In [1]:

```
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
# 解决坐标轴刻度负号乱码
plt.rcParams['axes.unicode_minus'] = False

# 解决中文乱码问题
plt.rcParams['font.sans-serif'] = ['Simhei']
```

1 散点图

1.1 plt.plot()

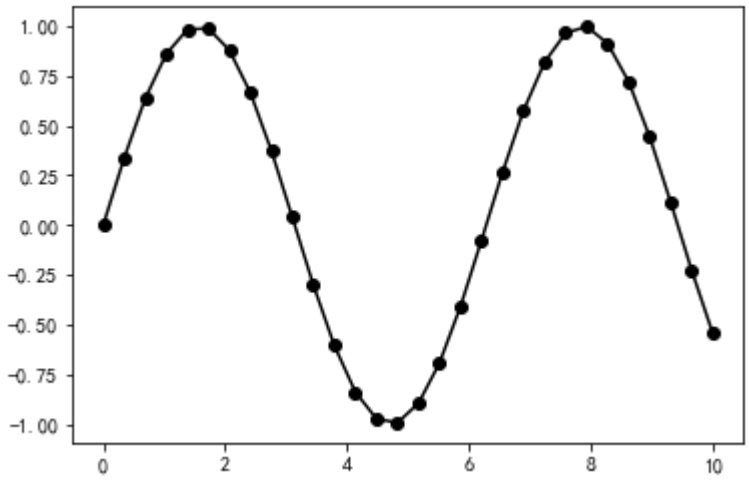
其实我们使用之前学习的plt.plot()函数是可以绘制简单的散点图的：

plt.plot(x, y, Markers) 第三个参数其实是散点的样式：

Markers	描述
"."	point marker
","	pixel marker
"o"	circle marker
"v"	triangle_down marker
"^"	triangle_up marker
"<"	triangle_left marker
">"	triangle_right marker
"1"	tri_down marker
"2"	tri_up marker
"3"	tri_left marker
"4"	tri_right marker
"s"	square marker
"p"	pentagon marker
"*"	star marker
"h"	hexagon1 marker
"H"	hexagon2 marker
"+"	plus marker
"x"	x marker
"D"	diamond marker
"d"	thin_diamond marker
"|"	vline marker
"_"	hline marker

In [2]:

```
x = np.linspace(0, 10, 30)
y = np.sin(x)

plt.plot(x, y, marker="o", color='black')

plt.show();
```



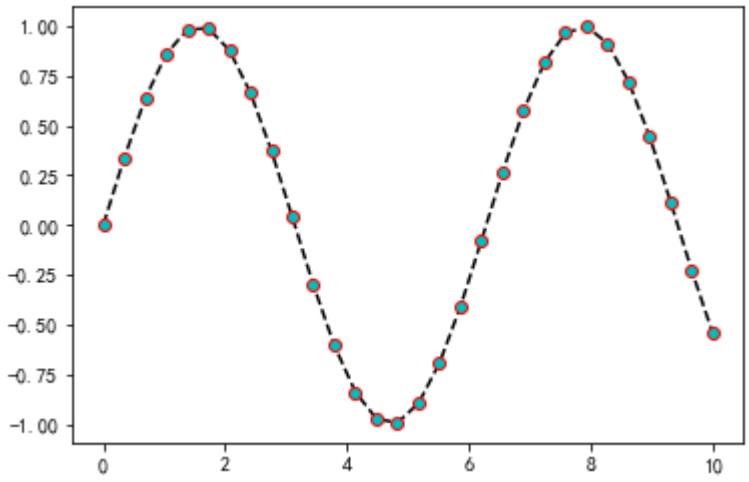
其实上面的markder可以搭配折线风格，比如可以写成"--o",折线风格就变成了虚线带点：

此外和markers相关的参数：

- markersize：标记的大小
- markerfacecolor：标记的颜色
- markeredgecolor：标记边缘的颜色
- markeredgewidth：标记边缘的宽度

In [3]:

```
x = np.linspace(0, 10, 30)
y = np.sin(x)

plt.plot(x, y, '--o', markerfacecolor="c",markeredgecolor="r",markeredgewidth=1,color='black

plt.show();
```



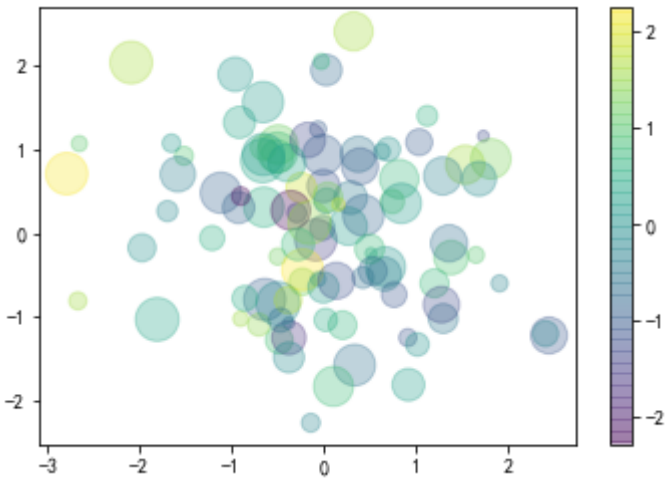
1.2 plt.scatter()

plt.scatter比plt.plot更加灵活，比如可以单独控制每个散点与数据匹配，也可以让每个散点具有不同的属性。

- 函数功能：寻找变量之间的关系
- 调用签名：plt.scatter(x,y,c="b",label="scatter figure")
- 参数说明：
  - x：x轴上的数值
  - y：y轴的数值
  - c：散点图中的标记的颜色
  - label：标记图形内容的标签文本

In [22]:

```
import matplotlib.pyplot as plt
import numpy as np
np.random.seed(9)

x=np.random.randn(100)
y=np.random.randn(100)
colors=np.random.randn(100)
size=np.random.randint(20,500,100)

plt.scatter(x,y,c=colors,s=size,alpha=0.3,cmap="viridis")

plt.colorbar()

plt.show();
```



因为plt.scatter要对每个散点进行单独的大小和颜色渲染，所以如果散点多达数千的时候，plt.plot会更快。

2 柱状图plt.bar()

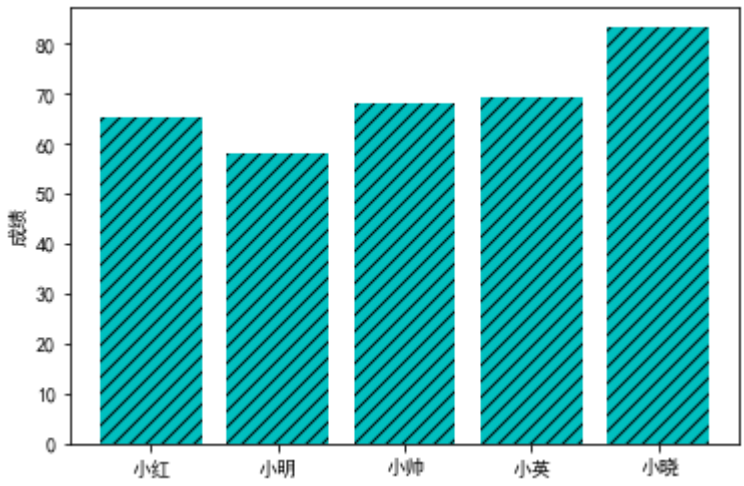
柱状图是描述统计中使用频率非常高的一种统计图形，主要应用在定性数据的可视化场景中，或者是离散型数据的分布展示。

- 函数功能：在x轴上绘制定性数据的分布特征。
- 调用签名：plt.bar(x,y)
- 主要参数：
  - x：标示在x轴上的定性数据的类别

    - y：每种定性数据的类别的数量
- 其他参数：
    - align:柱体对齐方式
    - color：柱体颜色
    - tick_label：刻度标签值
    - alpha：柱体的透明度
    - hatch：柱体填充的样式

In [21]:

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

x=[i for i in range(1,6)]   #可以是列表
y= np.random.randint(45,100,5)  #可以是数组
z=("小红","小明","小帅","小英","小晓")   #可以是元组

plt.ylabel("成绩")


plt.bar(x,y,align="center",color="c",tick_label=z,hatch="///")

plt.show();
#hatch参数还可以填充“/”、“///”“\\”、“|”、“-”、'+', 'x', 'o', 'O', '.', '*'
```

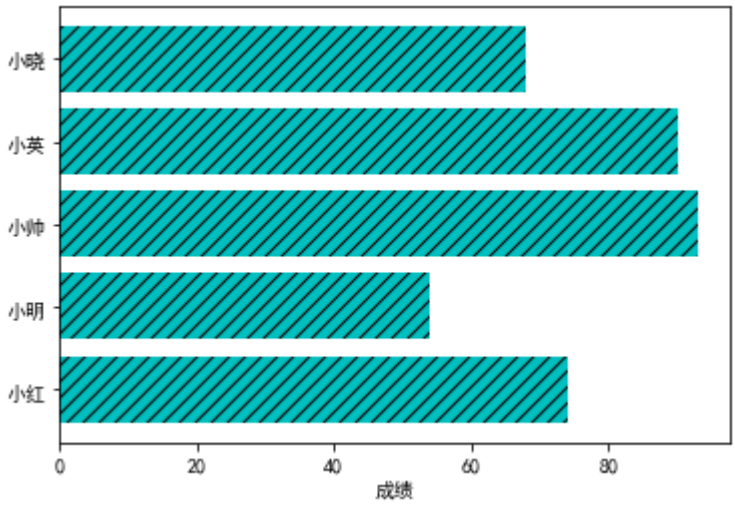


2.1 条形图plt.barh()

- 函数功能：在y轴上绘制定性数据的分布特征。
- 调用签名：plt.barh(x,y)
- 主要参数：
    - x：标示在y轴上的定性数据的类别。
    - y：每种定性数据的类别的数量。
- 其他参数：
    - align:柱体对齐方式
    - color：柱体颜色
    - tick_label：刻度标签值
    - alpha：柱体的透明度

In [20]:

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

x=[i for i in range(1,6)]   #可以是列表
y= np.random.randint(45,100,5)  #可以是数组
z=("小红","小明","小帅","小英","小晓")   #可以是元组

plt.xlabel("成绩")

plt.barh(x,y,align="center",color="c",tick_label=z,hatch="////")
plt.show();
# xx=np.random.randint(0,10,2000)

# plt.hist(xx,color="g",rwidth=1,alpha=0.6)
```

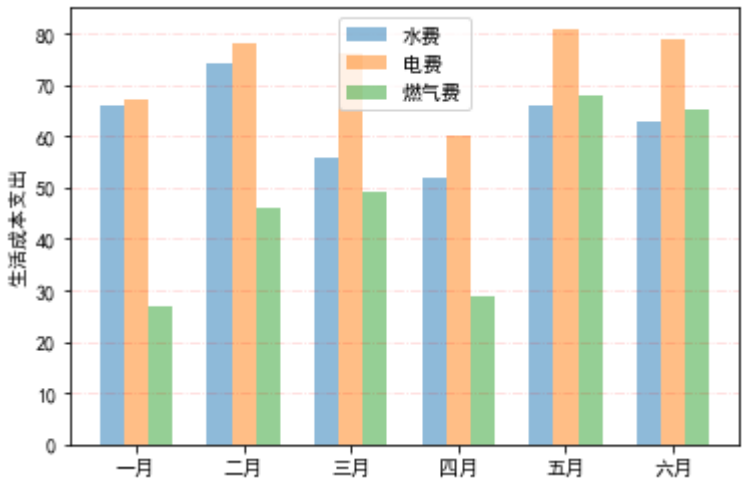


2.2 分块图

2.2.1 多数据并列柱状图

In [24]:

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

# np.random.seed(100)

x=np.arange(1,12,2)    #要设置宽一点

water= np.random.randint(20,80,6)
electricity = np.random.randint(60,90,6)
gas = np.random.randint(20,70,6)
bar_width=0.45

months=("一月","二月","三月","四月","五月","六月")

plt.ylabel("生活成本支出")

plt.bar(x,water,bar_width,align="center",label="水费",alpha=0.5)    #每组的第一列，就不要添加
plt.bar(x+bar_width,electricity,bar_width,align="center",tick_label=months,label="电费",alph
plt.bar(x+bar_width*2,gas,bar_width,align="center",label="燃气费",alpha=0.5)    #每组的第三列

plt.legend()

plt.grid(linestyle="-.",color="r",axis="y",alpha=0.15)     #设置红色的、虚线的网格线

plt.show();
```

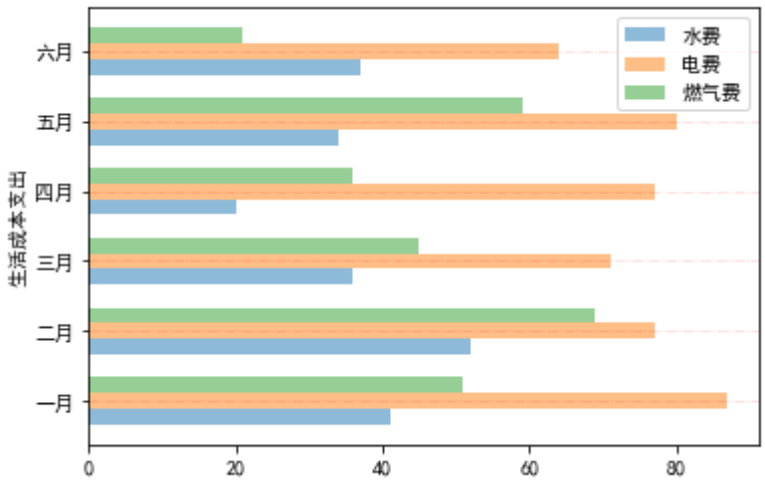


2.2.2 多数据平行条形图

将plt.bar改为plt.barh就可以：

In [154]:

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

# np.random.seed(100)

x=np.arange(1,12,2)    #要设置宽一点

water= np.random.randint(20,80,6)
electricity = np.random.randint(60,90,6)
gas = np.random.randint(20,70,6)
bar_width=0.45

months=("一月","二月","三月","四月","五月","六月")

plt.ylabel("生活成本支出")


plt.barh(x,water,bar_width,align="center",label="水费",alpha=0.5)    #每组的第一列，就不要添
plt.barh(x+bar_width,electricity,bar_width,align="center",tick_label=months,label="电费",al
plt.barh(x+bar_width+bar_width,gas,bar_width,align="center",label="燃气费",alpha=0.5)    #每

plt.legend()

plt.grid(linestyle="-.",color="r",axis="y",alpha=0.15)     #设置红色的、虚线的网格线

plt.show();
```



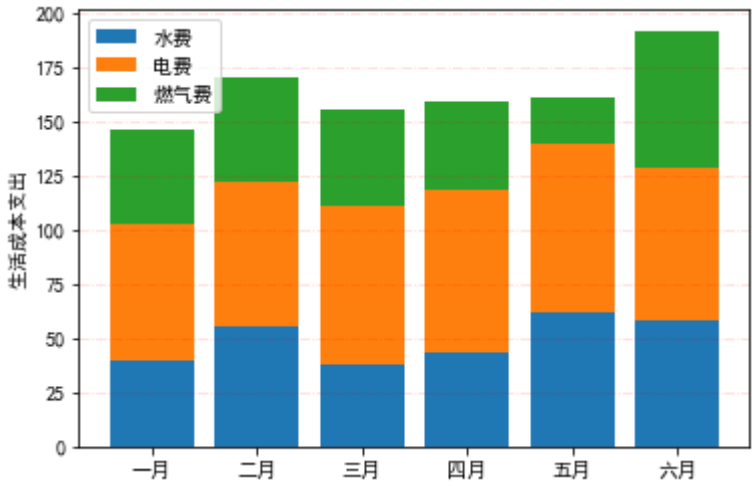
2.3 堆积柱状图

- 堆积图就是将若干统计图形堆叠起来的统计图形，是一种组合式图形。
- 其实是多个plt.bar()画图函数同时作图，只不过堆积在上面的柱状图函数plt.bar()中有bottom参数来设定。

In [134]:

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

# np.random.seed(100)

x=np.arange(6)

water= np.random.randint(20,80,6)
electricity = np.random.randint(60,90,6)
gas = np.random.randint(20,70,6)

months=("一月","二月","三月","四月","五月","六月")

plt.ylabel("生活成本支出")

plt.bar(x,water,align="center",tick_label=months,label="水费")
plt.bar(x,electricity,align="center",tick_label=months,label="电费",bottom=water)     #记得
plt.bar(x,gas,align="center",tick_label=months,label="燃气费",bottom=water+electricity)

plt.legend()

plt.grid(linestyle="-.",color="r",axis="y",alpha=0.15)     #设置红色的、虚线的网格线

plt.show();
```

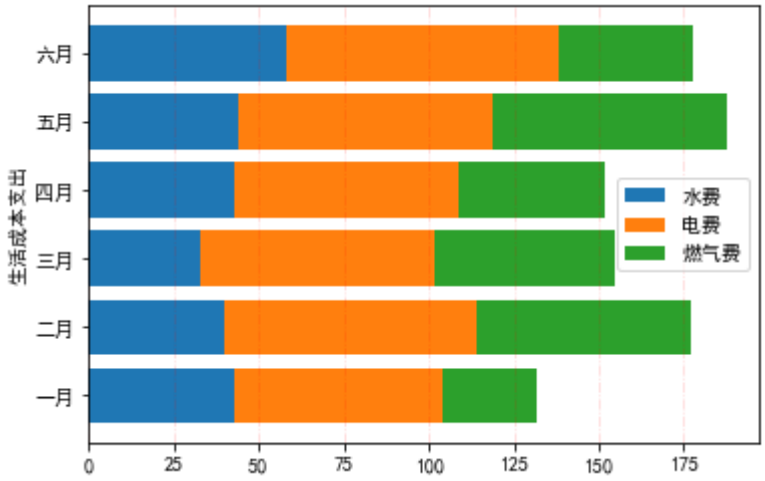
```
[40 55 38 43 62 58]
[63 67 73 75 78 71]
[43 48 44 41 21 63]
```



将上面的**plt.bar()**函数改为**plt.barh()**函数，其中，**bottom**参数改为**left**参数，有堆积条形图：

In [25]:

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

# np.random.seed(100)

x=np.arange(6)

water= np.random.randint(20,80,6)
electricity = np.random.randint(60,90,6)
gas = np.random.randint(20,70,6)


months=("一月","二月","三月","四月","五月","六月")

plt.ylabel("生活成本支出")

plt.barh(x,water,align="center",tick_label=months,label="水费")
plt.barh(x,electricity,align="center",tick_label=months,label="电费",left=water)
plt.barh(x,gas,align="center",tick_label=months,label="燃气费",left=water+electricity)

plt.legend()

plt.grid(linestyle="-.",color="r",axis="x",alpha=0.15)     #设置红色的、虚线的网格线

plt.show();
```



3 直方图plt.hist()

3.1 直方图

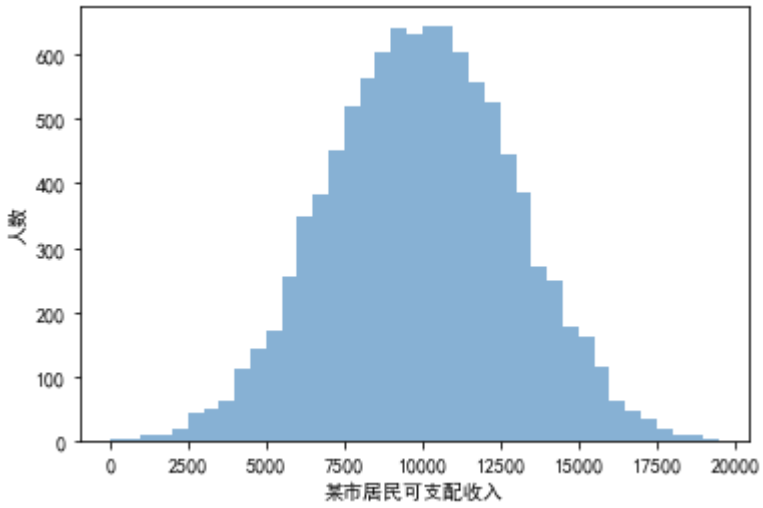
直方图主要是在应用在定量数据的可视化场景中，比如需要展现居民可支配收入的分布特征。

- 函数功能：在x轴上绘制定量数据的分布特征。
- 调用签名：plt.hist(x)
- 参数说明：
  - x：在x轴上绘制箱体的定量数据输入值。
  - bins:用于确定柱体的个数或柱体边缘范围,除了最后一个柱体左右都为闭区间，其他柱体为左闭右开区间。

- color:柱体颜色
- histtype:柱体类型
  - 'bar'：是传统的条形直方图。 如果是多个数据给出了并排排列的条形图。
  - 'step'：生成默认的线图填补。
  - 'stepfilled'：生成默认的线图填充。
- label:图例内容
- rwidth:柱体宽度

In [27]:

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

x=np.random.normal(loc=10000,scale=3000,size=10000)   #loc均值，scale标准差，size样本数量
bins=range(0,20000,500)  #一共40条柱体,相当于自动分箱40类

plt.hist(x,bins=bins,histtype="bar",color="#377eb8",alpha=0.6)

plt.xlabel("某市居民可支配收入")
plt.ylabel("人数")

plt.show();
```



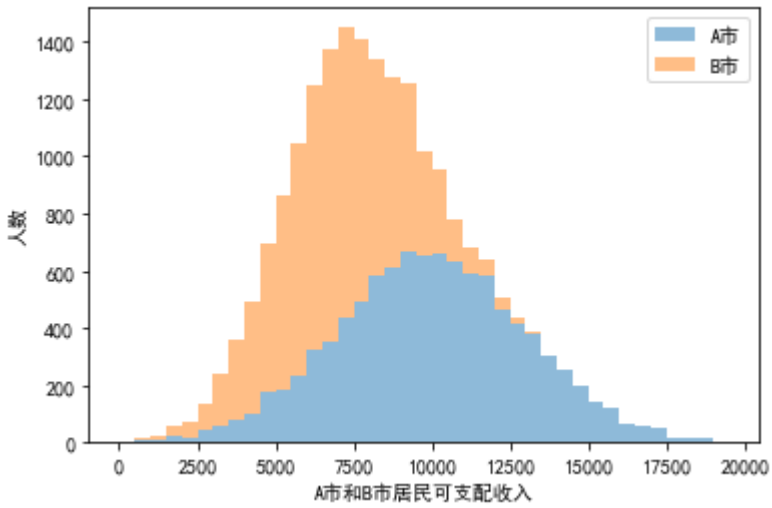
3.2 堆积直方图

和直方图相比，需要注意添加两个参数：

- x：参数x需要对应两个数据对象，可以用列表“装载”。
- stacked：True即两个直方图堆积，False即两个直方图并排放置。
- label：因为要展现两类数据，因此标签需要额外设置。

In [30]:

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

city_A=np.random.normal(loc=10000,scale=3000,size=10000)   #loc均值，scale标准差，size样本数
city_B=np.random.normal(loc=7000,scale=2000,size=10000)    #loc均值，scale标准差，size样本数
x=(city_A,city_B)
labels=["A市","B市"]

bins=range(0,20000,500)  #一共40条柱体,相当于自动分箱40类

plt.hist(x,bins=bins,histtype="bar",label=labels,stacked=True,alpha=0.5) #最后添加参数stack

plt.xlabel("A市和B市居民可支配收入")
plt.ylabel("人数")

plt.legend(loc="upper right")

plt.show();
```



4 饼图

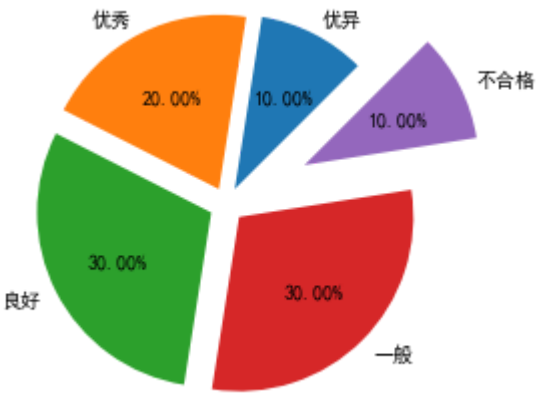
4.1 饼图plt.pie()

饼图是用来展示定性数据比例分布特征的统计图形。

- 函数功能：绘制定性数据的不同类别的百分比
- 调用签名：plt.pie(x)
- 主要参数：
  - x：定性数据的不同类别的百分比。
  - explode:饼片边缘偏离半径的百分比。（想制作分裂式饼图，添加上该参数即可）
  - labels:标记每份饼片的文本标签内容。
  - autopact:饼片文本标签内容对应的数值百分比样式。
  - startangle:从x轴作为起始位置，第一个饼片逆时针旋转的角度。
  - shadow:是否绘制饼片的阴影。
  - colors:饼片的颜色。
- 其他参数：
  - pctdistance：每个饼图的中心与开始之间的比率，默认值0.6。
  - labeldistance：绘制饼图标签的径向距离，默认值1.1。

In [32]:

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

percent="0.1","0.2","0.3","0.3","0.1"
explode=0.1,0.1,0.1,0.1,0.5   #比如我想突出不及格的部分，最后一个数值改为0.5
autopact="%.2f%%"
startangle=45
types="优异","优秀","良好","一般","不合格"

# colors=["r","g","b","c","y"]

plt.pie(percent,explode=explode,autopct=autopact,startangle=startangle,labels=types); #,colo
```



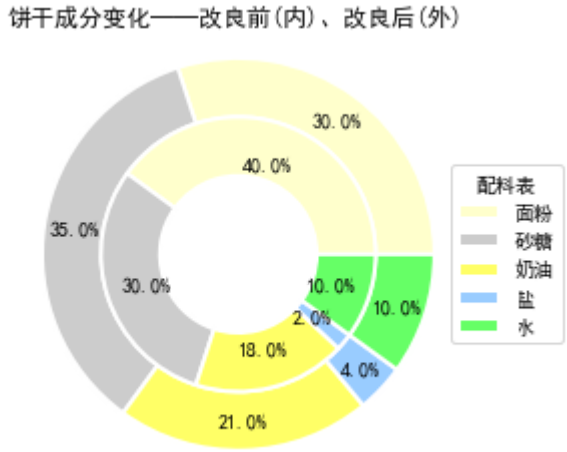
4.2 内嵌环状饼图

内嵌环装饼图可以对比多数据集的比例分布情况，与普通饼图制作相比，内嵌环状饼图需要注意以下要点：

- 内嵌环状饼图的本质是两个饼图嵌套，这就需要内层的饼图半径（radius）要比外层的要小。
- wedgeprops参数字典中里面，两个饼图的“饼边宽占比”（width）要恰当，最好两个饼图都一致。

In [71]:

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

#数据集，x1,x2分别对应外部、内部百分比例
outer=[30,35,21,4,10]
inner=[40,30,18,2,10]

#设置饼状图各个区块的颜色
color=['#FFFFCC','#CCCCCC','#FFFF66','#99CCFF','#66FF66']

plt.pie(outer,autopct='%.1f%%',radius=1,pctdistance=0.85,colors=color,wedgeprops=dict(linew
plt.pie(inner,autopct='%.1f%%',radius=0.7,pctdistance=0.7,colors=color,wedgeprops=dict(linew

#图例
legend_text=['面粉','砂糖','奶油',"盐",'水']
plt.legend(legend_text,title='配料表',loc='center right')#设置图例标题、位置
plt.axis('equal');#设置相等的缩放比例（特别对于圆形），改变轴限制。

plt.title("饼干成分变化——改良前(内)、改良后(外)");
```



In [70]:

```
legend_text
```

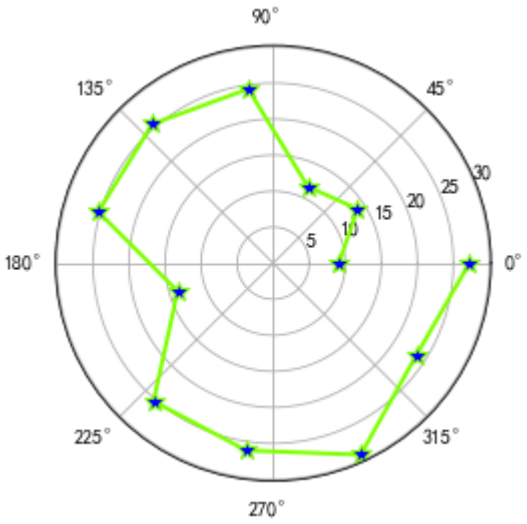
Out[70]: ['面粉', '砂糖', '奶油', '盐', '水']

5 极线图plt.polar()

- 函数功能：在极坐标上绘制折线图
- 调用签名：plt.polar(theta,r)
- 参数说明：
  - theta：每个标记所在射线与极径的夹角
  - r：每个标记到原点的距离

In [42]:

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

theta=np.linspace(0,2*np.pi,12)
r=30*np.random.rand(12)

plt.polar(theta,r,color="chartreuse",linewidth=2,marker="*",mfc="b",ms=10);
```

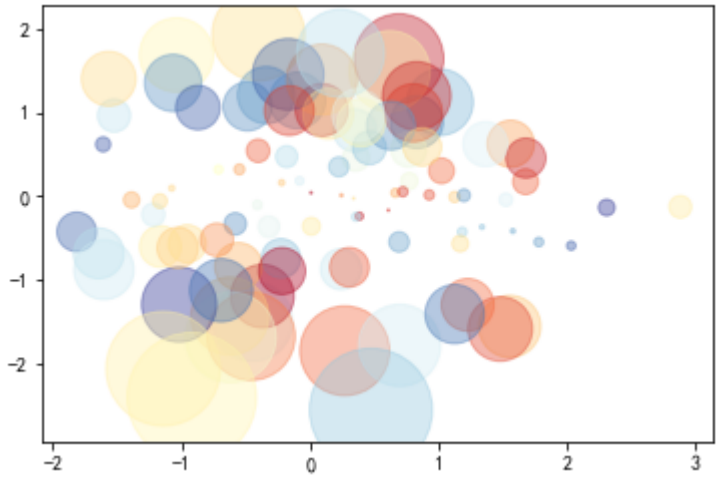


6 气泡图plt.scatter()

- 函数功能：二维数据借助气泡大小展示三维数据
- 调用签名：plt.scatter(x,y)
- 参数说明:
  - x：x轴上的数值
  - y：y轴上的数值
  - s：散点标记的大小
  - c：散点标记的颜色
  - cmap：将浮点数映射成颜色的颜色映射表

In [43]:

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

x=np.random.randn(100)
y=np.random.randn(100)
s=np.power(5*x+25*y,2)
c=np.random.rand(100)
cmap=mpl.cm.RdYlBu


plt.scatter(x,y,s,c,cmap=cmap,alpha=0.4,marker="o");
```

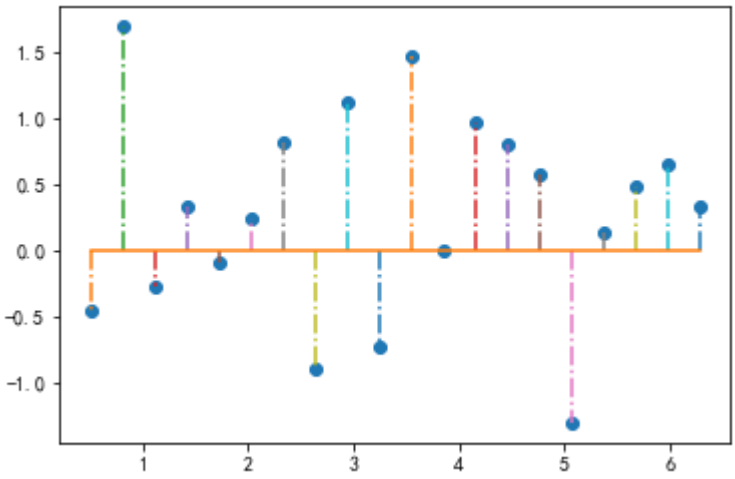


7 棉棒图plt.stem()

- 函数功能：绘制离散有序的数据
- 调用签名：plt.stem(x,y)
- 参数说明:
  - x：指定棉棒的x轴基线上的位置。
  - y：绘制棉棒的长度。
  - linefmt:棉棒的样式。
  - markerfmt:棉棒末端的样式
  - basefmt：指定基线的样式

In [119]:

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

x=np.linspace(0.5,2*np.pi,20)
y=np.random.randn(20)
#生成20个数，这些数服从标准正态分布 ， 区别np.random.rand——随机样本位于[0, 1)中

plt.stem(x,y,linefmt="-.",markerfmt="o",basefmt="-");
```



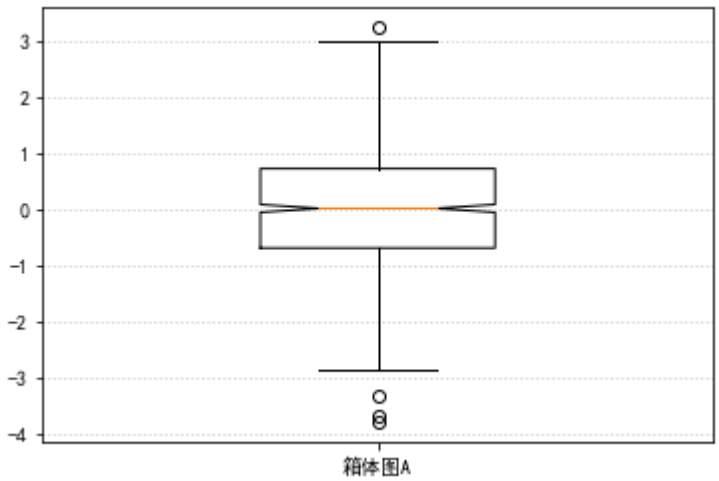
8 箱线图plt.boxplot()

箱体图是由一个箱体和一对箱须所组成的统计图形，箱体是由第一四分位数、中位数（第二四分位数）和第三四分位数所组成，箱须的末端之外为离群值。

- 函数功能：绘制箱线图
- 调用签名：plt.boxplot(x)
- 参数说明
  - x：绘制箱线图的输入数据
  - whis：四分位间距的倍数，用来确定箱须，包括数据的范围大小。
  - widths：设置箱体的宽度。
  - sym：离群值的标记样式。
  - labels:绘制每一个数据集的刻度标签。
  - patch_artist：是否给箱体添加颜色。
  - notch：如果notch为True，则箱体有凹痕。

In [44]:

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

x=np.random.randn(1000)

plt.boxplot(x,whis=1.63,widths=0.35,sym="o",labels=["箱体图A"],notch=True)

plt.grid(axis="y",ls=":",lw=1,color="gray",alpha=0.4)

plt.show();
```



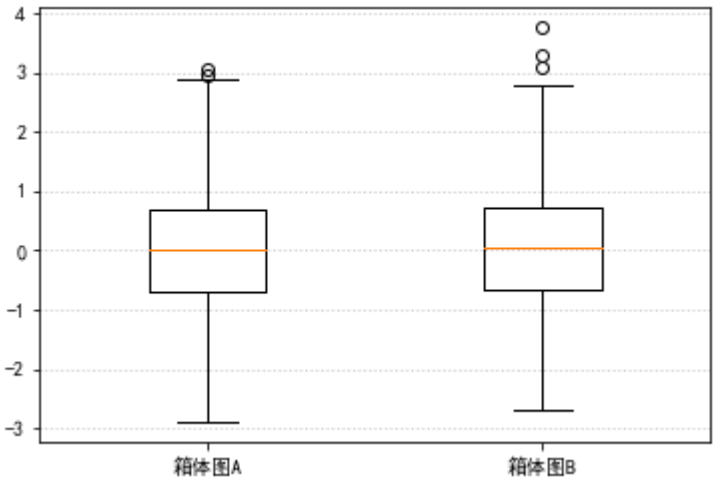
8.1 多个箱线图

相比单个箱线图，如果想制作多个箱线图，需要注意以下要点：

- 数据集参数x替换成包含多个数据集的对象。
- 标签参数labels的列表要包含多个标签字符串。

In [164]:

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

x=np.random.randn(1000)
y=np.random.randn(1000)
test_list=[x,y]

plt.boxplot(test_list,whis=1.63,widths=0.35,sym="o",labels=["箱体图A","箱体图B"])

plt.grid(axis="y",ls=":",lw=1,color="gray",alpha=0.4)

plt.show();
```

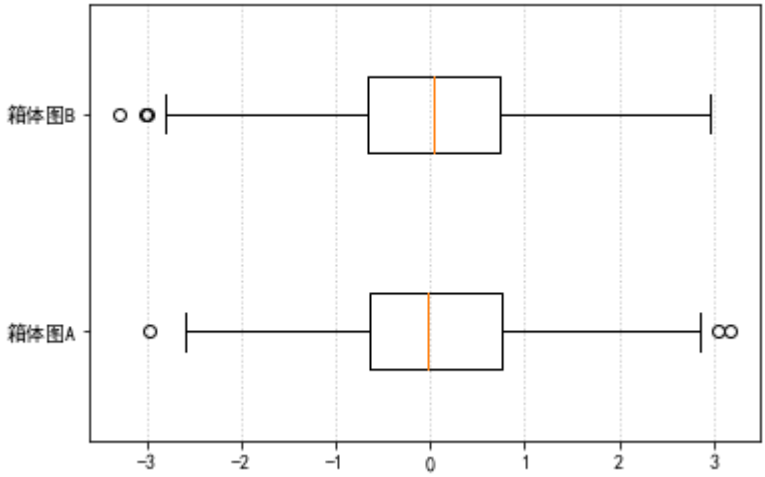


8.2 水平放置箱线图

只需要在plt.boxplot()中添加vert=False参数。

In [46]:

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

x=np.random.randn(1000)
y=np.random.randn(1000)
test_list=[x,y]

plt.boxplot(test_list,whis=1.63,widths=0.35,sym="o",labels=["箱体图A","箱体图B"],vert=False)

plt.grid(axis="x",ls=":",lw=1,color="gray",alpha=0.4)

plt.show();
```

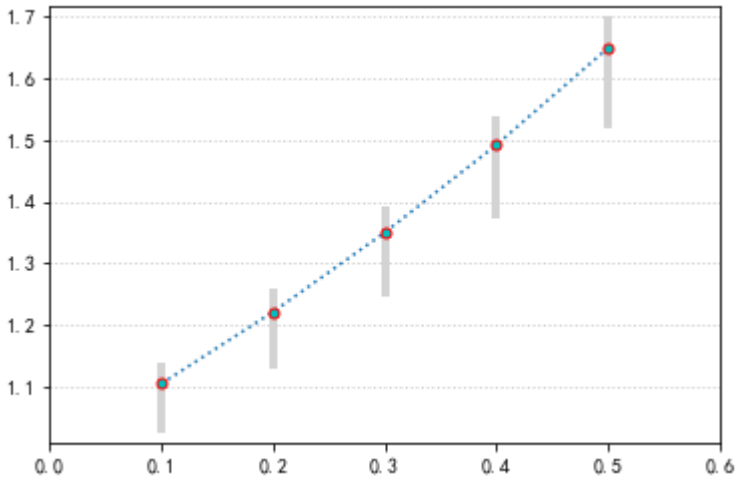


9 误差棒图plt.errorbar()

- 很多科学实验都存在测量误差或者是试验误差，这是无法控制的客观因素，误差棒图可以给试验结果增加观测结果的误差以表示测量偏差的客观存在。
- 误差棒可以很好地实现充当总体参数估计的置信区间的角色。
- 误差棒的计算方法：单一数值、置信区间、标准差和标准误等。
- 误差棒的可视化展示效果样式：水平误差棒、垂直误差棒、对称误差棒、和非对称误差棒。
- 函数功能：绘制y轴方向或是x轴方向的误差范围。
- 调用签名：plt.errorbar(x,y,yerr=a,xerr=b)
- 主要参数：
  - x：数据点的水平位置。
  - y：数据点的垂直位置。
  - yerr：y轴方向的数据点的误差计算方法。
  - xerr：x轴方向的数据点的误差计算方法。
- 其他参数：
  - fmt：数据点的标记样式和数据点标记的连接线样式。
  - ecolor：误差棒的线条颜色。
  - elinewidth：误差棒的线条粗细。
  - ms：数据点的大小
  - mfc：数据点的标记颜色。
  - mec：数据点的标记边缘颜色。
  - capthick：误差棒边界横杆的厚度。
  - capsize：误差棒边界横杆的大小。

In [50]:

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

x=np.linspace(0.1,0.5,5)
y=np.exp(x)

lower_error=0.05+0.13*x
upper_error=0.3*lower_error
error_range=[lower_error,upper_error]

plt.errorbar(x,y,yerr=error_range,fmt="o:",ecolor="lightgray",elinewidth=4,ms=5,mfc="c",mec

plt.xlim(0,0.6)

plt.grid(axis="y",ls=":",lw=1,color="gray",alpha=0.4)

plt.show();
```



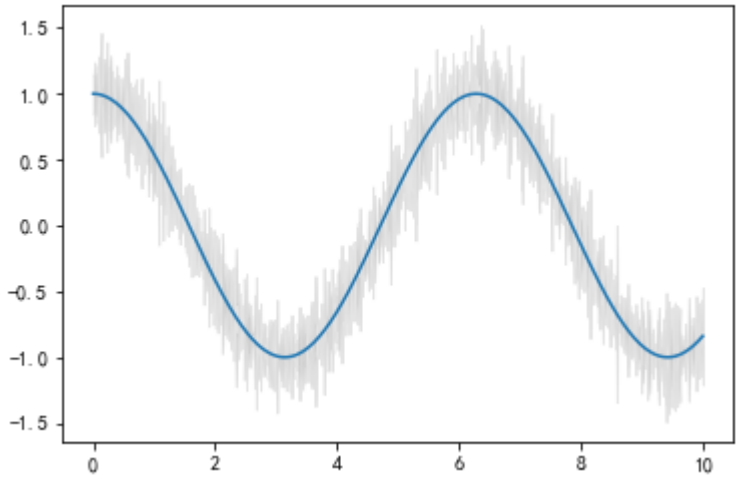
如果单一地使用误差棒图可能不会很好地发挥这种统计图形的实际应用价值，一般将误差棒图和其他统计图形结合起来展示数据集的测量误差等内容。

9.1 连续误差

- plt.fill_between(x,y_01,y_02)
  - x：定义曲线的节点的x坐标（数组）。
  - y_01：定义第一条曲线的节点的y坐标（数组或标量）。
  - y_02：定义第二条曲线的节点的y坐标（数组或标量、可选）。

In [53]:

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

x=np.linspace(0,10,1000)
y=np.cos(x)
yerror=np.random.randn(1000)*0.2

plt.plot(x,y,"-")
plt.fill_between(x,y-yerror,y+yerror,color="gray",alpha=0.2)

plt.show();
```

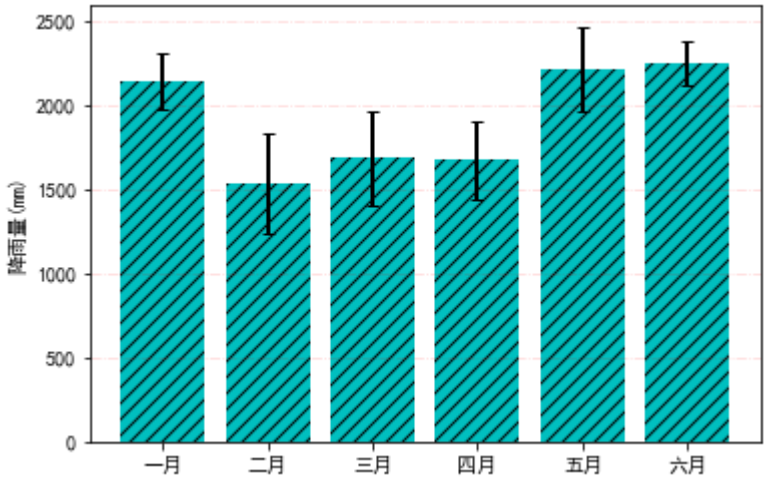


9.2 带误差棒柱状图

- 绘制带误差棒的柱状图，只需要在plt.bar()的参数中添加yerr参数即可。
- 如果你想对这些误差棒设定格式，在plot.bar()中添加error_kw参数即可，而error_kw需要对应一个格式字典，包括误差横帽capsize等。

In [203]:

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

np.random.seed(99)
x=[i for i in range(1,7)]
labels=("一月","二月","三月","四月","五月","六月")

y= np.random.randint(1500,2500,6)
std_err=np.random.randint(100,300,6)
error_attri=dict(elinewidth=2,ecolor="black",capsize=3)

plt.ylabel("降雨量(mm)")

plt.grid(linestyle="-.",color="r",axis="y",alpha=0.15)     #设置红色的、虚线的网格线

plt.bar(x,y,align="center",color="c",tick_label=labels,hatch="///",yerr=std_err,error_kw=e
#hatch参数还可以填充“/”、“///”“\\”、“|”、“-”、'+', 'x', 'o', 'O', '.', '*'

plt.show();
```



10 衍生统计图形

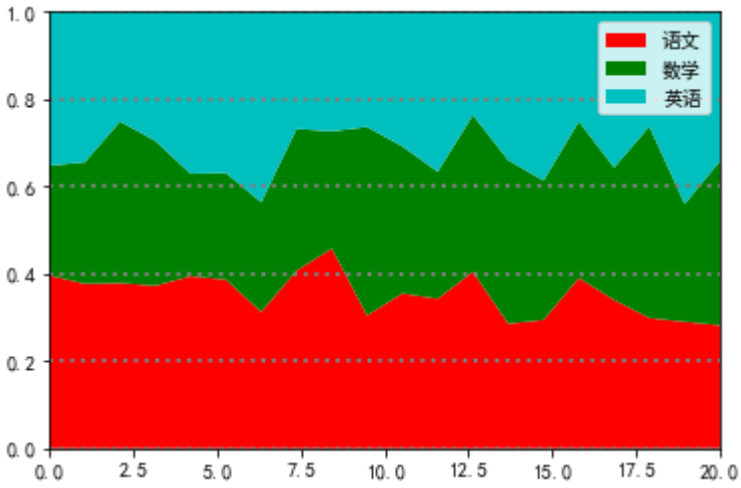
基于折线图、柱状图、条形图绘制原理，可以衍生出堆积折线图、间断条形图和阶梯图。

10.1 堆积折线图plt.stackplot()

堆积折线图是按照垂直方向上彼此堆叠且又不相互覆盖的排列顺序，绘制若干条折线图而形成的组合图形，其本质就是将若干条折线放在同一坐标轴上，以每条折线下部和下方折线作为填充边界，用一种颜色填充代表此条折线的数值区域。

In [54]:

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

# 生成数据
# x为小明20次摸底考试的成绩。
x = np.linspace(0, 20, 20)
y1 = np.random.randint(50, 100, 20)
y2 = np.random.randint(50, 100, 20)
y3 = np.random.randint(50, 100, 20)

# 计算百分比
y1p = y1 / (y1 + y2 + y3)
y2p = y2 / (y1 + y2 + y3)
y3p = y3 / (y1 + y2 + y3)


labels=['语文', '数学', '英语']
colors=['r', 'g', 'c']
# 比例堆积柱状图
plt.stackplot(x, y1p, y2p, y3p, baseline='zero', labels=labels, colors=colors)

# 显示范围
plt.xlim(0, 20)
plt.ylim(0, 1)

# 添加图例
plt.legend(loc='upper right')
plt.grid(axis='y', color='gray', linestyle=':', linewidth=2)

plt.show();
```

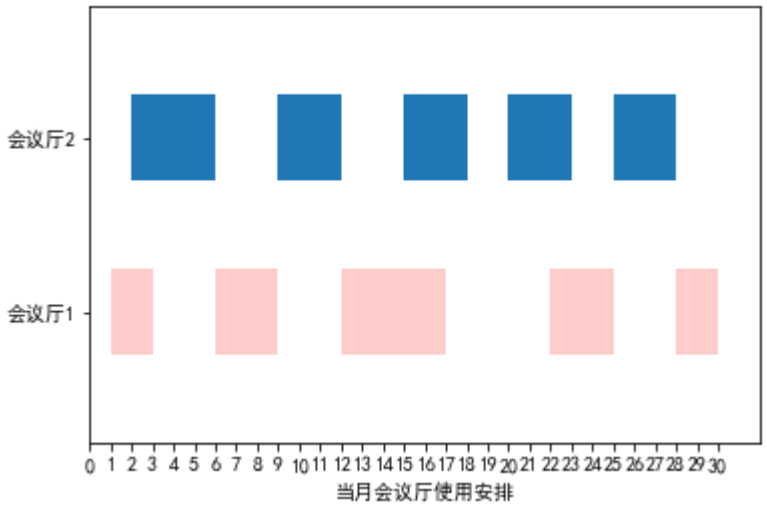


10.2 间断条形图plt.broken_barh()

间断条形图是在条形图的基础上绘制而成，主要用来可视化定性数据的相同指标在时间维度上的指标值的变化情况，实现定性数据的相同指标的变化情况的有效直观比较。

In [55]:

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

plt.broken_barh([(1,2),(6,3),(12,5),(22,3),(28,2)],(10,5),facecolors="#FFCCCC")  #[(1,2)...]
plt.broken_barh([(2,4),(9,3),(15,3),(20,3),(25,3)],(20,5))  #第二个参数（20,5）表示条形图从y

plt.xlim(0,32)
plt.ylim(5,30)
plt.xlabel("当月会议厅使用安排")

plt.xticks(np.arange(0,31,1))
plt.yticks([12.5,22.5],["会议厅1","会议厅2"])

plt.show();
```

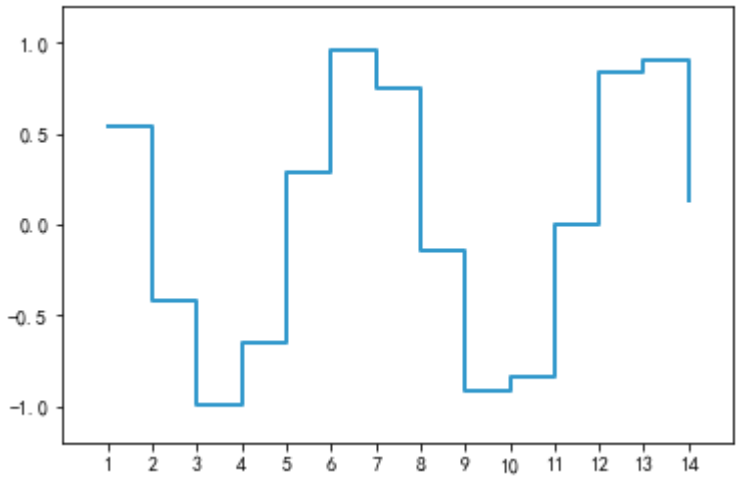


10.3 阶梯图plt.step()

阶梯图用来放映数据的趋势变化或是周期规律，经常用来使用在时间序列数据的可视化任务中，凸显时序数据的波动周期和规律。

In [56]:

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

x=np.arange(1,15)
y=np.cos(x)

plt.step(x,y,color="#3399CC",where="post",lw=2)    #where="pre"为数据点偏左侧绘制x垂直线到下-

plt.xlim(0,15)
plt.xticks(np.arange(1,15,1))
plt.ylim(-1.2,1.2)

plt.show();
```

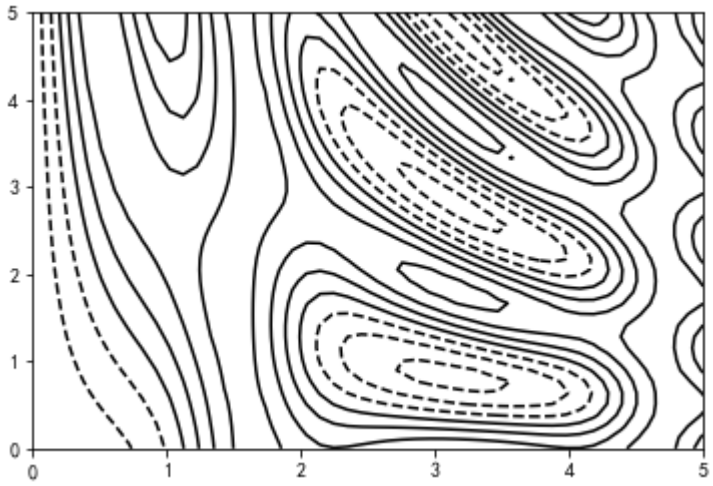


11 三维函数的可视化

- np.meshgrid(x,y)：从坐标向量返回坐标矩阵，可以用两个一维数组构建二维网格数据。
- plt.contour(X,Y,Z)：根据三个坐标轴构建等高图。

In [114]:

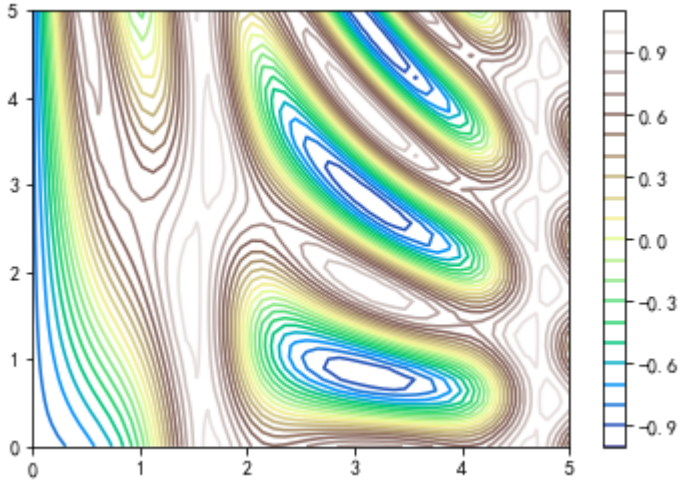
```
def f(x,y):
    return np.sin(x)**10+np.cos(10+y*x)*np.cos(x)

x=np.linspace(0,5,50)
y=np.linspace(0,5,40)

X,Y=np.meshgrid(x,y)
Z=f(X,Y)

plt.contour(X,Y,Z,colors='black')

plt.show();
```



- 需要注意的是，如果只使用一种颜色，会默认使用虚线表示负数，实线表示正数。
- 如果想绘制彩色的等高图，需要用cmap参数来设置线条的配色方案,常用的配色方案：

In [57]:

```
def f(x,y):
    return np.sin(x)**10+np.cos(10+y*x)*np.cos(x)

x=np.linspace(0,5,50)
y=np.linspace(0,5,40)

X,Y=np.meshgrid(x,y)
Z=f(X,Y)

plt.contour(X,Y,Z,20,cmap="terrain")    #讲数据范围等分为20份，用不同的颜色表示

plt.colorbar()

plt.show();
```



想进一步填充，可以使用plt.contourf()，其语法和plt.contour()一样。

In [127]:

```
plt.contourf(X,Y,Z,20,cmap="terrain")    #讲数据范围等分为20份，用不同的颜色表示

plt.colorbar();
```