2025年春

程序设计实习：Python程序设计

第十七讲 数据分析与可视化

刘家瑛

liujiaying@pku.edu.cn

01 多维数组库NumPy

NumPy简介

- **NumPy (Numerical Python)** 是Python中用于科学计算的基础库
- 提供了高性能的多维数组对象和处理这些数组的工具
- 可以替代多维列表, 速度更快
- 支持向量和矩阵的各种数学运算
- 所有元素类型必须相同

pip install numpy 安装

numpy创建数组的函数

函 数	功 能
array(x)	根据列表或元组x创建数组
arange(x, y, i)	创建一维数组, 元素等价于range(x, y, i)
linespace(x, y, n)	创建一个由区间[x, y]的n-1等分点构成的一维数组, 包含x和y
random.randint(…)	创建一个元素为随机整数的数组
zeros(n)	创建一个元素全为0.0的长度为n数组
ones(n)	创建一个元素全为1.0的长度为n数组

numpy创建数组示例

```
import numpy as np          #以后numpy简写为np
print(np.array([1,2,3]))  #>>输出[1 2 3]
```

将Python列表转换为NumPy数组

(注意NumPy数组显示时无逗号,与列表不同)

```
print(np.arange(1,9,2))     #>>输出[1 3 5 7]
```

生成从1到9(不含端点),步长为2的序列

```
print(np.linspace(1,10,4))#>>输出[ 1.  4.  7. 10.]
```

在1到10之间生成4个等间隔数(含端点)

numpy创建数组示例

```
import numpy as np

print(np.random.randint(10,20,[2,3]))
#生成2行3列的随机整数矩阵
#>>输出 [[12 19 12]
#        [19 13 10]]

print(np.random.randint(10,20,5))
#生成5个[10,20)的随机整数
#>>输出 [12 19 19 10 13]
```

随机数生成时区间是左闭右开[low,high)

numpy创建数组示例

```
import numpy as np
a = np.zeros(3)
print(a)    #>>输出 [ 0.  0.  0.]
print(list(a))
#转为Python列表
#>>输出 [0.0, 0.0, 0.0]
a = np.zeros((2,3), dtype=int)
#创建一个2行3列的元素都是整数0的数组
```


numpy数组常用属性和函数

函　数	功　能
dtype	数组元素的类型
ndim	数组是几维的
shape	数组每一维的长度
size	数组元素个数
argwhere(…)	查找元素
tolist()	转换为list
min()	求最小元素
max()	求最大元素
reshape(…)	改变数组的形状
flatten()	转换成一维数组

numpy数组常用属性和函数

```
import numpy as np
b = np.array([i for i in range(12)])
#b是[ 0  1  2  3  4  5  6  7  8  9 10 11]
a = b.reshape((3,4))  #转换成3行4列的数组,b不变
print(len(a))     #>>3  a有3行
注意: len(a)返回第一维大小(行数),而非总元素数
print(a.size)     #>>12  a的总元素个数是12
print(a.ndim)     #>>2   a维度数是二维数组
print(a.shape)    #>>(3, 4)  a的形状元组(行,列)是3行4列
print(a.dtype)    #>>int32   a的元素类型是32位的整数
```


numpy数组常用属性和函数

```
L = a.tolist()
#tolist()将NumPy数组转为Python原生嵌套列表,原数组a不变
print(L)
#>>[[0, 1, 2, 3], [4, 5, 6, 7], [8, 9, 10, 11]]
b = a.flatten()
#flatten()返回展开的一维数组(始终是拷贝,原数组a不变)
print(b)
#>>[ 0  1  2  3  4  5  6  7  8  9 10 11]
```


numpy数组元素增删

函　数	功　能
append(x,y)	若y是数组，列表或元组，就将y的元素添加进数组x得新数组；否则将y本身添加进数组x得新数组
concatenate(…)	拼接多个数组或列表
delete(…)	删除数组元素得新数组

numpy数组一旦生成，元素就不能增删；

上面函数返回一个新的数组

numpy添加数组元素

```
import numpy as np
a = np.array((1,2,3))          #a是[1 2 3]
b = np.append(a,10)            #a不会发生变化
print(b)                       #>>[ 1  2  3  10]
print(np.append(a,[10,20]))
#>>添加列表,[ 1  2  3  10  20]
c = np.zeros((2,3), dtype=int) #c是2行3列的全0数组
print(np.append(a, c))
#>>[1 2 3 0 0 0 0 0 0]
```

- np.append()会将所有输入强制展平为一维后再拼接

numpy添加数组元素

```
print(np.concatenate((a,[10,20],a)))
#>>[ 1  2  3 10 20  1  2  3]
#将多个一维数组按顺序拼接(支持混合列表和NumPy数组)
```

- 二维数组的行拼接(axis=0默认)

```
print(np.concatenate((c, np.array([[10,20,30]]))))
#c拼接一行[10,20,30]得新数组
```

```
[[ 0  0  0]
 [ 0  0  0]
 [10 20 30]]
```

- 二维数组的列拼接(axis=1)

```
print(np.concatenate((c,np.array([[1,2],[10,20]])),
axis=1))
#c的第0行拼接了[1,2]两个元素；
#第1行拼接了[10,20]两个新元素后得到新数组
```

```
[[ 0  0  0  1  2]
 [ 0  0  0 10 20]]
```

numpy删除数组元素

```
import numpy as np
a = np.array((1,2,3,4))
b = np.delete(a,1)      #删除a中下标为1的元素,a不会改变
print(b)                #>>[1 3 4]
b = np.array([[1,2,3,4],[5,6,7,8],[9,10,11,12]])
print(np.delete(b, 1, axis=0)) #删除b的第1行得新数组
#>>[[ 1  2  3  4]
    [ 9 10 11 12]]
print(np.delete(b, 1, axis=1)) #删除b的第1列得新数组
#>> [[ 1 3 4]
     [ 5 7 8]
     [ 9 11 12]]
```


numpy删除数组元素

```
import numpy as np
b = np.array([[1,2,3,4],[5,6,7,8],[9,10,11,12]])
print(np.delete(b, [1,2], axis=0))
#删除b的第1行和第2行得新数组
#>>输出：[[1 2 3 4]]
print(np.delete(b, [1,3], axis=1))
#删除b的第1列和第3列得新数组
#>>输出：[[ 1 3]
          [ 5 7]
          [ 9 11]]
```


在numpy数组中查找元素

```
import numpy as np
a = np.array((1,2,3,5,3,4))
pos = np.argwhere(a==3)   #pos是[[2] [4]]
a = np.array([[1,2,3],[4,5,2]])
print(2 in a)                    #>>True
pos = np.argwhere(a==2)   #pos是[[0 1] [1 2]]
b = a[a > 2]    #抽取a中大于2的元素形成一个一维数组
print(b)          #>>[3 4 5]
a[a > 2] = -1   #a变成[[ 1  2 -1] [-1 -1  2]]
```


numpy数组的数学运算

```
import numpy as np
a = np.array((1,2,3,4))
b = a + 1
print(b)     #>>[2 3 4 5]
print(a * b)       #>>[ 2  6 12 20]  a,b对应元素相乘
print(a + b)       #>>[3 5 7 9]      a,b对应元素相加
c = np.sqrt(a*10)  #a*10是[10 20 30 40]
print(c)
#>>[ 3.16227766  4.47213595  5.47722558  6.32455532]
```


numpy数组的切片

- numpy数组的切片是“视图”，是原数组的一部分，而非一部分的拷贝

```
import numpy as np
a = np.arange(8)        #a是[0 1 2 3 4 5 6 7]
b = a[3:6]              #注意：b是a的一部分
print(b)                #>>[3 4 5]
c = np.copy(a[3:6]) #c是a的一部分的拷贝
b[0] = 100              #会修改a
print(a)                #>>[  0  1  2  100  4  5  6  7]
print(c)                #>>[3 4 5]  c不受b影响
```


numpy数组的切片

- numpy数组的切片是“视图”，是原数组的一部分，而非一部分的拷贝

```
import numpy as np
a = np.array([[1,2,3,4],[5,6,7,8],
              [9,10,11,12],[13,14,15,16]])
b = a[1:3,1:4]
print(b)
#b是>>[[ 6  7 8] [10 11 12]]
```


02 | 数据分析库pandas

DataFrame的构造和访问

pandas简介

- 核心功能是在二维表格上做各种操作，如增删，修改，求一列数据的和，方差，中位数，平均数等
- 需要numpy支持
- 如果有openpyxl或xlrd或xlwt支持，还可以读写excel文档
- 最关键的类：DataFrame，表示二维表格

pip install pandas 安装

pandas的重要类: Series

- Series是一维表格, 每个元素带标签且有下标, 兼具列表和字典的访问形式

```
import pandas as pd
s = pd.Series(data=[80,90,100],index=['语文','数学','英语'])
#创建了一个Series对象
for x in s:
    print(x, end=" ")
#>>80 90 100, 直接遍历Series会输出所有的值(不包含标签)
print("")
print(s['语文'], s[1])
#>>80 90 标签和位置索引 都可以作为下标来访问元素
```


pandas的重要类: Series

- 切片操作

```
print(s[0:2]['数学']) #>>90

#位置切片s[0:2] 获取前两个元素,遵循前闭后开[start:end)规则

print(s['数学':'英语'][1]) #>>100

#标签切片'数学':'英语'获取两个元素,遵循闭区间[start:end]规则
```


pandas的重要类: Series

```
for i in range(len(s.index)):   #>>语文 数学 英语
      print(s.index[i], end = " ")
#s.index 返回 Series 的索引(标签)列表
s['体育'] = 110 #在尾部添加元素,标签为'体育',值为110
s.pop('数学')    #删除标签为'数学'的元素
s2 = s._append(pd.Series(120, index = ['政治']))
#不改变s
```


pandas的重要类: Series

```
print(s2['语文'], s2['政治']) #>>80 120
print(list(s2))               #>>[80, 100, 110, 120]
print(s.sum(), s.min(), s.mean(), s.median())
#>>290 80 96.66666666666667 100.0
# 输出和,最小值,平均值,中位数
print(s.idxmax(), s.argmax())
#>>体育 2  输出最大元素的标签和下标
```


DataFrame的构造和访问

- DataFrame是带行列标签的二维表格，每一列都是一个Series

```
import pandas as pd
pd.set_option('display.unicode.east_asian_width', True)
#输出对齐方面的设置
scores = [['男',108,115,97],['女',115,87,105],
          ['女',100,60,130],['男',112,80,50]]
names = ['刘一哥','王二姐','张三妹','李四弟']
courses = ['性别','语文','数学','英语']
df = pd.DataFrame(data=scores, index = names,
                  columns = courses)
print(df)
```

	性别	语文	数学	英语
刘一哥	男	108	115	97
王二姐	女	115	87	105
张三妹	女	100	60	130
李四弟	男	112	80	50

DataFrame的构造和访问

```
print(df.values[0][1], type(df.values))
#>>108 <class 'numpy.ndarray'>
print(list(df.index))
#>>['刘一哥','王二姐','张三妹','李四弟']
print(list(df.columns)) #>>['性别','语文','数学','英语']
print(df.index[2], df.columns[2]) #>>张三妹 数学
s1 = df['语文']                    #s1是个Series,代表'语文'那一列
print(s1['刘一哥'], s1[0])    #>>108 108 刘一哥语文成绩
print(df['语文']['刘一哥'])   #>>108 列索引先写
s2 = df.loc['王二姐']
#s2也是个Series,代表“王二姐”那一行
print(s2['性别'],s2['语文'],s2[2])
#>>女 115 87 王二姐的性别,语文和数学分数
```

	性别	语文	数学	英语
刘一哥	男	108	115	97
王二姐	女	115	87	105
张三妹	女	100	60	130
李四弟	男	112	80	50

数据分析库pandas

DataFrame的切片和统计

DataFrame的切片

```
#DataFrame的切片：
#iloc[行选择器,列选择器]  用下标做切片
#loc[行选择器,列选择器]   用标签做切片
#DataFrame的切片是视图
df2 = df.iloc[1:3]
#行切片(是视图),选1和2两行
df2 = df.loc['王二姐':'张三妹']
#和上一行等价
print(df2)
```

	性别	语文	数学	英语
刘一哥	男	108	115	97
王二姐	女	115	87	105
张三妹	女	100	60	130
李四弟	男	112	80	50

	性别	语文	数学	英语
王二姐	女	115	87	105
张三妹	女	100	60	130

DataFrame的切片

```
df2 = df.iloc[:, 0:3]
#列切片(是视图),选0,1,2三列
df2 = df.loc[:,'性别':'数学']
#和上一行等价
print(df2)
```

	性别	语文	数学	英语
刘一哥	男	108	115	97
王二姐	女	115	87	105
张三妹	女	100	60	130
李四弟	男	112	80	50

	性别	语文	数学
刘一哥	男	108	115
王二姐	女	115	87
张三妹	女	100	60
李四弟	男	112	80

DataFrame的切片

```
df2 = df.iloc[:2,[1,3]]
#行列切片
df2 = df.loc[:'王二姐',['语文','英语']]
#和上一行等价
print(df2)
```

	性别	语文	数学	英语
刘一哥	男	108	115	97
王二姐	女	115	87	105
张三妹	女	100	60	130
李四弟	男	112	80	50

	语文	英语
刘一哥	108	97
王二姐	115	105

DataFrame的切片

```
df2 = df.iloc[[1,3], 2:4]
#取第1和3行,第2和3列
df2 = df.loc[['王二姐','李四弟'],'数学':'英语']
#和上一行等价
print(df2)
```

	性别	语文	数学	英语
刘一哥	男	108	115	97
王二姐	女	115	87	105
张三妹	女	100	60	130
李四弟	男	112	80	50

	数学	英语
王二姐	87	105
李四弟	80	50

DataFrame的分析统计

```
print("---下面是DataFrame的分析和统计---")
print(df.T)
#df.T是df的转置矩阵,即行列互换的矩阵
```

	刘一哥	王二姐	张三妹	李四弟
性别	男	女	女	男
语文	108	115	100	112
数学	115	87	60	80
英语	97	105	130	50

```
df.drop('性别', axis=1, inplace=True)
#去除性别,方便计算数值统计
```

	语文	数学	英语
王二姐	115	87	105
李四弟	112	80	50
刘一哥	108	115	97
张三妹	100	60	130

```
print(df.sort_values('语文', ascending=False))
#按语文成绩降序排列
print(df.sum()['语文'],df.mean()['数学'],df.median()['英语'])
#>>435 85.5 101.0
#语文分数之和,数学平均分,英语中位数
```

	性别	语文	数学	英语
刘一哥	男	108	115	97
王二姐	女	115	87	105
张三妹	女	100	60	130
李四弟	男	112	80	50

DataFrame的分析统计

排序方法

sort_values(... inplace=True, axis=1...)

- **inplace=True**：表示直接在原DataFrame上进行修改，而不是返回一个新的排序后的DataFrame
- **axis=1**：表示按列进行排序（默认是axis=0，即按行索引排序）

```
import pandas as pd

df = pd.DataFrame({'B': [1, 2, 3], 'A': [4, 5, 6], 'C': [7, 8, 9]})

# 按列名排序（字母顺序）
df.sort_values(axis=1, inplace=True)
```


DataFrame的分析统计

```
print(df.min()['语文'], df.max()['数学'])
#>>100 115   语文最低分,数学最高分
print(df.max(axis = 1)['王二姐'])
#>>115   王二姐的最高分科目的分数
print(df['语文'].idxmax())
#>>王二姐   语文最高分所在行的标签
print(df['数学'].argmin()) #>>2   数学最低分所在行的行号
print(df.loc[(df['语文'] > 100)&(df['数学'] >= 85)])
```

	性别	语文	数学	英语
刘一哥	男	108	115	97
王二姐	女	115	87	105
张三妹	女	100	60	130
李四弟	男	112	80	50

	性别	语文	数学	英语
刘一哥	男	108	115	97
王二姐	女	115	87	105

DataFrame的修改和增删

```
print("---下面是DataFrame的增删和修改---")
df.loc['王二姐','英语'] = df.iloc[0,1] = 150
#修改王二姐英语和刘一哥语文成绩
df['物理'] = [80,70,90,100] #为所有人添加物理成绩这一列
df.insert(1,"体育",[89,77,76,45])#为所有人插入体育成绩到第1列
df.loc['李四弟'] = ['男',100,100,100,100,100]
#修改李四弟全部信息
df.loc[:,'语文'] = [20,20,20,20] #修改所有人语文成绩
df.loc['钱五叔'] =  ['男',100,100,100,100,100] #加一行
df.loc[:,'英语'] += 10                    #>>所有人英语加10分
df.columns = ['性别','体育','语文','数学','English','物理']
#改列标签
print(df)
```

	性别	体育	语文	数学	English	物理
刘一哥	男	89	20	115	107	80
王二姐	女	77	20	87	160	70
张三妹	女	76	20	60	140	90
李四弟	男	100	20	100	110	100
钱五叔	男	100	100	100	110	100

	性别	语文	数学	英语
刘一哥	男	108	115	97
王二姐	女	115	87	105
张三妹	女	100	60	130
李四弟	男	112	80	50

DataFrame的修改和增删

```
df.drop( ['体育','物理'], axis = 1, inplace = True)
#删除 体育和物理成绩
df.drop( '王二姐', axis = 0, inplace = True)
#删除 王二姐 那一行
print(df)
```

	性别	语文	数学	英语
刘一哥	男	108	115	97
王二姐	女	115	87	105
张三妹	女	100	60	130
李四弟	男	112	80	50

	性别	语文	数学	English
刘一哥	男	20	115	107
张三妹	女	20	60	140
李四弟	男	20	100	110
钱五叔	男	100	100	110

```
df.drop([df.index[i] for i in range(1,3)], axis = 0,
inplace = True) #删除第1,2行
df.drop([df.columns[i] for i in range(3)], axis = 1,
inplace = True) #删除第0到2列
```

	English
刘一哥	107
钱五叔	110

数据分析库pandas

读写excel和csv文档

用pandas读excel文档

- 需要Openpyxl (对.xlsx文件) 或 xlrd或xlwt支持 (老的.xls文件)

安装 pip install openpyxl

- 读取的每张工作表都是一个DataFrame

	A	B	C	D	E	F
1	产品类别	数量	销售额	成本	利润	
2	睡袋	4080	224, 192. 97	180, 501. 27	43, 691. 70	
3	彩盒	502		62, 452. 41	-62, 452. 41	
4	宠物用品	437	51, 558. 43		51, 558. 43	
5	警告标	382	36, 796. 62	32, 100. 23	4, 696. 40	
6	总计	5401	312548. 0199	275053. 904	37494. 11589	
7						

销售情况 | CVOID | odd

用pandas读excel文档

```
import pandas as pd
pd.set_option('display.unicode.east_asian_width', True)
dt = pd.read_excel("excel_sample.xlsx", sheet_name=['销售情况',1],
                   index_col=0)   #读取第0和第1张工作表
df = dt['销售情况']                #dt是字典,df是DataFrame
print(df.iloc[0,0], df.loc['睡袋','数量']) #>>4080 4080
print(df)
```

	数量	销售额	成本	利润
产品类别				
睡袋	4080	224192.9700	180501.270	43691.70000
彩盒	502	NaN	62452.410	-62452.41000
宠物用品	437	51558.4300	NaN	51558.43000
警告标	382	36796.6200	32100.230	4696.40000
总计	5401	312548.0199	275053.904	37494.11589

```
print(pd.isnull(df.loc['彩盒','销售额'])) #>>True
df.fillna(0, inplace=True)                #将所有NaN用0替换
print(df.loc['彩盒','销售额'], df.iloc[2,2])      #>>0.0 0.0
```


用pandas写excel文档

```
df.to_excel(filename, sheet_name="Sheet1",
na_rep= '', ...)
```

- 将DataFrame对象df中的数据写入excel文档filename中的"Sheet1"工作表,NaN用''代替
- 会覆盖原有的filename文件
- 如果要在一个excel文档中写入多个工作表,需要用ExcelWrite

用pandas写excel文档

```
#(接上面程序)
writer = pd.ExcelWriter("new.xlsx") #创建ExcelWriter对象
df.to_excel(writer, sheet_name="S1")
df.T.to_excel(writer, sheet_name="S2") #转置矩阵写入
df.sort_values('销售额', ascending = False).
                    to_excel(writer, sheet_name="S3")
#按销售额排序的新DataFrame写入工作表S3
df['销售额'].to_excel(writer, sheet_name="S4")#只写入一列
writer._save()
```


用pandas读写csv文件

```
df.to_csv("result.csv", sep =",", na_rep = 'NA',
          float_format = "%.2f", encoding = "gbk")
df = pd.read_csv("result.csv", encoding = "gbk")
```


03 | Matplotlib进行数据展示

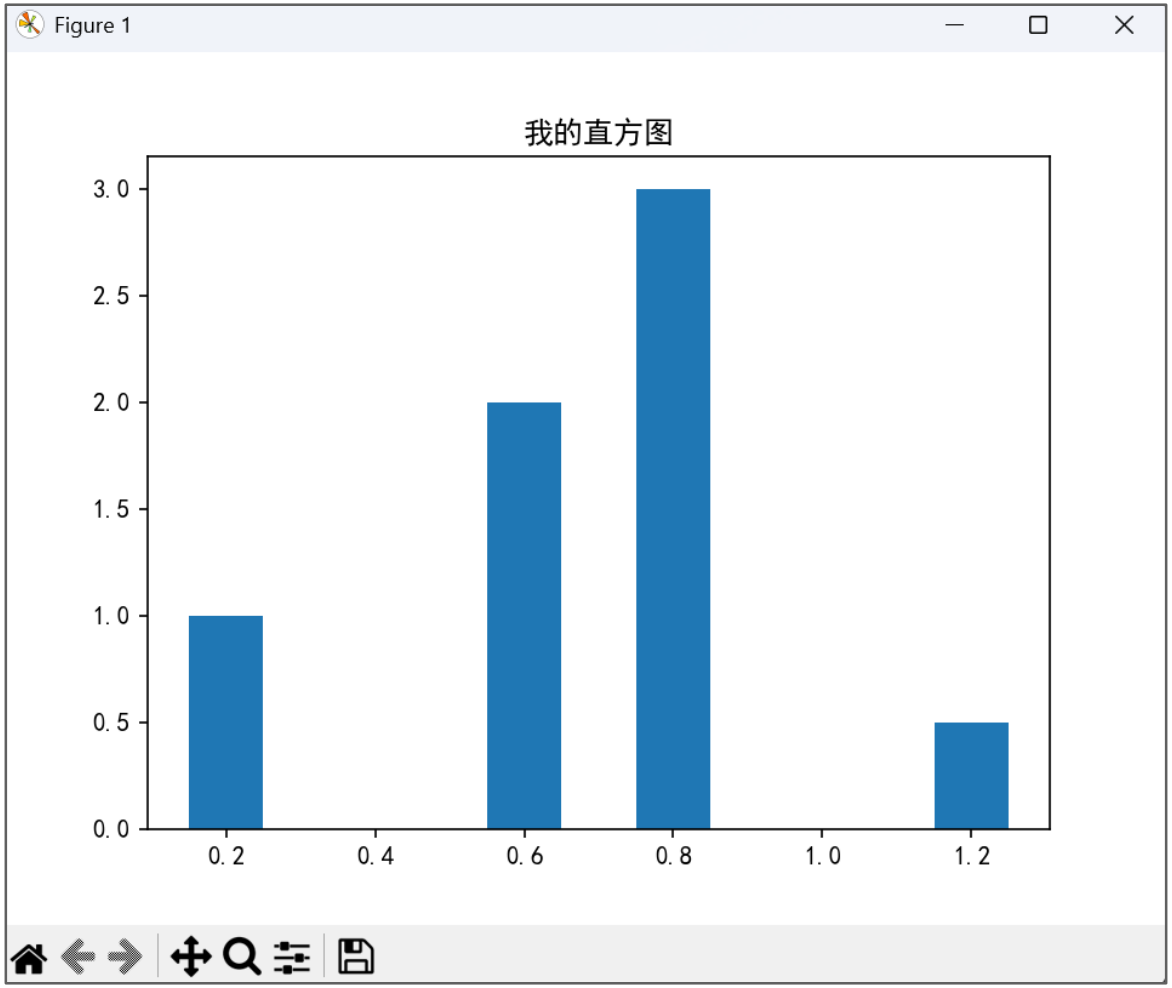
Matplotlib库

- Matplotlib 是 Python 中最流行的数据可视化库之一，由 John D. Hunter 于 2003 年创建
- 提供了绘图接口，能够创建各种静态、动态和交互式的 2D 和简单 3D 图形
- **核心特点**
  - 跨平台兼容：支持 Windows / Linux 和 macOS
  - 多种输出格式：可生成 PNG, PDF, SVG 等格式
  - 与 NumPy 无缝集成：完美处理数组数据
  - 多种图形类型：线图，柱状图，散点图，饼图等

使用安装 pip install matplotlib

绘制直方图

绘制基本直方图



绘制基本直方图

```
import matplotlib.pyplot as plt
#以后plt等价于matplotlib.pyplot
from matplotlib import rcParams
rcParams['font.family'] = rcParams['font.sans-serif'] =
'SimHei' #设置中文支持,中文字体为简体黑体
ax = plt.figure().add_subplot() #建图,获取子图对象ax
ax.bar(x = (0.2,0.6,0.8,1.2), height = (1,2,3,0.5),
width = 0.1) #x表示4个柱子中心横坐标分别是0.2,0.6,0.8, 1
#height表示4个柱子高度分别是1,2,3,0.5
#width表示柱子宽度0.1
ax.set_title ('我的直方图')                    #设置标题
plt.show()                                     #显示绘图结果
plt.savefig("c:/tmp/bar.png")                  #将图保存为文件
```

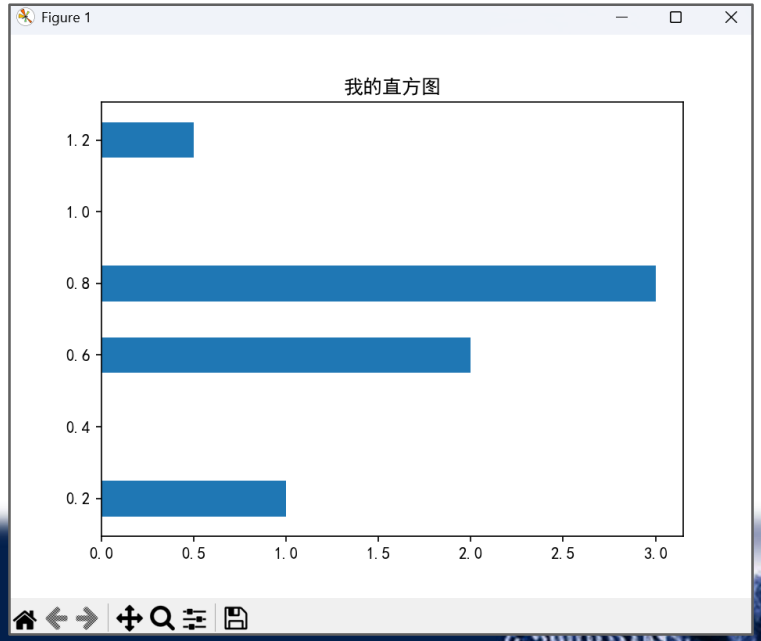

绘制横向直方图

- 纵向

```
ax.bar(x = (0.2,0.6,0.8,1.2), height = (1,2,3,0.5),
width = 0.1)
```

- 横向

```
ax.barh(y = (0.2,0.6,0.8,1.2), width = (1,2,3,0.5),
height = 0.1)
```



绘制堆叠直方图

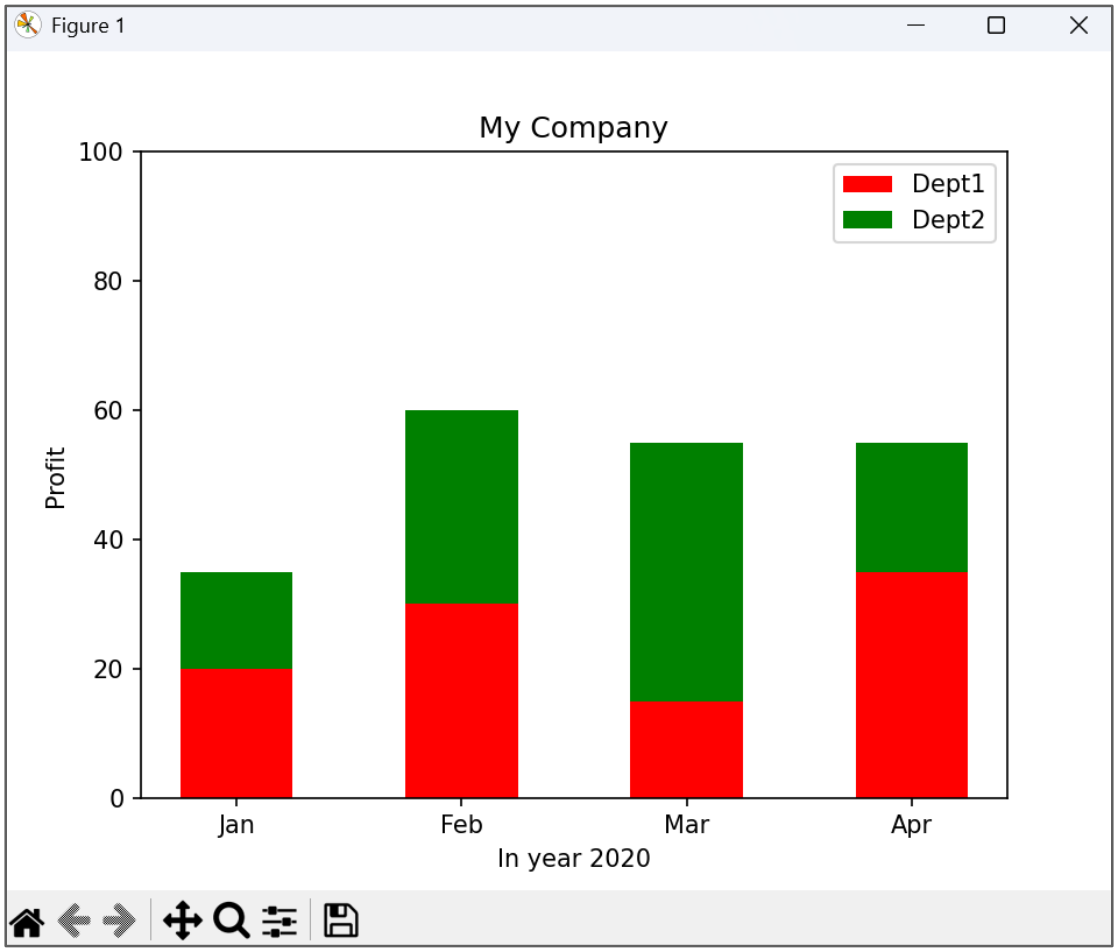


Figure 1
My Company
Dept1
Dept2
100
80
60
40
20
0
Profit
Jan
Feb
Mar
Apr
In year 2020

绘制堆叠直方图

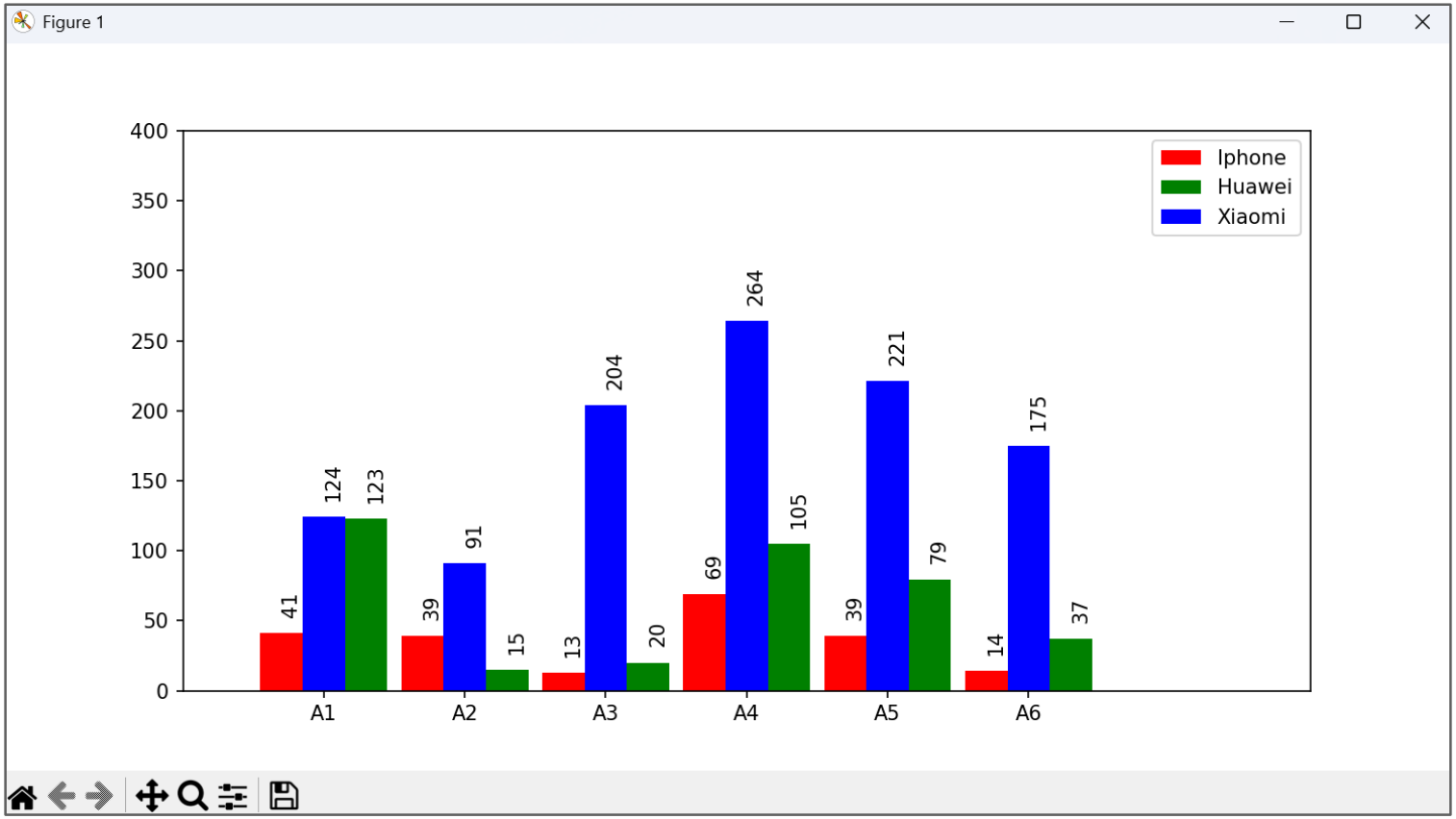
```
import matplotlib.pyplot as plt
ax = plt.figure().add_subplot() #添加一个子图
labels = ['Jan', 'Feb', 'Mar', 'Apr']
num1 = [20, 30, 15, 35]      #Dept1的数据
num2 = [15, 30, 40, 20]      #Dept2的数据
cordx = range(len(num1))     #x轴刻度位置 [0,1,2,3]
rects1 = ax.bar(x = cordx, height=num1, width=0.5,
color='red', label="Dept1")
rects2 = ax.bar(x = cordx, height=num2, width=0.5,
color='green', label="Dept2", bottom=num1)
#bottom=num1: 指定每个柱子的底部位置为Dept1的值,实现堆叠效果
```

绘制堆叠直方图

```
ax.set_ylim(0, 100)                #y轴坐标范围0-100
ax.set_ylabel("Profit")            #y轴含义(标签)
ax.set_xticks(cordx)               #设置x轴刻度位置[0,1,2,3]
ax.set_xticklabels(labels)         #设置x轴刻度标签(下方文字)
ax.set_xlabel("In year 2020")      #x轴含义(标签)
ax.set_title("My Company")
ax.legend()                        #在右上角显示图例说明
plt.show()
```


绘制对比直方图

- 多组数据



```
import matplotlib.pyplot as plt
ax = plt.figure(figsize=(10,5)).add_subplot()
#建图,获取子图对象ax
ax.set_ylim(0,400)    #指定y轴坐标范围
ax.set_xlim(0,80)     #指定x轴坐标范围
#以下是3组直方图的数据
x1 = [7, 17, 27, 37, 47, 57]
#第一组直方图每个柱子中心点的横坐标
x2 = [13, 23, 33, 43, 53, 63]
x3 = [10, 20, 30, 40, 50, 60]
y1 = [41, 39, 13, 69, 39, 14]
#第一组直方图每个柱子的高度
y2 = [123, 15, 20, 105, 79, 37]
y3 = [124, 91, 204, 264, 221, 175]
```


```
rects1 = ax.bar(x1, y1, facecolor='red', width=3,
label = 'Iphone')
rects2 = ax.bar(x2, y2, facecolor='green', width=3,
label = 'Huawei')
rects3 = ax.bar(x3, y3, facecolor='blue', width=3,
label = 'Xiaomi')
ax.set_xticks(x3)
#x轴在x3中的各坐标点下面加刻度
ax.set_xticklabels(('A1','A2','A3','A4','A5','A6'))
#指定x轴上每一刻度下方的文字
ax.legend()                    #显示右上角三组图的说明
```


```
def label(ax, rects):   #在rects的每个柱子顶端标注数值
        for rect in rects:
            height = rect.get_height()
            ax.text(rect.get_x() + rect.get_width()/2,
            height+14, str(height), rotation=90)
            #文字旋转90度
label(ax,rects1)
label(ax,rects2)
label(ax,rects3)
plt.show()
```


绘制散点，折线图

绘制折线图和散点图

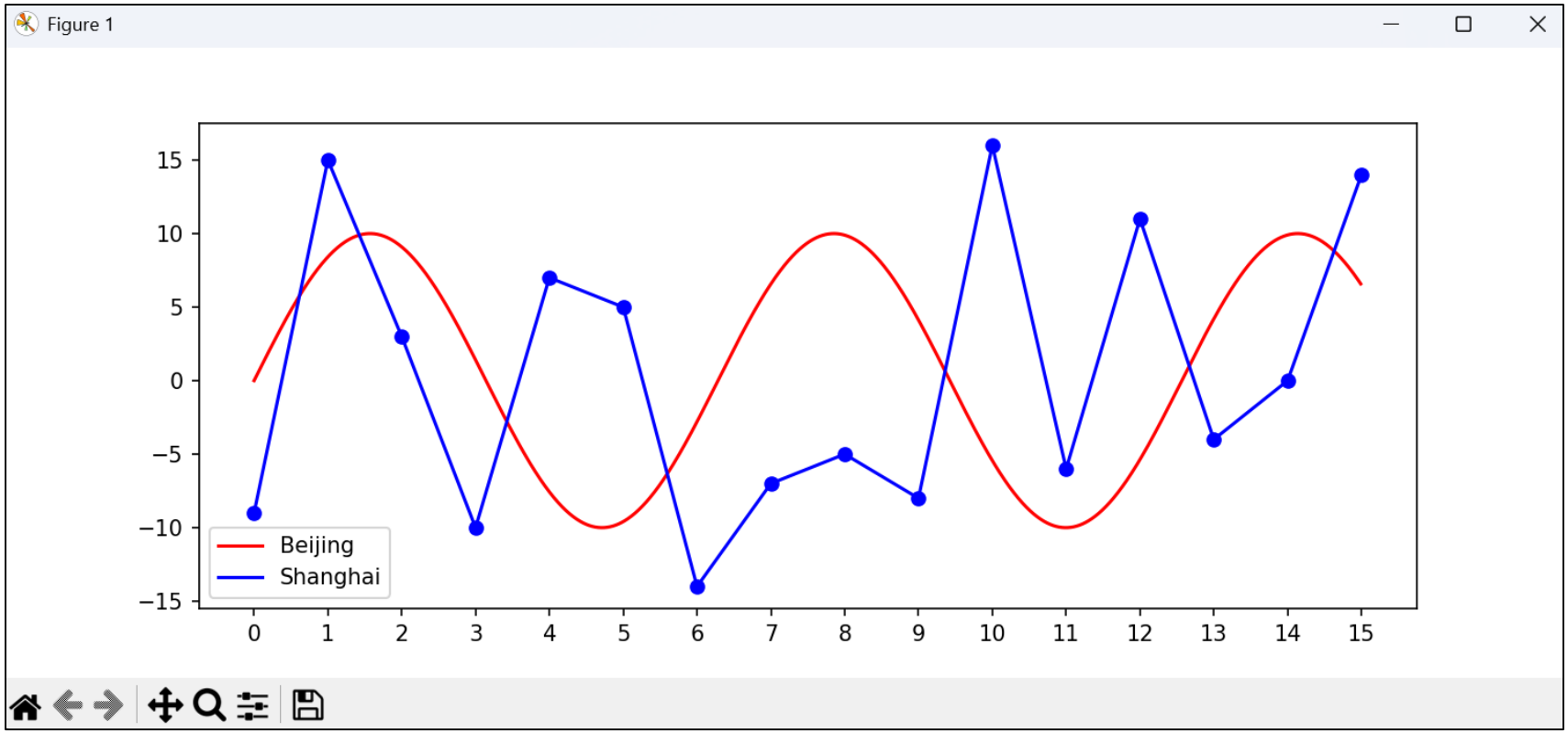


Figure 1
15
10
5
0
−5
−10
−15
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
Beijing
Shanghai

绘制折线图和散点图

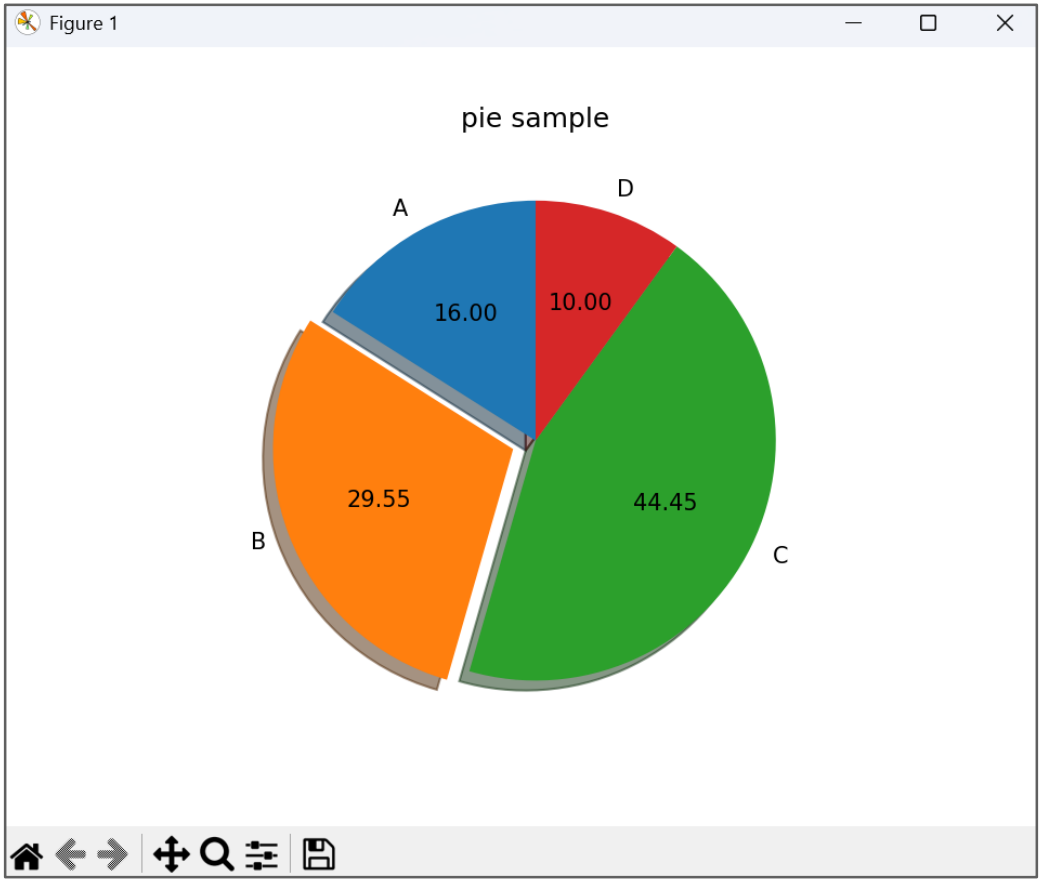
```
import math, random
import matplotlib.pyplot as plt
def drawPlot(ax):
        xs = [i / 100 for i in range(1500)]
        #1500个点的横坐标,间隔0.01
        ys = [10*math.sin(x) for x in xs]
        #对应曲线y=10*sin(x)上的1500个点的y坐标
        ax.plot(xs, ys, "red", label = "Beijing")#画曲线y=10*sin(x)
        ys = list(range(-18,18))
        random.shuffle(ys)
        ax.scatter(range(16), ys[:16], c = "blue")#画前16个点的散点图
        ax.plot(range(16), ys[:16], "blue", label="Shanghai")
        #使用相同的16个随机点绘制蓝色折线
        ax.legend()                          #显示右上角的各条折线说明
        ax.set_xticks(range(16))             #x轴在坐标0, 1, ..., 15处加刻度
        ax.set_xticklabels(range(16)) #指定x轴每个刻度下方显示的文字
```


绘制折线图和散点图

```
ax = plt.figure(figsize=(10,4), dpi=100).add_subplot()

#图像长宽和清晰度

drawPlot(ax)

plt.show()
```


绘制饼图

绘制饼图



绘制饼图

```
import matplotlib.pyplot as plt
def drawPie(ax):
      lbs = ('A', 'B', 'C', 'D')          #四个扇区的标签
      sectors = [16, 29.55, 44.45, 10]#四个扇区的份额(百分比)
      expl = [0, 0.1, 0, 0]               #四个扇区的突出程度
      ax.pie(x=sectors, labels=lbs, explode=expl,
            autopct='%.2f', shadow=True, labeldistance=1.1,
            pctdistance = 0.6, startangle = 90)
ax.set_title("pie sample")     #饼图标题
ax = plt.figure().add_subplot()
drawPie(ax)
plt.show()
```


绘制热力图

绘制热力图

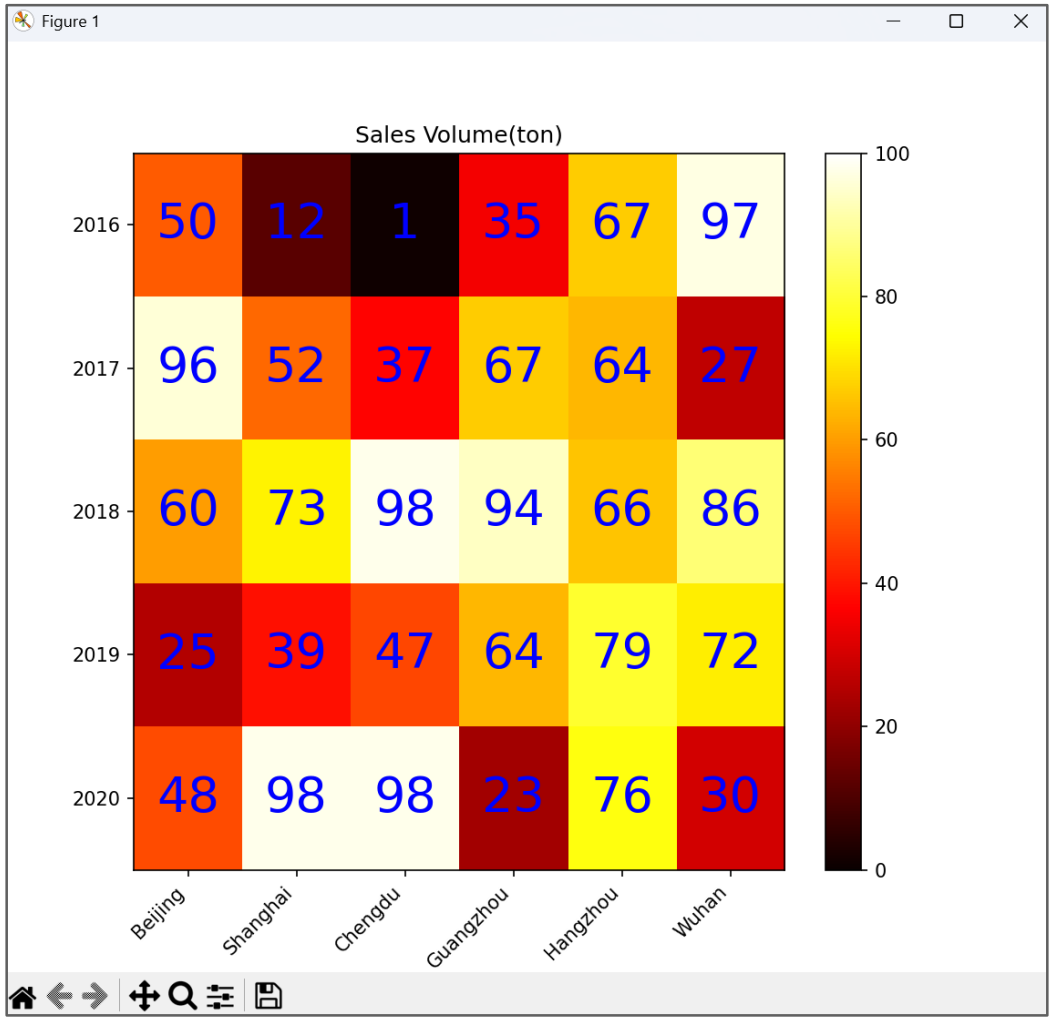


Figure 1
Sales Volume(ton)
2016
2017
2018
2019
2020
50 12 1 35 67 97
96 52 37 67 64 27
60 73 98 94 66 86
25 39 47 64 79 72
48 98 98 23 76 30
Beijing
Shanghai
Chengdu
Guangzhou
Hangzhou
Wuhan
100
80
60
40
20
0

绘制热力图

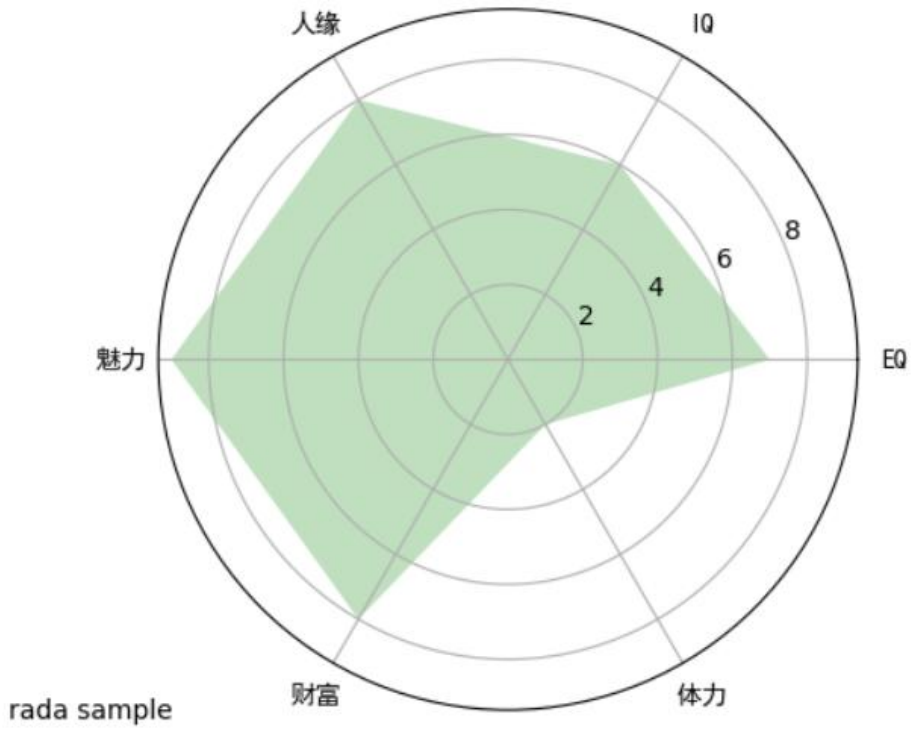
```
import random
from matplotlib import pyplot as plt
data = [[random.randint(0,100) for j in range(6)] for i
in range(5)]
#生成一个5行六列,元素[0,100]内的随机矩阵
xlabels = ['Beijing', 'Shanghai', 'Chengdu', 'Guangzhou',
'Hangzhou', 'Wuhan']
ylabels = ['2016', '2017', '2018', '2019', '2020']
ax = plt.figure(figsize=(10,8)).add_subplot()
ax.set_yticks(range(len(ylabels)))
#y轴在坐标[0,len(ylabels))处加刻度
ax.set_yticklabels(ylabels)    #设置y轴刻度文字
ax.set_xticks(range(len(xlabels)))
ax.set_xticklabels(xlabels)
```


绘制热力图

```
heatMp = ax.imshow(data, cmap = plt.cm.hot, aspect =
                   'auto', vmin = 0,vmax = 100)
for i in range(len(xlabels)):
      for j in range(len(ylabels)):
          ax.text(i, j, data[j][i], ha = "center",
                  va = "center", color = "blue", size=26)
plt.colorbar(heatMp)     #绘制右边的颜色-数值对照柱
plt.xticks(rotation=45, ha="right")
#将x轴刻度文字进行旋转，且水平方向右对齐
plt.title("Sales Volume(ton)")
plt.show()
```


绘制雷达图

绘制雷达图



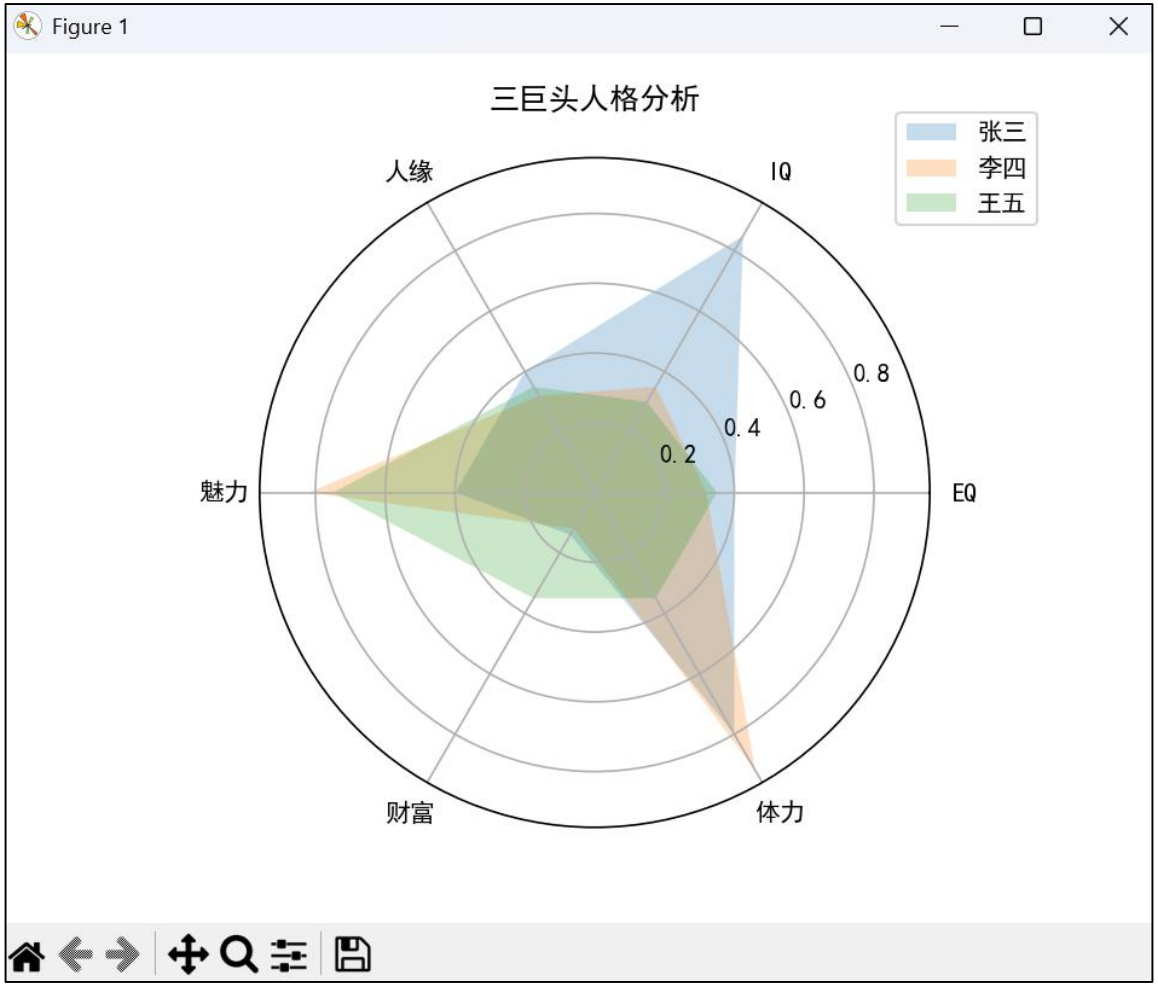
绘制雷达图

```
import matplotlib.pyplot as plt
from matplotlib import rcParams    #处理汉字用
def drawRadar(ax):
        pi = 3.1415926
        labels = ['EQ','IQ','人缘','魅力','财富','体力'] #6个属性的名称
        attrNum = len(labels)   #attrNum是属性种类数,此处等于6
        data = [7,6,8,9,8,2]    #六个属性的值
        angles = [2*pi*i/attrNum  for i in range(attrNum)]
        #angles是以弧度为单位的6个属性对应的6条半径线的角度
        angles2 = [x * 180/pi for x in angles]
        #angles2是以角度为单位的6个属性对应的半径线的角度
        ax.set_ylim(0, 10)      #限定半径线上的坐标范围
        ax.set_thetagrids(angles2, labels, fontproperties="SimHei")
        #绘制6个属性对应的6条半径
        ax.fill(angles, data, facecolor= 'g', alpha=0.25)
       #填充,alpha:透明度
```


绘制雷达图

```
rcParams['font.family'] = rcParams['font.sans-serif'] = 'SimHei'
#处理汉字
ax = plt.figure().add_subplot(projection = "polar")
#生成极坐标形式子图
drawRadar(ax)
plt.show()
```


绘制多层雷达图



绘制多层雷达图

```
import matplotlib.pyplot as plt
from matplotlib import rcParams
rcParams['font.family'] = rcParams['font.sans-serif'] =
'SimHei'
pi = 3.1415926
labels = ['EQ','IQ','人缘','魅力','财富','体力'] #6个属性的名称
attrNum = len(labels)
names = ('张三','李四','王五')
data = [[0.40,0.32,0.35], [0.85,0.35,0.30],
        [0.40,0.32,0.35], [0.40,0.82,0.75],
        [0.14,0.12,0.35], [0.80,0.92,0.35]]  #三个人的数据
angles = [2*pi*i/attrNum  for i in range(attrNum)]
angles2 = [x * 180/pi for x in angles]
ax = plt.figure().add_subplot(projection = "polar")
ax.fill(angles, data, alpha= 0.25) # 填充三个人的数据区域
```


绘制多层雷达图

```
ax.set_thetagrids(angles2, labels) # 设置半径标签
ax.set_title('三巨头人格分析', y = 1.05)
#y指明标题垂直位置
ax.legend(names, loc=(0.95, 0.9))
#画出右上角图例(不同人的颜色说明)
plt.show()
```