图标合集

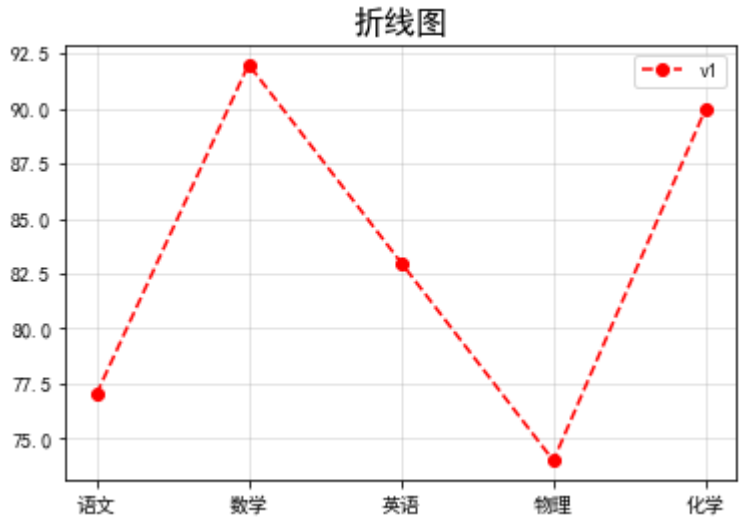
In [2]:

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

plt.rcParams['font.family'] = 'simhei'
plt.rcParams['axes.unicode_minus'] = False

font_title = {
    'size': 16,
}

subject = ['语文', '数学', '英语', '物理', '化学']
v1 = [77, 92, 83, 74, 90]
v2 = [63, 88, 99, 69, 66]
```

折线图

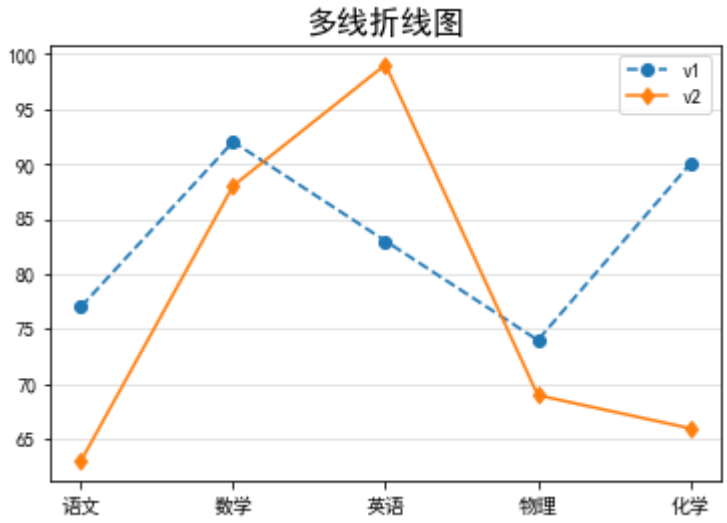
In [2]:

```
plt.title('折线图', fontdict=font_title)
plt.plot(subject, v1, 'ro--', label='v1')
plt.grid(axis='both', alpha=0.4)
plt.legend()
plt.show()
```



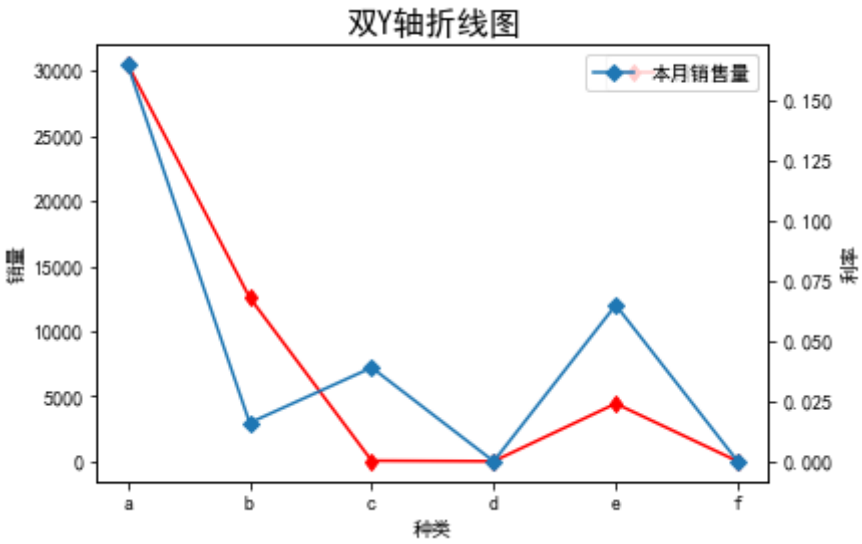
In [3]:

```
plt.title('多线折线图', fontdict=font_title)
plt.plot(subject, v1, 'o--', label='v1')
plt.plot(subject, v2, 'd-', label='v2')
plt.grid(axis='y', alpha=0.4)
plt.legend()
plt.show()
```



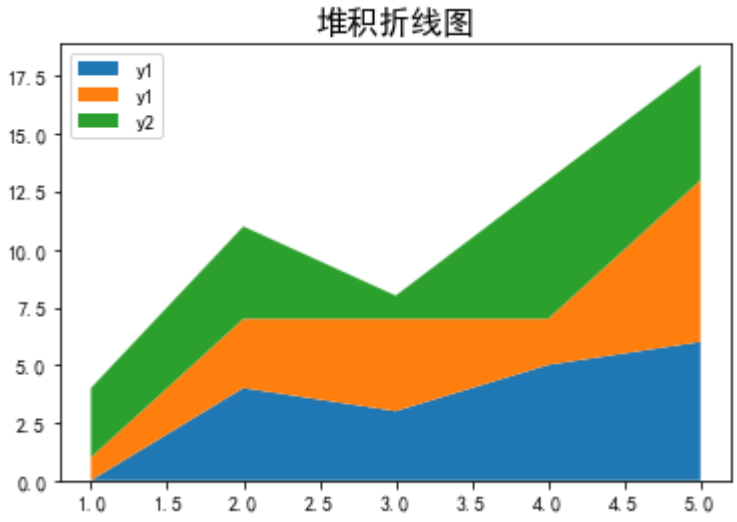
In [4]:

```
x = ['a', 'b', 'c', 'd', 'e', 'f']
y1 = [30481, 12583, 51, 9, 4442, 2]
y2 = [0.165, 0.016, 0.039, 0, 0.065, 0]

plt.title('双Y轴折线图', fontdict=font_title)
plt.plot(x, y1, 'r', marker='d', label='销售人次')
plt.xlabel('种类')
plt.ylabel('销量')
plt.legend()

ax2 = plt.twinx()
ax2.plot(x, y2, marker='D', label='本月销售量')
ax2.set_ylabel('利率')
ax2.legend()
plt.show()
```



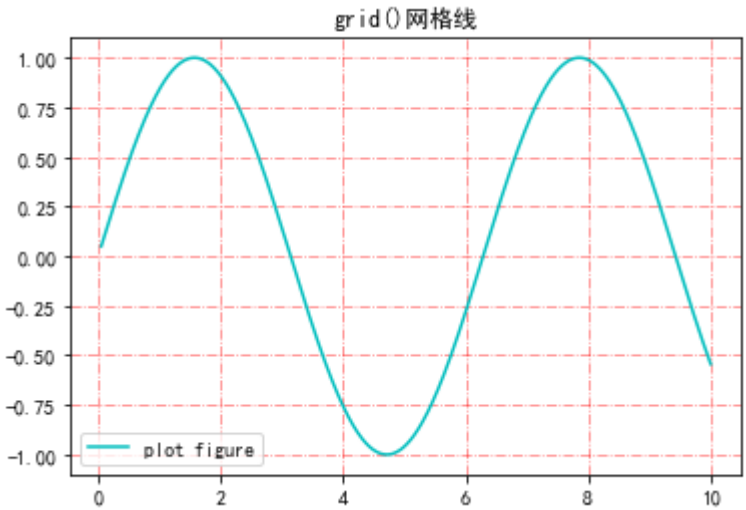
In [5]:

```
x = np.arange(1, 6, 1)
y = [0, 4, 3, 5, 6]
y1 = [1, 3, 4, 2, 7]
y2 = [3, 4, 1, 6, 5]

labels = ["y1", "y1", "y2"]

plt.title('面积图', fontdict=font_title)
plt.stackplot(x, y, y1, y2, labels=labels)
plt.legend(loc="upper left")
plt.show()
```



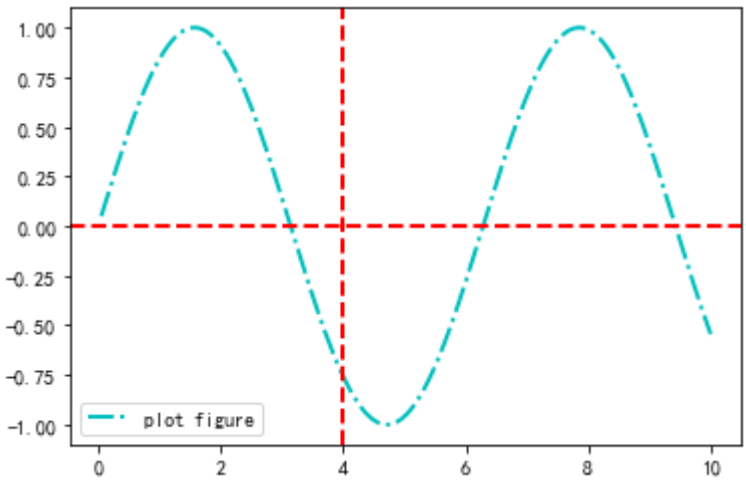
In [3]:

```
# 函数grid()—绘制刻度线的网格线

x = np.linspace(0.05, 10, 1000)
y = np.sin(x)

plt.title('grid()网格线')
plt.plot(x, y, c="c", label="plot figure")
plt.legend()
plt.grid(ls='-.', c='r', alpha=0.5)
plt.show()
```



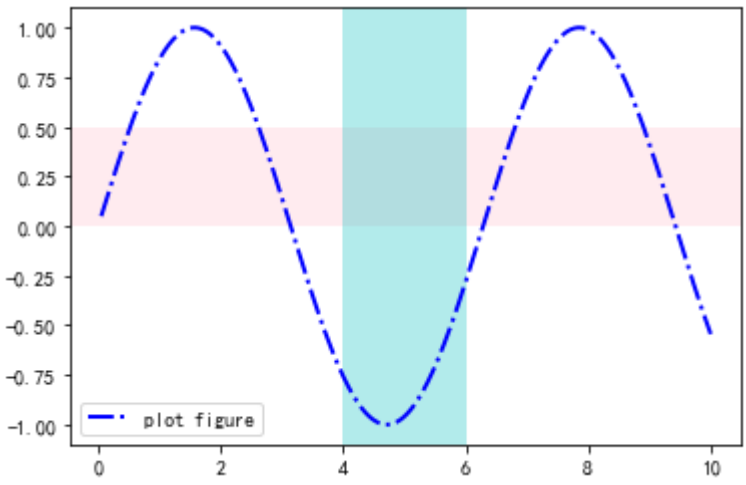
In [7]:

```
# axvline()函数绘制平行于x/y轴的水平参考线

x = np.linspace(0.05, 10, 1000)
y = np.sin(x)

plt.plot(x, y, ls="-.", lw=2, c="c", label="plot figure")
plt.legend()

plt.axhline(y=0.0, c='r', ls='--', lw=2)
plt.axvline(x=4, c='r', ls='--', lw=2)

plt.show()
```



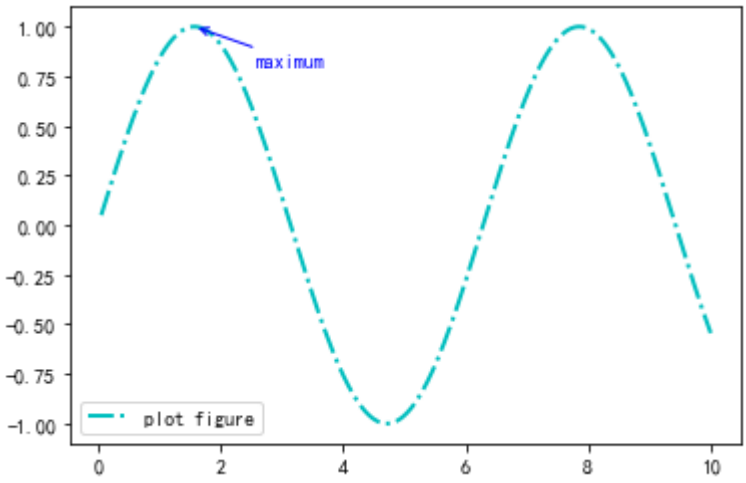
In [5]:

```
# axvspan()函数绘制垂直于x轴的参考区域

x = np.linspace(0.05, 10, 1000)
y = np.sin(x)

plt.plot(x, y, ls="-.", lw=2, c="b", label="plot figure")
plt.legend()

plt.axhspan(ymin=0.0, ymax=0.5, facecolor='pink', alpha=0.3)
plt.axvspan(xmin=4.0, xmax=6.0, facecolor='c', alpha=0.3)

plt.show()
```



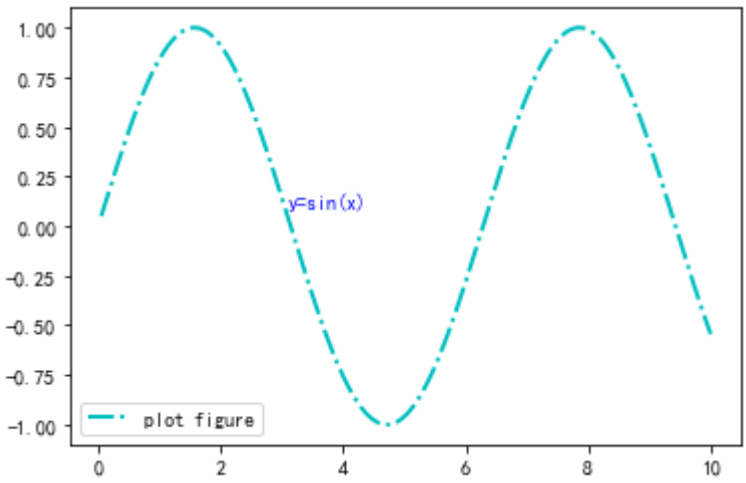
In [9]:

```
# annotate()函数添加图形内容细节的指向型注释文本

x = np.linspace(0.05, 10, 1000)
y = np.sin(x)

plt.plot(x, y, ls="-.", lw=2, c="c", label="plot figure")
plt.legend()
plt.annotate("maximum",
             xy=(np.pi / 2, 1.0),
             xytext=((np.pi / 2) + 1.0, .8),
             weight="bold",
             color="b",
             arrowprops=dict(arrowstyle="->", connectionstyle="arc3", color

plt.show()
```



In [10]:

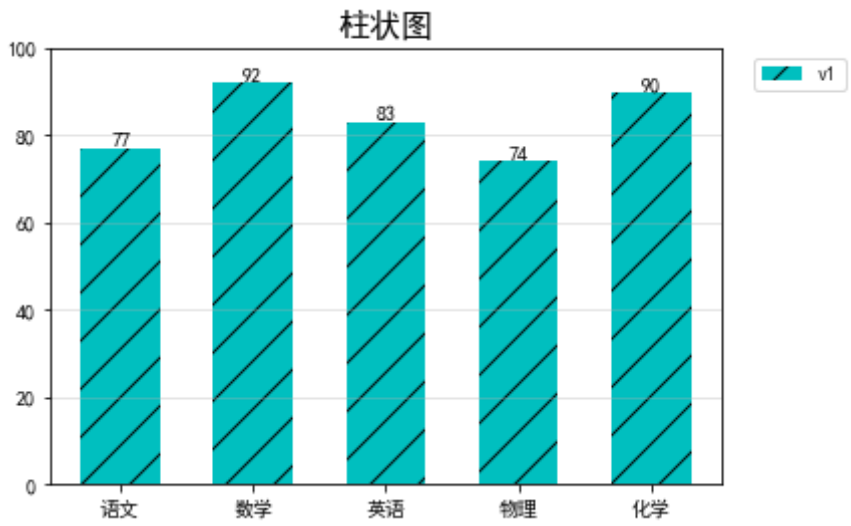
```
# 函数text()—添加图形内容细节的无指向型注释文本

x = np.linspace(0.05, 10, 1000)
y = np.sin(x)

plt.plot(x, y, ls="-.", lw=2, c="c", label="plot figure")
plt.legend()

plt.text(3.1, 0.09, "y=sin(x)", color="b")

plt.show()
```



柱状图

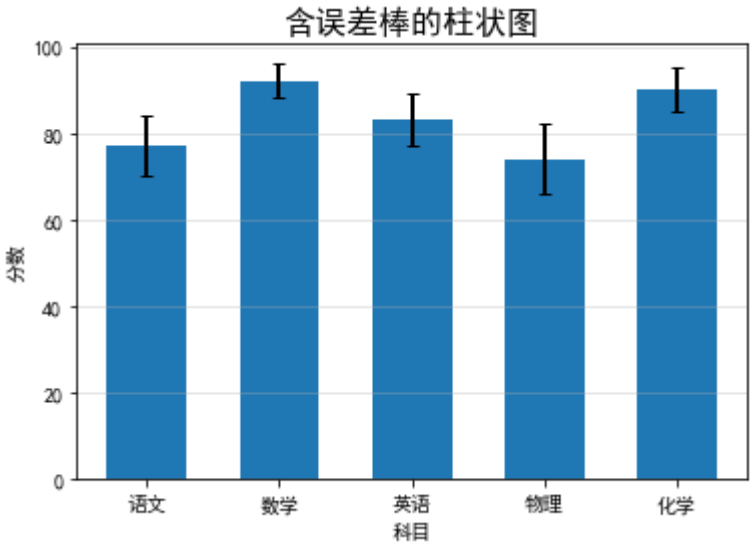
In [6]:

```
plt.title('柱状图', fontdict=font_title)
plt.bar(subject, v1, width=0.6, color='c', hatch="/", label='v1')
for x, y in enumerate(v1):
    plt.text(x, y, y, ha='center', va='bottom')
plt.grid(axis='y', alpha=0.4)
plt.legend(bbox_to_anchor=(0.7, 0, 0.5, 1))
plt.ylim(0, 100)
plt.show()
```



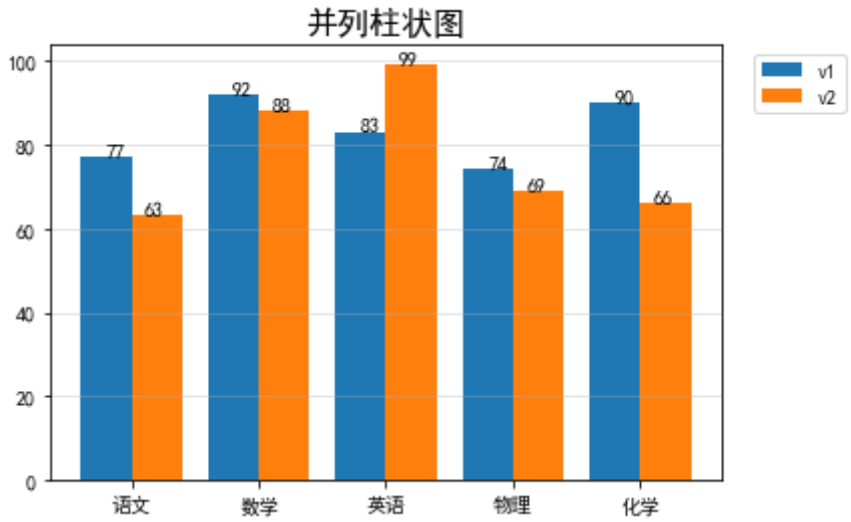
In [12]:

```
std_err = [7, 4, 6, 8, 5]  # 误差棒长度
err_style = dict(elinewidth=2, ecolor="black", capsize=3)  # 误差棒样式

plt.title("含误差棒的柱状图", fontdict=font_title)
plt.bar(subject, v1, width=0.6, yerr=std_err, error_kw=err_style)
plt.xlabel("科目")
plt.ylabel("分数")
plt.grid(axis="y", alpha=0.4)
plt.show()
```



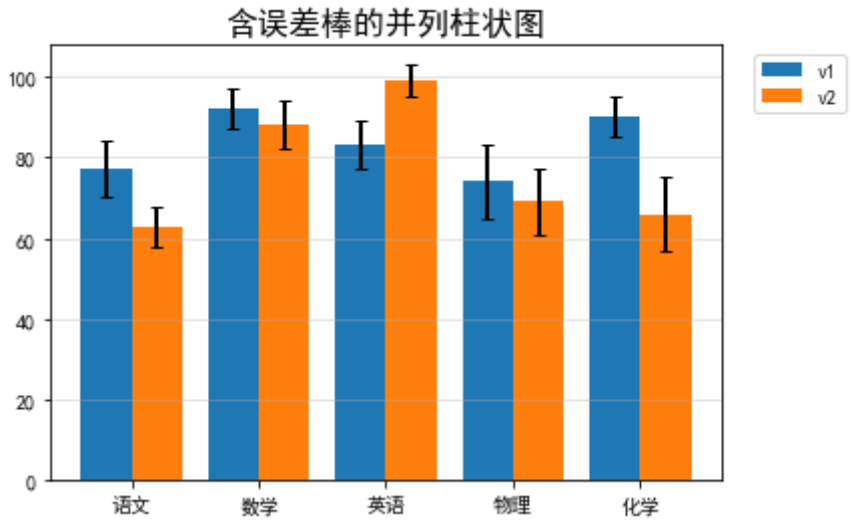
In [13]:

```
plt.title('并列柱状图', fontdict=font_title)
plt.bar(np.arange(len(subject)), v1, width=0.4, label='v1')
for x, y in enumerate(v1):
    plt.text(x, y, y, ha='center', va='bottom')
plt.bar(np.arange(len(subject)) + 0.4, v2, width=0.4, label='v2')
for x, y in enumerate(v2):
    plt.text(x + 0.3, y, y, ha='center', va='bottom')
plt.xticks(np.arange(len(subject)) + 0.4 / 2, subject)
plt.grid(axis='y', alpha=0.4)
plt.legend(bbox_to_anchor=(0.7, 0, 0.5, 1))
plt.show()
```



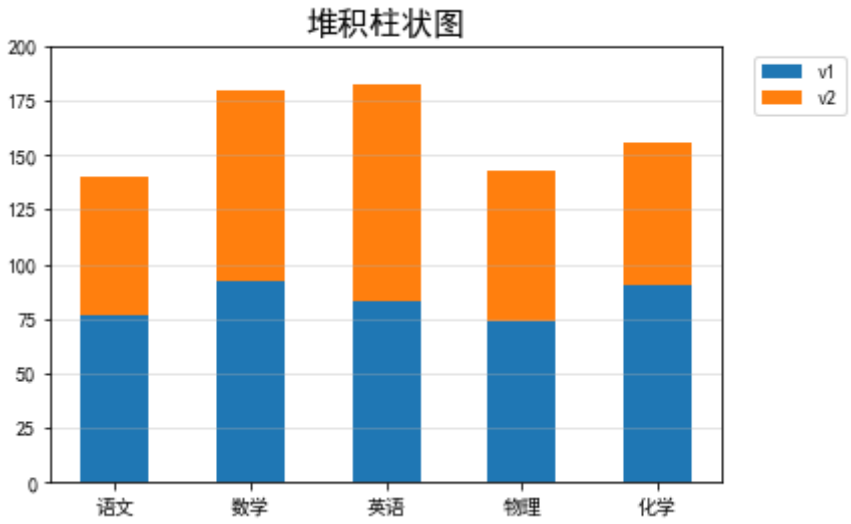
In [11]:

```
v1_err = [7, 5, 6, 9, 5]
v2_err = [5, 6, 4, 8, 9]
err_style = dict(elinewidth=2, ecolor="black", capsize=3)

plt.title("含误差棒的并列柱状图", fontdict=font_title)
plt.bar(subject, v1, width=0.4, yerr=v1_err, error_kw=err_style, label="v1"
plt.bar(np.arange(len(subject)) + 0.4, v2, width=0.4,
        yerr=v2_err, error_kw=err_style, label="v2")
plt.xticks(np.arange(len(subject)) + 0.4 / 2, subject)
plt.grid(axis="y", alpha=0.4)
plt.legend(bbox_to_anchor=(0.7, 0, 0.5, 1))
plt.show()
```



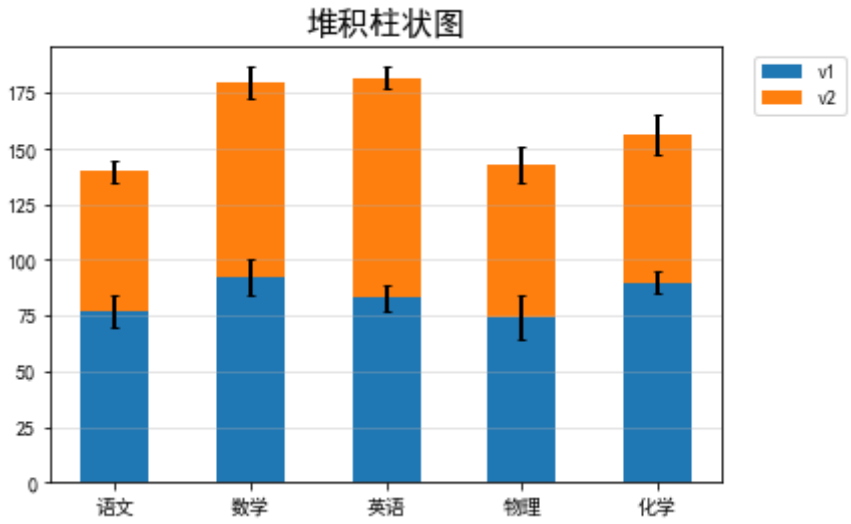
In [10]:

```
plt.title('堆积柱状图', fontdict=font_title)
plt.bar(subject, v1, width=0.5, label='v1')
plt.bar(np.arange(len(subject)), v2, width=0.5, bottom=v1, label='v2')
plt.grid(axis='y', alpha=0.4)
plt.legend(bbox_to_anchor=(0.7, 0, 0.5, 1))
plt.ylim(0,200)
plt.show()
```



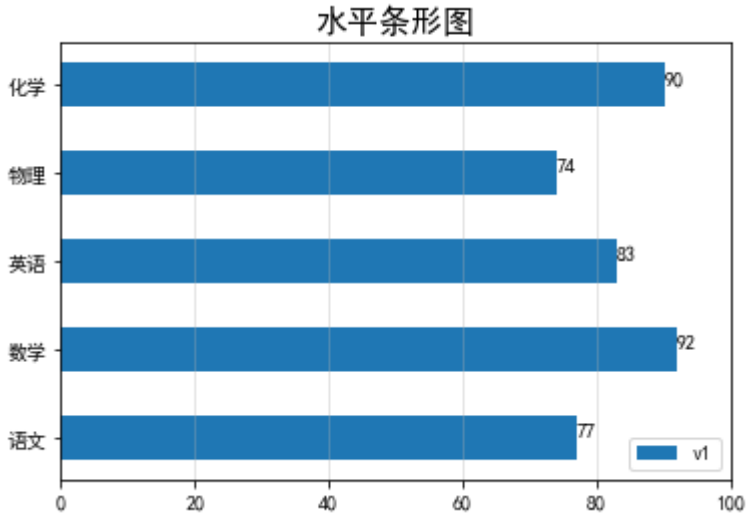
In [16]:

```
v1_err = [7, 8, 6, 10, 5]
v2_err = [5, 7, 5, 8, 9]
err_style = dict(color="black", elinewidth=2, capsize=2)

plt.title('堆积柱状图', fontdict=font_title)
plt.bar(subject, v1, width=0.5, yerr=v1_err, error_kw=err_style, label='v1'
plt.bar(np.arange(len(subject)), v2, width=0.5, yerr=v2_err,
        error_kw=err_style, bottom=v1, label='v2')
plt.grid(axis='y', alpha=0.4)
plt.legend(bbox_to_anchor=(0.7, 0, 0.5, 1))
plt.show()
```



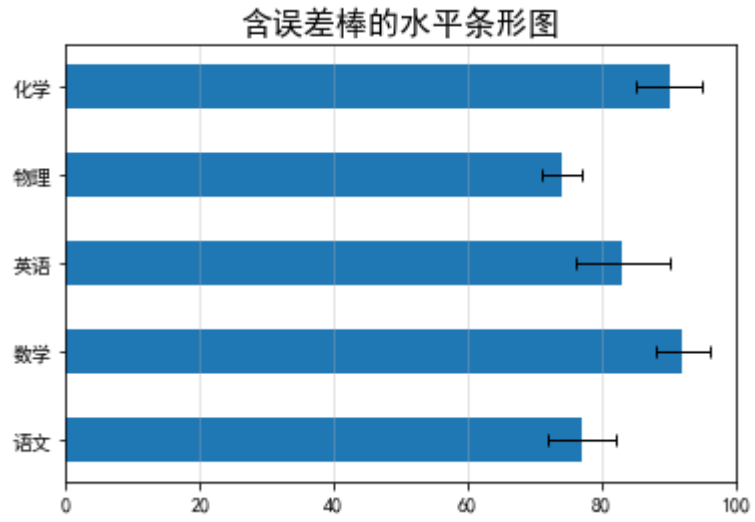
In [12]:

```
plt.title('水平条形图', fontdict=font_title)
plt.barh(subject, v1, height=0.5,label='v1')
for x, y in enumerate(v1):
    plt.text(y, x, y)
plt.legend()
plt.xlim(0, 100)
plt.grid(axis='x', alpha=0.4)
plt.show()
```



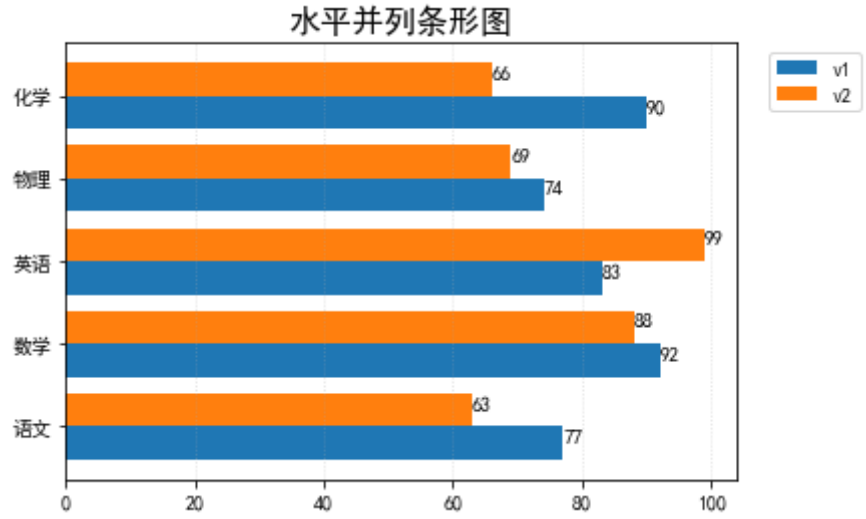
In [13]:

```
std_err = [5, 4, 7, 3, 5]  # 误差棒长度
err_style = dict(elinewidth=1, ecolor='black', capsize=3)

plt.title("含误差棒的水平条形图", fontdict=font_title)
plt.barh(subject, v1, height=0.5, xerr=std_err, error_kw=err_style)
plt.grid(axis="x", alpha=0.4)
plt.xlim(0, 100)
plt.show()
```



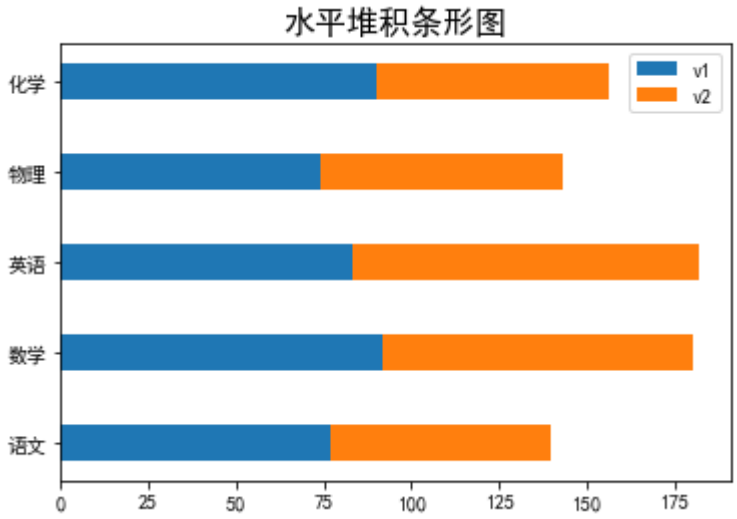
In [14]:

```
plt.title('水平并列条形图', fontdict=font_title)
plt.barh(subject, v1, height=0.4, label='v1')
for x, y in enumerate(v1):
    plt.text(y, x, y)
plt.barh(np.arange(len(subject)) + 0.4, v2, height=0.4, label='v2')
for x, y in enumerate(v2):
    plt.text(y, x + 0.4, y)
plt.legend(bbox_to_anchor=(0.7, 0, 0.5, 1))
plt.yticks(np.arange(len(v1)) + 0.4 / 2, subject)
plt.grid(axis='x', ls=':',alpha=0.4)
plt.show()
```



In [15]:

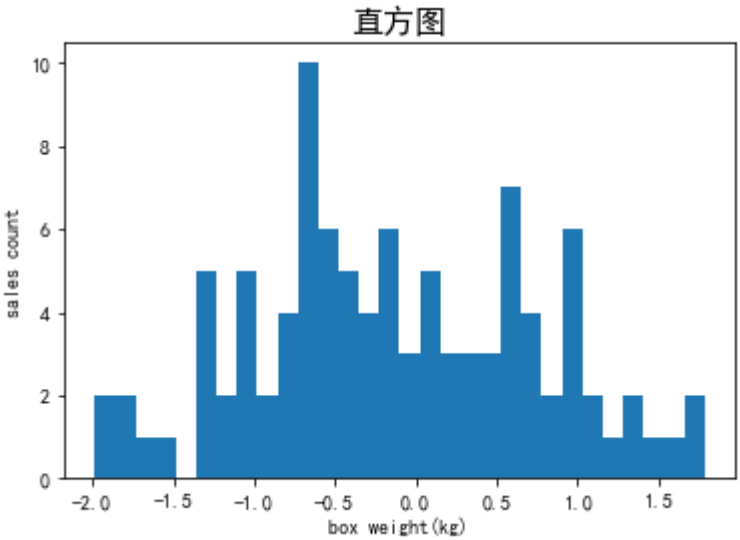
```
plt.title('水平堆积条形图', fontdict=font_title)
plt.barh(subject, v1, height=0.4, label='v1')
plt.barh(subject, v2, height=0.4, left=v1, label='v2')

plt.legend()
plt.show()
```



直方图

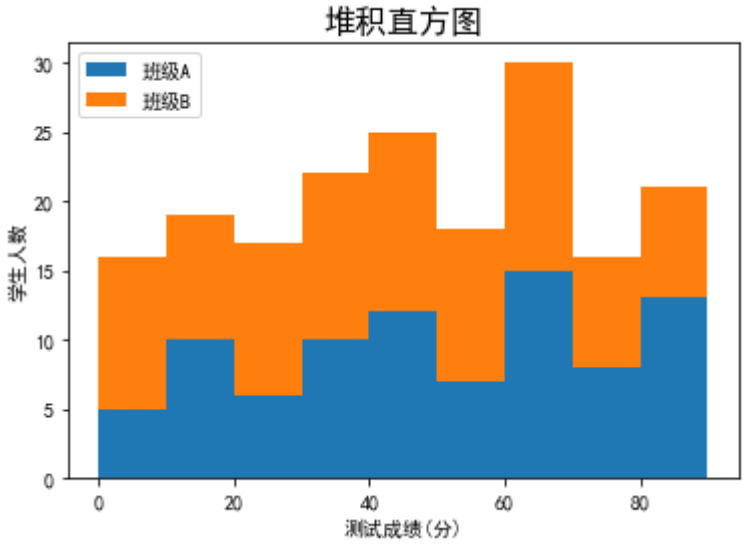
In [21]:

```
plt.title('直方图', fontdict=font_title)
x = np.random.normal(size=100)
plt.hist(x, bins=30)

plt.xlabel("box weight(kg)")
plt.ylabel("sales count")
plt.show()
```



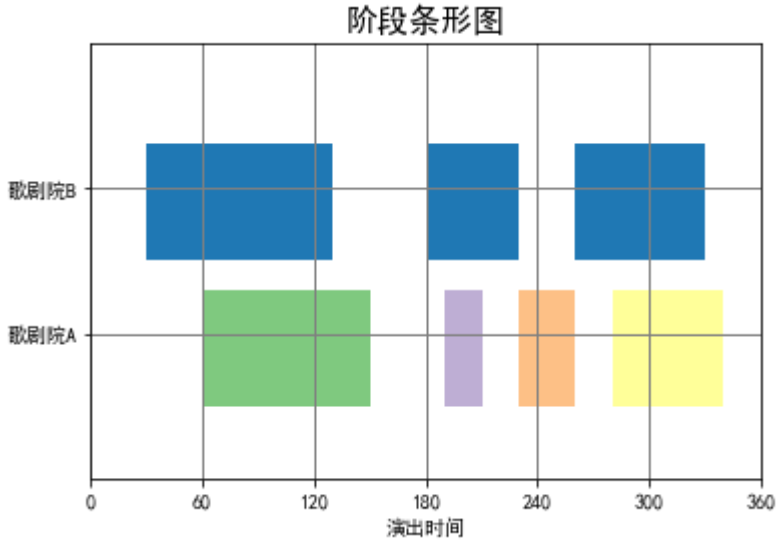
In [19]:

```
scoresT1 = np.random.randint(0, 100, 100)
scoresT2 = np.random.randint(0, 100, 100)
x = [scoresT1, scoresT2]

labels = ["班级A", "班级B"]
bins = range(0, 100, 10)

plt.title("堆积直方图", fontdict=font_title)
plt.hist(x, bins=bins, histtype="bar", stacked=True, label=labels)
plt.xlabel("测试成绩(分)")
plt.ylabel("学生人数")
plt.legend(loc="upper left")
plt.show()
```



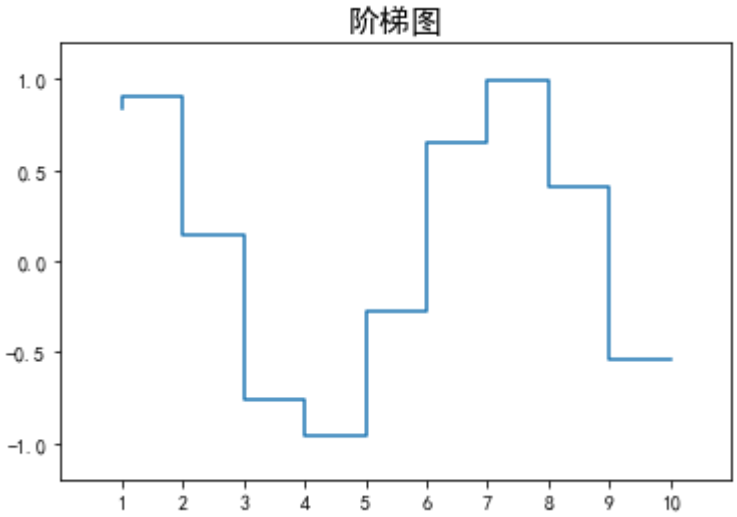
In [17]:

```
x = [(30, 100), (180, 50), (260, 70)]
y = [(60, 90), (190, 20), (230, 30), (280, 60)]

plt.title("阶段条形图", fontdict=font_title)
plt.broken_barh(x, (20, 8), facecolors="#1f78b4")
plt.broken_barh(y, (10, 8), facecolors=(
    "#7fc97f", "#beaed4", "#fdc086", "#ffff99"))
plt.xlim(0, 360)
plt.ylim(5, 35)
plt.xlabel("演出时间")
plt.xticks(np.arange(0, 361, 60))
plt.yticks([15, 25], ["歌剧院A", "歌剧院B"])
plt.grid(ls="-", lw=1, color="gray")
plt.show()
```



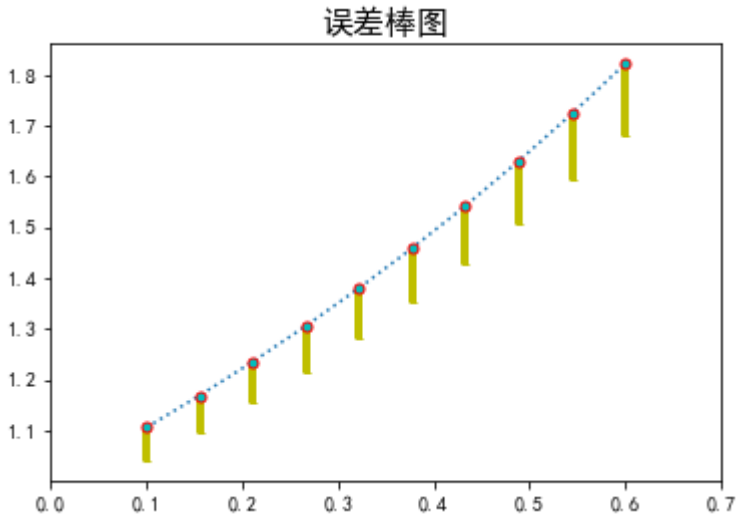
In [20]:

```
x = np.linspace(1, 10, 10)
y = np.sin(x)

plt.title('阶梯图', fontdict=font_title)
plt.step(x, y, where="pre")
plt.xlim(0, 11)
plt.xticks(np.arange(1, 11, 1))
plt.ylim(-1.2, 1.2)
plt.show()
```



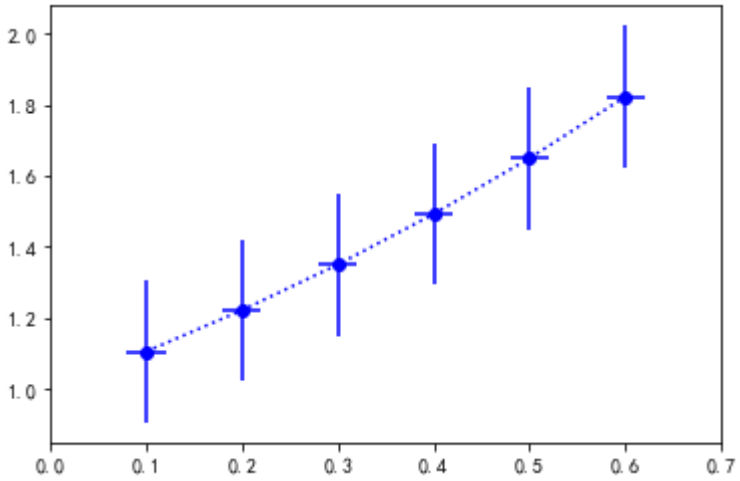
In [25]:

```
x = np.linspace(0.1, 0.6, 10)
y = np.exp(x)
error = 0.05 + 0.15 * x
lower_error = error
upper_error = 0. * error
error_limit = [lower_error, upper_error]

plt.title('误差棒图', fontdict=font_title)
plt.errorbar(x, y, yerr=error_limit, fmt=":o", ecolor="y",
             elinewidth=4, ms=5, mfc="c", mec="r", capthick=1, capsize=2)
plt.xlim(0, 0.7)
plt.show()
```



In [26]:

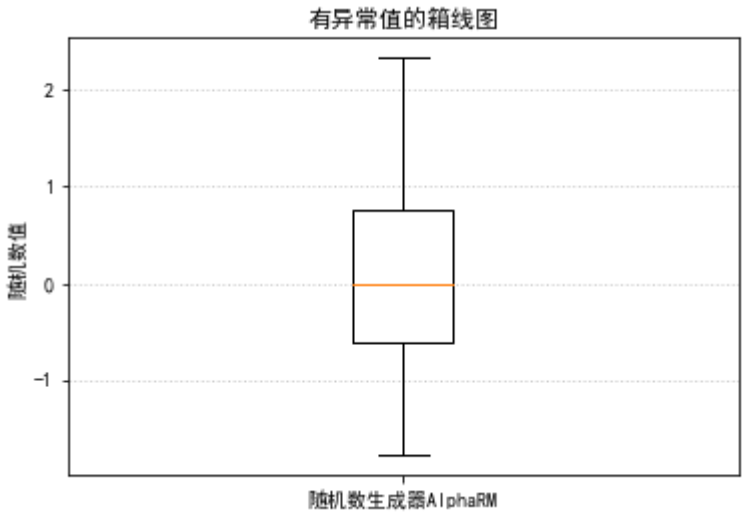
```
# 函数errorbar()——用于绘制误差棒图

x = np.linspace(0.1, 0.6, 6)
y = np.exp(x)

plt.errorbar(x, y, fmt="bo:", yerr=0.2, xerr=0.02)
plt.xlim(0, 0.7)
plt.show()
```



箱线图

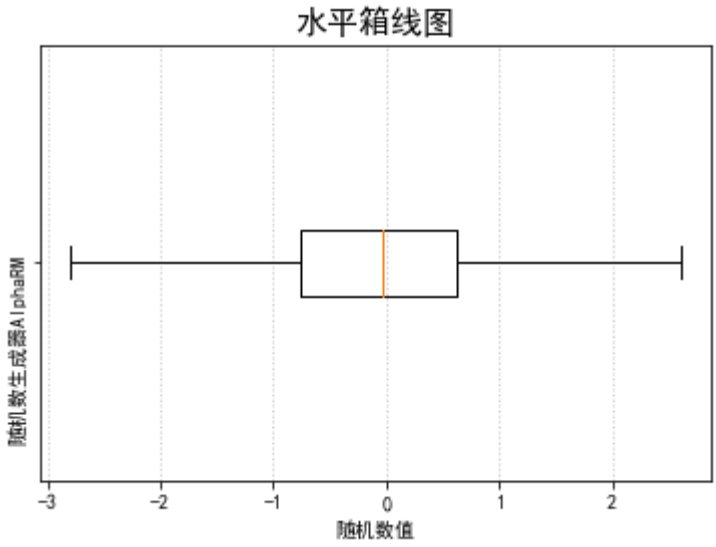
In [21]:

```
# 函数boxplot()——用于绘制箱线图

x = np.random.randn(100)

plt.boxplot(x)
plt.xticks([1], ["随机数生成器AlphaRM"])
plt.ylabel("随机数值")
plt.title("有异常值的箱线图")
plt.grid(axis="y", ls=":", lw=1, color="gray", alpha=0.4)
plt.show()
```



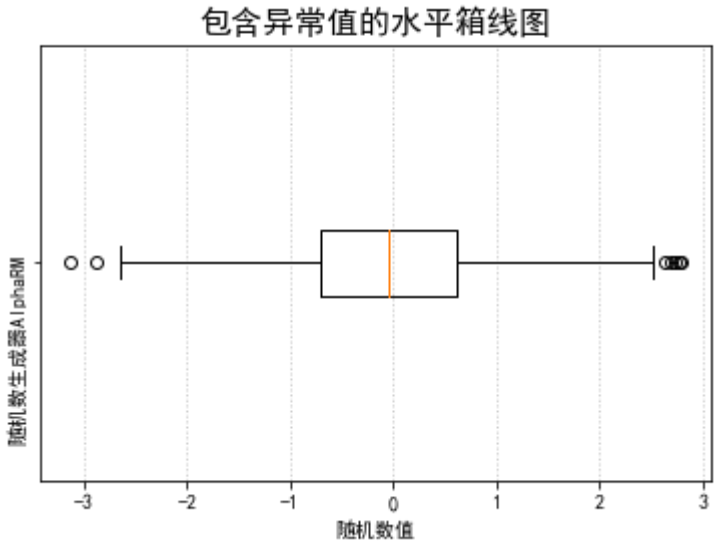
In [28]:

```
# vert: 是否需要将箱线图垂直摆放，默认垂直摆放
# showfliers: 是否显示异常值，默认显示

x = np.random.randn(1000)

plt.title("水平箱线图", fontdict=font_title)
plt.boxplot(x, vert=False, showfliers=False)
plt.xlabel("随机数值")
plt.yticks([1], ["随机数生成器AlphaRM"], rotation=90)
plt.grid(axis="x", ls=":", lw=1, color="gray", alpha=0.4)
plt.show()
```



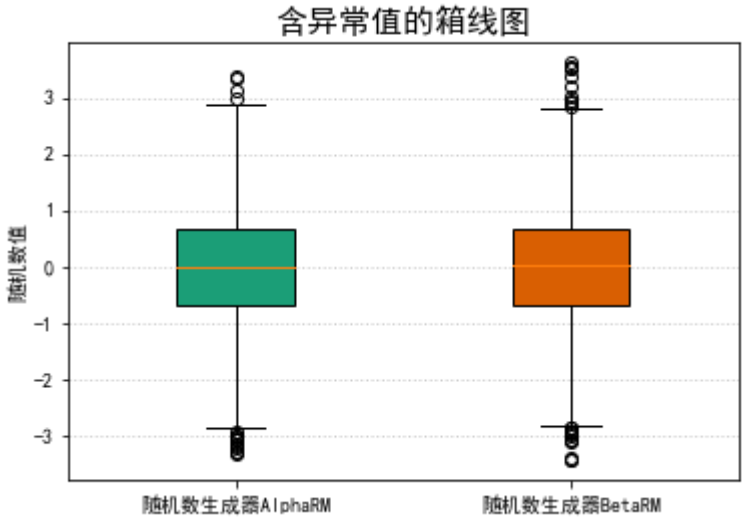
In [29]:

```
x = np.random.randn(1000)

plt.title("包含异常值的水平箱线图", fontdict=font_title)
plt.boxplot(x, vert=False)
plt.xlabel("随机数值")
plt.yticks([1], ["随机数生成器AlphaRM"], rotation=90)
plt.grid(axis="x", ls=":", lw=1, color="gray", alpha=0.4)
plt.show()
```



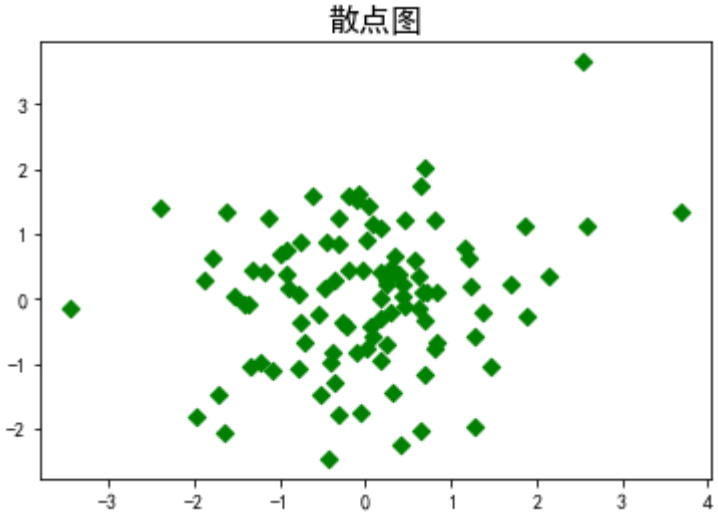
In [30]:

```
testA = np.random.randn(5000)
testB = np.random.randn(5000)
testList = [testA, testB]
labels = ["随机数生成器AlphaRM", "随机数生成器BetaRM"]
colors = ["#1b9e77", "#d95f02"]
whis = 1.6
width = 0.35

plt.title("含异常值的箱线图", fontdict=font_title)
bplot = plt.boxplot(testList, whis=whis, widths=width,
                    sym="o", labels=labels, patch_artist=True)
for patch, color in zip(bplot["boxes"], colors):
    patch.set_facecolor(color)
plt.ylabel("随机数值")

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



In [31]:

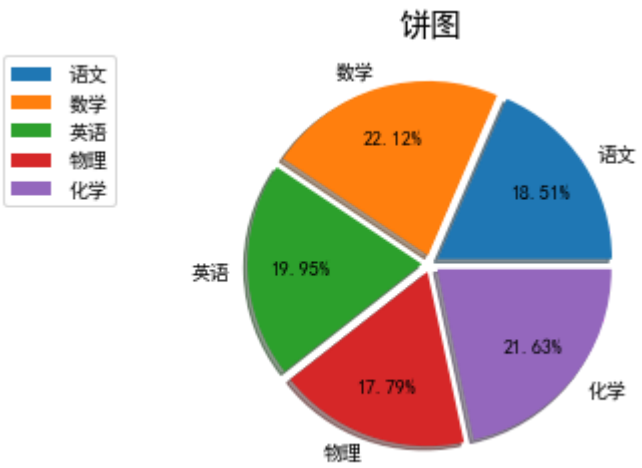
```
plt.title('散点图', fontdict=font_title)
X = np.random.randn(100)
Y = np.random.randn(100)
plt.scatter(X, Y, color='green', marker='D')
plt.show()
```



饼图

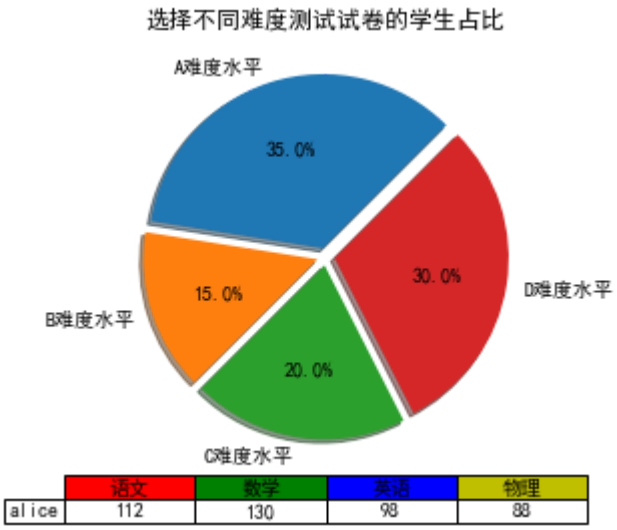
In [23]:

```
plt.title('饼图', fontdict=font_title)
explode = [0.05] * len(subject)
plt.pie(v1, explode, subject, autopct='%.2f%%', pctdistance=0.7,shadow=True
plt.legend(subject, bbox_to_anchor=(-0.7, 0, 0.5, 1))
plt.show()
```



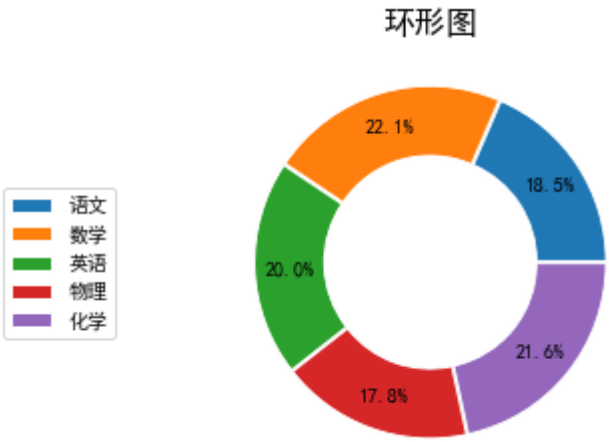
In [33]:

```
# 向统计图添加表格

# 绘图
labels = "A难度水平", "B难度水平", "C难度水平", "D难度水平"
students = [0.35, 0.15, 0.20, 0.30]
explode = [0.05] * 4

plt.title("选择不同难度测试试卷的学生占比")
plt.pie(students, explode=explode, labels=labels, autopct="%1.1f%%",
        startangle=45, shadow=True)

# 制表
colLabels = ["语文", "数学", "英语", "物理"]
rowLabels = ["alice"]
Values = [[112, 130, 98, 88]]
colColors = ["r", "g", "b", "y"]

plt.table(cellText=Values,
          cellLoc="center",
          colWidths=[0.3] * 4,
          colLabels=colLabels,
          colColours=colColors,
          rowLabels=rowLabels,
          rowLoc="center",
          loc="bottom")
plt.show()
```



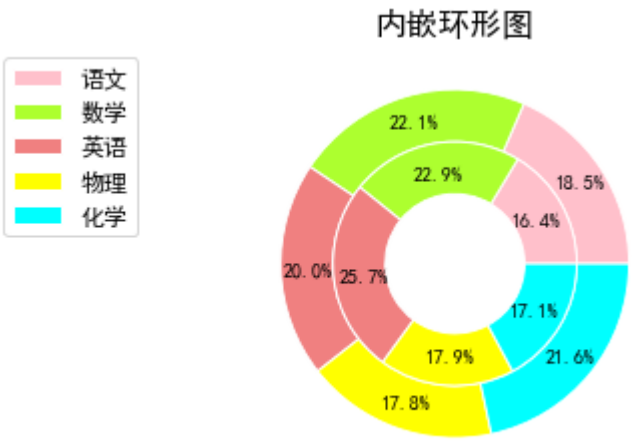
In [24]:

```
plt.title("环形图", fontdict=font_title)
explode = [0.01] * len(subject)
plt.pie(v1, explode, autopct="%.1f%%", pctdistance=0.8,
        wedgeprops=dict(width=0.4, edgecolor='w'))
plt.legend(subject, loc="center right",
           bbox_to_anchor=(-0.7, 0, 0.5, 1))
plt.show()
```



In [25]:

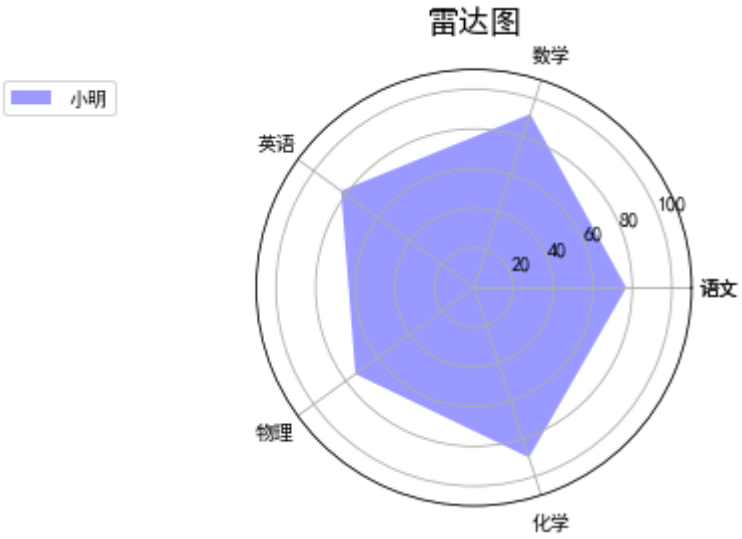
```
'''
radius: 饼图半径，默认值为1
pctdistance: 饼块内标签与圆心的距离,默认值为0.6, autopct不为None该参数生效
shadow: 饼图是否有阴影，默认值为False
wedgeprops: 饼块属性。字典。默认值为None
wedgeprops -> width: 饼块宽度
wedgeprops -> edgecolor: 饼块边缘线颜色
'''

color = ['pink', 'greenyellow', 'lightcoral', 'yellow', 'cyan']

plt.title("内嵌环形图", fontdict=font_title)
# 外环
plt.pie(v1, radius=1, pctdistance=0.85, autopct="%.1f%%", colors=color,
        wedgeprops=dict(width=0.3, edgecolor='w'))

# 内环
plt.pie(v2, radius=0.7, pctdistance=0.75, autopct="%.1f%%", colors=color,
        wedgeprops=dict(width=0.3, edgecolor='w'))

plt.legend(subject, fontsize=12, bbox_to_anchor=(-0.7, 0, 0.5, 1))
plt.show()
```



雷达图

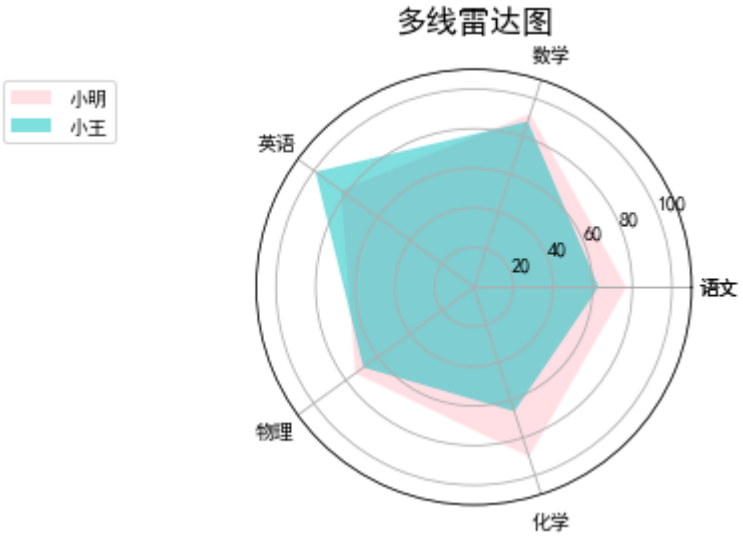
In [26]:

```
subject = ['语文', '数学', '英语', '物理', '化学']
v1 = [77, 92, 83, 74, 90]
v2 = [63, 88, 99, 69, 66]

theta = np.linspace(0, 2 * np.pi, len(v1), endpoint=False)
theta = np.concatenate((theta, [theta[0]]))
subject = np.concatenate((subject, [subject[0]]))
v1 = np.concatenate((v1, [v1[0]]))

plt.subplot(projection='polar')
plt.title('雷达图', fontdict=font_title)
plt.polar(theta, v1, 'bo-')
plt.fill(theta, v1, 'b', alpha=0.4, label='小明')
plt.legend(bbox_to_anchor=(-0.8, 0, 0.5, 1))
plt.thetagrids(theta * 180 / np.pi, subject)
plt.ylim(0, 110)
plt.show()
```



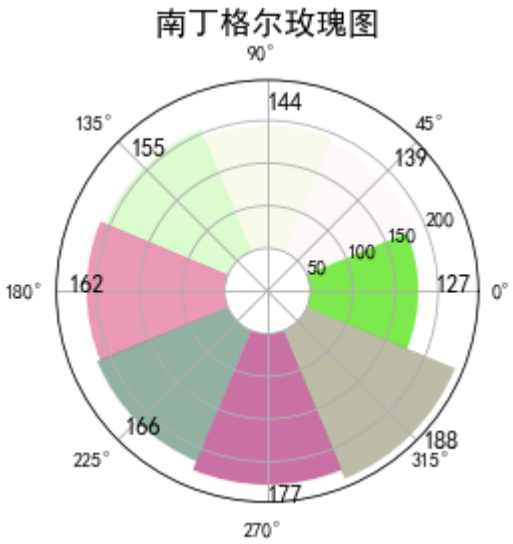
In [27]:

```
subject = ['语文', '数学', '英语', '物理', '化学']
v1 = [77, 92, 83, 74, 90]
v2 = [63, 88, 99, 69, 66]

theta = np.linspace(0, 2 * np.pi, len(v1), endpoint=False)
theta = np.concatenate((theta, [theta[0]]))
subject = np.concatenate((subject, [subject[0]]))
v1 = np.concatenate((v1, [v1[0]]))
v2 = np.concatenate((v2, [v2[0]]))

plt.subplot(projection='polar')
plt.title('多线雷达图', fontdict=font_title)
plt.polar(theta, v1)
plt.fill(theta, v1, 'pink', alpha=0.5, label='小明')
plt.polar(theta, v2)
plt.fill(theta, v2, 'c', alpha=0.5, label='小王')
plt.thetagrids(theta * 180 / np.pi, subject)
plt.legend(bbox_to_anchor=(-0.8, 0, 0.5, 1))
plt.ylim(0, 110)
plt.show()
```



In [28]:

```
title = ['福建', '辽宁', '河北', '安徽', '广东', '山东', '陕西', '广西']
value = [144, 127, 188, 162, 139, 155, 177, 166]
value = np.sort(value)

theta = np.linspace(0, 2 * np.pi, len(title), endpoint=False)

plt.polar()
plt.bar(theta, value,
        width=0.79,  # 控制宽度
        color=np.random.random((len(value), 4)),
        bottom=50,  # 控制中间空白区域大小
        )
# plt.text(1 * np.pi, 0, '南丁格尔', fontsize=13)  # 中间空白区域文字

for t, v in zip(theta, value):  # 柱状图数量
    plt.text(t, v + 70, v, fontsize=12)

plt.title('南丁格尔玫瑰图', fontdict=font_title)
# plt.axis(False)  # 关闭极坐标轴
plt.tight_layout()  # 自动调节子图大小
plt.show()
```

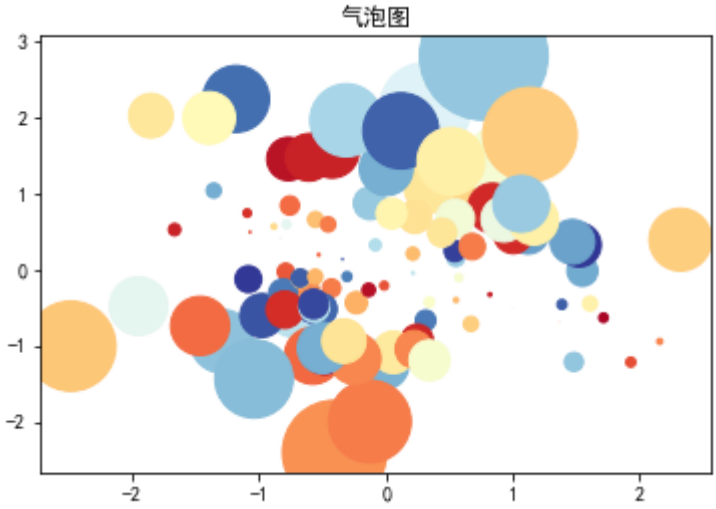


In [39]:

```
subject = ['语文', '数学', '英语', '物理', '化学']
v1 = [77, 92, 83, 74, 90]
v2 = [63, 88, 99, 69, 66]
```

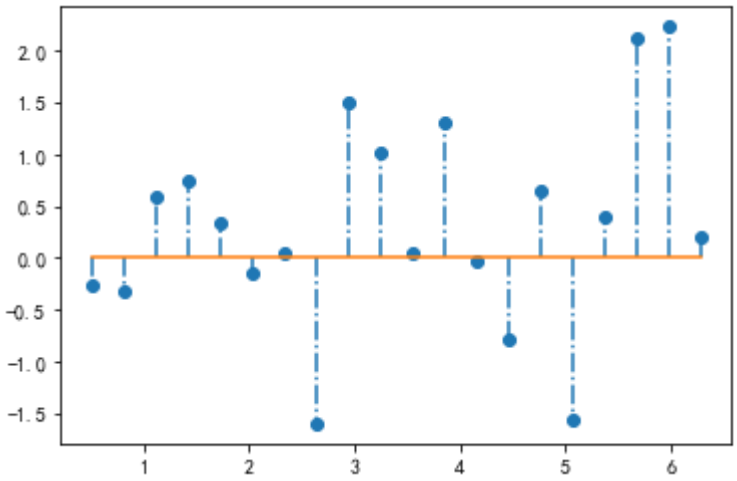
In [40]:

```
# 函数scatter()—用于绘制气泡图

a = np.random.randn(100)
b = np.random.randn(100)

plt.title('气泡图')
plt.scatter(a, b, s=np.power(10 * a + 20 * b, 2),
            c=np.random.rand(100),
            cmap=plt.cm.RdYlBu,
            marker="o")
plt.show()
```



In [41]:

```
# 函数stem()—用于绘制棉棒图

x = np.linspace(0.5, 2 * np.pi, 20)
y = np.random.randn(20)

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



In [29]:

```
# 函数polar()—用于绘制极线图

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

plt.title('极线图')
plt.polar(theta, r, color="chartreuse",
          linewidth=2, marker="*", mfc="b", ms=10)
plt.show()
```

```
<ipython-input-29-51c13baeed39>:8: UserWarning: Trying to create polar plot on an Ax
es that does not have a polar projection.
  plt.polar(theta, r, color="chartreuse",
```