



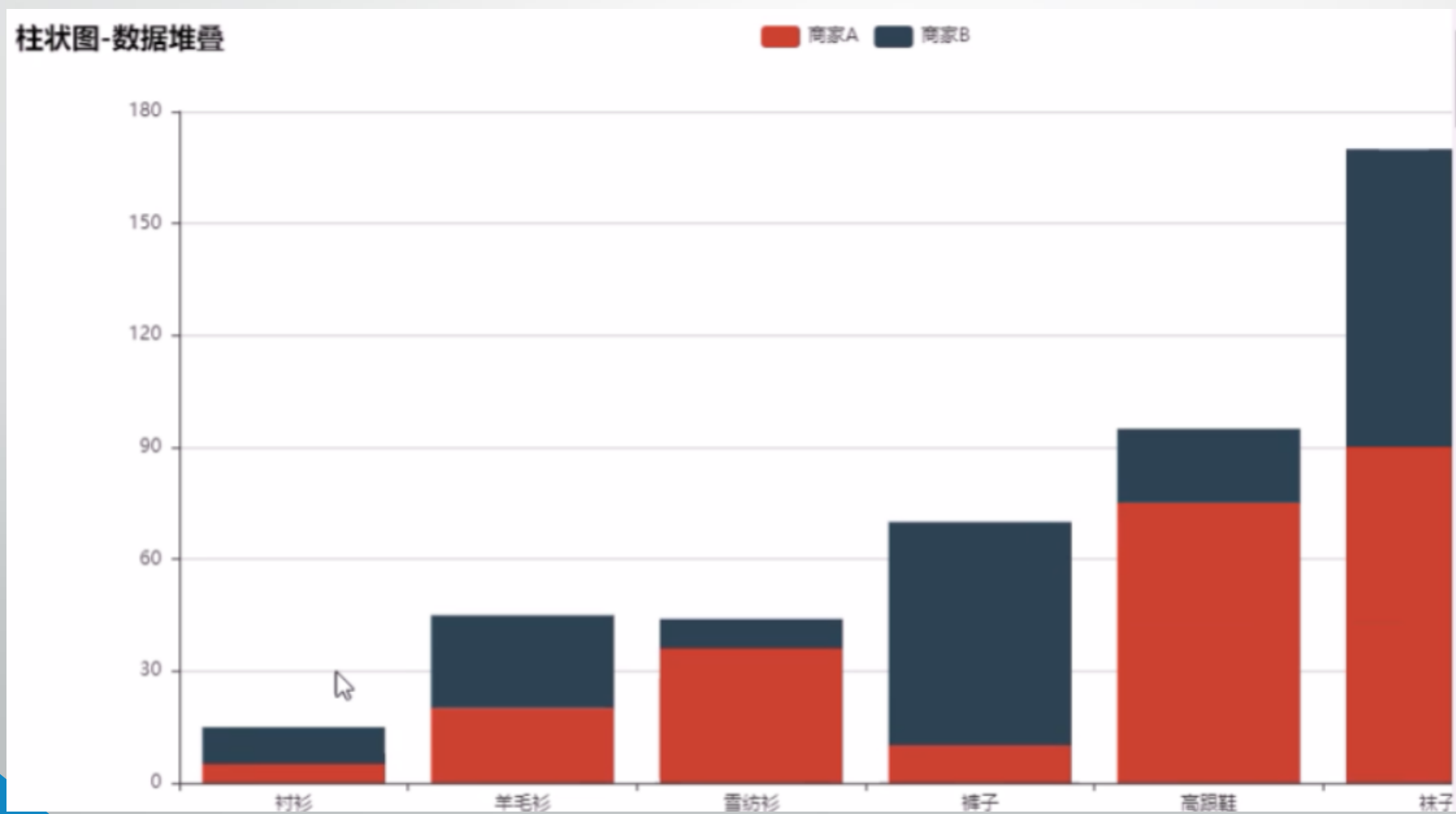
Python数据分析



matplotlib模块

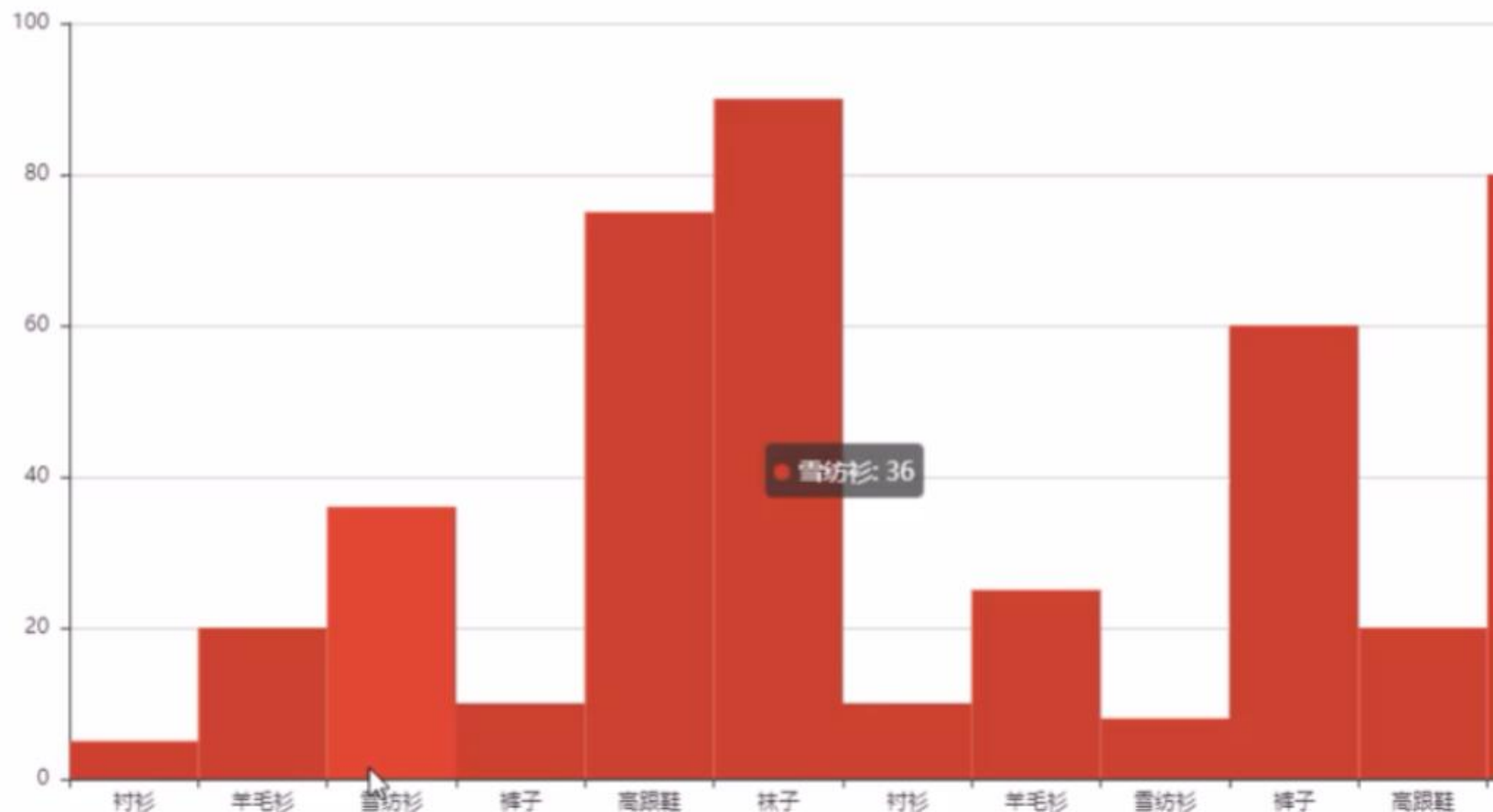


绘图

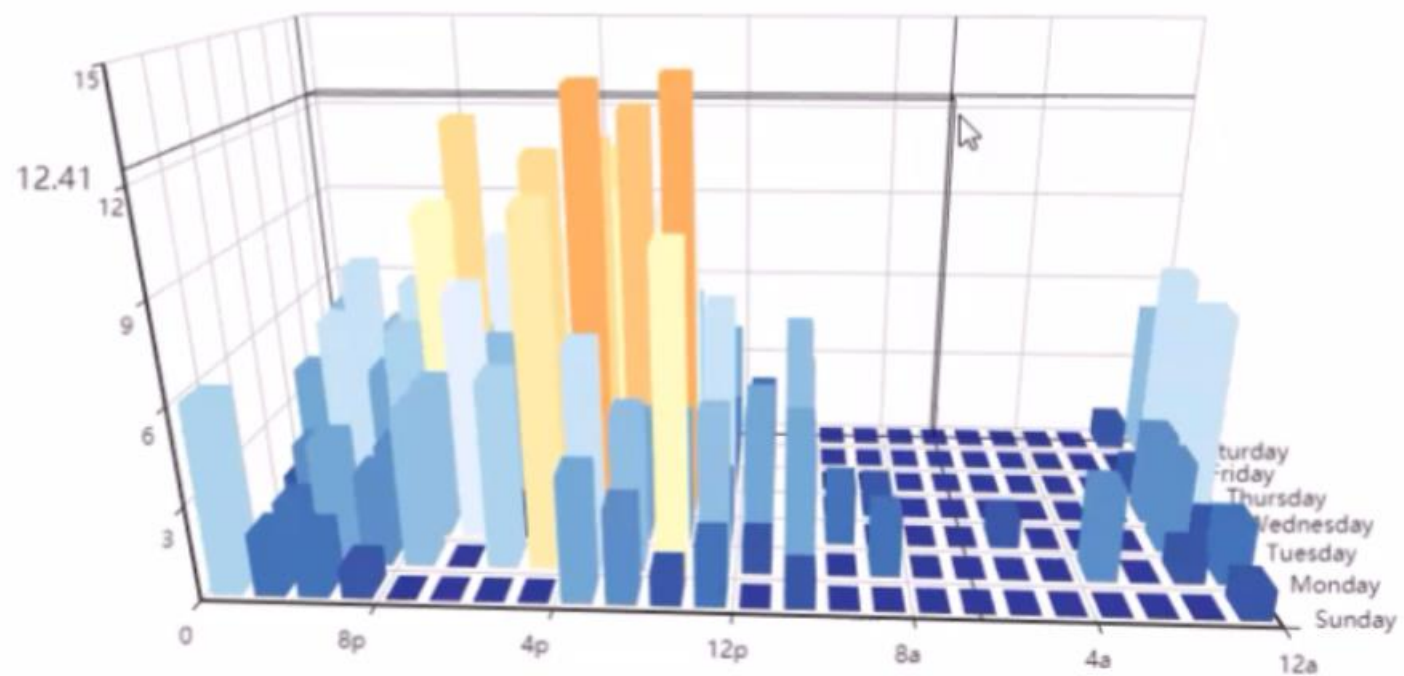


绘图

柱状图-直方图



3D 柱状图-默认





CONTENTS

1

3.1 matplotlib概述及功能介绍

2

3.2 基本绘图

3

3.3 图形对象

3.1 matplotlib概述

概述：

matplotlib概述

概述：

matplotlib是python的一个绘图库。使用它可以很方便的绘制出版质量级别的图形。

概述：

功能介绍：

matplotlib基本功能

1. 基本绘图（在二维平面坐标系中绘制连续的线）
 1. 设置线型、线宽和颜色
 2. 设置坐标轴范围
 3. 设置坐标刻度
 4. 设置坐标轴
 5. 图例
 6. 特殊点
 7. 备注

I

概述:

功能介绍:

2. 图形对象(图形窗口)

1. 子图 I
2. 刻度定位器
3. 刻度网格线
4. 半对数坐标
5. 散点图
6. 填充
7. 条形图
8. 饼图
9. 等高线图
10. 热成像图
11. 极坐标系
12. 三维曲面
13. 简单动画

3.2 基本绘图

3.2 基本绘图

绘图核心API:

案例：绘制一条余弦曲线

```
1 import numpy as np
2 import matplotlib.pyplot as mp
3
4 # xarray: <序列> 水平坐标序列
5 # yarray: <序列> 垂直坐标序列
6 mp.plot(xarray, yarray)
7 #显示图表
8 mp.show()
```

3.2 基本绘图

绘图核心API:

绘制水平线与垂直线：

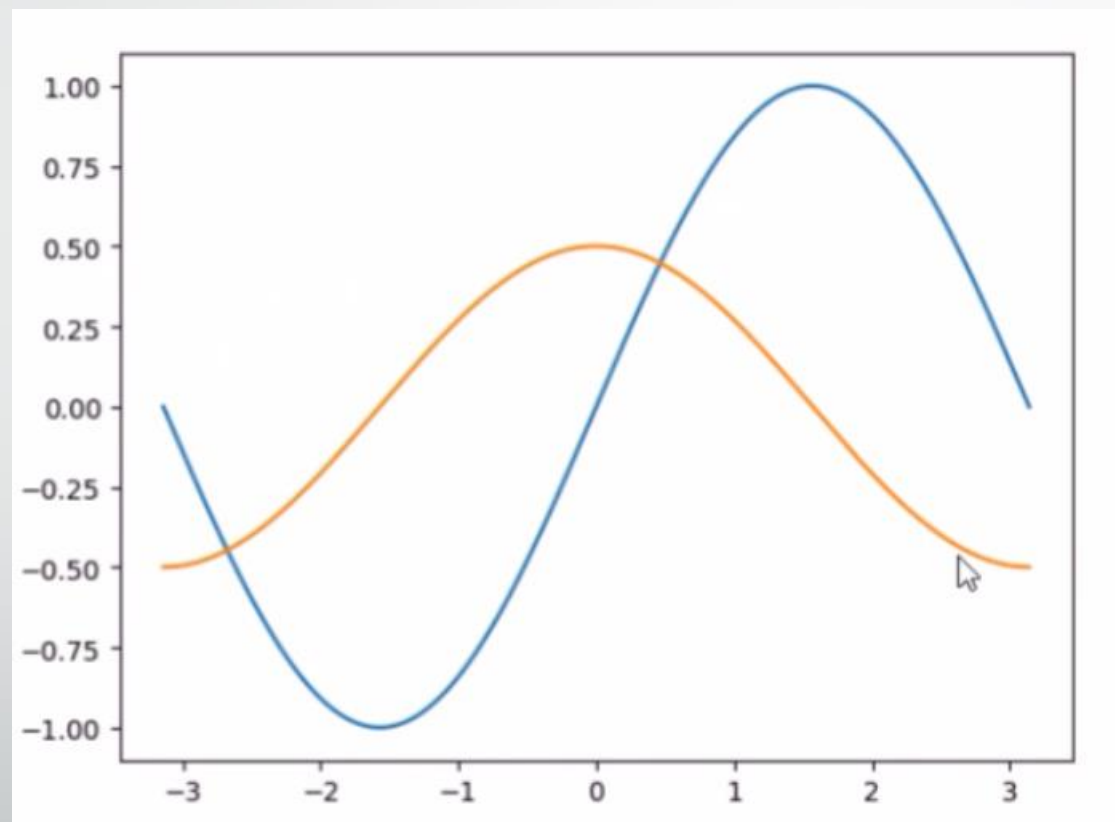
```
1 import numpy as np
2 import matplotlib.pyplot as mp
3
4 # vertical 绘制垂直线
5 mp.vlines(vval, ymin, ymax, ...)
6 # horizontal 绘制水平线
7 mp.hlines(xval, xmin, xmax, ...)
8 #显示图表
9 mp.show()
```

3.2 基本绘图

```
"""  
绘制正弦曲线  
"""  
# 创建-pi 到 pi 1000个点  
x = np.linspace(-np.pi,np.pi,1000)  
  
# 调用sin功能： 矢量化的sin方法会返回每一个x对应的y点  
y = np.sin(x)  
# y = np.cos(x) 余弦  
mp.plot(x,y)  
mp.show()
```

3.2 基本绘图

设置颜色:



3.2 基本绘图

设置颜色:

```
1 #linestyle: 线型    "-"    "--"    ":"    "-."
2 #linewidth: 线宽
3     #    数字
4 #color: <关键字参数> 颜色
5     #    英文颜色单词 或 常见颜色英文单词首字母 或 #495434 或
        (1,1,1) 或 (1,1,1,1)
6 #alpha: <关键字参数> 透明度
7     #    浮点数值
8 mp.plot(xarray, yarray, linestyle='', linewidth=1,
        color='', alpha=0.5)
```


3.2 基本绘图

设置坐标轴范围

案例：把坐标轴范围设置为 $-\pi \sim \pi$

范围：

```
1 #x_limt_min:    <float> x轴范围最小值
2 #x_limit_max:   <float> x轴范围最大值
3 mp.xlim(x_limt_min, x_limit_max)
4 #y_limt_min:    <float> y轴范围最小值
5 #y_limit_max:   <float> y轴范围最大值
6 mp.ylim(y_limt_min, y_limit_max)
```

3.2 基本绘图

设置坐标刻度

案例：把横坐标的刻度显示为： $0, \pi/2, \pi, 3\pi/2, 2\pi$

刻度：

```
1 #x_val_list:      x轴刻度值序列
2 #x_text_list:     x轴刻度标签文本序列 [可选]
3 mp.xticks(x_val_list , x_text_list )
4 #y_val_list:      y轴刻度值序列
5 #y_text_list:     y轴刻度标签文本序列 [可选]
6 mp.yticks(y_val_list , y_text_list )
```

3.2 基本绘图

刻度文本的特殊语法 -- LaTeX排版语法字符串

刻度:

```
1 | r'$x^n+y^n=z^n$',    r'$\int\frac{1}{x} dx = \ln |x| + C$',  
   | r'$-\frac{\pi}{2}$'
```

$$x^2 + y^2 = z^2, \int \frac{1}{x} dx = \ln |x| + C, -\frac{\pi}{2} \quad (1)$$

3.2 基本绘图

刻度:

表 3.1: 数学模式重音符

\hat{a}	<code>\hat{a}</code>	\check{a}	<code>\check{a}</code>	\tilde{a}	<code>\tilde{a}</code>	\acute{a}	<code>\acute{a}</code>
\grave{a}	<code>\grave{a}</code>	\dot{a}	<code>\dot{a}</code>	\ddot{a}	<code>\ddot{a}</code>	\breve{a}	<code>\breve{a}</code>
\bar{a}	<code>\bar{a}</code>	\vec{a}	<code>\vec{a}</code>	\widehat{A}	<code>\widehat{A}</code>	\widetilde{A}	<code>\widetilde{A}</code>

3.2 基本绘图

刻度:

表 3.2: 小写希腊字母

α	<code>\alpha</code>	θ	<code>\theta</code>	o	<code>o</code>	v	<code>\upsilon</code>
β	<code>\beta</code>	ϑ	<code>\vartheta</code>	π	<code>\pi</code>	ϕ	<code>\phi</code>
γ	<code>\gamma</code>	ι	<code>\iota</code>	ϖ	<code>\varpi</code>	φ	<code>\varphi</code>
δ	<code>\delta</code>	κ	<code>\kappa</code>	ρ	<code>\rho</code>	χ	<code>\chi</code>
ϵ	<code>\epsilon</code>	λ	<code>\lambda</code>	ϱ	<code>\varrho</code>	ψ	<code>\psi</code>
ε	<code>\varepsilon</code>	μ	<code>\mu</code>	σ	<code>\sigma</code>	ω	<code>\omega</code>
ζ	<code>\zeta</code>	ν	<code>\nu</code>	ς	<code>\varsigma</code>		
η	<code>\eta</code>	ξ	<code>\xi</code>	τ	<code>\tau</code>		

3.2 基本绘图

刻度:

表 3.3: 大写希腊字母

Γ	<code>\Gamma</code>	Λ	<code>\Lambda</code>	Σ	<code>\Sigma</code>	Ψ	<code>\Psi</code>
Δ	<code>\Delta</code>	Ξ	<code>\Xi</code>	Υ	<code>\Upsilon</code>	Ω	<code>\Omega</code>
Θ	<code>\Theta</code>	Π	<code>\Pi</code>	Φ	<code>\Phi</code>		

3.2 基本绘图

设置坐标轴

坐标轴名：left / right / bottom / top

```
1 # 获取当前坐标轴字典, {'left':左轴, 'right':右轴, 'bottom':下  
  轴, 'top':上轴 }  
2 ax = mp.gca()  
3 # 获取其中某个坐标轴  
4 axis = ax.spines['坐标轴名']  
5 # 设置坐标轴的位置。 该方法需要传入2个元素的元组作为参数  
6 # type: <str> 移动坐标轴的参照类型 一般为'data' (以数据的值  
  作为移动参照值)  
7 # val: 参照值  
8 axis.set_position((type, val))  
9 # 设置坐标轴的颜色  
10 # color: <str> 颜色值字符串  
11 axis.set_color(color)
```


3.2 基本绘图

设置图例

图例

显示两条曲线的图例，并测试loc属性。

```
1 # 再绘制曲线时定义曲线的label
2 # label: <关键字参数 str> 支持LaTeX排版语法字符串I
3 mp.plot(xarray, yarray ... label='', ...)
4 # 设置图例的位置
5 # loc: <关键字参数> 制定图例的显示位置 (若不设置loc, 则显示默认位置)
```


3.2 基本绘图

设置图例

```
6 # =====
7 # Location String Location Code
8 # =====
9 # 'best' 0
0 # 'upper right' 1
1 # 'upper left' 2
2 # 'lower left' 3
3 # 'lower right' 4
4 # 'right' 5
5 # 'center left' 6
6 # 'center right' 7
7 # 'lower center' 8
8 # 'upper center' 9
9 # 'center' 10
0 # =====
```

```
mp.legend(loc='')
```

3.2 基本绘图

设置特殊点


























特殊点

案例：绘制当 $x=3\pi/4$ 时两条曲线上的特殊点。

```
1 # xarray: <序列> 所有需要标注点的水平坐标组成的序列
2 # yarray: <序列> 所有需要标注点的垂直坐标组成的序列
3 mp.scatter(xarray, yarray,
4             marker='',          #点型 ~ matplotlib.markers
5             s='',              #大小
6             edgecolor='',      #边缘色
7             facecolor='',      #填充色
8             zorder=3           #绘制图层编号 (编号越大, 图层越
9             靠上)
10 )
```

3.2 基本绘图

设置特殊点

^ : triangle_up 	3 : tri_left 	P : plus (filled) 	x : x 	_ : hline 
v : triangle_down 	2 : tri_up 	p : pentagon 	+ : plus 	: vline 
o : circle 	1 : tri_down 	s : square 	H : hexagon2 	d : thin_diamond 
, : pixel 	> : triangle_right 	8 : octagon 	h : hexagon1 	D : diamond 
. : point 	< : triangle_left 	4 : tri_right 	* : star 	X : x (filled) 

3.2 基本绘图

备注

案例：为在某条曲线上的点添加备注，指明函数方程与值。

```
1 # 在图表中为某个点添加备注。包含备注文本，备注箭头等图像的设置。
2 mp.annotate(
3     r'$\frac{\pi}{2}$',          #备注中显示的文本内容
4     xycoords='data',          #备注目标点所使用的坐标系
5     xy=(x, y),                #备注目标点的坐标
6     textcoords='offset points', #备注文本所使用的坐标系
7     xytext=(x, y),            #备注文本的坐标
8     fontsize=14,              #备注文本的字体大小
9     arrowprops=dict()         #使用字典定义文本指向目标
10 )
```

3.2 基本绘图

备注

arrowprops参数使用字典定义指向目标点的箭头样式

```
1 #arrowprops字典参数的常用key
2 arrowprops=dict(
3     arrowstyle='',          #定义箭头样式
4     connectionstyle=''      #定义连接线的样式
5 )
```

3.2 基本绘图

箭
头
样
式

```

=====
2  Name          Attrs
3  =====
=====
4  '-'           None
5  '->'          head_length=0.4,head_width=0.2
6  '-[|'         widthB=1.0,lengthB=0.2,angleB=None
7  '|-|'         widthA=1.0,widthB=1.0
8  '-|>'         head_length=0.4,head_width=0.2
9  '<-'          head_length=0.4,head_width=0.2
10 '<->'         head_length=0.4,head_width=0.2
11 '<|-|'        head_length=0.4,head_width=0.2
12 '<|-|>'       head_length=0.4,head_width=0.2
13 'fancy'
    head_length=0.4,head_width=0.4,tail_width=0.4
14 'simple'
  
```


3.2 基本绘图

连接
线
样
式

```

=====
2  Name          Attrs
3  =====
4  'angle'       angleA=90,angleB=0,rad=0.0
5  'angle3'      angleA=90,angleB=0`
6  'arc'
   angleA=0,angleB=0,armA=None,armB=None,rad=0.0
7  'arc3'        rad=0.0
8  'bar'
   armA=0.0,armB=0.0,fraction=0.3,angle=None
9  =====
=====

```

3.3 图形对象

3.3 图形对象

图形窗口

图形对象（图形窗口）

案例：绘制两个窗口，一起显示。

I

```
1 # 手动构建 matplotlib 窗口
2 mp.figure(
3     '',                                #窗口标题栏文本
4     figsize=(4, 3),                  #窗口大小 <元组>
5     dpi=120,                          #像素密度
6     facecolor=''                      #图表背景色
7 )
8 mp.show()
```

3.3 图形对象

图形窗口

```
1 # 手动构建 matplotlib 窗口
2 mp.figure(
3     'A',                      #窗口标题栏文本
4     facecolor=''             #图表背景色
5 )
6 mp.figure('B')
7 mp.figure('A') # 把A创建置为当前窗口
8 mp.plot(...) # 将会作用在A窗口中
9 mp.show()
```

python

mp.figure方法不仅可以构建一个新窗口，如果已经构建过title='xxx'的窗口，又使用figure方法构建了title='xxx'的窗口的话，mp将不会创建新的窗口，而是把title='xxx'的窗口置为当前操作窗口。

3.3 图形对象

图形窗口

设置当前窗口的参数

案例：测试窗口相关参数

```
1 # 设置图表标题 显示在图表上方
2 mp.title(title, fontsize=12)
3 # 设置水平轴的文本
4 mp.xlabel(x_label_str, fontsize=12)
5 # 设置垂直轴的文本
6 mp.ylabel(y_label_str, fontsize=12)
7 # 设置刻度参数 labelsz设置刻度字体大小
8 mp.tick_params(..., labelsz=8, ...)
```

3.3 图形对象

图形窗口

```
9  # 设置图表网格线  linestyle设置网格线的样式
10     #   -   or solid 粗线
11     #   --  or dashed 虚线
12     #   -.  or dashdot 点虚线
13     #   :   or dotted 点线
14  mp.grid(linestyle='')
15  # 设置紧凑布局，把图表相关参数都显示在窗口中
16  mp.tight_layout()
```

3.3 图形对象

子图：
矩阵式布局

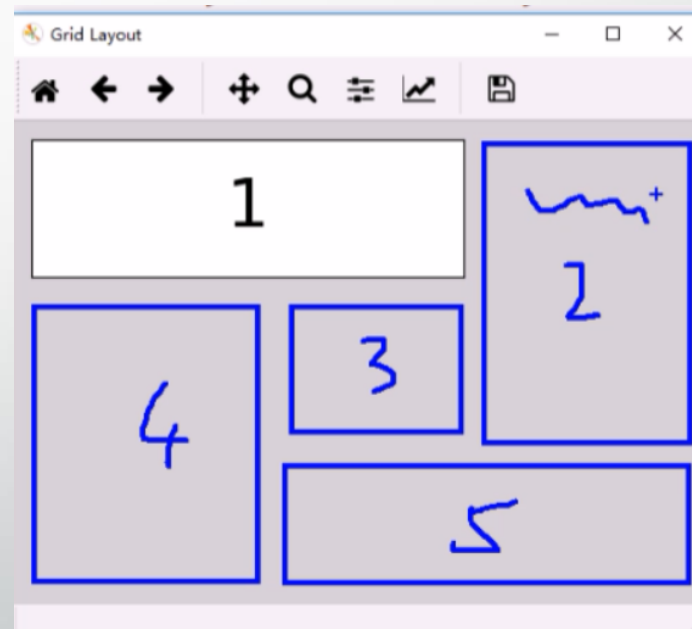
```
1 mp.figure('Subplot Layout', facecolor='lightgray')
2 # 拆分矩阵+
3     # rows: 行数
4     # cols: 列数
5     # num: 编号
6 mp.subplot(rows, cols, num)
7     #   1 2 3
8     #   4 5 6
9     #   7 8 9
10 mp.subplot(3, 3, 5)      #操作3*3的矩阵中编号为5的子图
11 mp.subplot(335)          #简写
```

3.3 图形对象

子图:

网格式布局 (支持单元格合并)

```
1 import matplotlib.gridspec as mg
2 mp.figure('Grid Layout', facecolor='lightgray')
3 # 调用GridSpec方法拆分网格式布局
4 # rows: 行数
5 # cols: 列数
6 # gs = mg.GridSpec(rows, cols) 拆分成3行3列
7 gs = mg.GridSpec(3, 3)
8 # 合并0行与0、1列为一个子图表
9 mp.subplot(gs[0, :2])
10 mp.text(0.5, 0.5, '1', ha='center', va='center',
11        size=36)
11 mp.show()
```



3.3 图形对象

子图:

自由式布局

```
1  mp.figure('Flow Layout', facecolor='lightgray')
2  # 设置图标的位置, 给出左下角点坐标与宽高即可
3  # left_bottom_x: 坐下角点x坐标
4  # left_bottom_x: 坐下角点y坐标
5  # width:          宽度
6  # height:         高度
7  # mp.axes([left_bottom_x, left_bottom_y, width,
8             height])
9  mp.axes([0.03, 0.03, 0.94, 0.94])
10 mp.text(0.5, 0.5, '1', ha='center', va='center',
11         size=36)
10 mp.show()
```

3.3 图形对象

刻度定位器

刻度定位器相关API：

```
1 # 获取当前坐标轴
2 ax = mp.gca()
3 # 设置水平坐标轴的主刻度定位器
4 ax.xaxis.set_major_locator(mp.NullLocator())
5 # 设置水平坐标轴的次刻度定位器为多点定位器，间隔0.1
6 ax.xaxis.set_minor_locator(mp.MultipleLocator(0.1))
```


3.3 图形对象

刻度定位器

案例：绘制一个数轴。

```
1 mp.figure('Locators', facecolor='lightgray')
2 # 获取当前坐标轴
3 ax = mp.gca()
4 # 隐藏除底轴以外的所有坐标轴
5 ax.spines['left'].set_color('none')
6 ax.spines['top'].set_color('none')
7 ax.spines['right'].set_color('none')
8 # 将底坐标轴调整到子图中心位置
9 ax.spines['bottom'].set_position(('data', 0))
10 # 设置水平坐标轴的主刻度定位器
11 ax.xaxis.set_major_locator(mp.NullLocator())
```

3.3 图形对象

常用
刻度定位器

```
1 # 空定位器：不绘制刻度
2 mp.NullLocator()
3 # 最大值定位器：
4 # 最多绘制nbins+1个刻度
5 mp.MaxNLocator(nbins=3)
6 # 定点定位器：根据locs参数中的位置绘制刻度
7 mp.FixedLocator(locs=[0, 2.5, 5, 7.5, 10])
8 # 自动定位器：由系统自动选择刻度的绘制位置
9 mp.AutoLocator()
10 # 索引定位器：由offset确定起始刻度，由base确定相邻刻度的间隔
11 mp.IndexLocator(offset=0.5, base=1.5)
12 # 多点定位器：从0开始，按照参数指定的间隔(缺省1)绘制刻度
13 mp.MultipleLocator()
14 # 线性定位器：等分numticks-1份，绘制numticks个刻度
15 mp.LinearLocator(numticks=21)

16 # 对数定位器：以base为底，绘制刻度
17 mp.LogLocator(base=2)
```

3.3 图形对象

刻度网格线

刻度网格线

绘制刻度网格线的相关API：

I

```
1 ax = mp.gca()
2 #绘制刻度网格线
3 ax.grid(
4     which='',          # 'major'/'minor' <-> '主刻度'/'次刻
                        度'
5     axis='',          # 'x'/'y'/'both' <-> 绘制x或y轴
6     linewidth=1,      # 线宽
7     linestyle='',     # 线型
8     color='',         # 颜色
9     alpha=0.5         # 透明度
```

3.3 图形对象

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

y = np.array([1,10,100,1000,100,10,1])
mp.figure('Grid Line', facecolor='lightgray')
# 设置刻度器
ax = mp.gca()
# 设置 X 轴主、次刻度
ax.xaxis.set_major_locator(mp.MultipleLocator(1))
ax.xaxis.set_minor_locator(mp.MultipleLocator(0.1))
# 设置 Y 轴主、次刻度
ax.yaxis.set_major_locator(mp.MultipleLocator(250))
ax.yaxis.set_minor_locator(mp.MultipleLocator(50))
# 设置网格线
ax.grid(
    which='major', axis='both', linewidth=0.7, color='red', linestyle='-'
)
ax.grid(
    which='minor', axis='both', linewidth=0.2, color='red', linestyle='-'
)
# 画线
mp.plot(y,'o-',color='blue')
mp.show()
```

3.3 图形对象

散点图

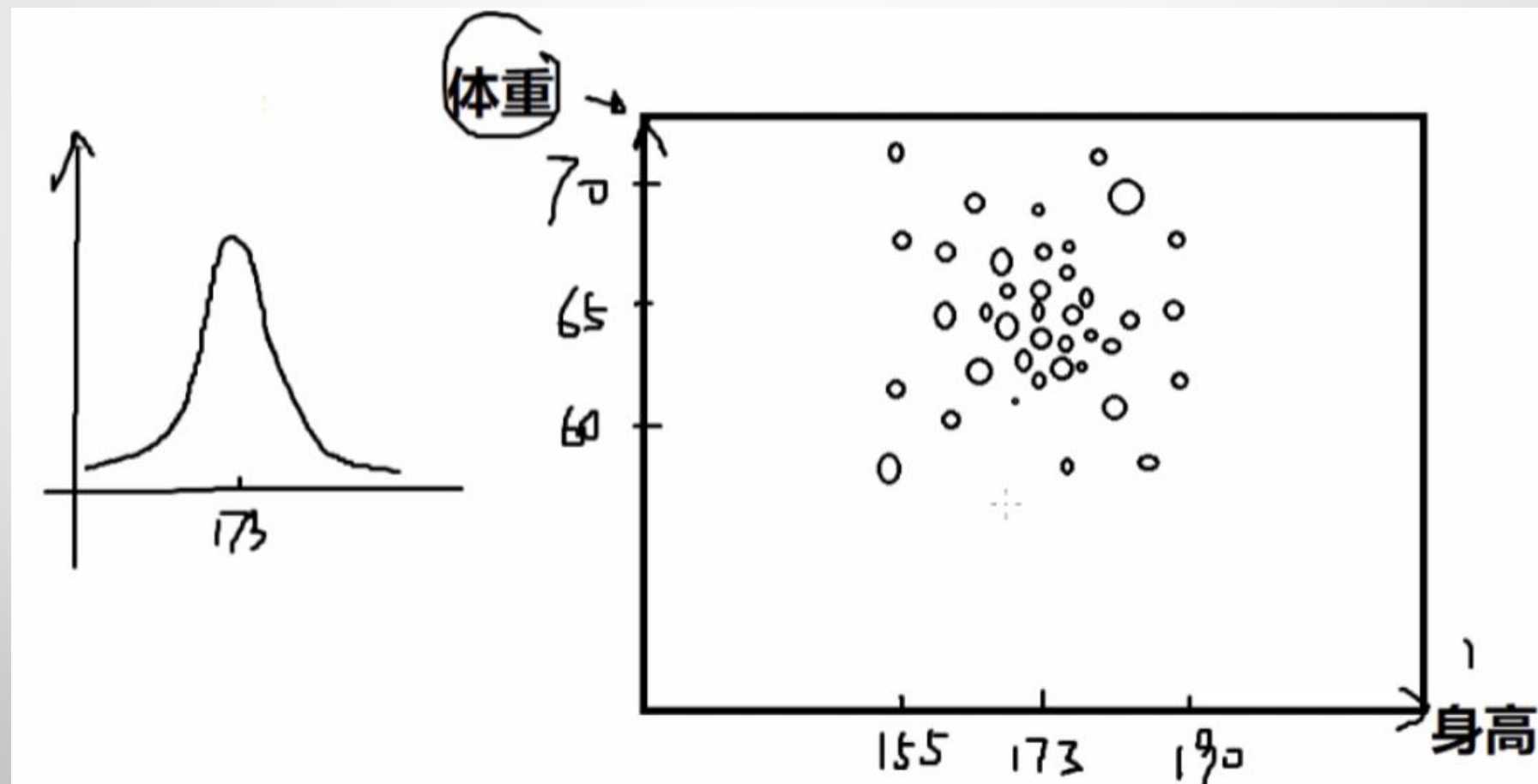
+

可以通过每个点的坐标、颜色、大小和形状表示不同的特征值。

身高	体重	性别	年龄段	种族
180	80	男	中年	亚洲
160	50	女	青少	美洲

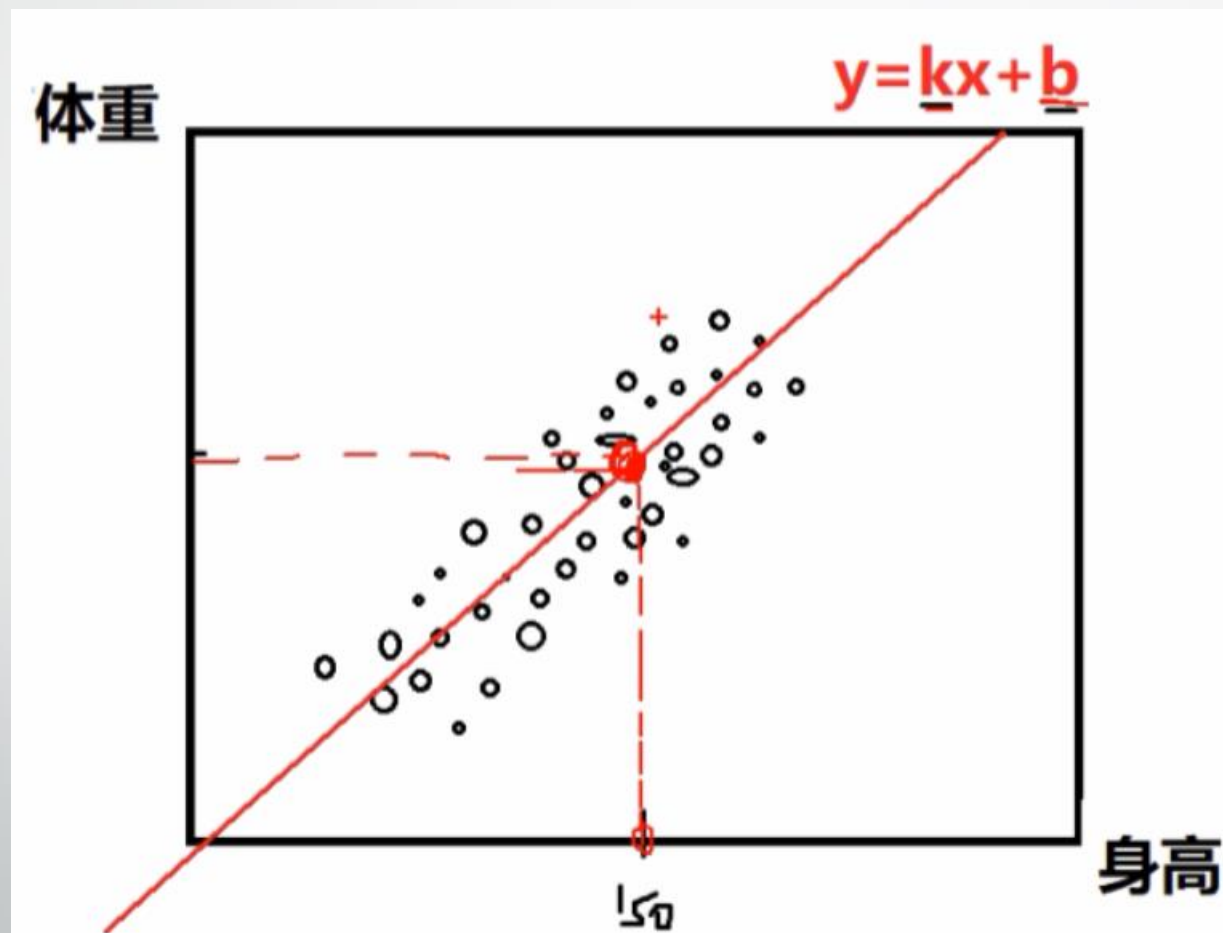
3.3 图形对象

散点图



3.3 图形对象

散点图



3.3 图形对象

散点图

绘制散点图的相关API：

```
1 mp.scatter(  
2     x,                # x轴坐标数组  
3     y,                # y轴坐标数组  
4     marker='',        # 点型  
5     s=10,             # 大小  
6     color='',         # 颜色  
7     edgecolor='',     # 边缘颜色  
8     facecolor='',     # 填充色  
9     zorder=''         # 图层序号  
10 )
```


3.3 图形对象

散点图

numpy.random提供了normal函数用于产生符合 正态分布 的随机数

```
1 n = 100
2 # 172: 期望值
3 # 10: 标准差
4 # n: 数字生成数量
5 x = np.random.normal(172, 20, n)
6 y = np.random.normal(60, 10, n)
```

3.3 图形对象

散点图

案例：绘制平面散点图。

python

```
1 mp.figure('scatter', facecolor='lightgray')
2 mp.title('scatter')
3 mp.scatter(x, y)
4 mp.show()
```

设置点的颜色

```
1 mp.scatter(x, y, c='red')           #直接设置颜色
2 d = (x-172)**2 + (y-60)**2
3 mp.scatter(x, y, c=d, cmap='jet')   #以c作为参数，取cmap
                                     颜色映射表中的颜色值
```

3.3 图形对象

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

# cos、 sin
x = np.linspace(0,8*np.pi,1000)
sin_x = np.sin(x)
con_x = np.cos(x/2)/2

# 窗口
mp.figure('Fill', facecolor='lightgray')
mp.plot(x,sin_x,color='red',label='sin(x)')
mp.plot(x,con_x,color='green',label='cos(x)')

'''
设置填充
'''
mp.fill_between(x,sin_x,con_x, sin_x<con_x,color='dodgerblue',)
mp.fill_between(x,sin_x,con_x,sin_x>con_x,color='orangered',)
# 显示图例
mp.legend()
mp.show()
```

3.3 图形对象

形状

条形图（柱状图）

绘制柱状图的相关API：

```
1 mp.figure('Bar', facecolor='lightgray')
2 mp.bar(
3     x,                # 水平坐标数组
4     y,                # 柱状图高度数组
5     width,            # 柱子的宽度
6     color='',         # 填充颜色
7     label='',         #
8     alpha=0.2         #
9 )
```

3.3 图形对象

形状

height : scalar or sequence of scalars
the height(s) of the bars

width : scalar or array-like, optional
the width(s) of the bars
default: 0.8

bottom : scalar or array-like, optional
the y coordinate(s) of the bars
default: None

align : {'center', 'edge'}, optional, default: 'center'
If 'center', interpret the *x* argument as the coordinates
of the centers of the bars. If 'edge', **aligns** bars by
their left edges

To align the bars on the right edge pass a negative
width and ``align='edge'``

3.3 图形对象

形状

案例：先以柱状图绘制苹果12个月的销量，然后再绘制橘子的销量。

```
1 apples = np.array([30, 25, 22, 36, 21, 29, 20, 24, 33,
2 19, 27, 15])
3 oranges = np.array([24, 33, 19, 27, 35, 20, 15, 27,
4 20, 32, 20, 22])
5 mp.figure('Bar' , facecolor='lightgray')
6 mp.title('Bar', font size=20)
7 mp.xlabel('Month', fontsize=14)
8 mp.ylabel('Price', fontsize=14)
9 mp.tick_params(labelsize=10)
10 mp.grid(axis='y', linestyle=':')
```

3.3 图形对象

设置柱状图

```
x = np.arange(1,13)
```

```
mp.bar(x-0.2, apples, width=0.4,color='dodgerblue',label='Apple',align='center')
```

```
mp.bar(x+0.2, oranges, width=0.4,color='orangered',label='Orange',align='center')
```

设置x的刻度文本

```
mp.xticks(x,['Jan','Feb','Mar','Apr','Mar','Jun','Jul','Aug','Sep','Oct','Nov','Dec'])
```

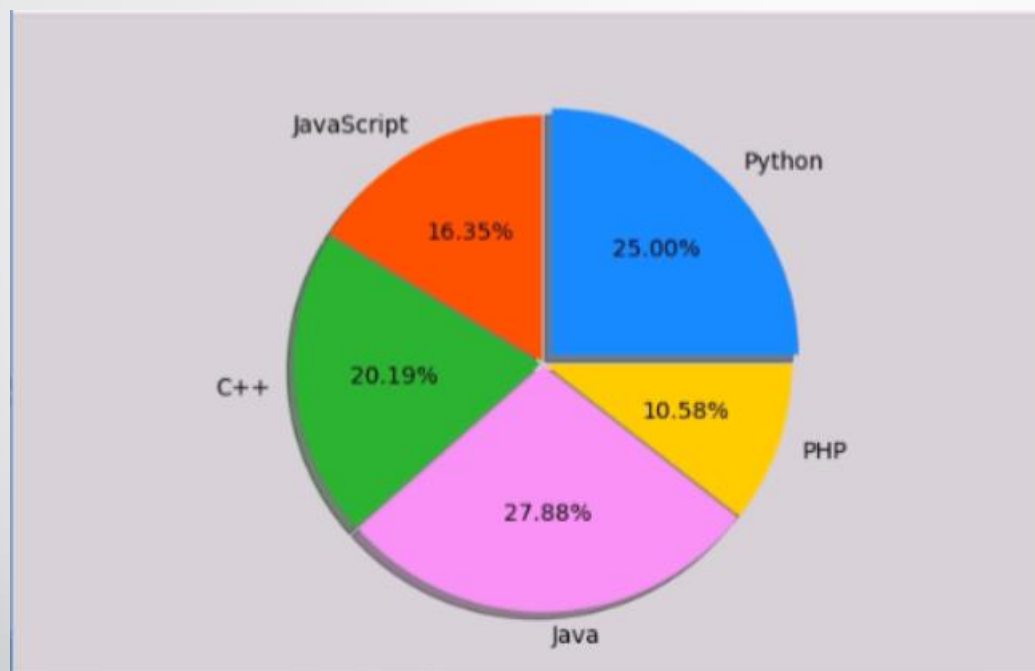
设置图例

```
mp.legend(loc='upper right')
```

```
mp.show()
```

3.3 图形对象

饼状图



3.3 图形对象

饼状图

绘制饼状图的基本API：

```
1 mp.pie(  
2     values,          # 值列表  
3     spaces,          # 扇形之间的间距列表  
4     labels,          # 标签列表  
5     colors,          # 颜色列表  
6     '%d%%',          # 标签所占比例格式  
7     shadow=True,     # 是否显示阴影  
8     startangle=90    # 逆时针绘制饼状图时的起始角度  
9     radius=1         # 半径  
10 )
```

3.3 图形对象

饼状图

案例：绘制饼状图显示5门语言的流程度：

```
1 mp.figure('pie', facecolor='lightgray')
2 #整理数据
3 values = [26, 17, 21, 29, 11]
4 spaces = [0.05, 0.01, 0.01, 0.01, 0.01]
5 labels = ['Python', 'JavaScript',
6           'C++', 'Java', 'PHP']
7 colors = ['dodgerblue', 'orangered',
8           'limegreen', 'violet', 'gold']
9 mp.figure('Pie', facecolor='lightgray')
10 mp.title('Pie', fontsize=20)
11 # 等轴比例
12 mp.axis('equal')
```

用python挖掘客户数据

3.3 图形对象

子图：
矩阵式布局

案例：绘制9宫格矩阵式子图，每个子图中写一个数字。

```
1  mp.figure('Subplot Layout', facecolor='lightgray')
2
3  for i in range(9):
4      mp.subplot(3, 3, i+1)
5      mp.text(
6          0.5, 0.5, i+1,
7          ha='center',
8          va='center',
9          size=36,
10         alpha=0.5,
11         withdash=False
12     )
13     mp.xticks([])
14     mp.yticks([])
15
```

RFM模型

RFM模型

- RMF模型的内容
- 根据美国数据库营销研究所Arthur Hughes的研究, [客户数据库](#)中有三个神奇的要素, 这三个要素构成了数据分析最好的[指标](#):
- 最近一次消费(Recency)
- 消费频率(Frequency)
- 消费金额(Monetary)

RFM模型

- RFM模型的应用意义

- 在众多的[客户关系管理](#)(CRM)的分析模式中，RFM模型是被广泛提到的。RFM模型是衡量客户价值和客户创利能力的重要工具和手段。该模型通过一个客户的近期购买行为、购买的总体频率以及花了多少钱三项指标来描述该客户的价值状况。

聚类分析后产生的 8个客户类别

客户类别	客户数量	近度/d	频度	值度/元	比较结果	客户级别
1	168	44.68	6.66	881.82	R ↓ F ↑ M ↑	重要保持客户
2	115	34.10	4.48	692.01	R ↓ F ↓ M ↓	重要发展客户
3	81	58.60	9.17	770.34	R ↓ F ↑ M ↑	重要保持客户
4	134	47.75	7.19	454.25	R ↓ F ↑ M ↓	一般重要客户
5	156	81.63	6.74	643.77	R ↑ F ↑ M ↓	一般客户
6	127	69.20	4.36	448.57	R ↑ F ↓ M ↓	无价值客户
7	90	77.98	6.27	1004.17	R ↑ F ↑ M ↑	重要挽留客户
8	155	67.85	4.05	801.98	R ↑ F ↓ M ↑	一般客户
总均值		60.07	5.98	704.75		

Apriori 算法

Apriori 算法

- 这里主要介绍的是叫做Apriori的‘一个先验’算法，通过该算法我们可以对数据集做关联分析——在大规模的数据中寻找有趣关系的任务，本文主要介绍使用Apriori算法发现数据的（频繁项集、关联规则）。
- 这些关系可以有两种形式：频繁项集、关联规则。
- 频繁项集：经常出现在一块的物品的集合
- 关联规则：暗示两种物品之间可能存在很强的关系

交易号码	商品
0	豆奶, 莴苣
1	莴苣, 尿布, 葡萄酒, 甜菜
2	豆奶, 尿布, 葡萄酒, 橙汁
3	莴苣, 豆奶, 尿布, 葡萄酒
4	莴苣, 豆奶, 尿布, 橙汁

图11-1 一个来自Hole Foods天然食品店的简单交易清单

Apriori 算法

- 频繁项集是指那些经常出现在一起的物品，例如上图的{葡萄酒、尿布、豆奶}，从上面的数据集中也可以找到尿布->葡萄酒的关联规则，这意味着有人买了尿布，那很有可能他也会购买葡萄酒。那如何定义和表示频繁项集和关联规则呢？这里引入支持度和可信度（置信度）。
- **支持度**：一个项集的支持度被定义为数据集中包含该项集的记录所占的比例，上图中，豆奶的支持度为4/5，（豆奶、尿布）为3/5。支持度是针对项集来说的，因此可以定义一个最小支持度，只保留最小支持度的项集。
- **可信度（置信度）**：针对如{尿布}->{葡萄酒}这样的关联规则来定义的。计算为 支持度{尿布，葡萄酒}/支持度{尿布}，其中{尿布，葡萄酒}的支持度为3/5，{尿布}的支持度为4/5，所以“尿布->葡萄酒”的可行度为 $3/4=0.75$ ，这意味着尿布的记录中，我们的规则有75%都适用。
- **Apriori 的原理**：如果某个项集是频繁项集，那么它所有的子集也是频繁的。即如果 {0,1} 是频繁的，那么 {0}, {1} 也一定是频繁的。