提示：重点在于如何完美的拼接两侧的条图。

请尝试实现分城市比较分学历消费者信心指数均值的多重雷达图。

请实现受教育程度和职业交叉情况下消费者信心指数均值的热图，并且将单元格做行、列排序，将热图左上角设定为均值高的区域，右下角设定为均值低的区域。

11 自由绘图

11.1 区域填充

在指定区域之间上色填充属于基本的绘图功能，在matplotlib中则可以借此实现其他方式难于绘制的一些特殊效果/图形。

11.1.1 填充两个水平曲线间的区域

matplotlib.pyplot.fill_between(

```
    x : 曲线的x坐标，array (length N)
    y1 : 第一条曲线的y坐标，array (length N)
    y2 = 0 : 第二条曲线的y坐标，array (length N)
    where = None : 绘制时是否包括所对应的区域，array of bool (length N)
    interpolate = False : 当使用where参数并且两曲线有交叉时有效
        当两条曲线比较接近时，要求计算精确的交叉位置并加以绘制
    step : 当各x的数值之间y应当是常数时，确定具体的常数取值方式
        "pre": y值恒等于左侧的上一个x,y数据对应的y值
        "post": y值恒等于右侧的下一个x,y数据对应的y值
        "mid": 取两侧y值的平均
```

)

In [ ]:

```
# 一个简单的填充示例，填充和横轴间的空间
plt.fill_between([1,2,3,4,5], [1,2,1,2,1])
```

In [ ]:

```
# 一个简单的填充示例，填充和横轴间的空间
plt.fill_between([1,2,3,4,5], [1,2,1,2,1],
                 where = [False, True, True, True, False])
```

In [ ]:

```
# 一个简单的填充示例
plt.fill_between([1,2,3,4,5], [1,2,1,2,1], [2,1,2,1,2], color = 'g')
```

In [ ]:

```
# 组合填充
plt.fill_between([1,2,3,4,5], [1,2,1,2,1], [2,1,2,1,2], color = 'g')
plt.fill_between([1,2,3,4,5], [2,1,2,1,2], color = 'b')
```

11.1.2 填充密闭的多边形区域

matplotlib.pyplot.fill()

```
    ax.fill(x, y)                    # a polygon with default color
    ax.fill(x, y, "b")               # a blue polygon
    ax.fill(x, y, x2, y2)            # two polygons
    ax.fill(x, y, "b", x2, y2, "r")  # a blue and a red polygon
```

In [ ]:

```
# 来点复杂的图形
import numpy as np
import matplotlib.pyplot as plt

t = np.linspace(-3, 3, 100)

x = 16*(np.sin(t))**3
y = 13*np.cos(t)-5*np.cos(2*t)-2*np.cos(3*t)-np.cos(4*t)

a = plt.figure(figsize = (5, 5))
# plt.plot(x, y)
plt.fill(x, y, 'b')
```

In [ ]:

```
# 多边形填充组合
plt.figure(figsize = (5, 5))
plt.fill(x, y, 'b',
         x/1.5, y/1.5, 'r',
         x/3, y/3, 'y',
         x/15, - y/15, 'c')
sns.set_style('white')
sns.despine(left = True, bottom = True)
plt.xticks([])
plt.yticks([])
```

In [ ]:

```
plt.figure(figsize = (5, 5))
plt.fill(x, y, 'b',
         x/1.5, y/1.5, 'r',
         x/3, y/3, 'y',
         x/15, -y/15, 'c')
plt.axis('off')
```

11.2 绘制多边形

绘制多边形属于matplotlib的底层功能，使得用户可以在其它简单方式无法完成需求的情况下，完全按照自己的想法绘制出所需的图像。

matplotlib的patches模块中提供了许多种绘制多边形的工具，所绘制的多边形均需要使用ax.add_patch()命令添加到Axes对象中才能显示出来。

完整列表：https://matplotlib.org/api/patches_api.html (https://matplotlib.org/api/patches_api.html)

11.2.1 绘制各种常见多边形

matplotlib.patches.circle(center = (0.0, 0.0), radius = 1.0, readonly = False)

圆形：中心坐标，半径，是否只读

matplotlib.patches.Ellipse(xy, width, height, angle = 0.0)

椭圆：中心坐标，宽度，高度，起始角度

matplotlib.patches.Rectangle(xy, width, height, angle = 0.0)

矩形：中心坐标，宽度，高度，起始角度

matplotlib.patches.RegularPolygon(xy, numVertices, radius = 5, orientation = 0)

正多边形：中心坐标，边数，距离中心的长度，起始弧度

matplotlib.patches.Wedge(center, r, theta1, theta2, width = None)

扇形：中心坐标，半径，起始角度，终止角度，弧形宽度

In [ ]:

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

fig, ax = plt.subplots()

# 圆的圆心坐标
xy1 = np.array([0.2, 0.2])
# 矩形的左下角
xy2 = np.array([0.1, 0.7])
# 多边形的中心
xy3 = np.array([1.0, 0.2])
# 椭圆的中心
xy4 = np.array([1.0, 0.8])
# 多边形的中心
xy5 = np.array([0.6, 0.6])

# 第二个参数是圆的半径
circle = mpatches.Circle(xy1, 0.1, color = "r")
ax.add_patch(circle)

# 第二个参数和第三个参数是先长后宽
rectangle = mpatches.Rectangle(xy2, 0.2, 0.2, color= "g")
ax.add_patch(rectangle)

# 第二个参数是边数，第三个是距离中心的位置
polygon1 = mpatches.RegularPolygon(xy3, 5, 0.1, color= "y")
ax.add_patch(polygon1)

# 绘制椭圆，第二个参数是椭圆的长直径，第三个是短直径
# 切记，是直径
ellipse = mpatches.Ellipse(xy4, 0.4, 0.2, color= "b")
ax.add_patch(ellipse)

polygon2 = mpatches.RegularPolygon(xy5, 6, 0.2, color= "c")
ax.add_patch(polygon2)

ax.add_patch(mpatches.RegularPolygon((1,1), numVertices=3,
                                     radius = .2, orientation= 3.14))
ax.add_patch(mpatches.Wedge((0.5, 0), 0.3, 10, 45))
ax.add_patch(mpatches.Wedge((0,0), 0.5, 10, 45, width = 0.1))

# 设置图形显示的时候x轴和y轴等比例
ax.axis("equal")

# 添加背景网格
ax.grid()
```

11.2.2 自由绘制多边形

matplotlib.patches.Polygon(xy, closed = True)

多边形各顶点坐标数组（Nx2格式），多边形是否自动闭合

In [ ]:

```
[[1,1], [1,2], [2,2], [2,1]]
```

In [ ]:

```
import matplotlib.patches as mpatches

fig, ax = plt.subplots()
pol = mpatches.Polygon([[1,1], [1,2], [2,2], [2,1]])
ax.add_patch(pol)
```

In [ ]:

```
import matplotlib.patches as mpatches

fig, ax = plt.subplots(figsize = (5, 5))
pol = mpatches.Polygon([[1,1], [1,2], [2,2], [2,1]], color = 'g')
ax.add_patch(pol)

plt.xlim(0,3)
plt.ylim(0,3)
```

11.3 使用外部图片

pyplot中的imread()和imshow()提供了简单的图像载入和显示功能。

python中有好几个模块都提供图像载入函数，这里只介绍比较简单的一种。

11.3.1 读入图片

plt.imread()：从图像文件读入数据

第一个参数是文件名或文件对象。
format参数指定图像类型，如果省略，就由文件的扩展名决定图像类型。
返回的是一个表示图像的NumPy数组。
对于灰度图像，它返回一个形状为(M，N)的数组
对于彩色图像，返回形状为(M，N，C)的数组。
M为图像的高度,N为图像的宽度,C为3或4,表示图像的通道数。

In [ ]:

```
img = plt.imread("射雕背景0.jpg")
print(img.shape)
img
```

11.3.2 显示图片

matplotlib.pyplot.imshow(

```
    X : 希望进行显示的图片数据
        array_like, shape (n, m) or (n, m, 3) or (n, m, 4)
    cmap : 使用的颜色配置，默认为rc image.cmap中的设定
    aspect = None : 长宽比，["auto" | "equal" | scalar], optional
    interpolation = None : 内插线的绘制方式
        "none", "nearest", "bilinear", "bicubic", "spline16",
        "spline36", "hanning", "hamming", "hermite", "kaiser",
        "quadric", "catrom", "gaussian", "bessel", "mitchell",
        "sinc", "lanczos"
    norm = None : 是否对数据进行标准化，以确保能够正常显示颜色
    vmin, vmax = None : 和norm一起使用，设置数据的最大/最小值范围
    origin = None : 图形是从上往下，还是从下往上开始绘制
        ["upper" | "lower"], optional, 缺省值为rc image.origin
)
```

In [ ]:

```
# 隐藏数轴，只显示图片
plt.imshow(img)
plt.axis('off')
```

In [ ]:

```
# 反向加载图片
plt.imshow(img, origin = 'lower')
```

加载的图片显示必然会占据一个Axes对象，但可以通过使用多个Axes对象的方式，将图像加载到绘图区域的某一部分，形成复合的效果。

In [ ]:

```
img = plt.imread("射雕背景0.jpg")
fig = plt.figure()
# 绘制主体统计图
plt.plot([1,2,2,1])
# 建立绘图所需的Axes对象，事先确认好位置和大小
ax = fig.add_axes([0.65,0.65,0.3,0.2])
# 加载所需图像
ax.imshow(img)
plt.axis('off') # 该命令只对当前使用的Axes对象生效
```

11.4 实战练习

尝试用绘制多边形的方式来实现饼图/圆环图。

思考对于风玫瑰图这种特殊的统计图形，在matplotlib中都有哪些实现方式，其优缺点各是什么。

尝试在统计图的适当位置加上公司logo。

尝试能否做到将公司logo作为统计图背景出现。