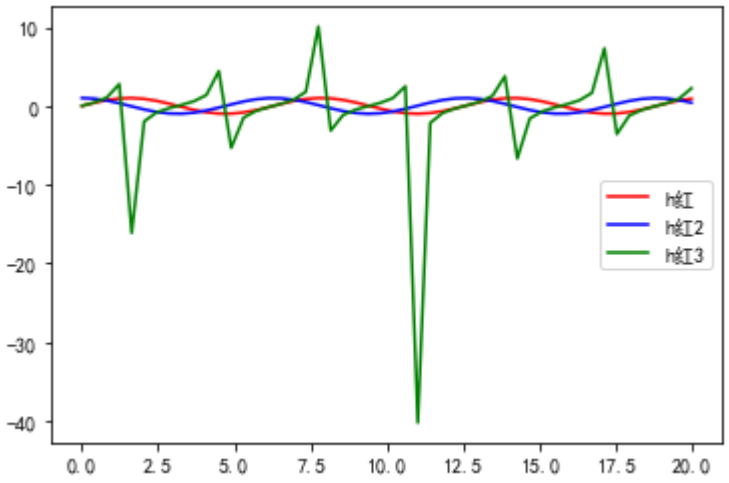
In [36]:
```
data = [3, 5, 8, 9, 2]
for a, b in enumerate(data):
    print(a, b)
```

```
0 3
1 5
2 8
3 9
4 2
```

绘制多个子图

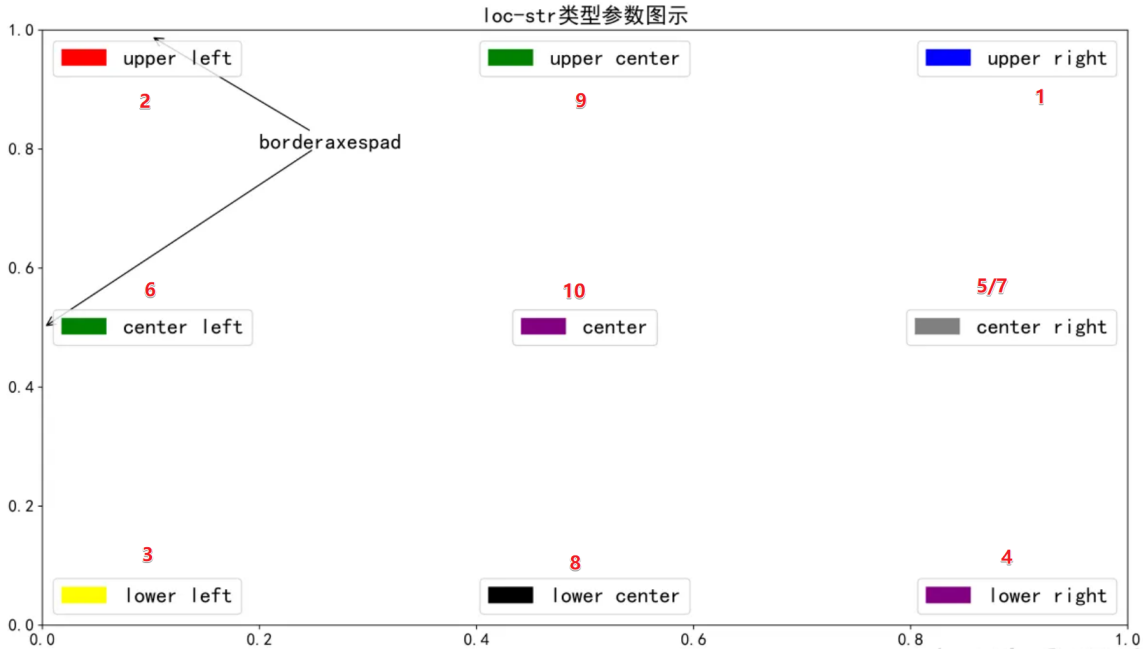
多根线条在同一个坐标系中，只要准备好画布，重新使用plt.plot()绘制即可
在绘图时,每个plot()中加入 label="图像名称" 的方式为每个图形命名。
最后通过 plt.legend() 显示图例

In [37]:
```
x = np.linspace(0, 20)
y1 = np.sin(x)
y2 = np.cos(x)
plt.figure()
plt.plot(x, y1, c='r',label='h红')
plt.plot(x, y2, c='b',label='h红2')
plt.plot(x, np.tan(x) , c='g',label='h红3')
# plt.ylim((-1.3,1.3))
plt.legend(loc=5)
plt.show()
```



设置图例plt.legend()

plt.legend(loc='string'或Number)
loc参数表示图例放置位置,可以是字符串，数字或者坐标点
（绘图函数需加入label参数标注）



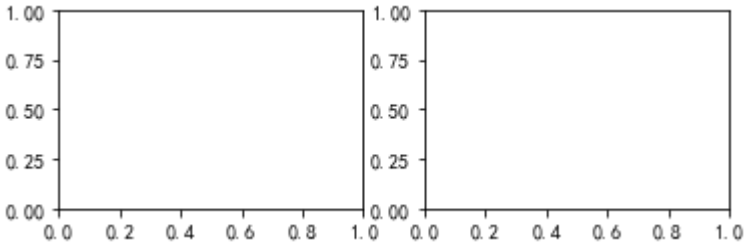
绘制多个坐标子图plt.subplot()

plt.subplot(221) :表示1个画布中共生成2行2列4个子图，现在是绘制第1个子图；

1. 第一个数字 2代表子图排列的行数，
   第二个数字 2代表子图排列的列数，
   第三个数字 1代表当前第几个子图（默认从1开始，从左往右依次表示子图数）。
2. subplot()后面所操作代码都是作用于在当前子图上，直到出现新的subplot()标识。

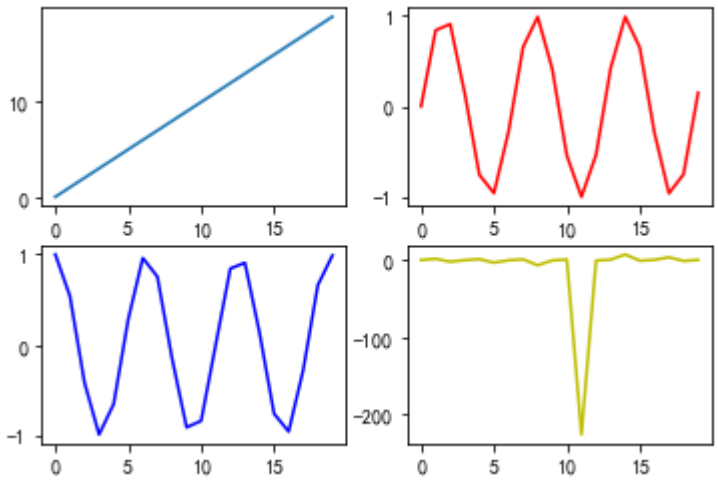
In [38]:
```
plt.subplot(221)
plt.subplot(222)
```

Out[38]: <AxesSubplot:>



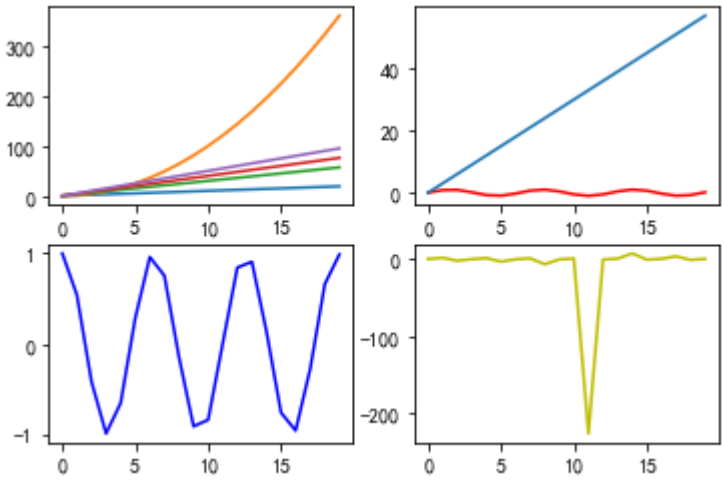
In [39]:
```
# 绘制子图方法1：plt.subplot(行数，列数，第几个图)
values = np.arange(20)
plt.figure()
plt.subplot(221)
plt.plot(values)
plt.subplot(222)
plt.plot(np.sin(values), "r")
plt.subplot(223)
plt.plot(np.cos(values), "b")
plt.subplot(224)
plt.plot(np.tan(values), "y")
```

Out[39]: [<matplotlib.lines.Line2D at 0x2a72bdcbf10>]



In [40]:
```
values = np.arange(20)
plt.figure()
plt.subplot(221)
plt.plot(values)
plt.plot(values ** 2)
plt.plot(values * 3)
plt.plot(values * 4)
plt.plot(values * 5)
plt.subplot(222)
plt.plot(np.sin(values), "r")
plt.plot(values * 3)
plt.subplot(223)
plt.plot(np.cos(values), "b")
plt.subplot(224)
plt.plot(np.tan(values), "y")
```

Out[40]: [<matplotlib.lines.Line2D at 0x2a72bae23d0>]



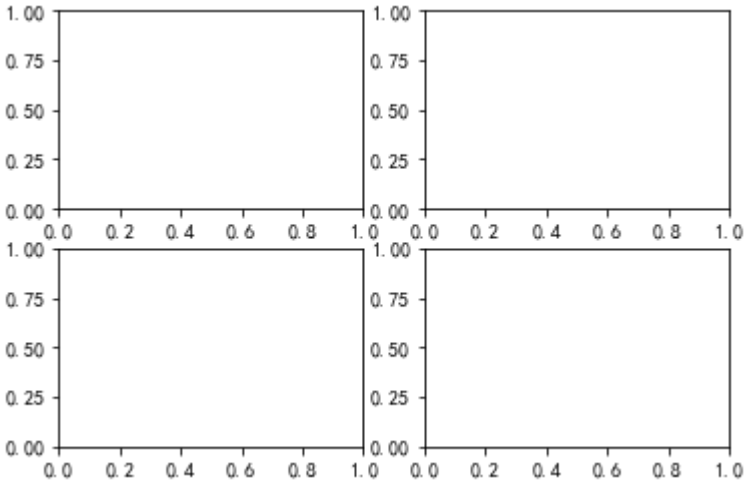
绘制多个坐标子图plt.subplots()

`fig,axes =plt.subplots(2,2,sharex=True,sharey=True)`：

1. 第一个数字 2代表子图排列的行数，
   第二个数字 2代表子图排列的列数。
2. `sharex/sharey` 两个参数表示:所有坐标轴是否需要共享同一个X轴和Y轴。
3. 使用fig,axs=plt.subplots()，返回值是一个元组，元组解包之后：
   `fig` 参数是figure对象， `axs` 是axes对象的array。
4. 这个方法是先一次性绘制所有空子图（空坐标系），通过axes索引到各自子图对象的坐标系，再在各自坐标系中对子图对象进行操作。

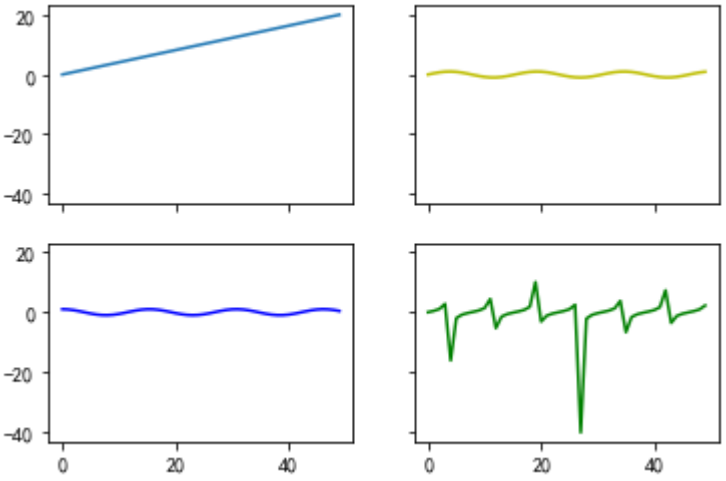
In [41]:
```
fig, axes = plt.subplots(2, 2)
axes
```

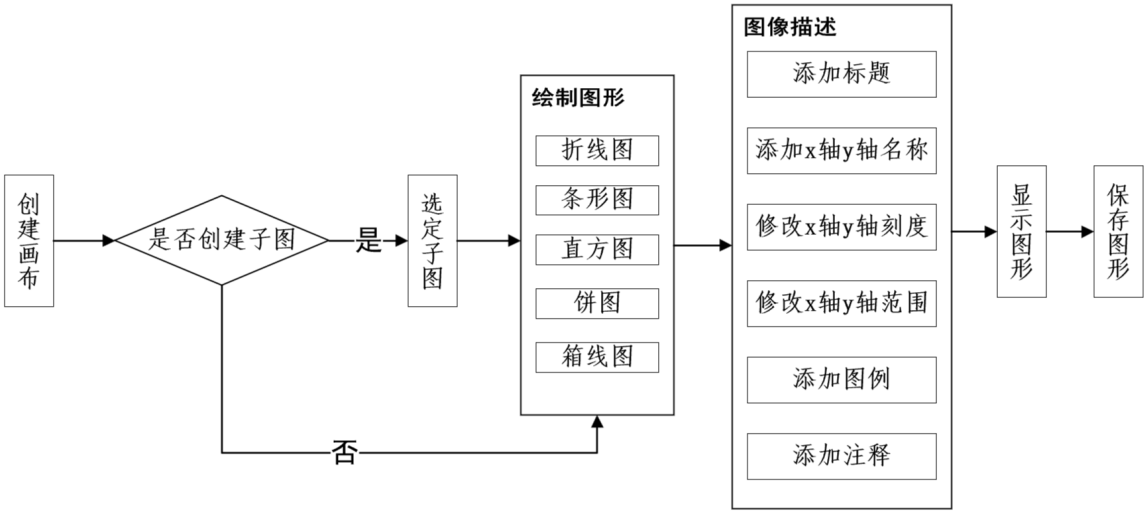
Out[41]:
```
array([[<AxesSubplot:>, <AxesSubplot:>],
       [<AxesSubplot:>, <AxesSubplot:>]], dtype=object)
```



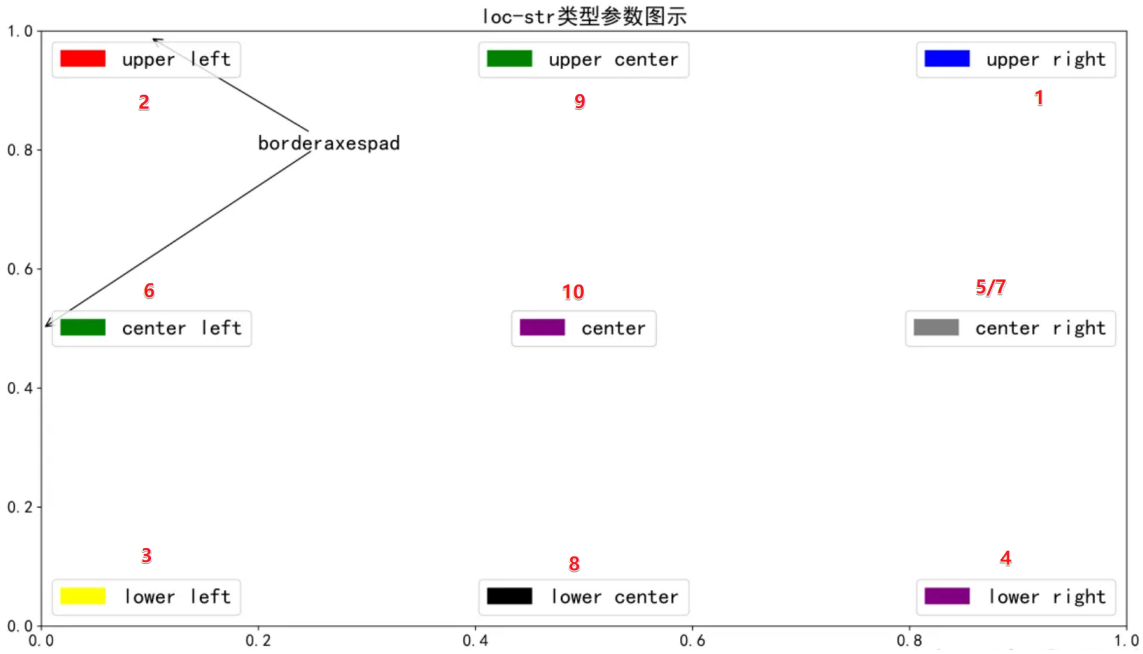
In [42]:
```
# 绘制子图方法2：plt.subplots(2,2)
values = np.linspace(0, 20)
fig, axes = plt.subplots(2, 2, sharex=True, sharey=True)
ax1 = axes[0, 0]
ax1.plot(values)



ax2 = axes[0, 1]
ax2.plot(np.sin(values), c="y")
ax3 = axes[1, 0]
ax3.plot(np.cos(values), c="b")
ax4 = axes[1, 1]
ax4.plot(np.tan(values), c="g")
```

Out[42]:
```
[<matplotlib.lines.Line2D at 0x2a72cfb5f70>]
```



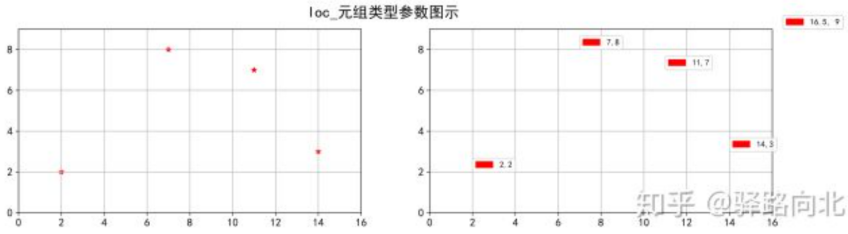


图例位置参数loc补充：



- **一对浮点数(x,y)，保存在元组中. 示例：loc = (0, 0)**

  （x, y）:此种形式是将legend的**左下角**放置在点（x, y）上，可参考下图：



左图中红点坐标分别为(2,2), (7,8) , (11,7) , (14,5)

通过左右两张图比较，很容易看出，右图中legend的左下角与点(x, y)重合。

注意：当使用loc=(x,y)时，x, y并不是轴域中实际的x, y的值，而是将x轴, y轴分别看成1, 即：(x/(x_max-x_min), y/(y_max-y_min))（归1处理）；

在绘制上图时，其中xlim=(0, 16), ylim=(0, 9),所以如果将legend放置到点(2, 2)上，那loc实际要写成：

```
.legend(loc=(2/16,  2/9))
```

In [ ]: