Matplotlib数据可视化基础

2019/12/9

Matplotlib数据可视化基础

Matplotlib简介

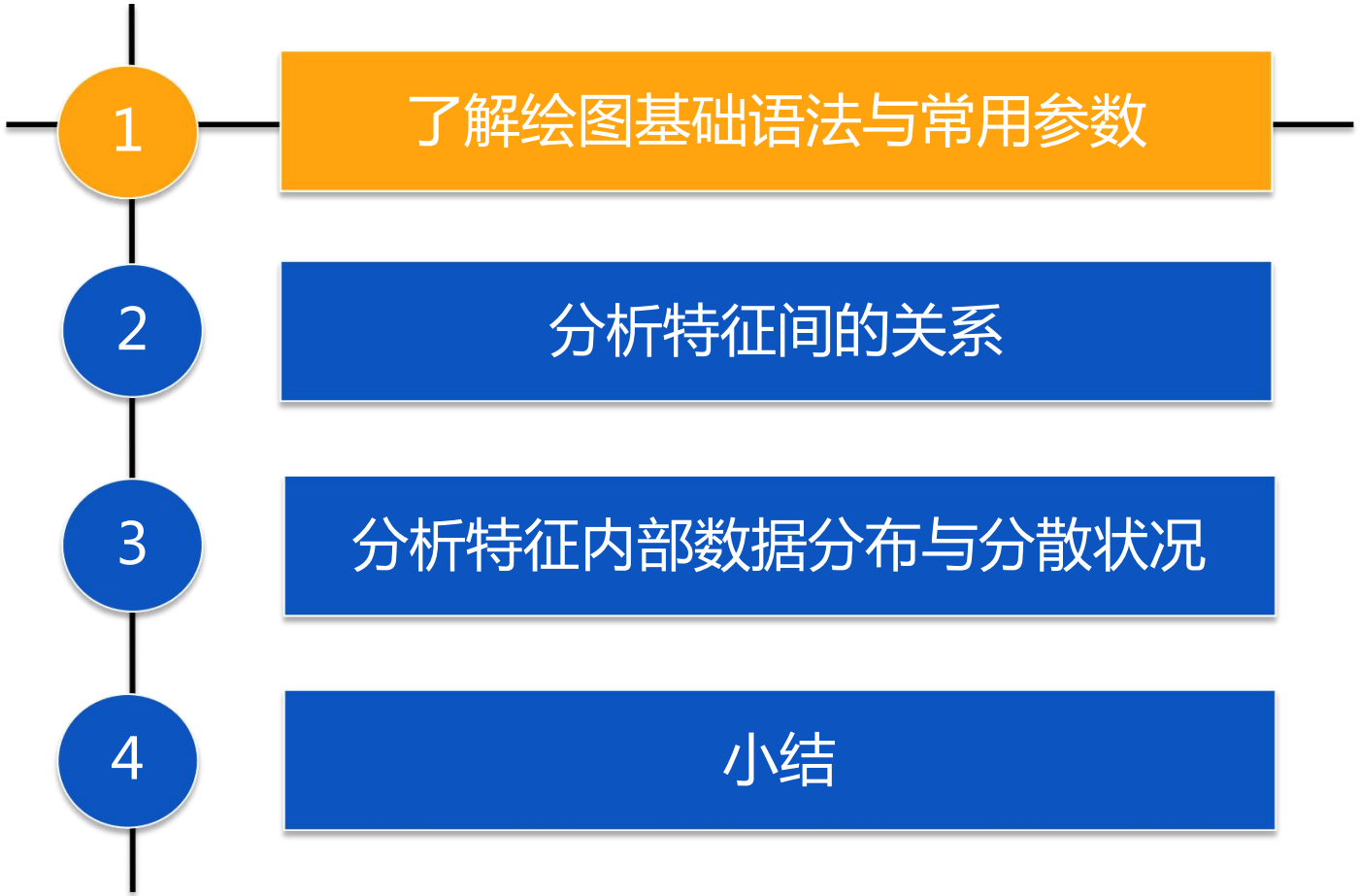
- Matplotlib首次发表于2007年，由于在函数设计上参考了MATLAB，所以其名字以"Mat”开头，中间的"plot”表示绘图这一作用， 而结尾的"lib” 则表示它是一个集合。
- 近年来，Matplotlib在开源社区的推动下，在科学计算领域得到了广泛的应用，成为了Python中应用非常广的绘图工具包之一。
- **Matplotlib 中应用最广的是matplotlib.pyplot 模块。**

Matplotlib数据可视化基础

pyplot简介

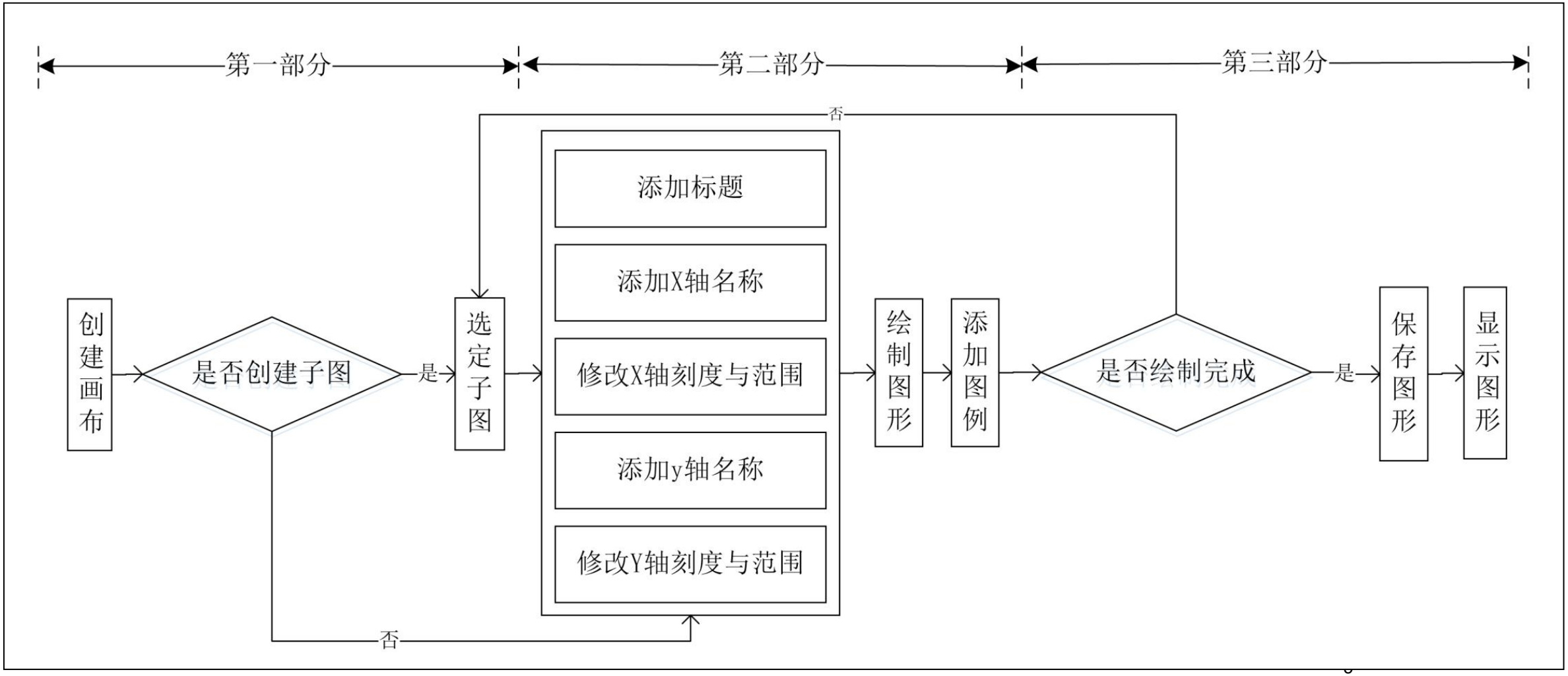
- matplotlib.pyplot (以下简称pyplot)是一个命令风格函数的集合，使Matplotib的机制更像MATLAB。
- 每个绘图函数都可对图形进行一些更改，如创建图形，在图形中创建绘图区域，在绘图区域绘制-些线条，使用标签装饰绘图等。
- 在pyplot中，各种状态跨函数调用保存，以便跟踪诸如当前图形和绘图区域之类的东西，并且绘图函数始终指向当前轴域。
- 本章将以pyplot为基础，讲述5种基础统计图形的绘制方法。
- 参考：https://matplotlib.org/api/pyplot_api.html

目录



掌握pyplot基础语法

基本绘图流程



掌握pyplot基础语法

1. 创建画布与创建子图

- 第一部分主要作用是构建出一张空白的画布，并可以选择是否将整个画布划分为多个部分，方便在同一幅图上绘制多个图形。
- 最简单的绘图可以省略第一部分，而后直接在默认的画布上进行图形绘制。

函数名称	函数作用
plt.figure	创建一个空白画布，可以指定画布大小，像素。
figure.add_subplot	创建并选中子图，可以指定子图的行数，列数，与选中图片编号。

掌握pyplot基础语法

2. 添加画布内容

- 第二部分是绘图的主体部分。其中添加标题、坐标轴名称、绘制图形等步骤是并列的，没有先后顺序，可以先绘制图形，也可以先添加各类标签。**但是添加图例一定要在绘制图形之后。**

函数名称	函数作用
plt.title	在当前图形中添加标题，可以指定标题的名称、位置、颜色、字体大小等参数。
plt.xlabel	在当前图形中添加x轴名称，可以指定位置、颜色、字体大小等参数。
plt.ylabel	在当前图形中添加y轴名称，可以指定位置、颜色、字体大小等参数。
plt.xlim	指定当前图形x轴的范围，只能确定一个数值区间，而无法使用字符串标识。
plt.ylim	指定当前图形y轴的范围，只能确定一个数值区间，而无法使用字符串标识。
plt.xticks	指定x轴刻度的数目与取值。
plt.yticks	指定y轴刻度的数目与取值。
plt.legend	指定当前图形的图例，可以指定图例的大小、位置、标签。

掌握pyplot基础语法

3. 保存与展示图形

- 第三部分主要用于保存和显示图形。

函数名称	函数作用
plt.savafig	保存绘制的图片，可以指定图片的分辨率、边缘的颜色等参数。
plt.show	在本机显示图形。

掌握pyplot基础语法

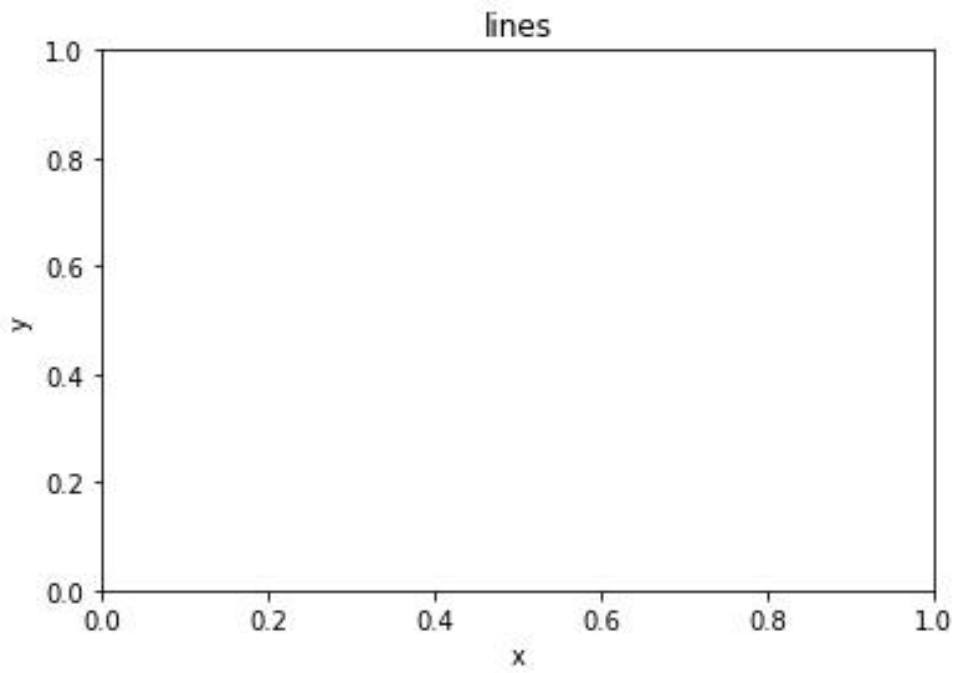
示例：pyplot基础绘图语法

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline      #表示在行中显示图片，在命令行运行报错

plt.title('lines')      # 添加标题
plt.xlabel('x')         # 添加x轴的名称
plt.ylabel('y')         # 添加y轴的名称
plt.xlim((0,1))         # 确定x轴范围
plt.ylim((0,1))         # 确定y轴范围
```

使用%matplotlib命令可以将matplotlib的图表直接嵌入到Notebook之中，或者使用指定的界面库显示图表，它有一个参数指定matplotlib图表的显示方式。inline表示将图表嵌入到Notebook中。

省略了第一步（创建空白画布），直接在默认的画布上进行图形绘制

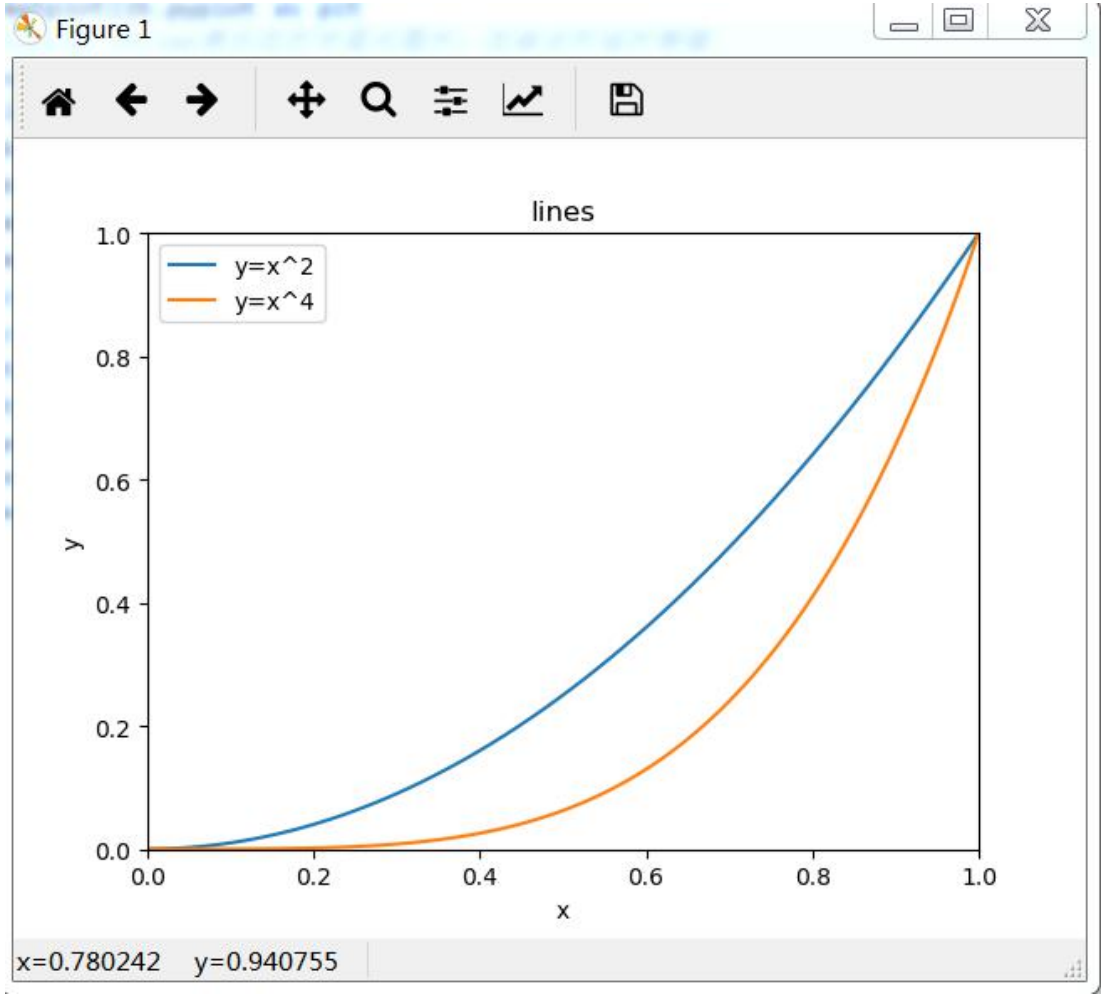


掌握pyplot基础语法

示例：pyplot基础绘图语法

```
plt.xticks([0,0.2,0.4,0.6,0.8,1])          # 规定x轴刻度
plt.yticks([0,0.2,0.4,0.6,0.8,1])          # 确定y轴刻度
data = np.arange(0,1.1,0.01)
plt.plot(data,data**2)                     # 添加y=x^2曲线
plt.plot(data,data**4)                     # 添加y=x^4曲线
plt.legend(['y=x^2','y=x^4'])              #添加图例

plt.savefig('../tmp/y=x^2.png')            #保存图片
plt.show()
```

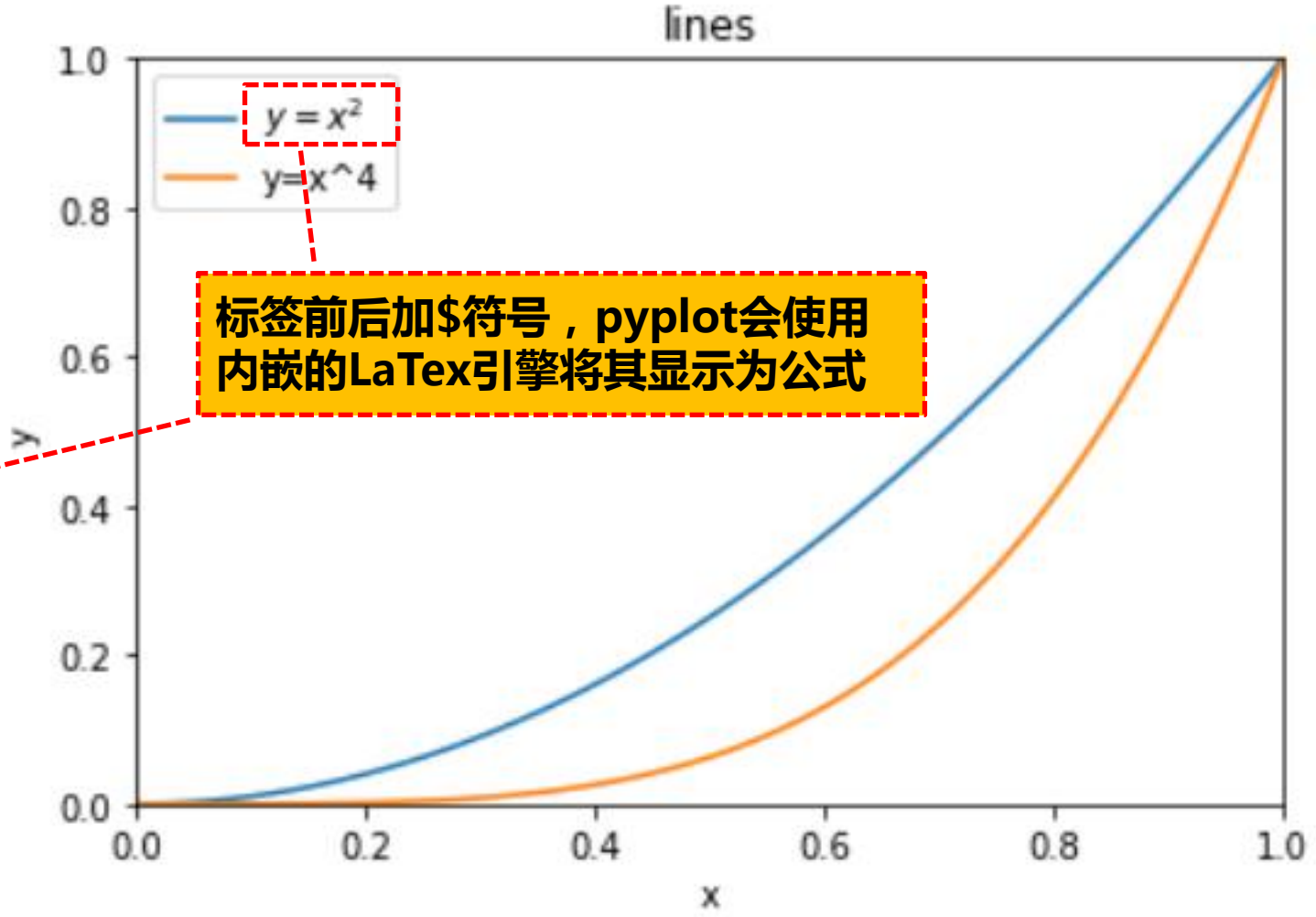


掌握pyplot基础语法

示例：pyplot基础绘图语法

```
plt.xticks([0,0.2,0.4,0.6,0.8,1])
plt.yticks([0,0.2,0.4,0.6,0.8,1])
data = np.arange(0,1.1,0.01)
plt.plot(data,data**2)          # 添加y
plt.plot(data,data**4)          # 添加y
plt.legend(['$y=x^2$','y=x^4'])


plt.savefig('../tmp/y=x^2.png')
plt.show()
```

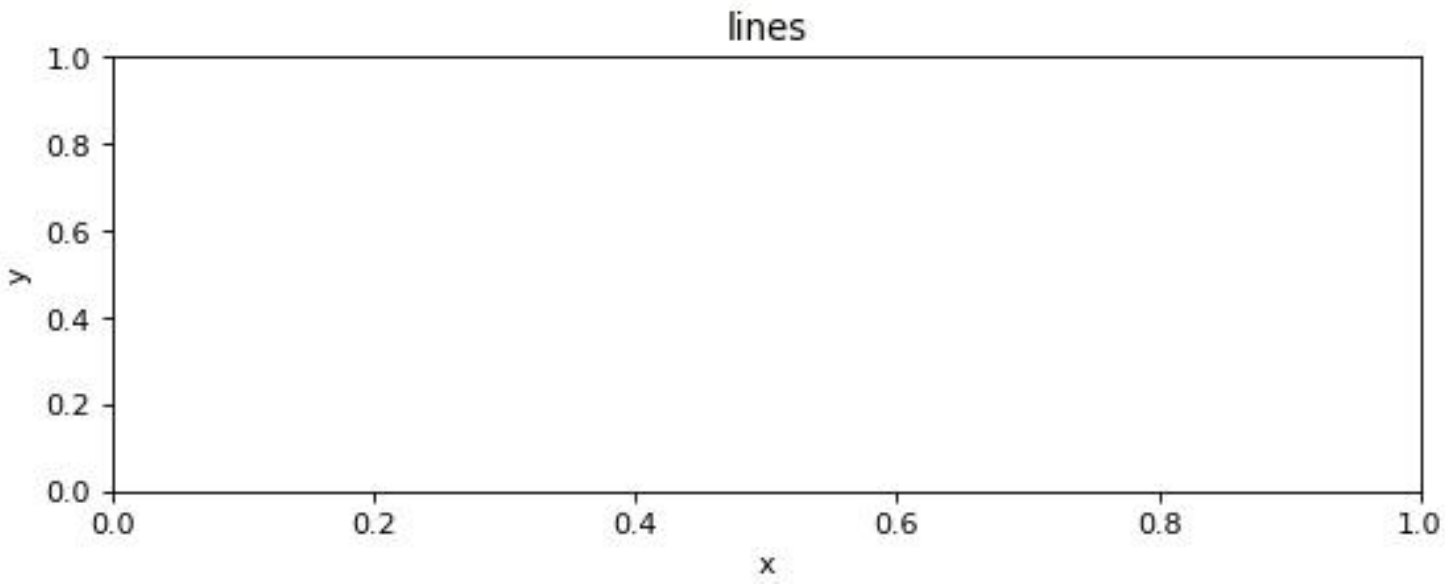


掌握pyplot基础语法

示例：包含子图绘制的基础绘图语法

```
p1 = plt.figure(figsize=(8,6),dpi=80)      # 确定画布大小
ax1 = p1.add_subplot(2,1,1)                # 创建一个两行1列的子图1，返回axes对象
ax2 = p1.add_subplot(2,1,2)                # 创建一个两行1列的子图2，返回axes对象
```

```
plt.sca(ax1)                # 第一幅子图
plt.title('lines')          # 添加标题
plt.xlabel('x')             # 添加x轴的名称
plt.ylabel('y')             # 添加y轴的名称
plt.xlim((0,1))             # 确定x轴范围
plt.ylim((0,1))             # 确定y轴范围
```

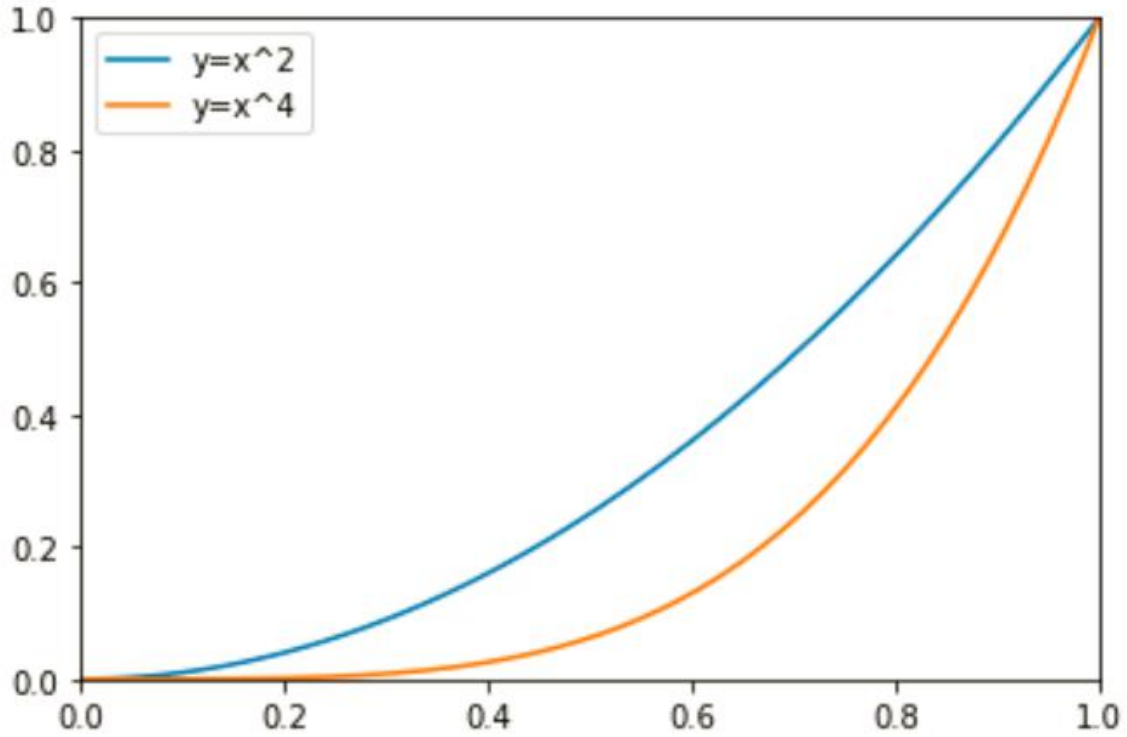


掌握pyplot基础语法

示例：包含子图绘制的基础绘图语法

```
plt.xticks([0,0.2,0.4,0.6,0.8,1])        # 规定x轴刻度
plt.yticks([0,0.2,0.4,0.6,0.8,1])        # 确定y轴刻度
```

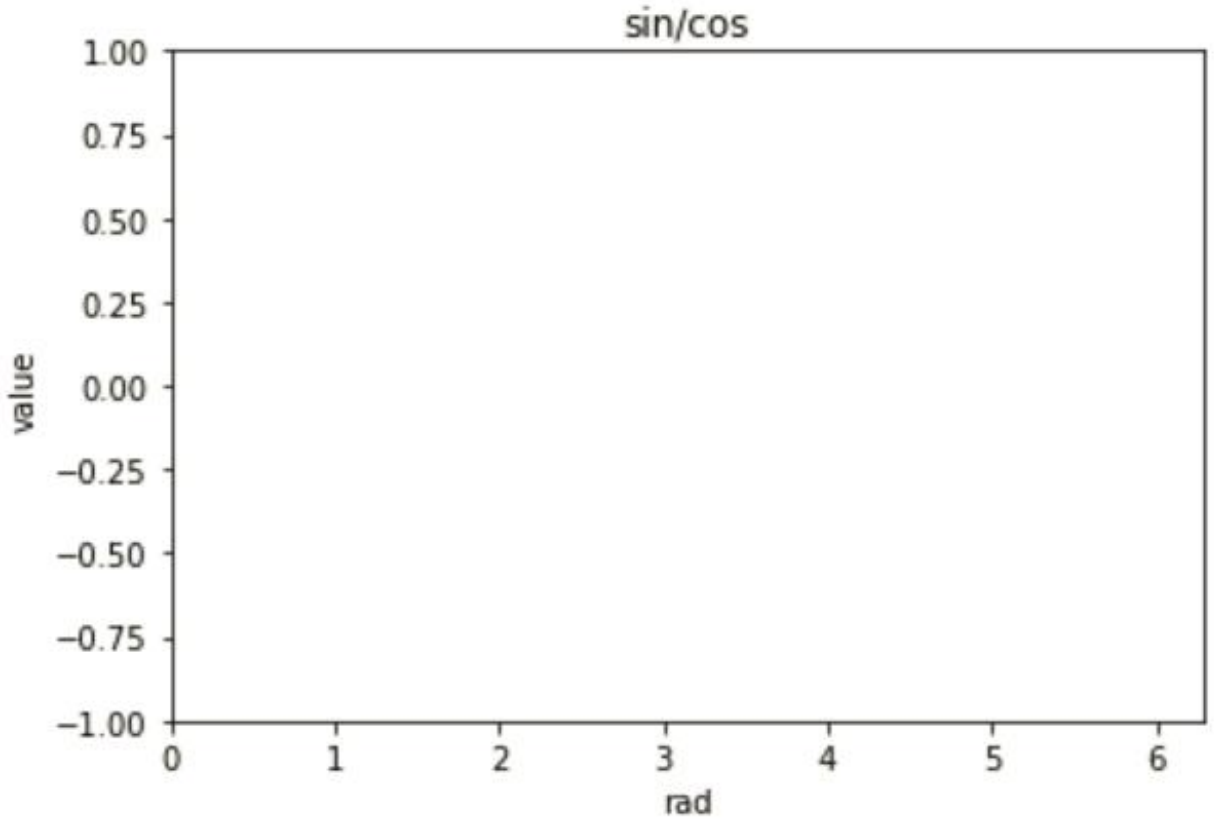
```
rad = np.arange(0,np.pi*2,0.01)
plt.plot(rad,rad**2)    # 添加y=x^2曲线
plt.plot(rad,rad**4)    # 添加y=x^4曲线
plt.legend(['y=x^2','y=x^4'])
```



掌握pyplot基础语法

示例：包含子图绘制的基础绘图语法

```
##第二幅子图
plt.sca(ax2)          # 开始绘制第2幅
plt.title('sin/cos')  # 添加标题
plt.xlabel('rad')     # 添加x轴的名称
plt.ylabel('value')   # 添加y轴的名称
plt.xlim((0,np.pi*2)) # 确定x轴范围
plt.ylim((-1,1))      # 确定y轴范围
```

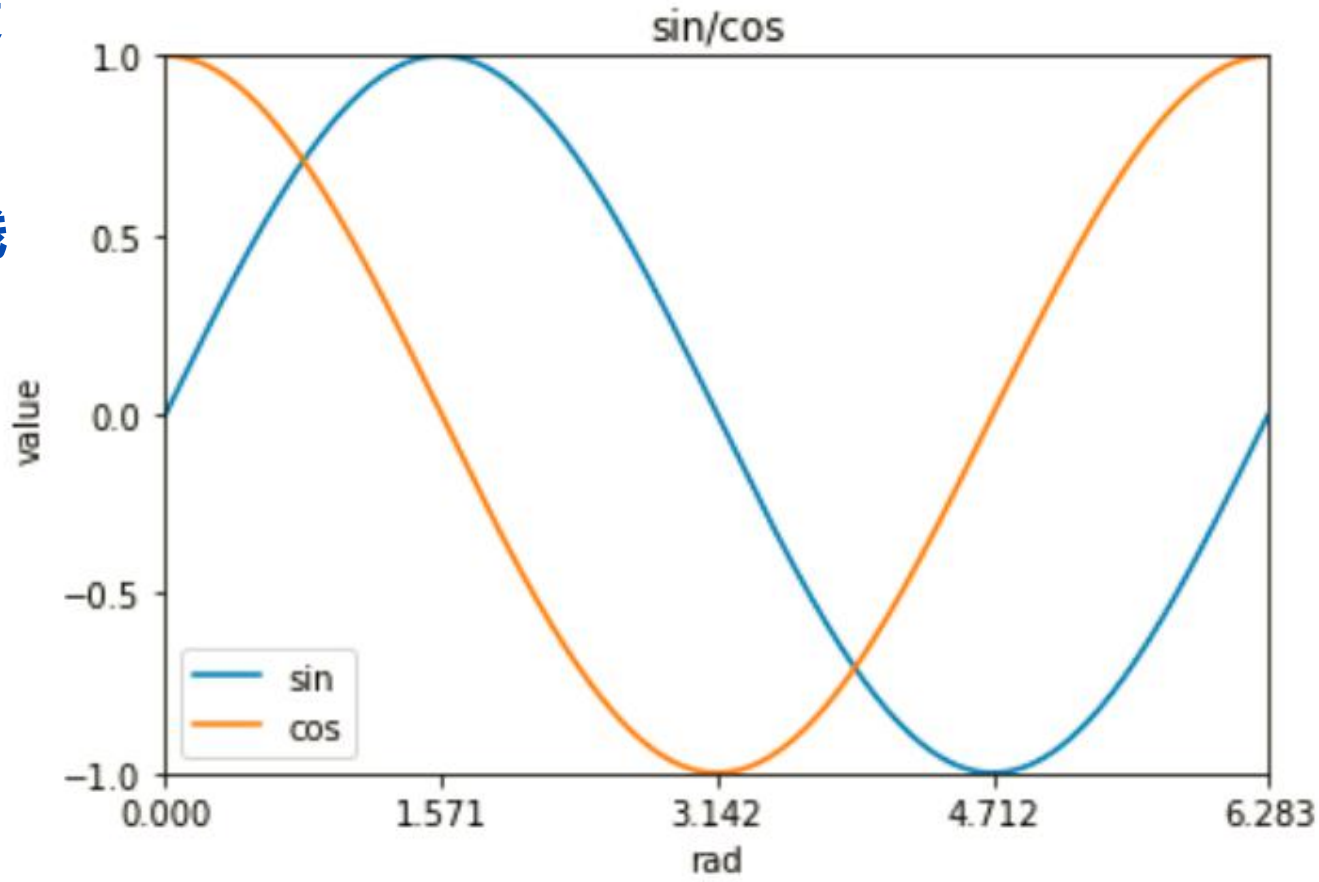


掌握pyplot基础语法

示例：包含子图绘制的基础绘图语法

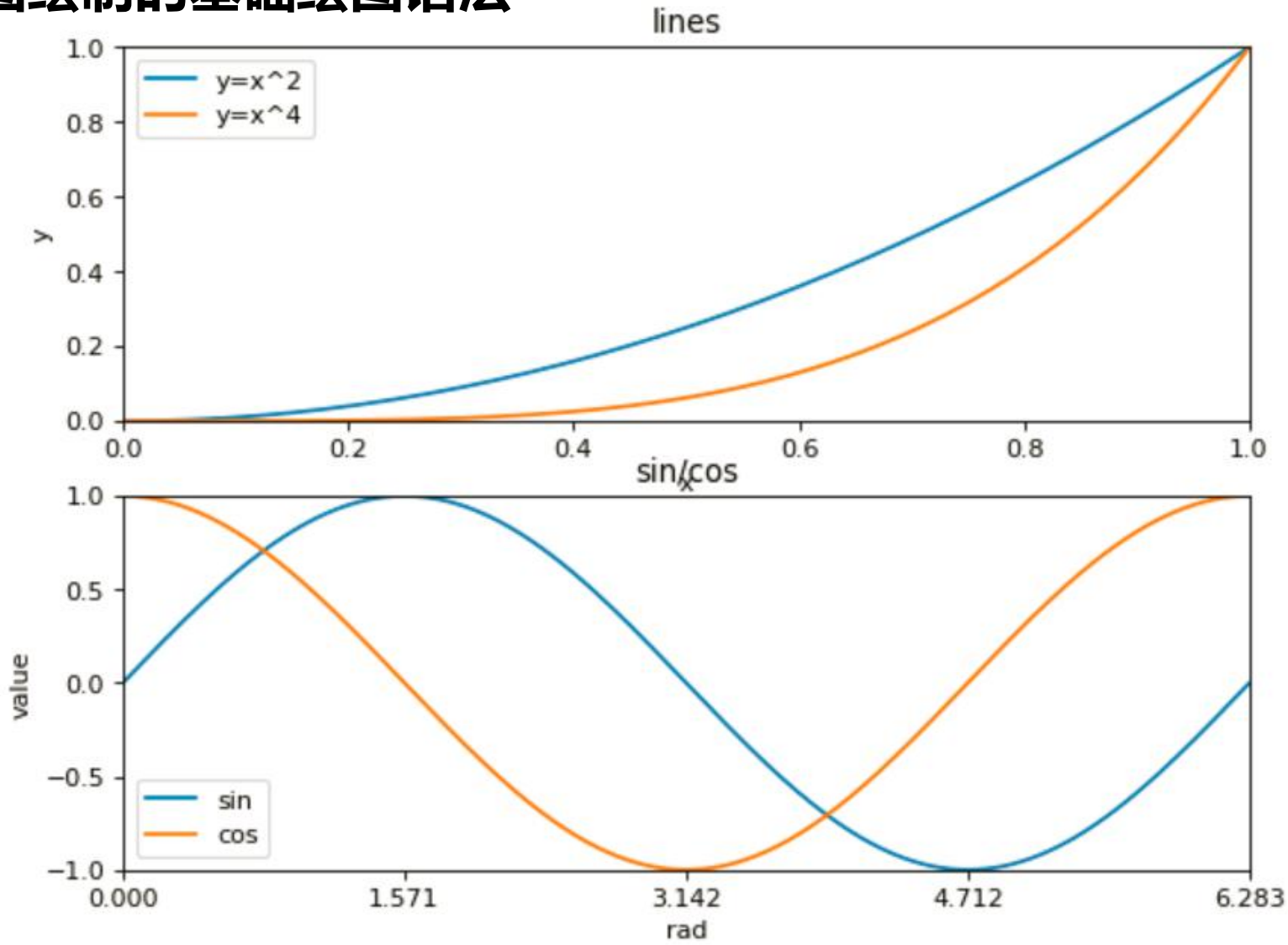
```
plt.xticks([0,np.pi/2,np.pi,np.pi*1.5,np.pi*2])      # 规定x轴刻度
plt.yticks([-1,-0.5,0,0.5,1])     # 确定y轴刻度
plt.plot(rad,np.sin(rad))         # 添加sin曲线
plt.plot(rad,np.cos(rad))         # 添加cos曲线
plt.legend(['sin','cos'])


plt.savefig('../tmp/sincos.png')
plt.show()
```



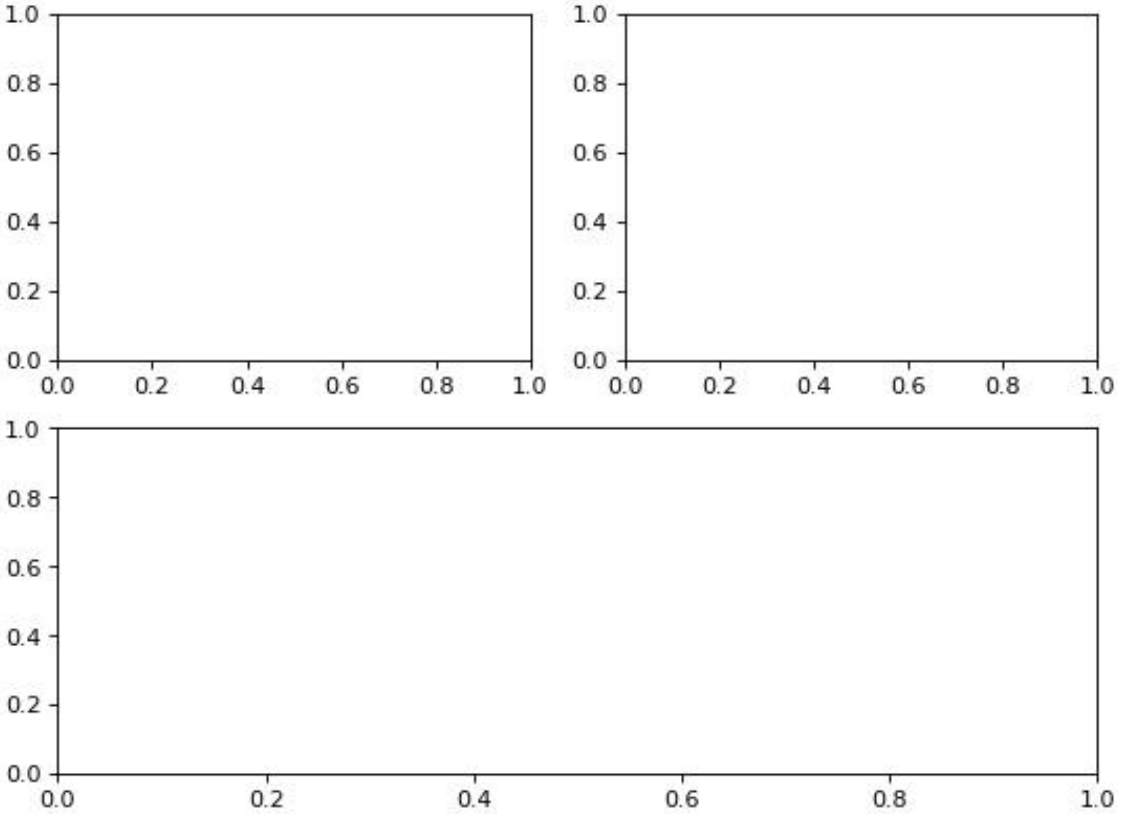
掌握pyplot基础语法

示例：包含子图绘制的基础绘图语法



掌握pyplot基础语法

示例：包含子图绘制的基础绘图语法



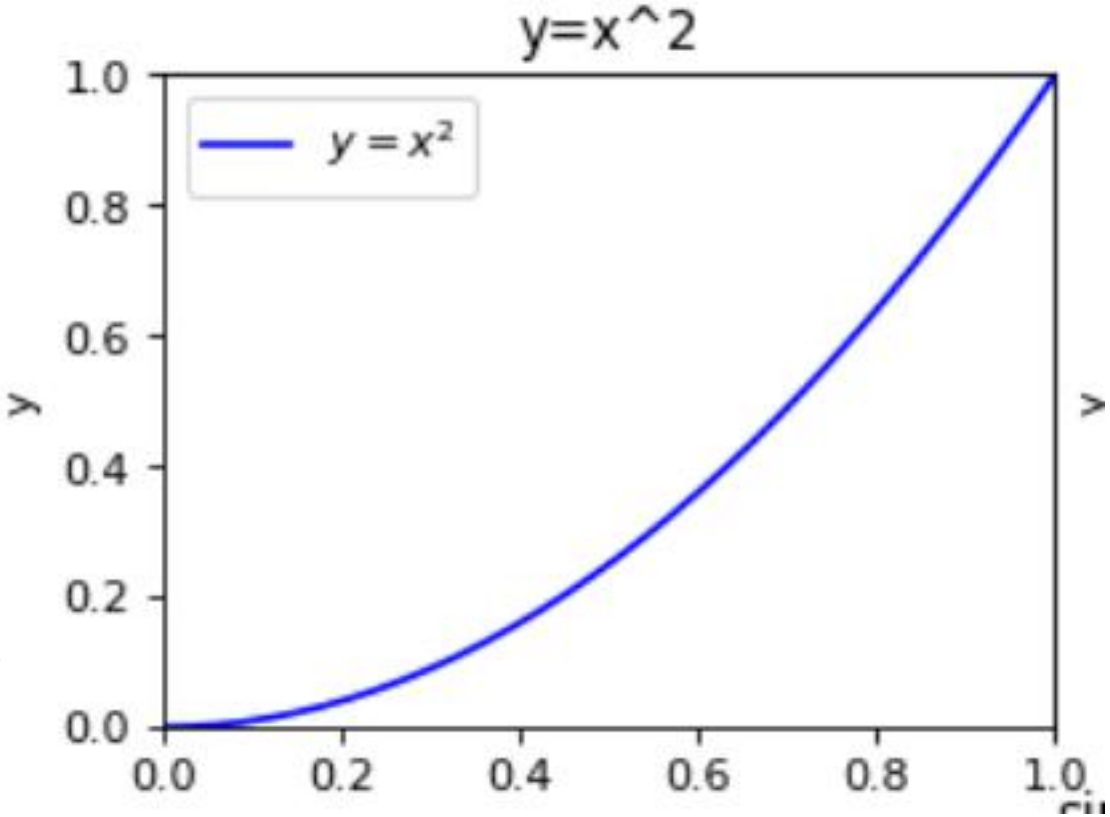

```
import numpy as np
import matplotlib.pyplot as plt
```

```
p1 = plt.figure(figsize=(8,6),dpi=80)        # 确定画布大小
ax1 = p1.add_subplot(2,2,1)                  # 创建一个两行2列的第1行子图1，返回axes对象
ax2 = p1.add_subplot(2,2,2)                  # 创建一个两行2列的第1行子图2，返回axes对象
ax3 = p1.add_subplot(2,1,2)                  # 创建一个两行1列的第2行子图，返回axes对象
```

掌握pyplot基础语法

示例：包含子图绘制的基础绘图语法

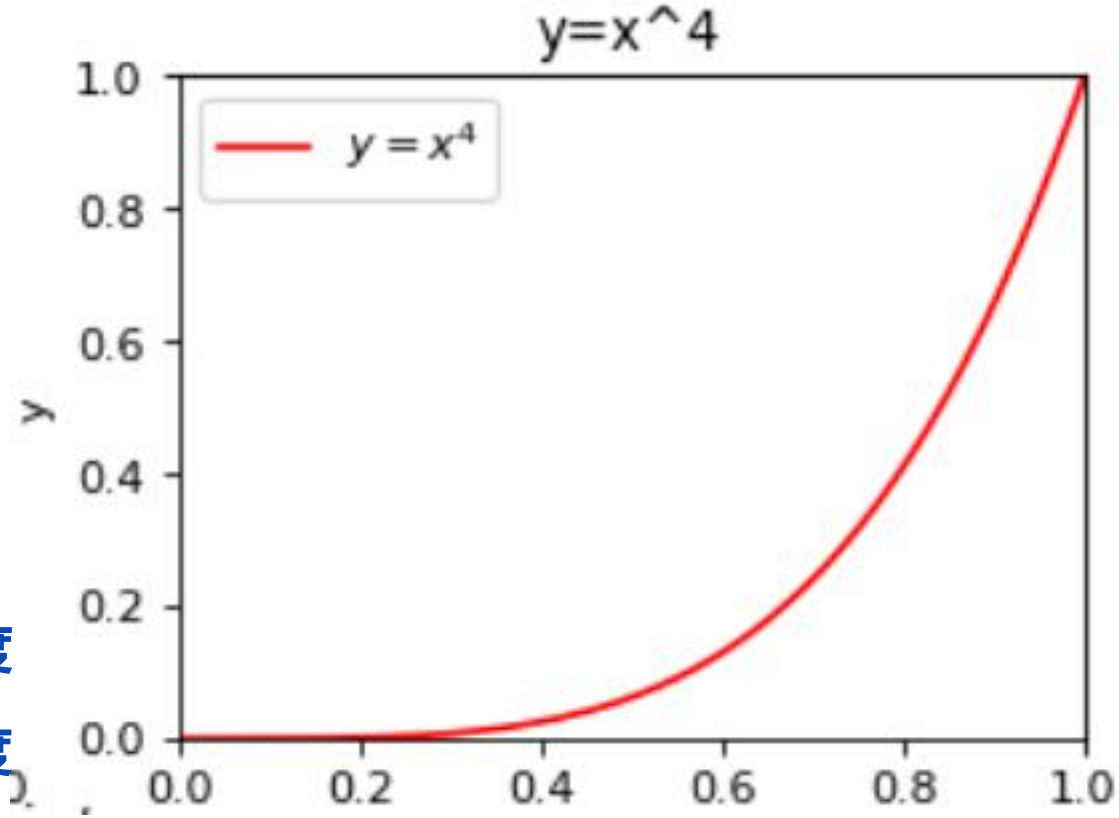
```
plt.sca(ax1)                    # 第一幅子图
plt.title('y=x^2')              # 添加标题
plt.xlabel('x')                 # 添加x轴的名称
plt.ylabel('y')                 # 添加y轴的名称
plt.xlim((0,1))                 # 确定x轴范围
plt.ylim((0,1))                 # 确定y轴范围
plt.xticks([0,0.2,0.4,0.6,0.8,1])       # 规定x轴刻度
plt.yticks([0,0.2,0.4,0.6,0.8,1])       # 确定y轴刻度
rad = np.arange(0,np.pi*2,0.01)
plt.plot(rad,rad**2,color='blue')       # 添加y=x^2曲线
plt.legend( '$y=x^2$' )
```



掌握pyplot基础语法

示例：包含子图绘制的基础绘图语法

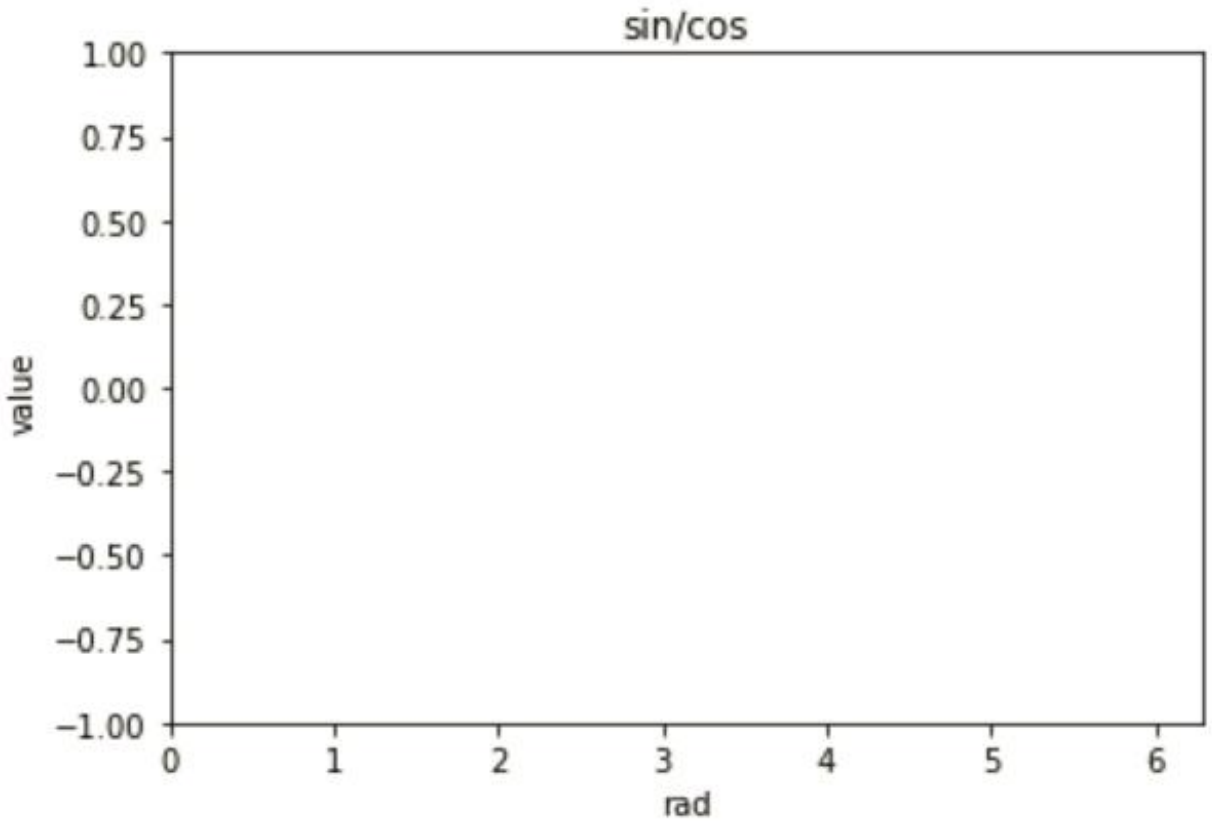
```
plt.sca(ax2)                    # 第二幅子图
plt.title('y=x^4')              # 添加标题
plt.xlabel('x')                 # 添加x轴的名称
plt.ylabel('y')                 # 添加y轴的名称
plt.xlim((0,1))                 # 确定x轴范围
plt.ylim((0,1))                 # 确定y轴范围
plt.xticks([0,0.2,0.4,0.6,0.8,1])       # 规定x轴刻度
plt.yticks([0,0.2,0.4,0.6,0.8,1])       # 确定y轴刻度
rad = np.arange(0,np.pi*2,0.01)
plt.plot(rad,rad**4,color='red')        # 添加y=x^4曲线
plt.legend('$y=x^4$' )
```



掌握pyplot基础语法

示例：包含子图绘制的基础绘图语法

```
plt.sca(ax3)              # 开始绘制第3幅
plt.title('sin/cos')      # 添加标题
plt.xlabel('rad')         # 添加x轴的名称
plt.ylabel('value')       # 添加y轴的名称
plt.xlim((0,np.pi*2))     # 确定x轴范围
plt.ylim((-1,1))          # 确定y轴范围
```

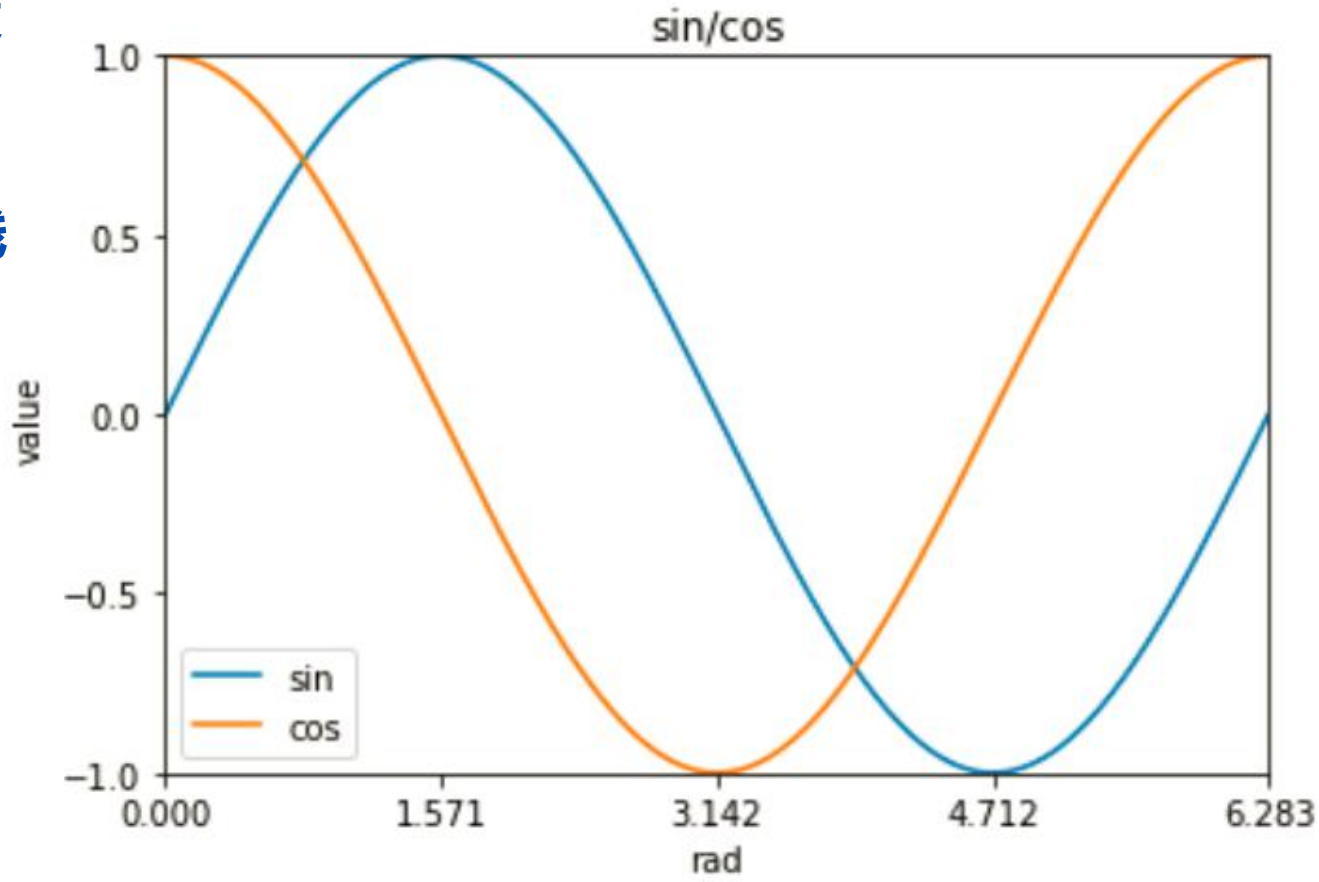


掌握pyplot基础语法

示例：包含子图绘制的基础绘图语法

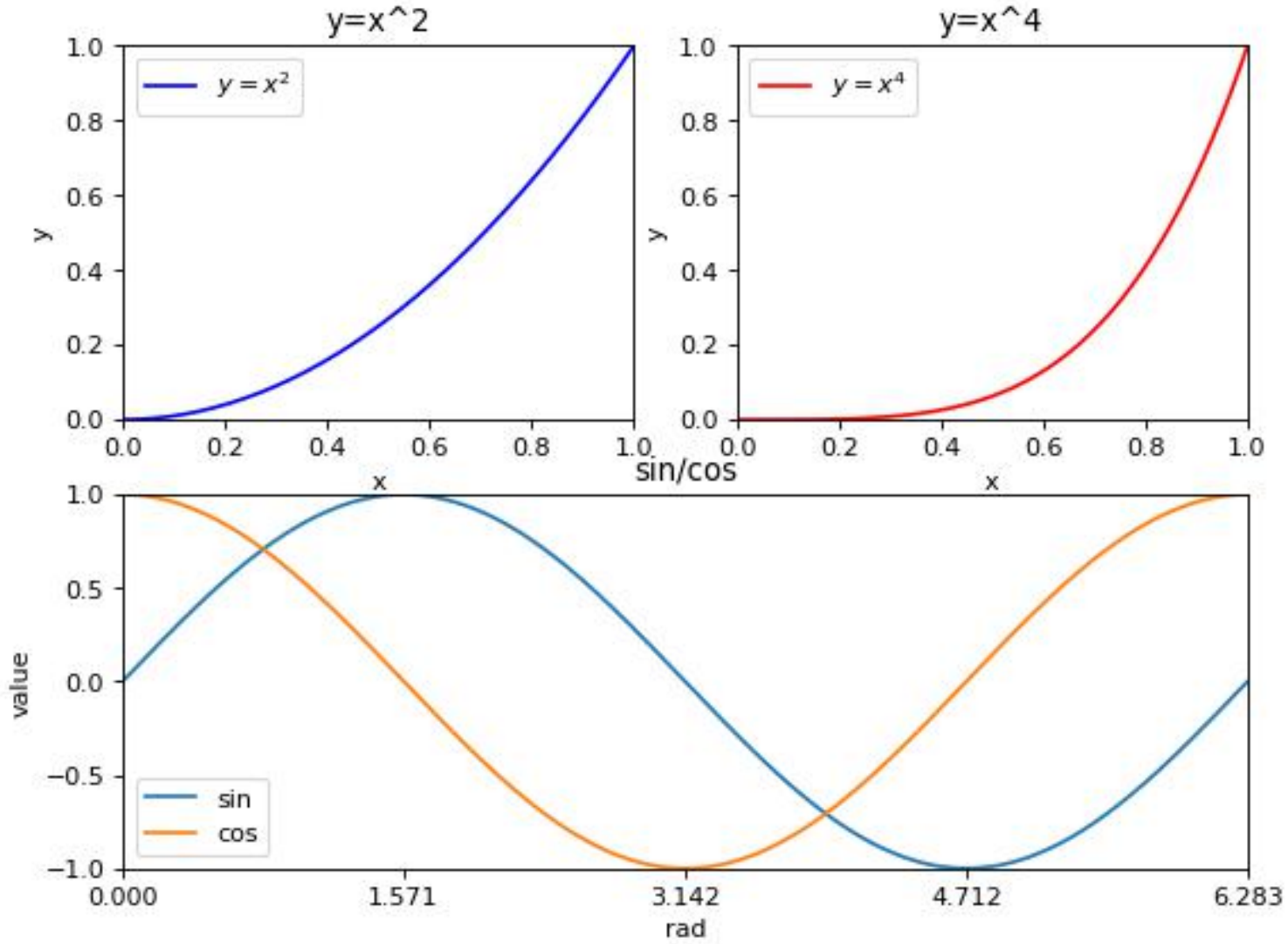
```
plt.xticks([0,np.pi/2,np.pi,np.pi*1.5,np.pi*2])      # 规定x轴刻度
plt.yticks([-1,-0.5,0,0.5,1])     # 确定y轴刻度
plt.plot(rad,np.sin(rad))         # 添加sin曲线
plt.plot(rad,np.cos(rad))         # 添加cos曲线
plt.legend(['sin','cos'])


plt.savefig('../tmp/sincos.png')
plt.show()
```



掌握pyplot基础语法

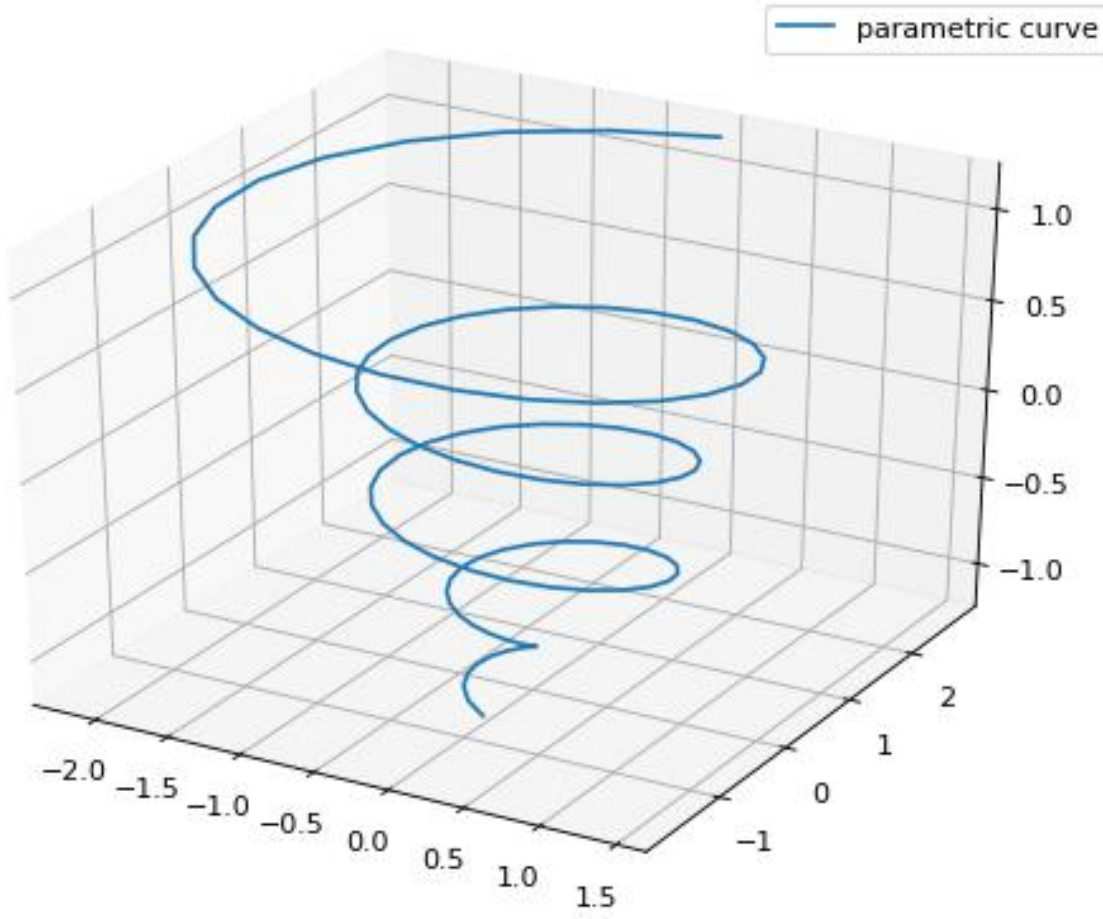
示例：包含子图绘制的基础绘图语法



掌握pyplot基础语法

示例：绘制三维参数曲线

```
import matplotlib as mpl
from mpl_toolkits.mplot3d import Axes3D
import numpy as np
import matplotlib.pyplot as plt
fig = plt.figure(figsize=(8,6),dpi=80)      # 确定画布大小
ax = fig.gca(projection='3d')               # 三维图形
theta = np.linspace(-4*np.pi, 4*np.pi, 100)
z = np.linspace(-4, 4, 100)*0.3
r = z**3+1
x = r * np.sin(theta)
y = r * np.cos(theta)
ax.plot(x,y,z, label='parametric curve')
ax.legend()
plt.show()
```

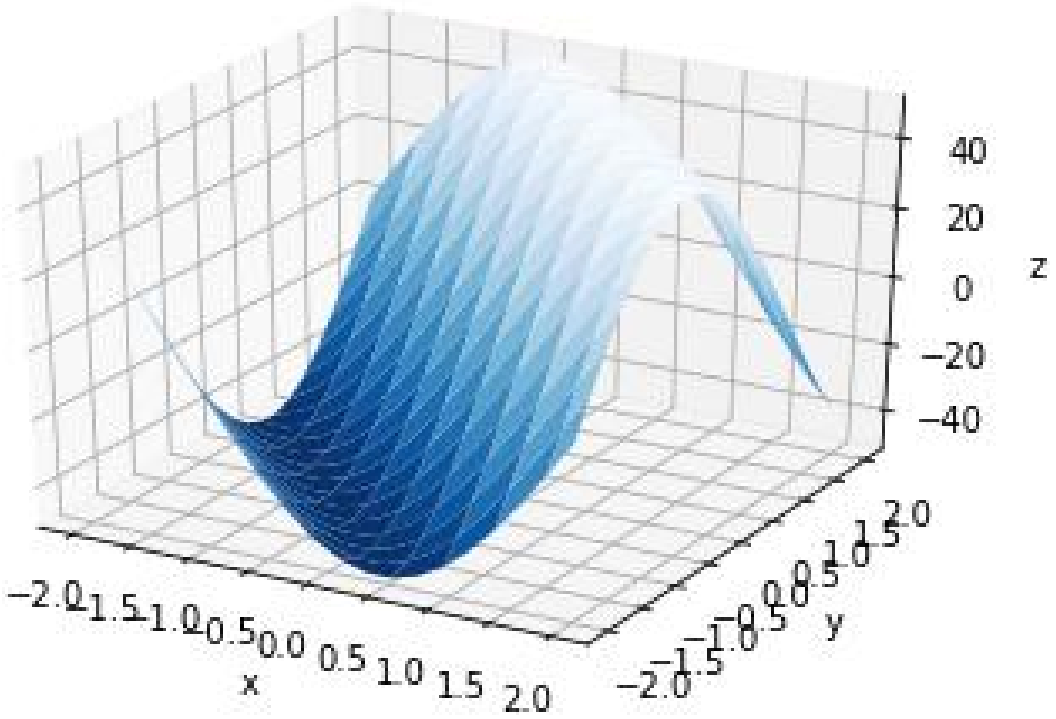


掌握pyplot基础语法

示例：绘制三维参数曲线

```
import numpy as np
import matplotlib.pyplot as plt
import mpl_toolkits.mplot3d

x, y =np.mgrid[-2:2:20j, -2:2:20j]      #测试数据
z = 50 * np.sin(x+y)
ax = plt.subplot(111, projection='3d')
ax.plot_surface(x,y,z, rstride=2, cstride=1, cmap=plt.cm.Blues_r)
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('z')
plt.show()
```

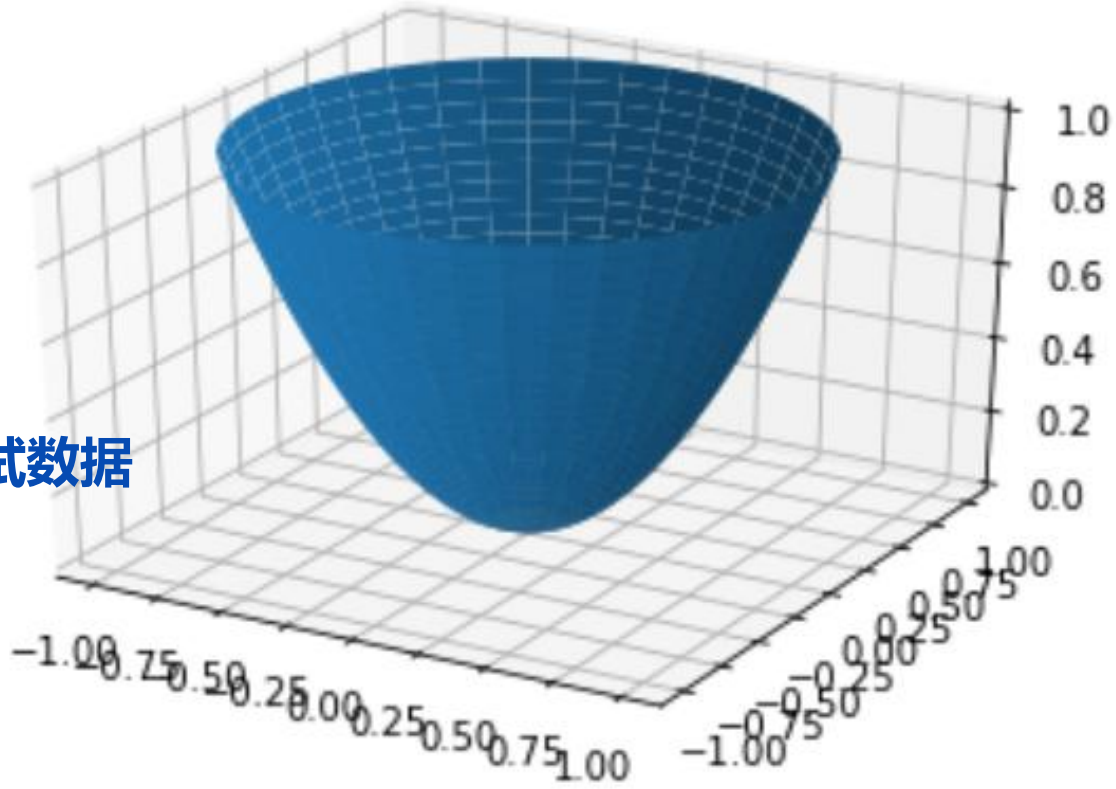


掌握pyplot基础语法

示例：绘制三维参数曲线

```
import numpy as np
import matplotlib.pyplot as plt
import mpl_toolkits.mplot3d

rho, theta =np.mgrid[0:1:40j, 0:2*np.pi:40j]  #测试数据
z = rho ** 2
x = rho * np.cos(theta)
y = rho * np.sin(theta)
ax = plt.subplot(111, projection='3d')
ax.plot_surface(x,y,z)
plt.show()
```



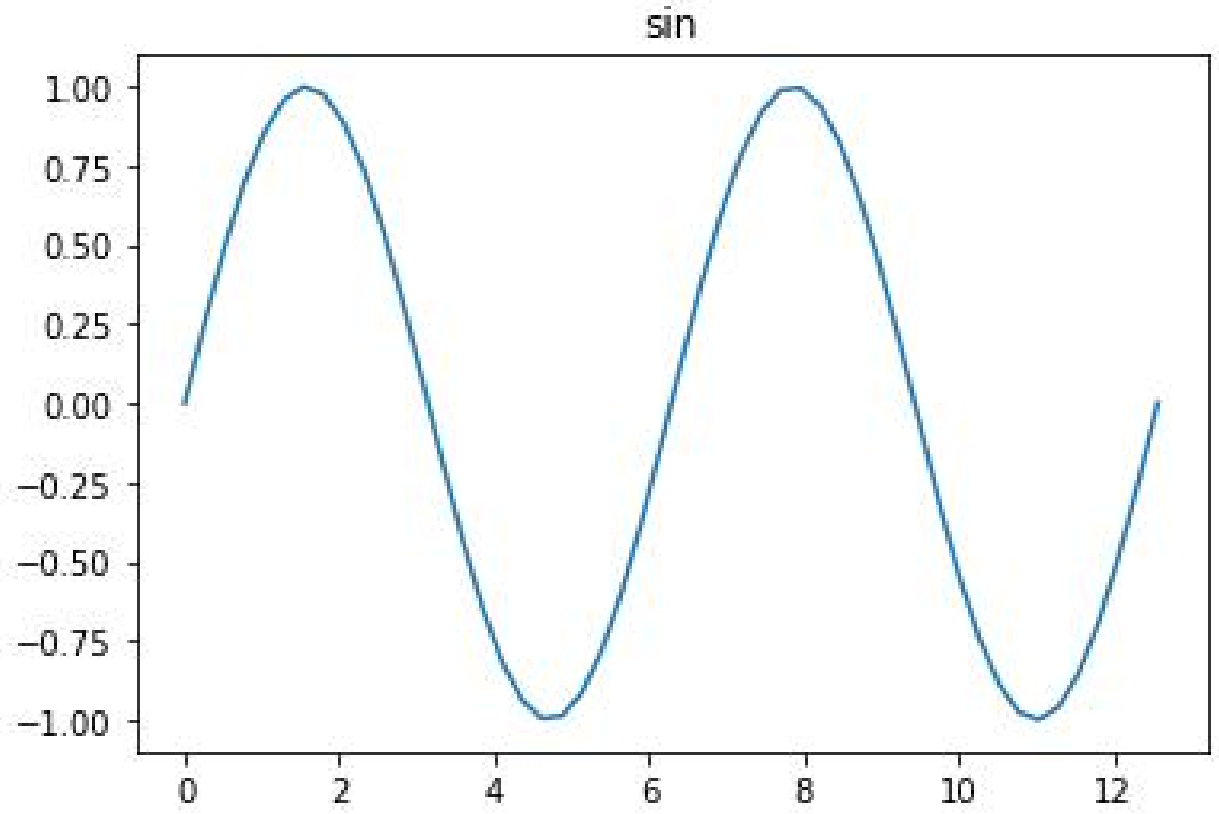
设置pyplot的动态rc参数

- pyplot使用**rc配置文件**来**自定义图形的各种默认属性**，被称为rc配置或rc参数。
- 在pyplot中几乎所有的默认属性都是可以控制的，例如视图窗口大小以及每英寸点数、线条宽度、颜色和样式、坐标轴、坐标和网格属性、文本、字体等。
- 默认rc参数可以在Python交互环境中动态更改
- 实际上，在代码执行过程中，有两种方式更改参数：
  - 使用参数字典(rcParams)：所有存储在字典变量中的rc参数，都被称为rcParams；
  - 调用matplotlib.rc()命令 通过传入关键字元组，修改参数；

设置pyplot的动态rc参数

示例：线条rc参数的修改

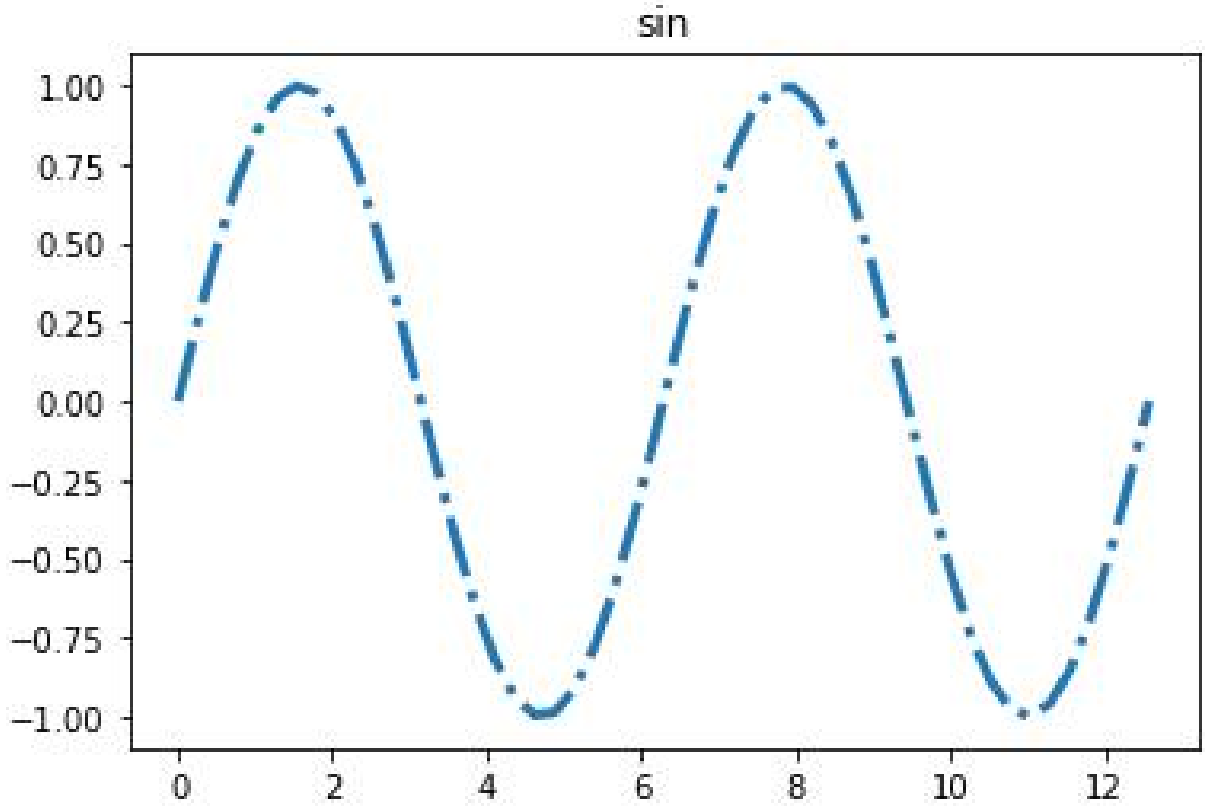
```
## 原图
x = np.linspace(0, 4*np.pi)    # 生成x轴数据
y = np.sin(x)                  # 生成y轴数据
plt.plot(x,y,label="$sin(x)$")# 绘制sin曲线图
plt.title('sin')
plt.savefig('../tmp/默认sin曲线.png')
plt.show()
```



设置pyplot的动态rc参数

示例：线条rc参数的修改

```
## 修改rc参数后的图
plt.rcParams['lines.linestyle'] = '-.'
plt.rcParams['lines.linewidth'] = 3
plt.plot(x,y,label="$sin(x)$")# 绘制sin曲线图
plt.title('sin')
plt.savefig('../tmp/修改rc参数后sin曲线.png')
plt.show()
```



设置pyplot的动态rc参数

线条的常用rc参数名称、解释与取值

rc参数名称	解释	取值
lines.linewidth	线条宽度	取0-10之间的数值，默认为1.5。
lines.linestyle	线条样式	可取“-” “--” “-.” “：” 四种。默认为“-”。
lines.marker	线条上点的形状	可取“o” “D” “h” “.” “,” “S”等20种，默认为None。
lines.markersize	点的大小	取0-10之间的数值，默认为1。

设置pyplot的动态rc参数

常用线条类型解释

linestyle取值	意义
-	实线
--	长虚线
-·	点线
:	短虚线

线条标记解释

marker取值	意义	marker取值	意义
‘o’	圆圈	‘.’	点
‘D’	菱形	‘s’	正方形
‘h’	六边形1	‘*’	星号
‘H’	六边形2	‘d’	小菱形
‘-’	水平线	‘v’	一角朝下的三角形
‘8’	八边形	‘<’	一角朝左的三角形
‘p’	五边形	‘>’	一角朝右的三角形
‘ , ’	像素	‘^’	一角朝上的三角形
‘+’	加号	‘\’	竖线
‘None’	无	‘x’	X

设置pyplot的动态rc参数

注意事项

- 由于**默认的pyplot字体并不支持中文**字符的显示，因此需要通过设置font.sans-serif参数改变绘图时的字体，使得图形可以正常显示中文。
- 同时，由于更改字体后，会导致坐标轴中的部分字符无法显示（例如负号），因此需要同时更改axes.unicode_minus参数，以正常显示负号。

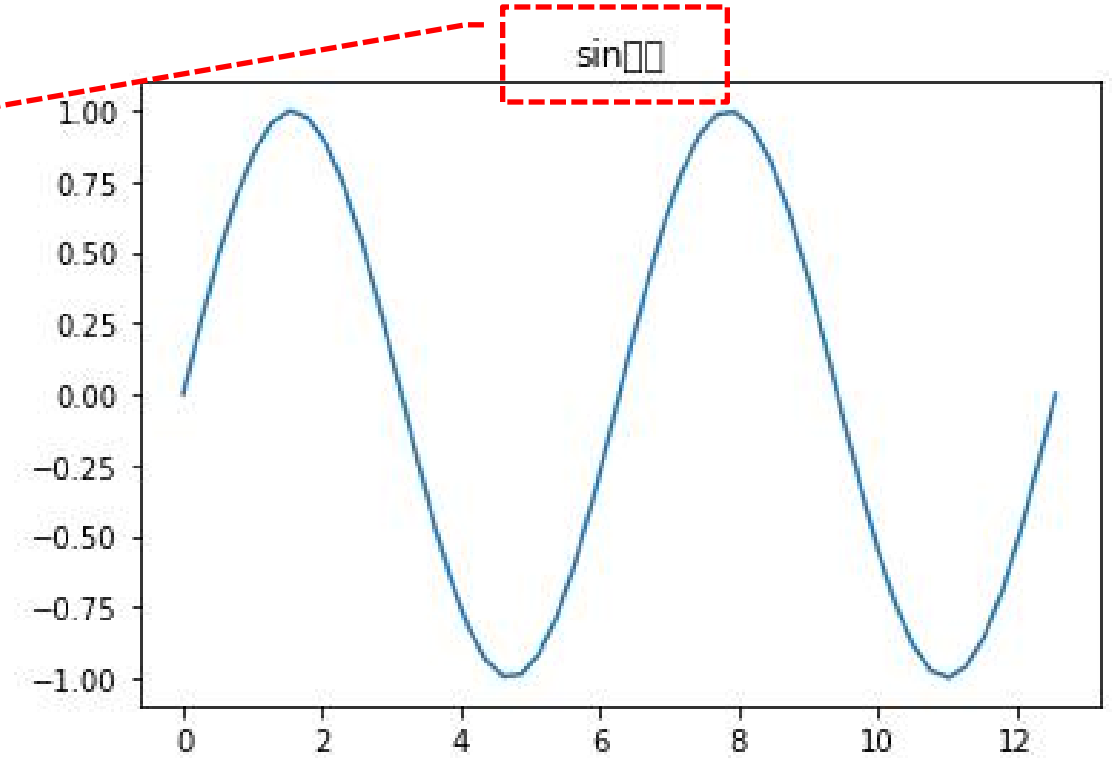
- 除了设置线条和字体的rc参数外，还有设置文本、箱线图、坐标轴、刻度、图例、标记、图片、图像保存等rc参数。具体参数与取值可以参考官方文档。

设置pyplot的动态rc参数

实例：显示中文字体

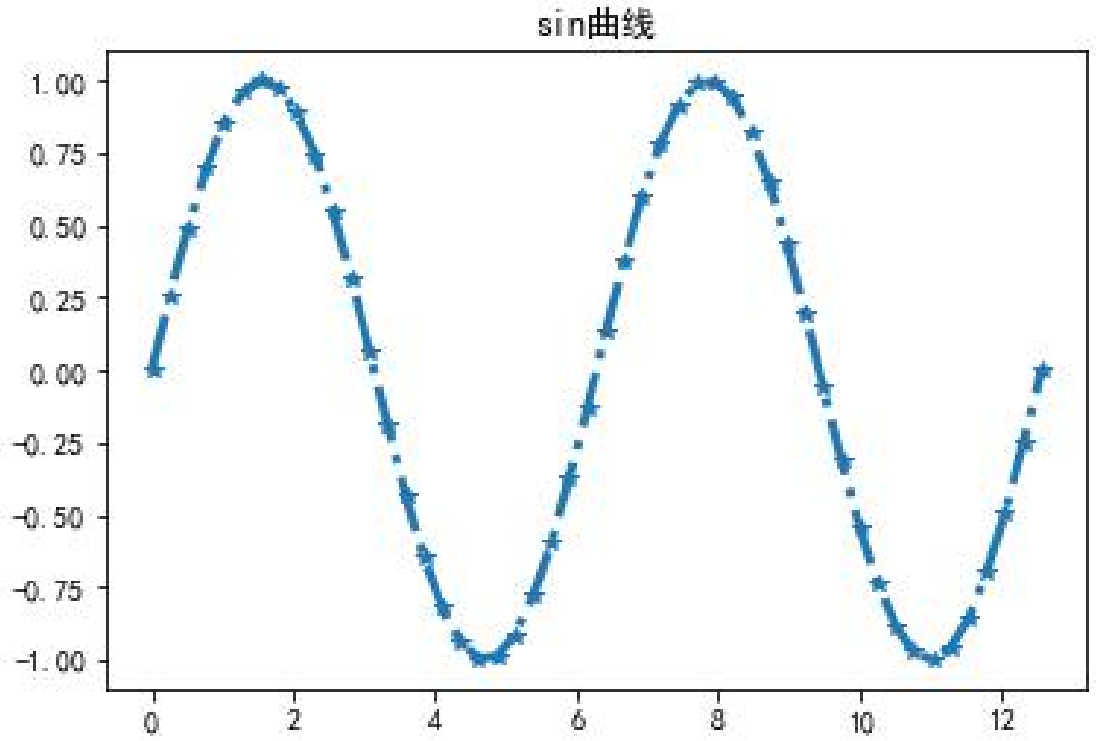
```
## 无法显示中文标题
plt.plot(x,y,label="$sin(x)$")    # 绘制三角函数
plt.title('sin曲线')


plt.savefig('../tmp/无法显示中文标题sin曲线.png')
plt.show()
```

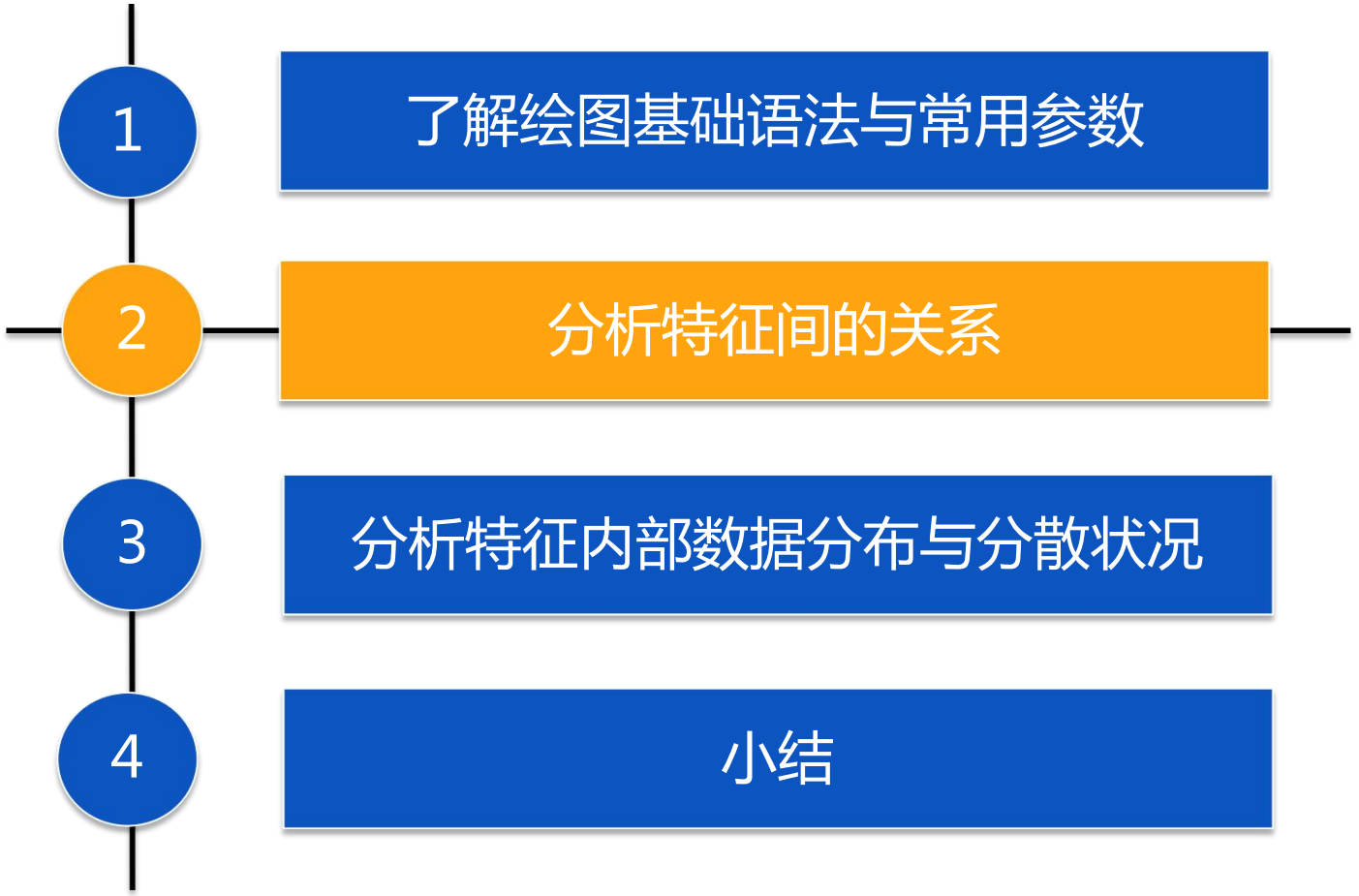


设置pyplot的动态rc参数

```
##设置rc参数显示中文标题
## 设置字体为SimHei显示中文
plt.rcParams['font.sans-serif'] = 'SimHei'
## 设置正常显示符号
plt.rcParams['axes.unicode_minus'] = False
plt.rcParams['lines.marker'] = '*'
plt.plot(x,y,label="$sin(x)$")    # 绘制三角函数
plt.title('sin曲线')
plt.savefig('../tmp/显示中文标题sin曲线.png')
plt.show()
```

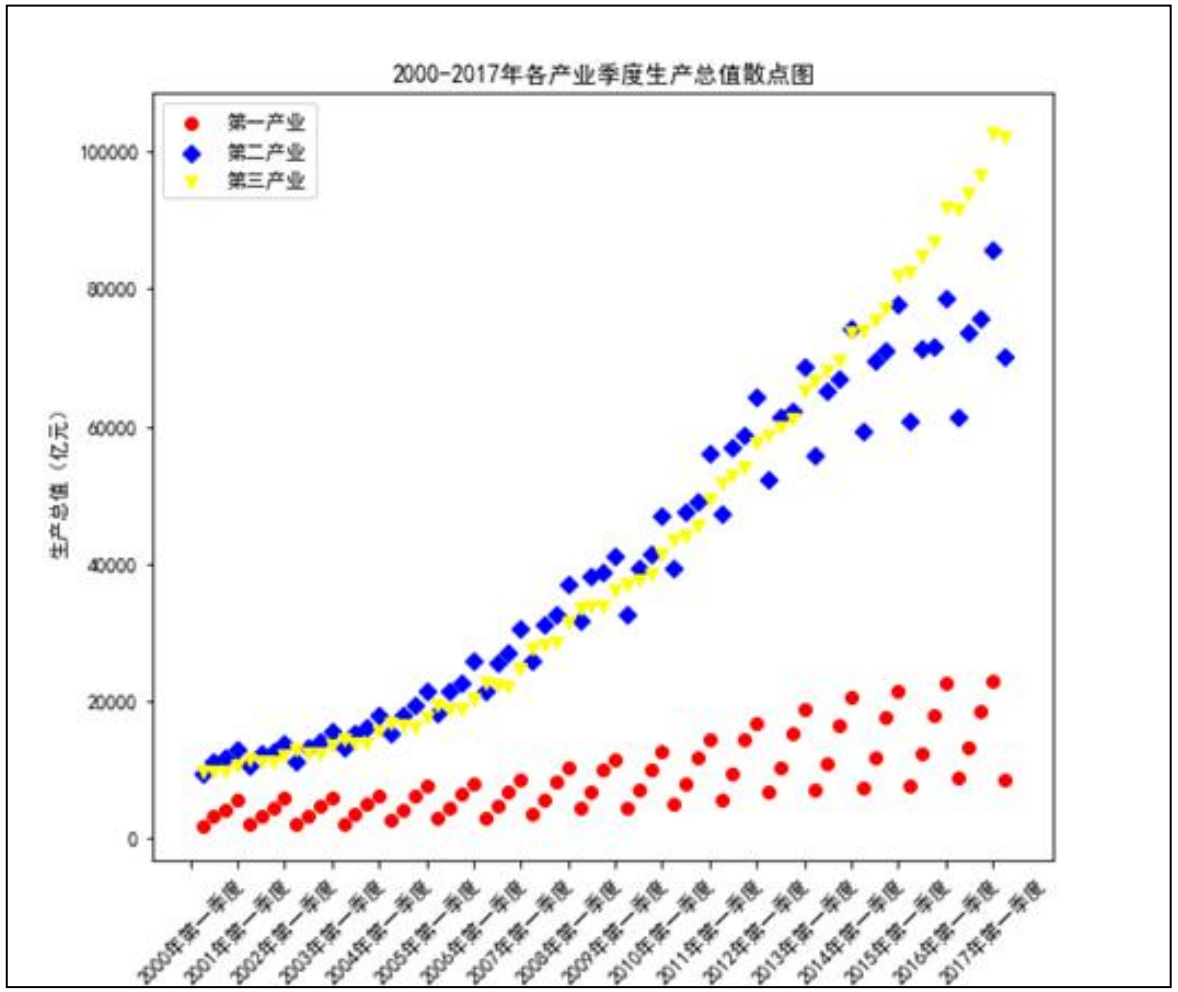


目录

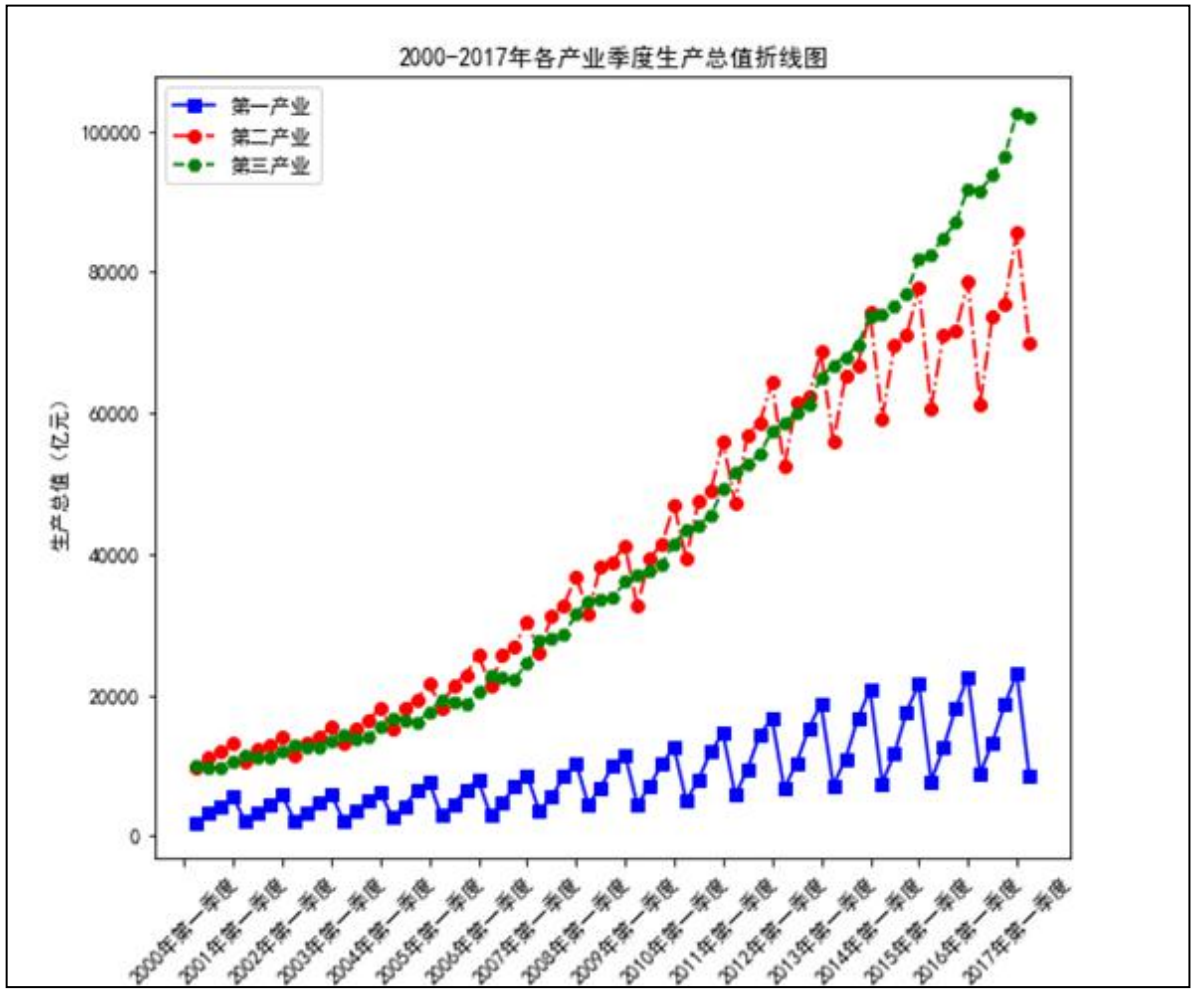


分析特征间的关系

散点图



折线图

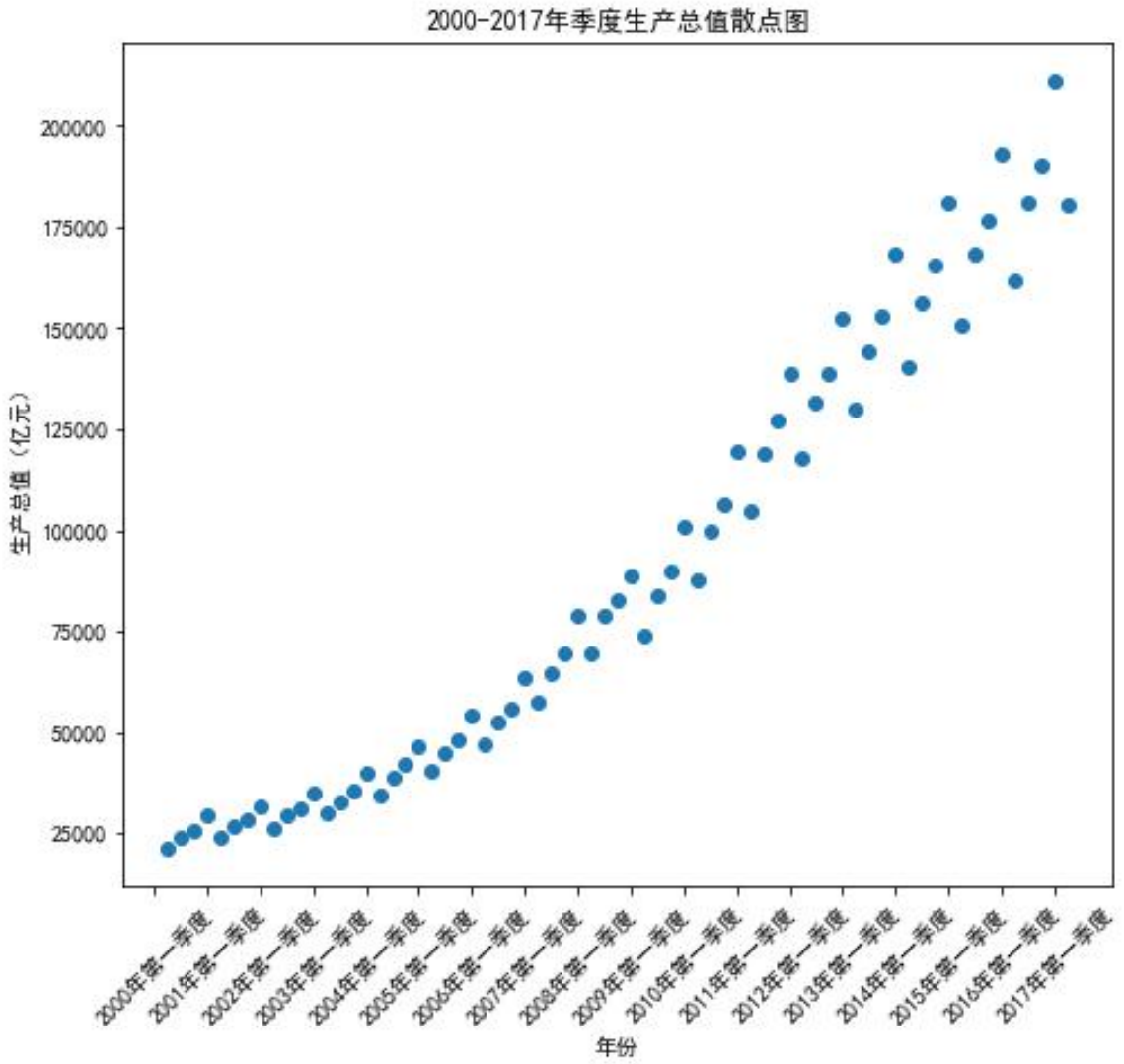


绘制散点图

散点图

- 散点图（scatter diagram）又称为散点分布图，是以一个特征为横坐标，另一个特征为纵坐标，利用坐标点（散点）的分布形态反映特征间的统计关系的一种图形。

- 值是由点在图表中的位置表示，类别是由图表中的不同标记表示，通常用于比较跨类别的数据。



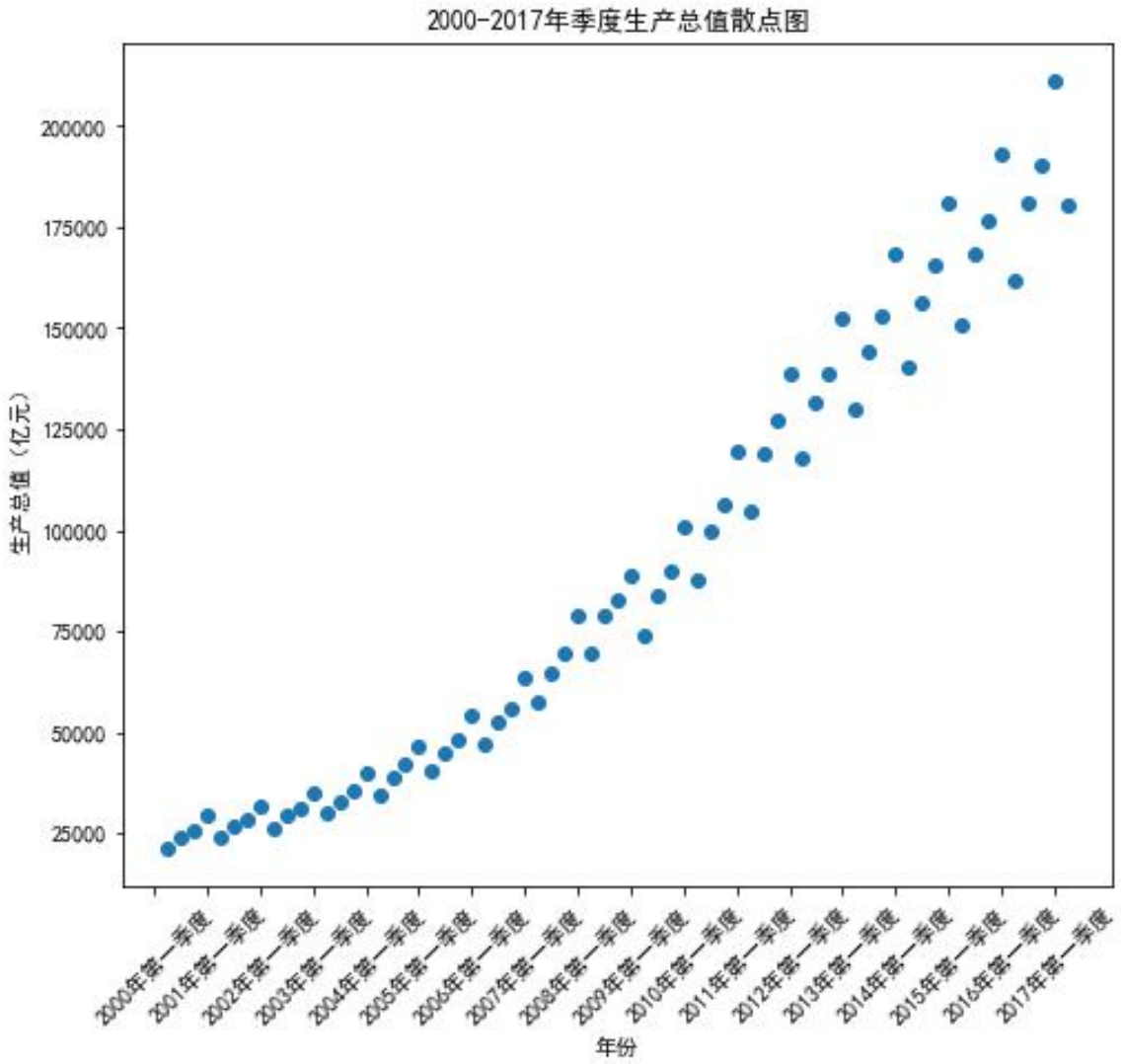
绘制散点图

散点图

- 散点图可以提供两类关键信息。

(1)特征之间是否存在数值或者数量的关联趋势，关联趋势是线性的还是非线性的。

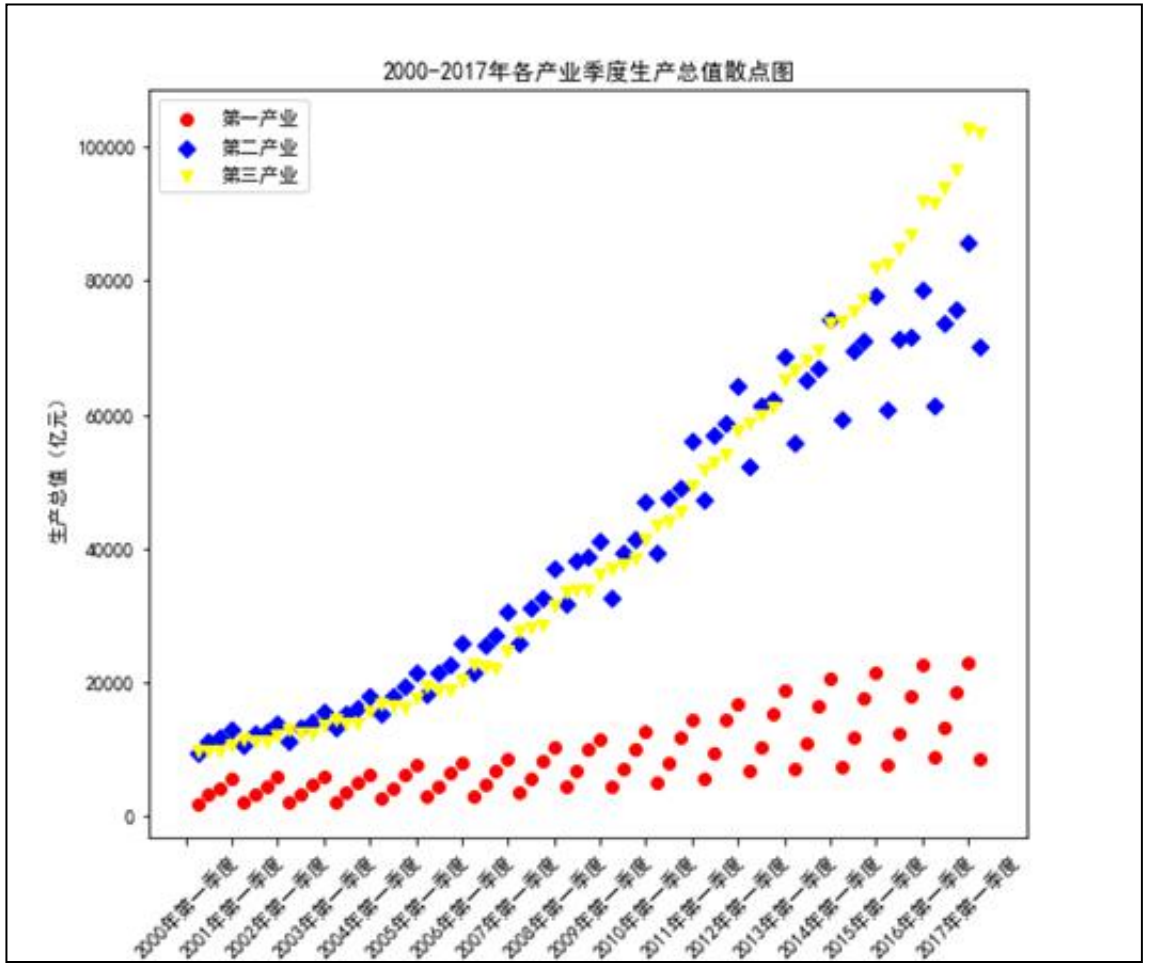
(2)如果某一个点或者某几个点偏离大多数点，则这些点就是离群值、通过散点图可以一自了然，从而可以进一步分析这些离群值是否在建模分析中产生很大的影响。



绘制散点图

散点图

- 散点图通过散点的疏密程度和变化趋势表示两个特征的数量关系。
- 如果有3个特征、其中一个特征为类别型，散点图改变不同特征的点的形状或者颜色，即可了解两个数值型特征和这个类别型之间的关系。



绘制散点图

scatter函数

matplotlib.pyplot.scatter(x, y, s=None, c=None, marker=None, alpha=None, **kwargs)

- 常用参数及说明如下表所示：

参数名称	说明
x，y	接收array。表示x轴和y轴对应的数据。无默认。
s	接收数值或者一维的array。指定点的大小，若传入一维array则表示每个点的大小。默认为None。
c	接收颜色或者一维的array。指定点的颜色，若传入一维array则表示每个点的颜色。默认为None
marker	接收特定string。表示绘制的点的类型。默认为None。
alpha	接收0-1的小数。表示点的透明度。默认为None。

绘制散点图

示例：使用scatter函数绘制2000~2017年各季度国民生产总值散点图

```
import numpy as np
import matplotlib.pyplot as plt
plt.rcParams['font.sans-serif'] = 'SimHei'              # 设置中文显示
plt.rcParams['axes.unicode_minus'] = False              # 设置负号可显示
data = np.load('../data/国民经济核算季度数据.npz')
name = data['columns']                  # 提取其中的columns数组，视为数据的标签
values = data['values']                 # 提取其中的values数组，数据的存在位置
```

绘制散点图

示例：使用scatter函数绘制2000~2017年各季度国民生产总值散点图

In[1]:	**name**
Out[1]:	array(['序号', '时间', '国内生产总值_当季值(亿元)', '第一产业增加值_当季值(亿元)', '第二产业增加值_当季值(亿元)', '第三产业增加值_当季值(亿元)', '农林牧渔业增加值_当季值(亿元)', '工业增加值_当季值(亿元)', '建筑业增加值_当季值(亿元)', '批发和零售业增加值_当季值(亿元)', '交通运输、仓储和邮政业增加值_当季值(亿元)', '住宿和餐饮业增加值_当季值(亿元)', '金融业增加值_当季值(亿元)', '房地产业增加值_当季值(亿元)', '其他行业增加值_当季值(亿元)'], dtype=object)
In[2]:	**values**
Out[2]:	array([[1, '2000年第一季度', 21329.9, ..., 1235.9, 933.7, 3586.1], [2, '2000年第二季度', 24043.4, ..., 1124.0, 904.7, 3464.9], [3, '2000年第三季度', 25712.5, ..., 1170.4, 1070.9, 3518.2], ..., [67, '2016年第三季度', 190529.5, ..., 15472.5, 12164.1, 37964.1], [68, '2016年第四季度', 211281.3, ..., 15548.7, 13214.9, 39848.4], [69, '2017年第一季度', 180682.7, ..., 17213.5, 12393.4, 42443.1]], dtype=object)

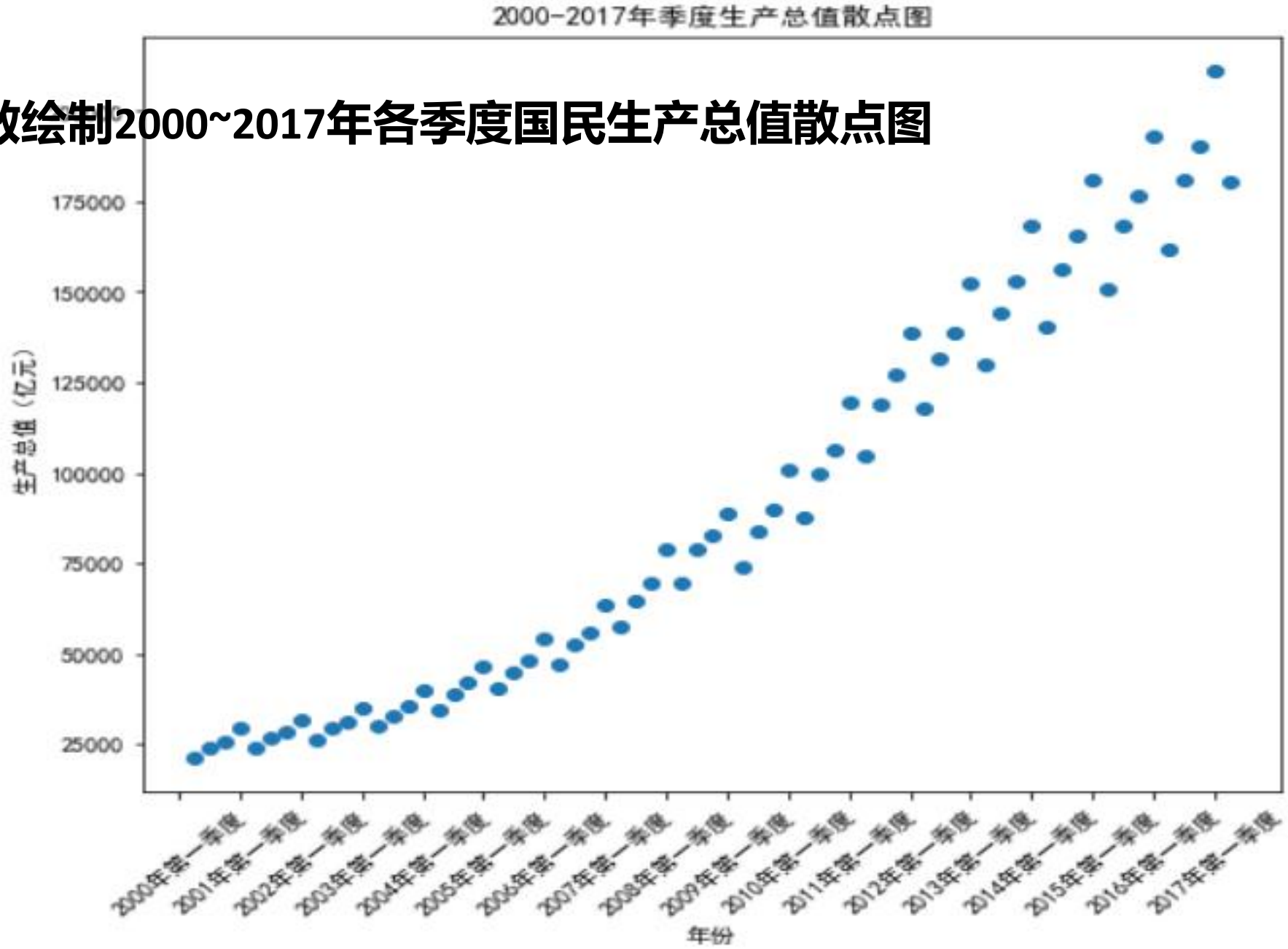
绘制散点图

示例：使用scatter函数绘制2000~2017年各季度国民生产总值散点图

```
plt.figure(figsize=(8,7))                                    # 设置画布
plt.scatter(values[:,0],values[:,2], marker='o')             # 绘制散点图
plt.xlabel('年份')                                           # 添加横轴标签
plt.ylabel('生产总值（亿元）')                               # 添加y轴名称
plt.xticks(range(0,70,4),values[range(0,70,4),1],rotation=45)    #指定x轴刻度的数目与取值
plt.title('2000-2017年季度生产总值散点图')                   # 添加图表标题
plt.savefig('../tmp/2000-2017年季度生产总值散点图.png')
plt.show()
```

绘制散点图

示例：使用scatter函数绘制2000~2017年各季度国民生产总值散点图



绘制散点图

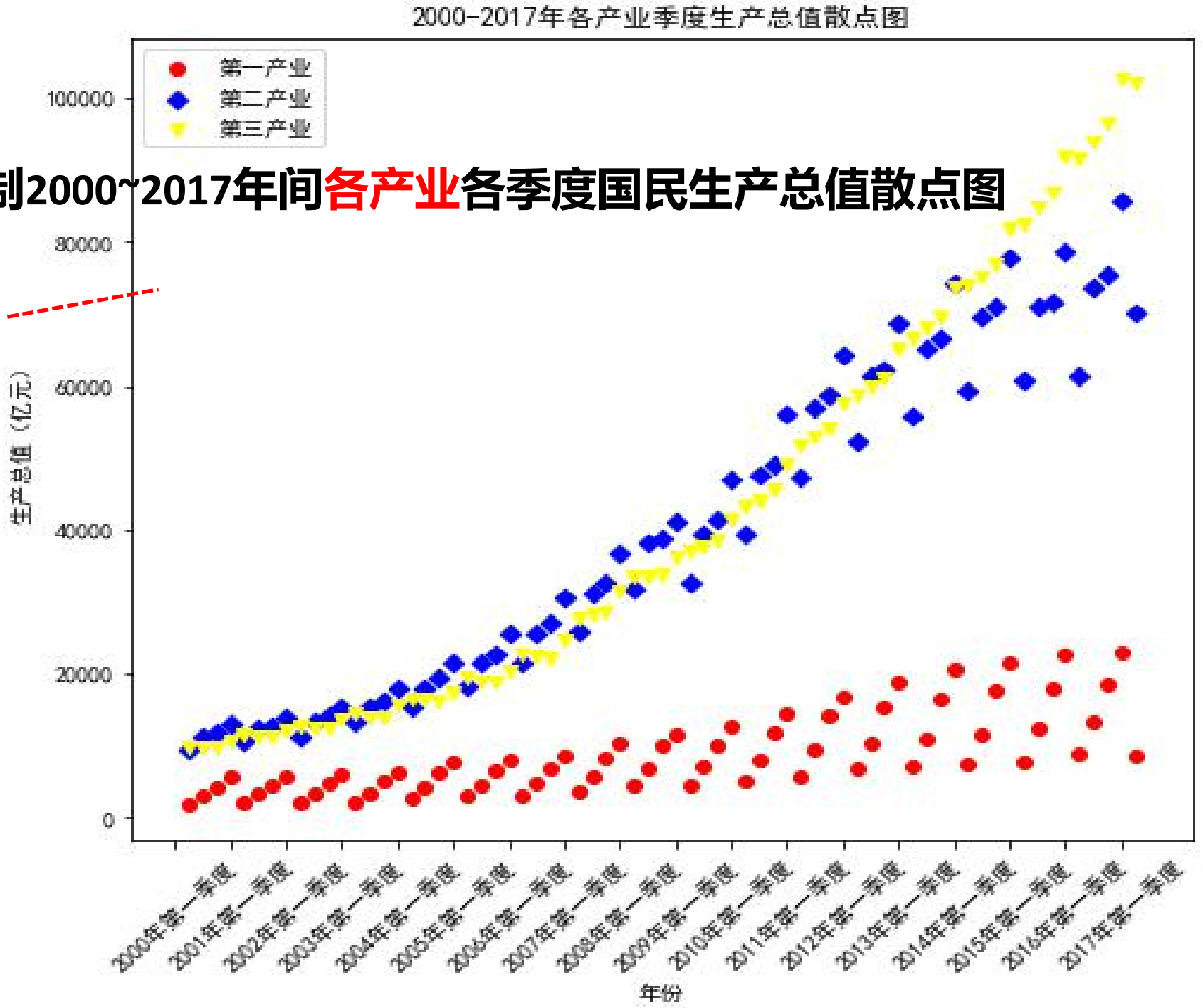
示例：使用scatter函数绘制2000~2017年间各产业各季度国民生产总值散点图

```
plt.figure(figsize=(8,7))                                    # 设置画布
plt.scatter(values[:,0],values[:,3], marker='o',c='red')     # 绘制散点1
plt.scatter(values[:,0],values[:,4], marker='D',c='blue')    # 绘制散点2
plt.scatter(values[:,0],values[:,5], marker='v',c='yellow')  # 绘制散点3
plt.xlabel('年份')                                           # 添加横轴标签
plt.ylabel('生产总值（亿元）')                               # 添加纵轴标签
plt.xticks(range(0,70,4),values[range(0,70,4),1],rotation=45)      #指定x轴刻度的数目与取值
plt.title('2000-2017年各产业季度生产总值散点图')             # 添加图表标题
plt.legend(['第一产业','第二产业','第三产业'])               # 添加图例
plt.savefig('../tmp/2000-2017年各产业季度生产总值散点图.png')
plt.show()
```

绘制散点图

示例：使用scatter函数绘制2000~2017年间各产业各季度国民生产总值散点图

- 通过这段代码绘制的散点图中点的颜色和形状的区别可以看出：
  - 第一产业增长平缓，
  - 第三产业呈现指数型增长，
  - 第二产业每年会根据季度呈现周期性的波动。
- 总体看来，我国近18年的各个产业都在持续增长中，并且第二和第三产业增长幅度非常大，18年间增长了400%以上。

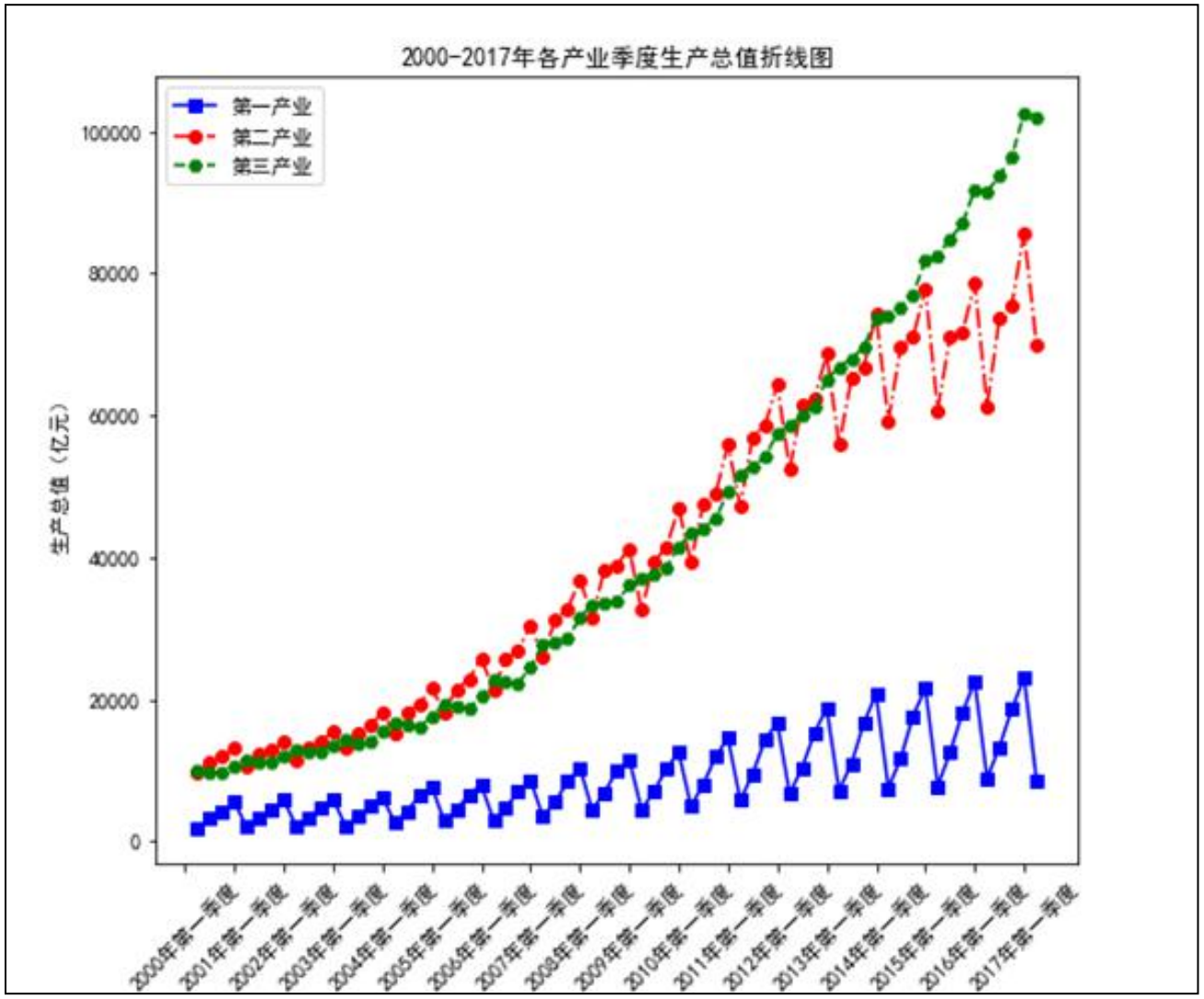


绘制折线图

折线图

- 折线图（line chart）是一种将数据点按照顺序连接起来的图形。可以看作是将散点图，按照x轴坐标顺序连接起来的图形。

- 折线图的主要功能是查看因变量y随着自变量x改变的趋势，最适合用于显示随时间（根据常用比例设置）而变化的连续数据。同时还可以看出数量的差异，增长趋势的变化。



绘制折线图

plot函数

matplotlib.pyplot.plot(*args, **kwargs)

- plot函数在官方文档的语法中只要求填入不定长参数，实际可以填入的主要参数主要如下。

参数名称	说明
x，y	接收array。表示x轴和y轴对应的数据。无默认。
color	接收特定string。指定线条的颜色。默认为None。
linestyle	接收特定string。指定线条类型。默认为“-”。
marker	接收特定string。表示绘制的点的类型。默认为None。
alpha	接收0-1的小数。表示点的透明度。默认为None。

绘制折线图

plot函数

- 其中，color参数的8种常用颜色的缩写。

颜色缩写	代表的颜色	颜色缩写	代表的颜色
b	蓝色	m	品红
g	绿色	y	黄色
r	红色	k	黑色
c	青色	w	白色

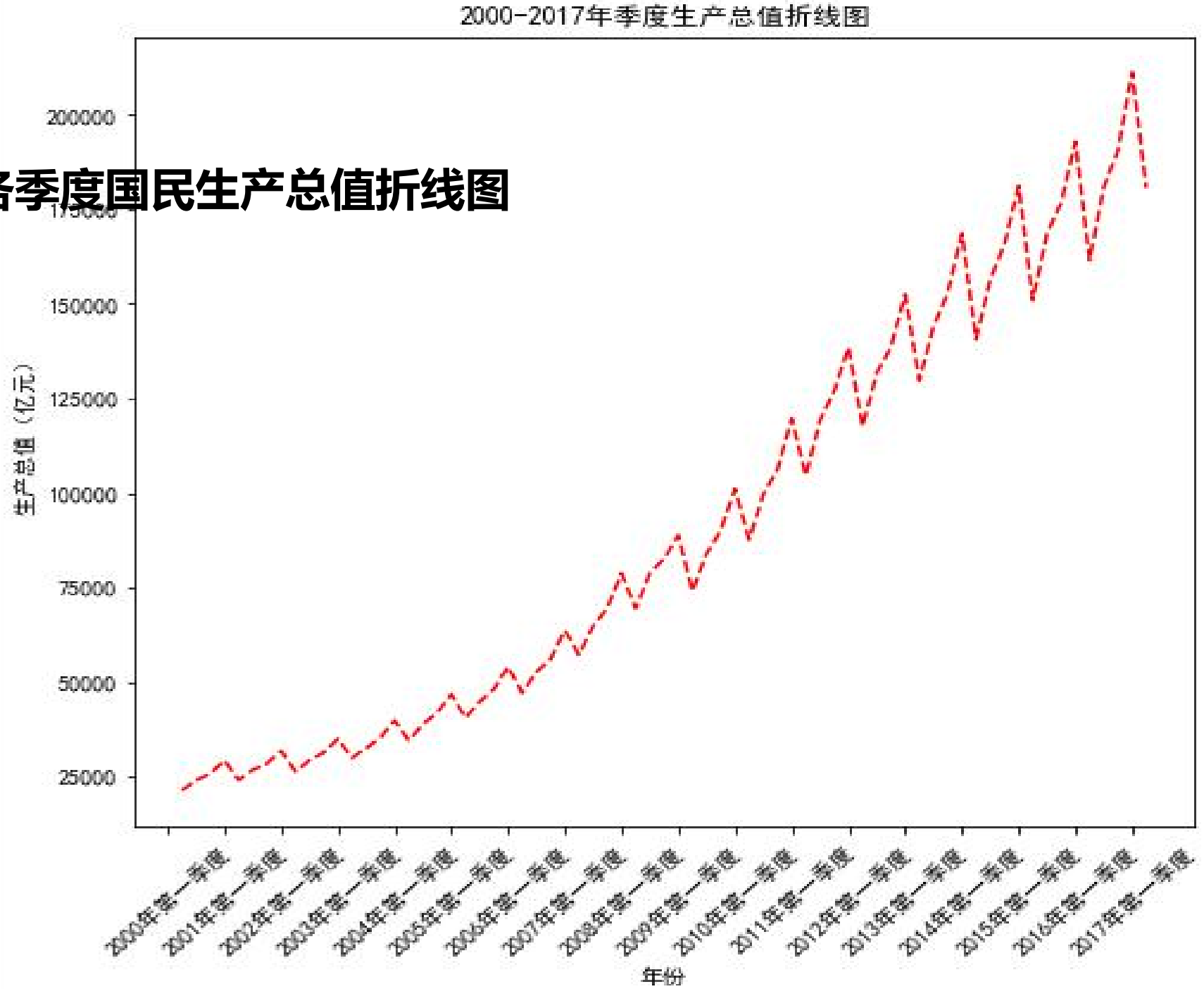
绘制折线图

示例：绘制2000-2017年各季度国民生产总值折线图

```
plt.figure(figsize=(8,7))                                          # 设置画布
plt.plot(values[:,0],values[:,2],color = 'r',linestyle = '--')     # 绘制折线图
plt.xlabel('年份')                                                  # 添加横轴标签
plt.ylabel('生产总值（亿元）')                                       # 添加y轴名称
plt.xticks(range(0,70,4),values[range(0,70,4),1],rotation=45)      #指定x轴刻度的数目与取值
plt.title('2000-2017年季度生产总值折线图')                           # 添加图表标题
plt.savefig('../tmp/2000-2017年季度生产总值折线图.png')
plt.show()
```

绘制折线图

示例：绘制2000-2017年各季度国民生产总值折线图



绘制折线图

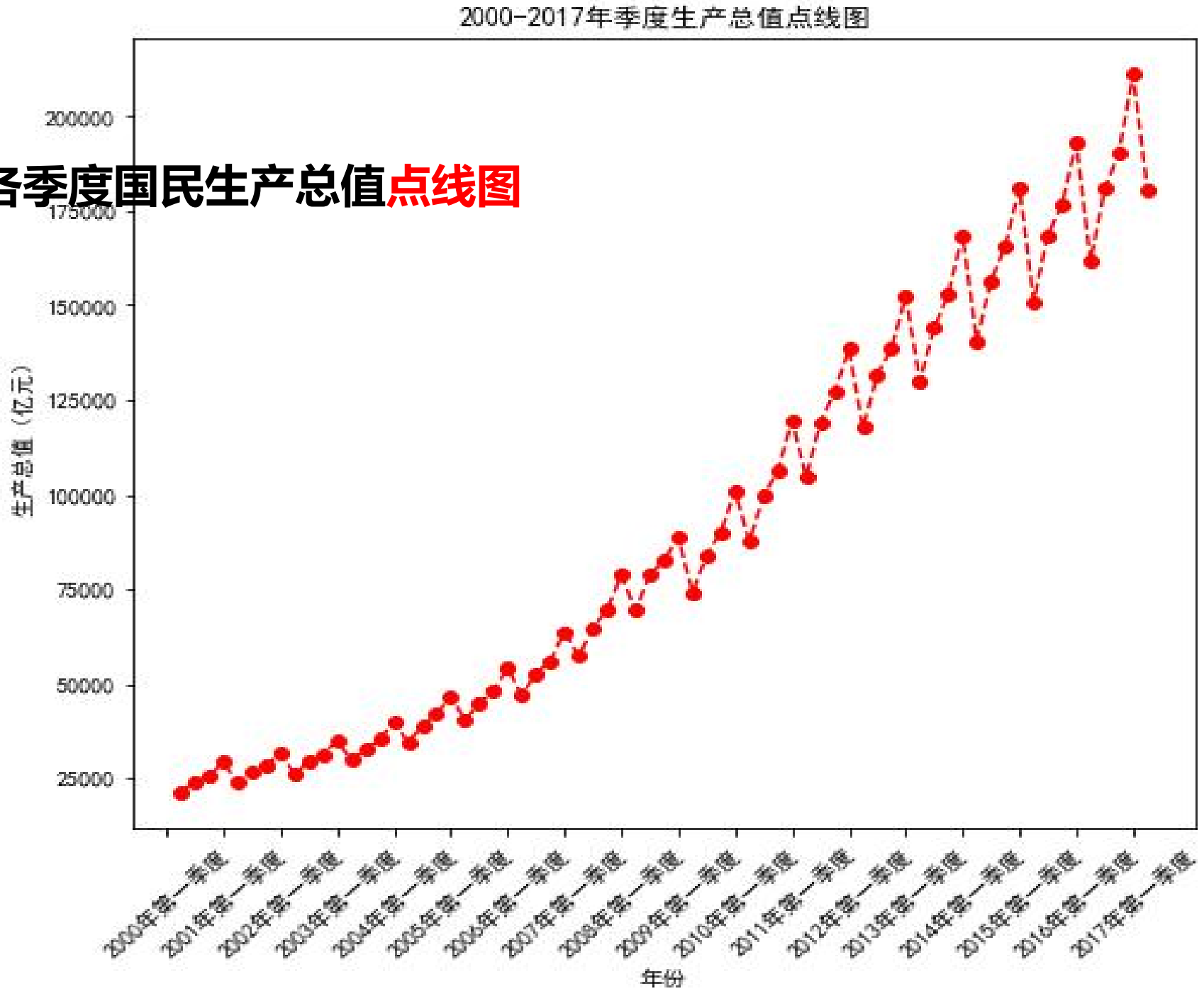
示例：绘制2000-2017年各季度国民生产总值点线图

- 使用marker参数可以绘制点线图，能够使图形更加丰富。

```
plt.figure(figsize=(8,7))## 设置画布
plt.plot(values[:,0],values[:,2],color = 'r',linestyle = '--',marker = 'o')      # 绘制折线图
plt.xlabel('年份')                                                               # 添加横轴标签
plt.ylabel('生产总值（亿元）')                                                    # 添加y轴名称
plt.xticks(range(0,70,4),values[range(0,70,4),1],rotation=45)
plt.title('2000-2017年季度生产总值点线图')                                        # 添加图表标题
plt.savefig('../tmp/2000-2017年季度生产总值点线图.png')
plt.show()
```

绘制折线图

示例：绘制2000-2017年各季度国民生产总值**点线图**



绘制折线图

示例：绘制2000-2017年各季度国民生产总值点线图

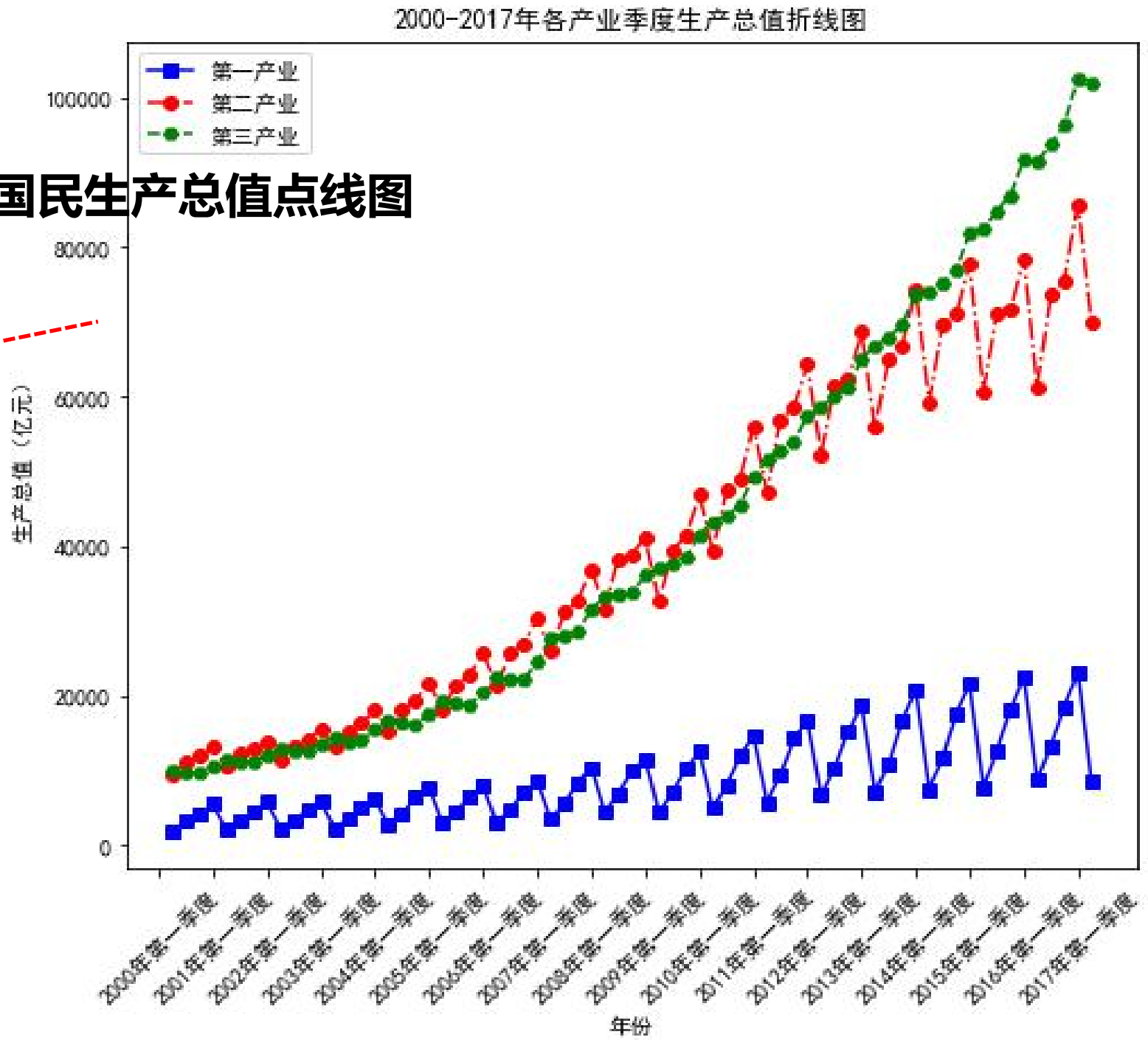
```
plt.figure(figsize=(8,7))                    # 设置画布
plt.plot(values[:,0],values[:,3],'bs-',
        values[:,0],values[:,4],'ro-.',
        values[:,0],values[:,5],'gH--')      # 绘制折线图
plt.xlabel('年份')                           # 添加横轴标签
plt.ylabel('生产总值（亿元）')               # 添加y轴名称
plt.xticks(range(0,70,4),values[range(0,70,4),1],rotation=45)
plt.title('2000-2017年各产业季度生产总值折线图')     # 添加图表标题
plt.legend(['第一产业','第二产业','第三产业'])       #添加图例
plt.savefig('../tmp/2000-2017年季度各产业生产总值折线图.png')
plt.show()
```

- plot函数一次可以接收多组数据，添加多条折线图
- 同时可以分别定义每条折线的颜色、点的形状和类型
- 还可以将这3个参数连接在一起，用一个字符串表示。

绘制折线图

示例：绘制2000-2017年各季度国民生产总值点线图

- 通过这张折线图看出，我国现阶段国民经济增长的主要动力是第三产业，其次是第二产业。



任务实现

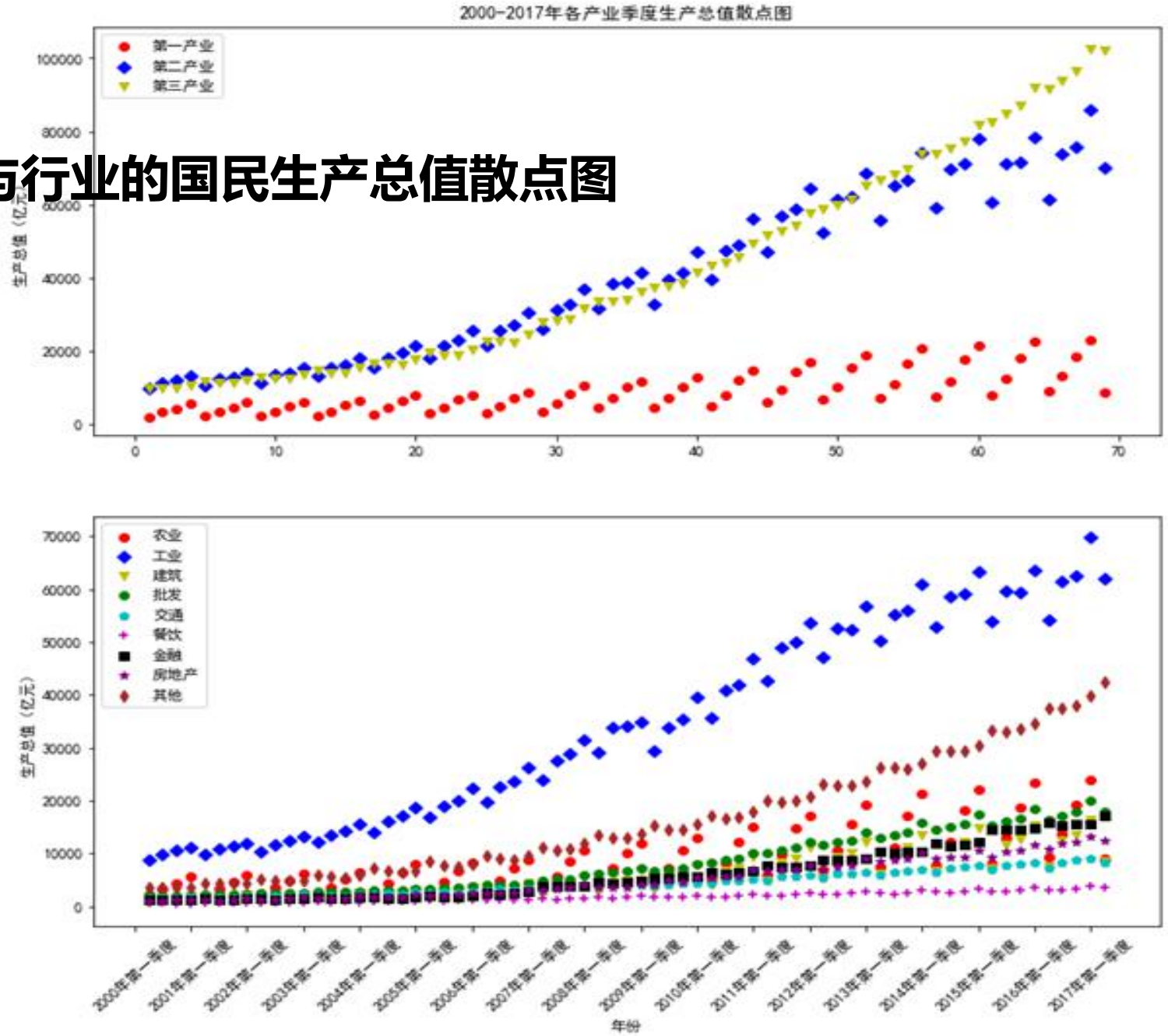
1. 绘制2000-2017各产业与行业的国民生产总值散点图

- 国民生产总值数据总共有三大产业的国民生产总值，以及农业、工业、建筑、批发、交通、餐饮、金融、房地产和其他行业各个季度的增加值。

- 通过散点图分析三大行业的国民生产总值可以发现我国产业结构。
- 通过比较各行业间季度的增加值则可以发现国民经济的主要贡献行业。

任务实现

1. 绘制2000-2017各产业与行业的国民生产总值散点图



任务实现

1. 绘制2000-2017各产业与行业的国民生产总值散点图

```
import numpy as np
import matplotlib.pyplot as plt
plt.rcParams['font.sans-serif'] = 'SimHei'              # 设置中文显示
plt.rcParams['axes.unicode_minus'] = False              # 设置负号显示
data = np.load('../data/国民经济核算季度数据.npz')  # 读取数据文件
name = data['columns']                                  # 提取其中的columns数组，视为数据的标签
values = data['values']                                 # 提取其中的values数组，数据的存在位置
p = plt.figure(figsize=(12,12))                         #设置画布
```

任务实现

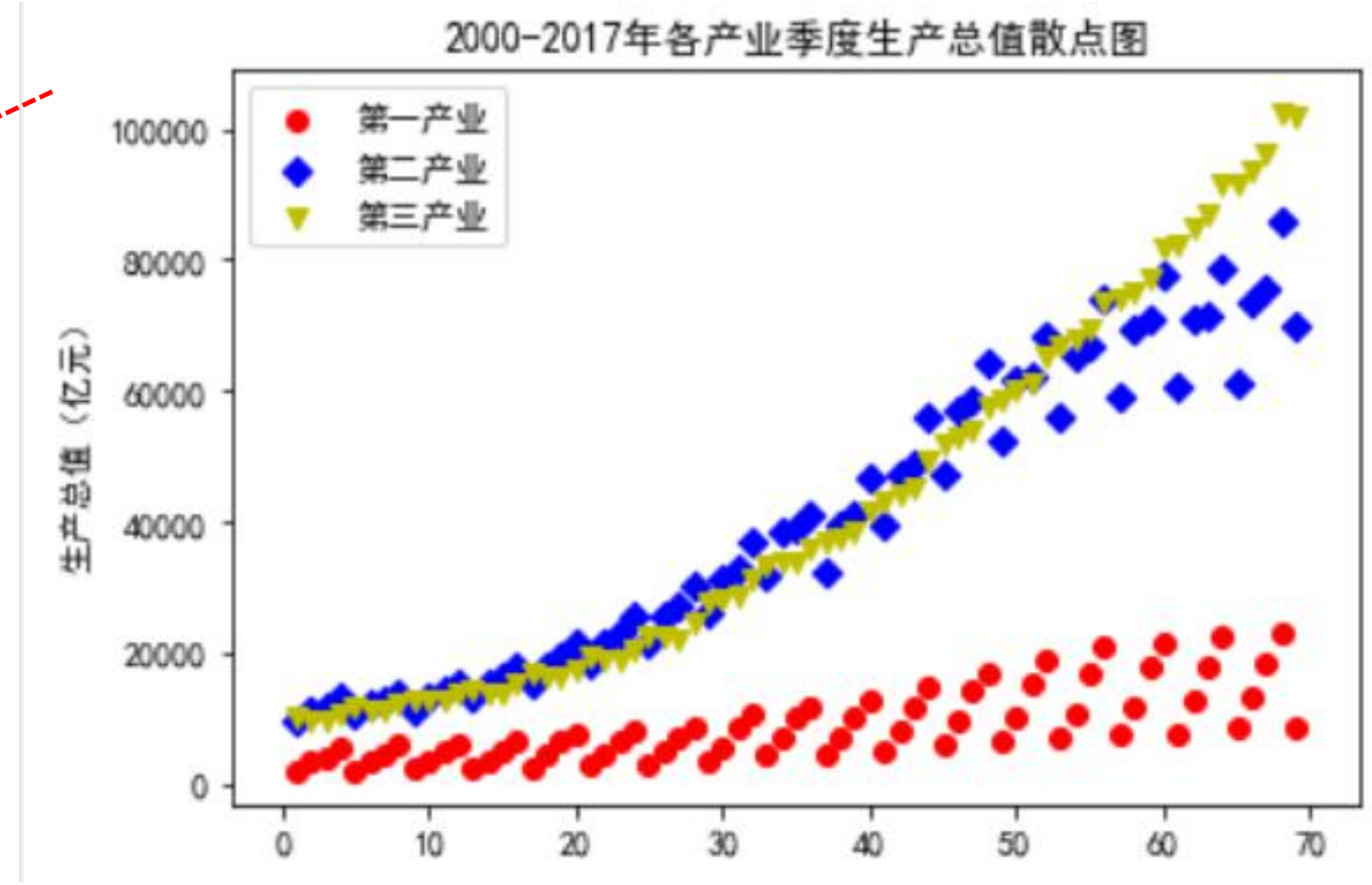
1. 绘制2000-2017各产业与行业的国民生产总值散点图

```
## 子图1
ax1 = p.add_subplot(2,1,1)                                  #创建2行1列的子图，并绘制第1幅
plt.scatter(values[:,0],values[:,3], marker='o',c='r')      # 绘制散点
plt.scatter(values[:,0],values[:,4], marker='D',c='b')      # 绘制散点
plt.scatter(values[:,0],values[:,5], marker='v',c='y')      # 绘制散点
plt.ylabel('生产总值（亿元）')                                # 添加纵轴标签
plt.title('2000-2017年各产业季度生产总值散点图')               # 添加图表标题
plt.legend(['第一产业','第二产业','第三产业'])                 # 添加图例
```

任务实现

1. 绘制2000-2017各产业与行业的国民生产总值散点图

- 通过这张散点图看出，我国现阶段国民经济增长的主要动力是第三产业，其次是第二产业。



任务实现

1.绘制2000-2017各产业与行业的国民生产总值散点图

```
## 子图2
ax2 = p.add_subplot(2,1,2)
plt.scatter(values[:,0],values[:,6], marker='o',c='r')          # 绘制散点
plt.scatter(values[:,0],values[:,7], marker='D',c='b')          # 绘制散点
plt.scatter(values[:,0],values[:,8], marker='v',c='y')          # 绘制散点
plt.scatter(values[:,0],values[:,9], marker='8',c='g')          # 绘制散点
plt.scatter(values[:,0],values[:,10], marker='p',c='c')         # 绘制散点
plt.scatter(values[:,0],values[:,11], marker='+',c='m')         # 绘制散点
plt.scatter(values[:,0],values[:,12], marker='s',c='k')         # 绘制散点
```

任务实现

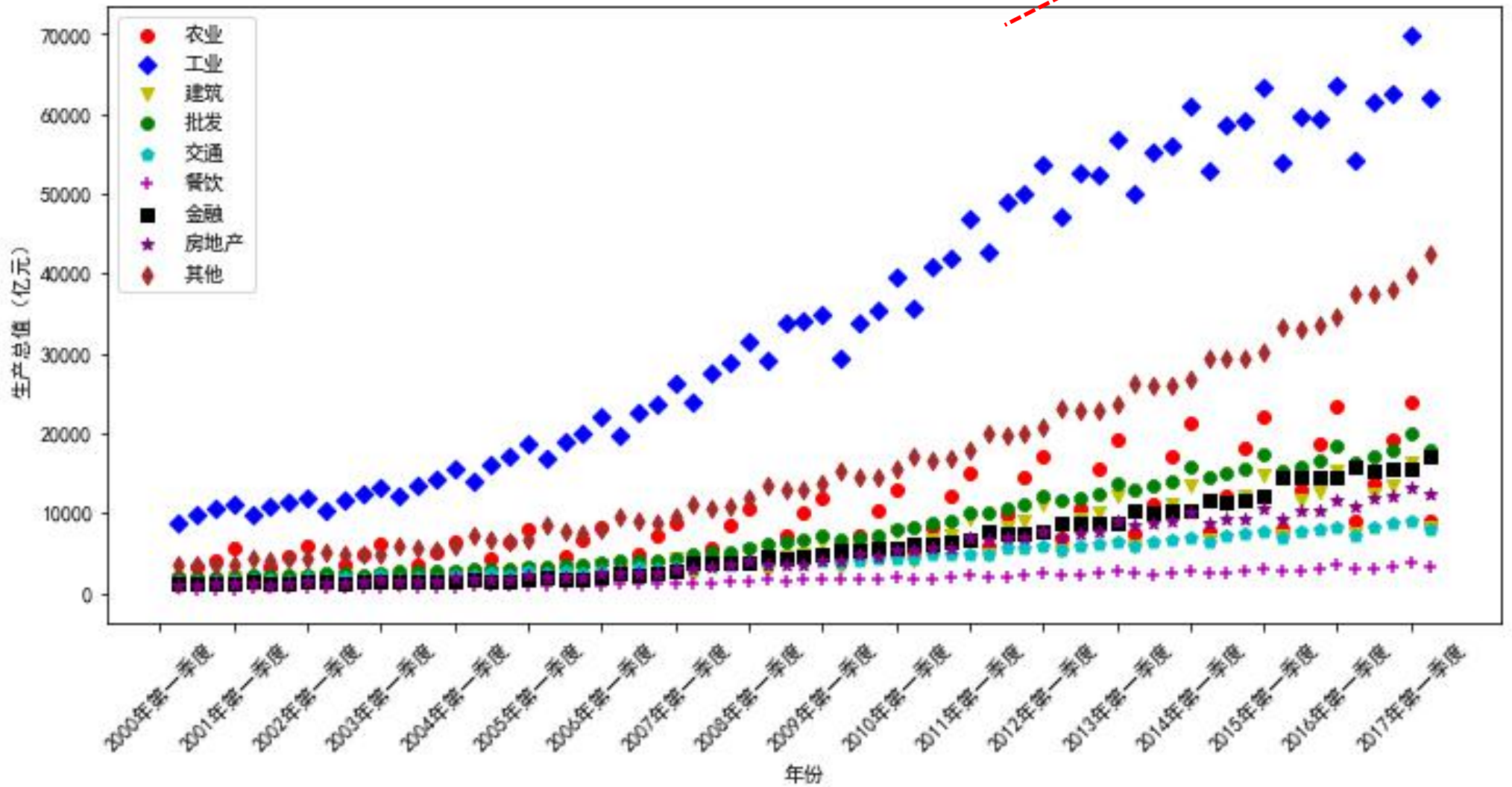
1.绘制2000-2017各产业与行业的国民生产总值散点图

```
plt.scatter(values[:,0],values[:,13], marker='*',c='purple')          # 绘制散点
plt.scatter(values[:,0],values[:,14], marker='d',c='brown')           # 绘制散点
plt.legend(['农业','工业','建筑','批发','交通', '餐饮','金融','房地产','其他'])
plt.xlabel('年份')                                                    # 添加横轴标签
plt.ylabel('生产总值（亿元）')                                        # 添加纵轴标签
plt.xticks(range(0,70,4),values[range(0,70,4),1],rotation=45)
plt.savefig('../tmp/2000-2017年季度各行业生产总值散点子图.png')
plt.show()
```

任务实现

1. 绘制2000-2017各产业与行业的国民生产总值散点图

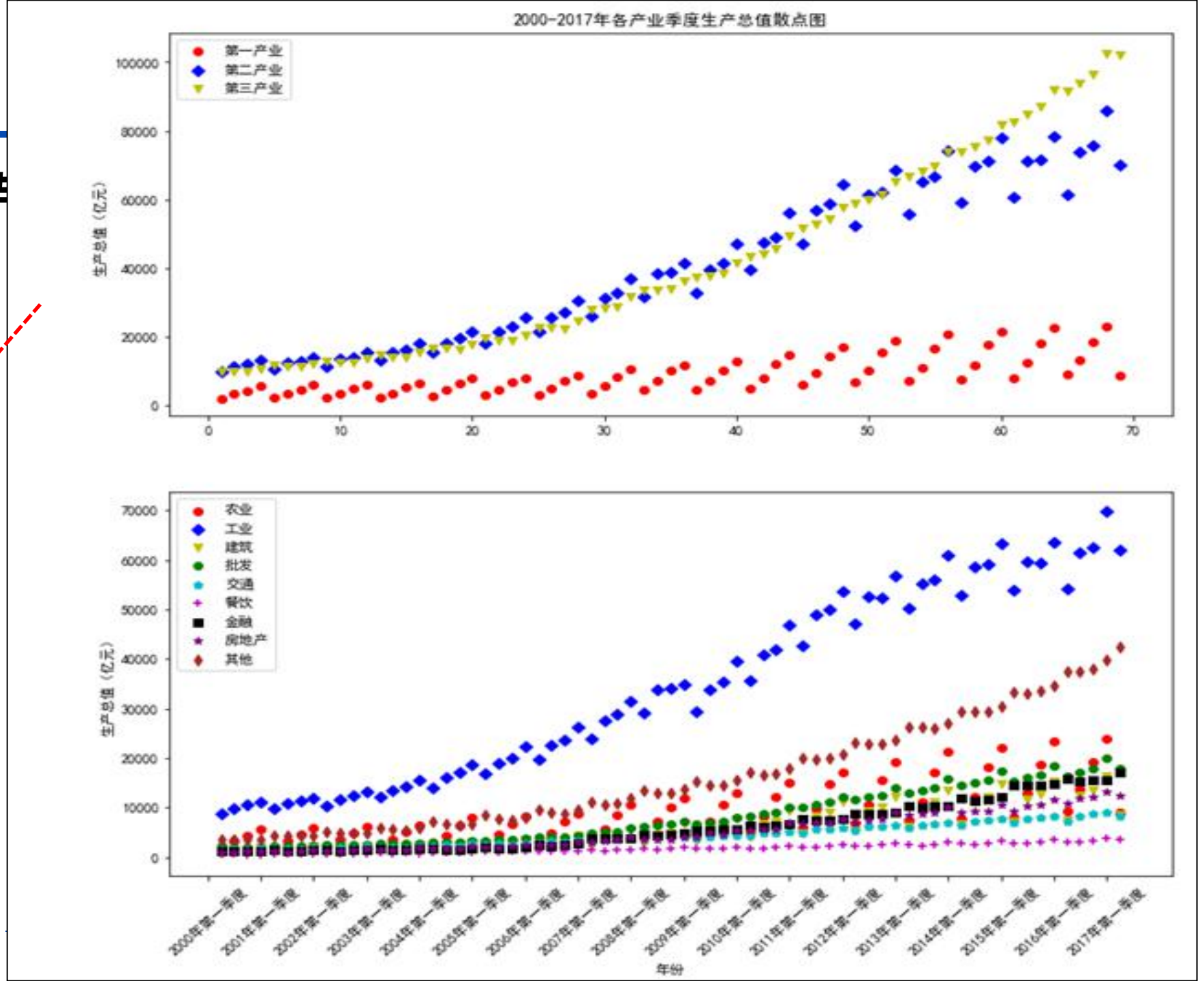
- 从行业来看，工业、其他行业和农业对整体国民经济贡献度最大。



任务实现

1.绘制2000-2017各产业与

- 通过这个散点图可以看出，我国现阶段国民经济增长的主要动力是第三产业，其次是第二产业。
- 从行业来看，工业、其他行业和农业对整体国民经济贡献度最大。



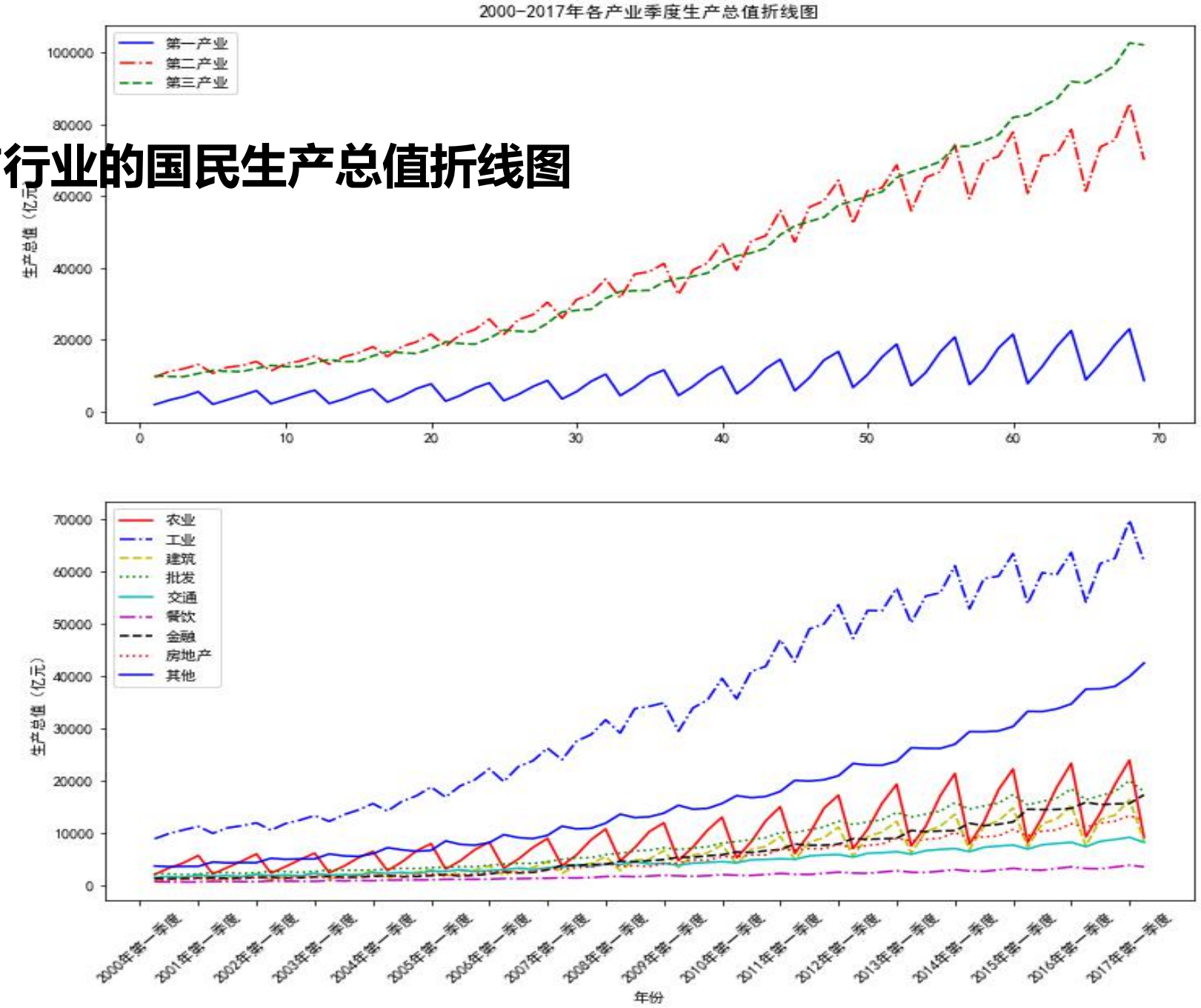
任务实现

2. 绘制2000-2017各产业与行业的国民生产总值折线图

- 通过绘制2000-2017各产业与行业的国民生产总值折线图，分别能够发现我国经济各产业与各行业增长趋势。

任务实现

2. 绘制2000-2017各产业与行业的国民生产总值折线图



任务实现

2. 绘制2000-2017各产业与行业的国民生产总值折线图

```
import numpy as np
import matplotlib.pyplot as plt
plt.rcParams['font.sans-serif'] = 'SimHei'              # 设置中文显示
plt.rcParams['axes.unicode_minus'] = False              # 设置负号显示
data = np.load('../data/国民经济核算季度数据.npz')
name = data['columns']                                  # 提取其中的columns数组，视为数据的标签
values = data['values']                                 # 提取其中的values数组，数据的存在位置
p = plt.figure(figsize=(8,7))                           #设置画布
```

任务实现

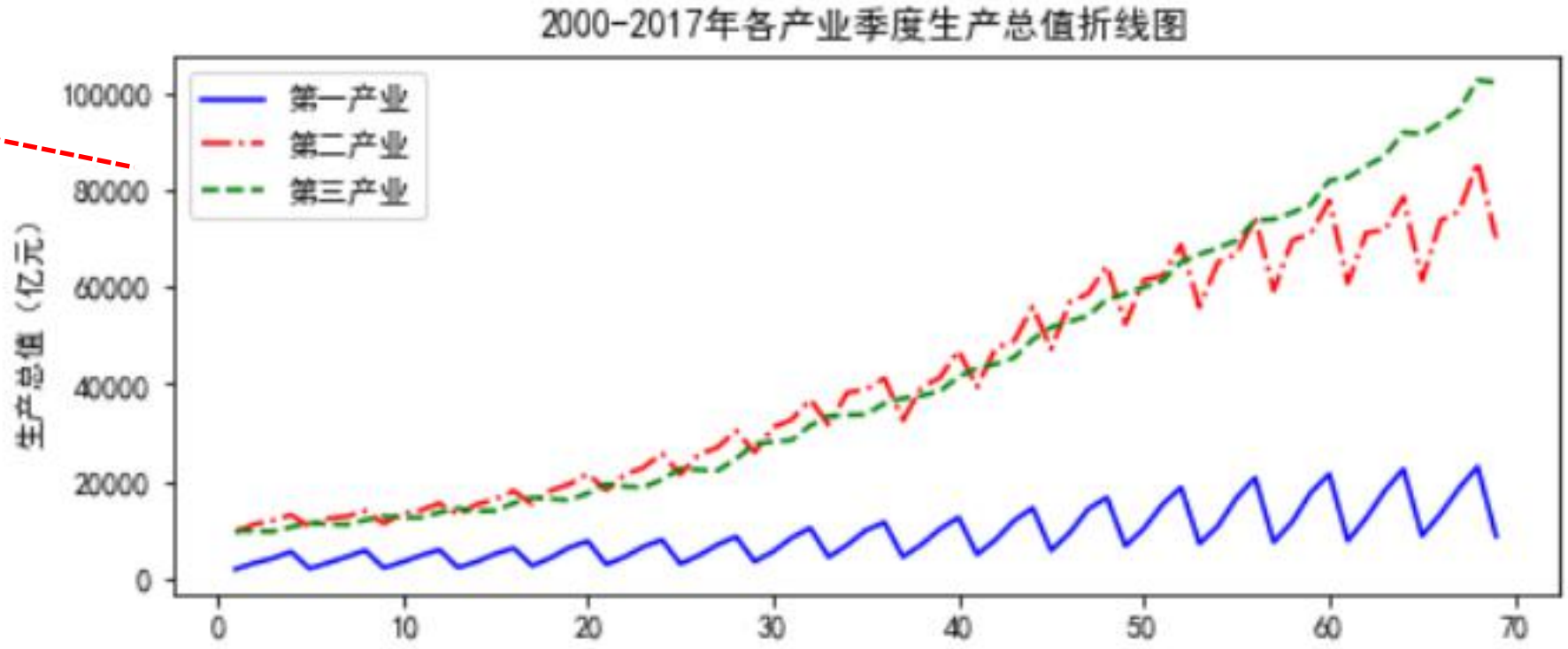
2. 绘制2000-2017各产业与行业的国民生产总值折线图

```
## 子图1
ax3 = p1.add_subplot(2,1,1)                          #创建2行1列的子图，并绘制第1幅
plt.plot(values[:,0],values[:,3],'b-',
        values[:,0],values[:,4],'r-.',
        values[:,0],values[:,5],'g--')               # 绘制折线图
plt.ylabel('生产总值（亿元）')                          # 添加纵轴标签
plt.title('2000-2017年各产业季度生产总值折线图')          # 添加图表标题
plt.legend(['第一产业','第二产业','第三产业'])           # 添加图例
```

任务实现

2. 绘制2000-2017各产业与行业的国民生产总值折线图

- 通过折线图看出，我国整体经济呈现增长趋势，其中第一产业增长相对较慢，但是周期性最明显



任务实现

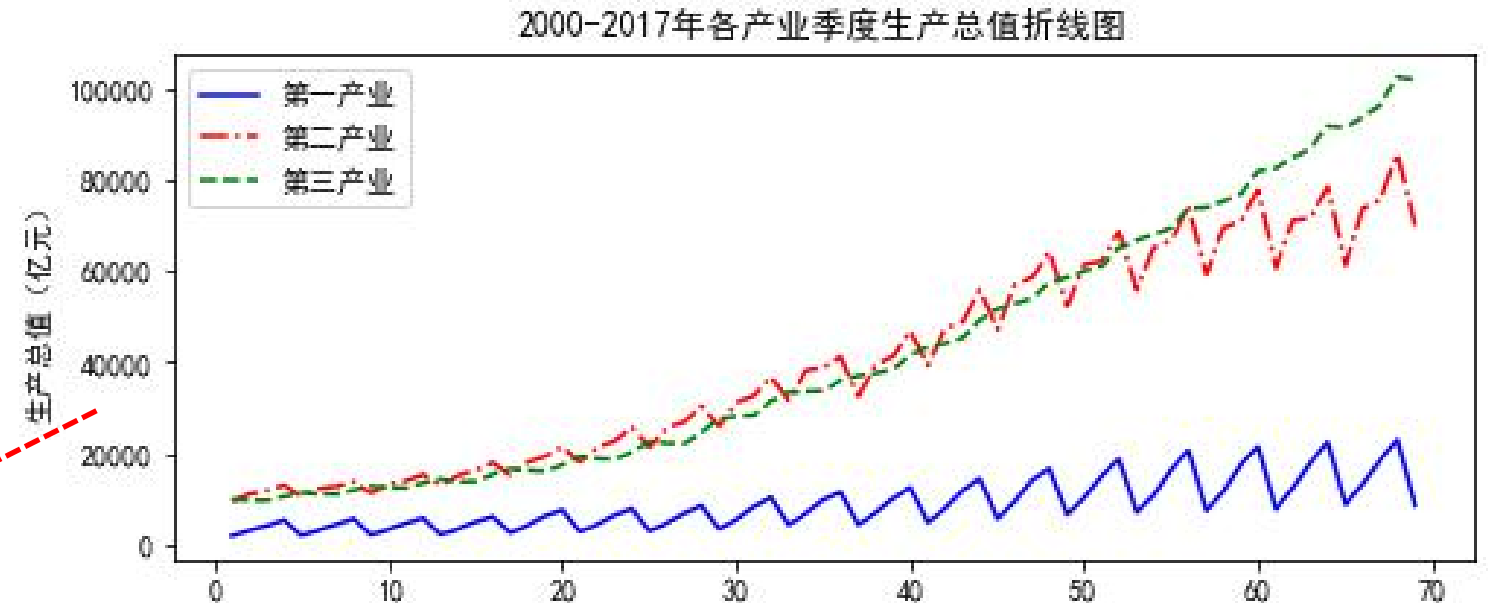
2. 绘制2000-2017各产业与行业的国民生产总值折线图

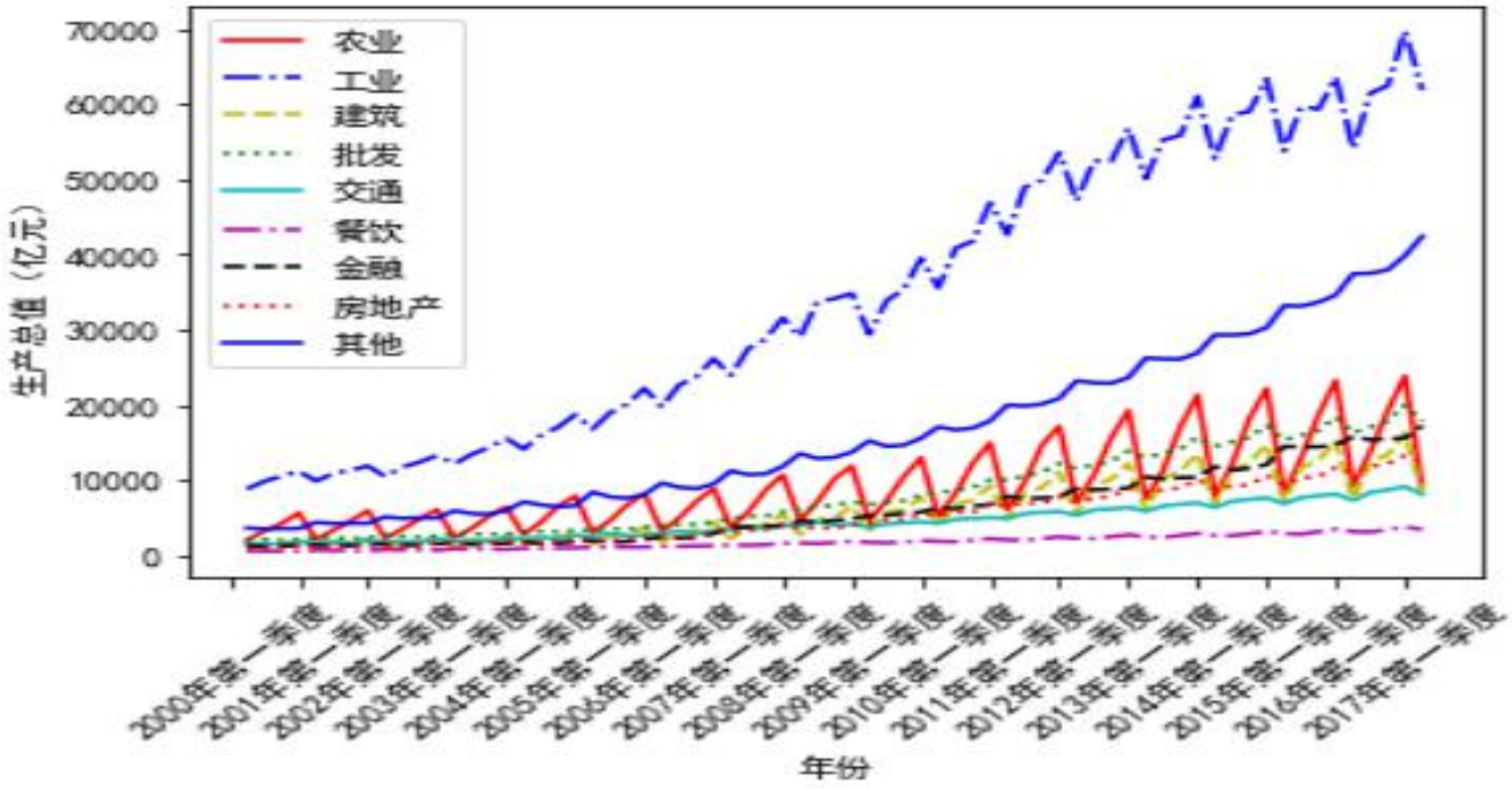
```
## 子图2
ax4 = p1.add_subplot(2,1,2)
plt.plot(values[:,0],values[:,6], 'r-', values[:,0],values[:,7], 'b-.',
        values[:,0],values[:,8],'y--', values[:,0],values[:,9], 'g:', values[:,0],values[:,10], 'c-',
        values[:,0],values[:,11], 'm-.', values[:,0],values[:,12], 'k--', values[:,0],values[:,13], 'r:',
        values[:,0],values[:,14], 'b-')                    # 绘制折线图
plt.legend(['农业','工业','建筑','批发','交通','餐饮','金融','房地产','其他'])        # 添加图例
plt.xlabel('年份')                                                                  # 添加横轴标签
plt.ylabel('生产总值（亿元）')                                                       # 添加纵轴标签
plt.xticks(range(0,70,4),values[range(0,70,4),1],rotation=45)       # 添加x轴刻度数目与取值
plt.savefig('../tmp/2000-2017年季度各行业生产总值折线子图.png')
plt.show()
```

任务实现

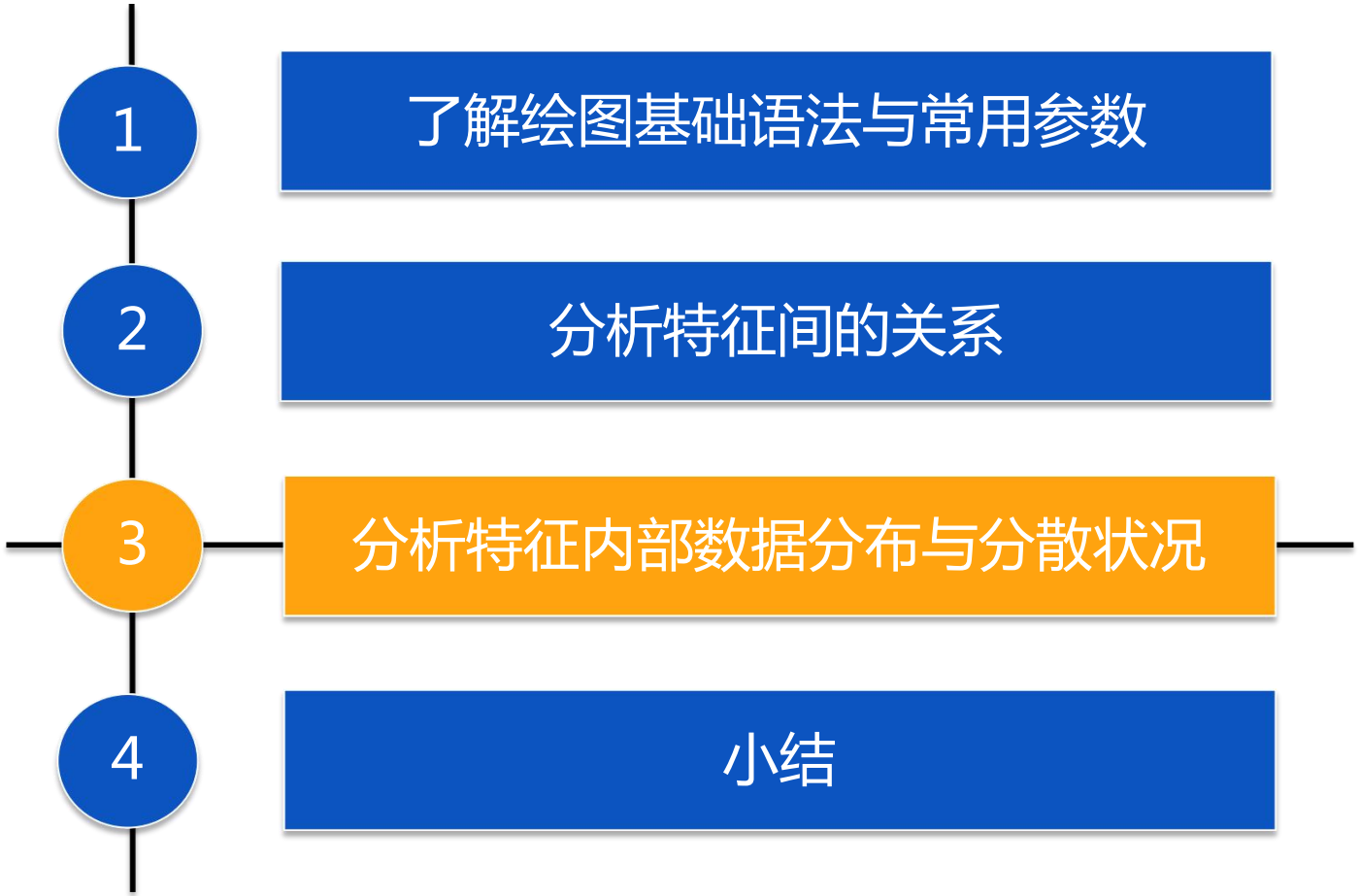
2. 绘制2000-2017各产业与行业的图

- 通过折线图看出，我国整体经济呈现增长趋势，其中第一产业增长相对较慢，但是周期性最明显
- 农业的周期性和第一产业的周期性基本吻合。
- 工业和第二产业的增长趋势基本一致。
- 同时，除了餐饮行业外，其他行业均呈现较为明显的增长趋势。



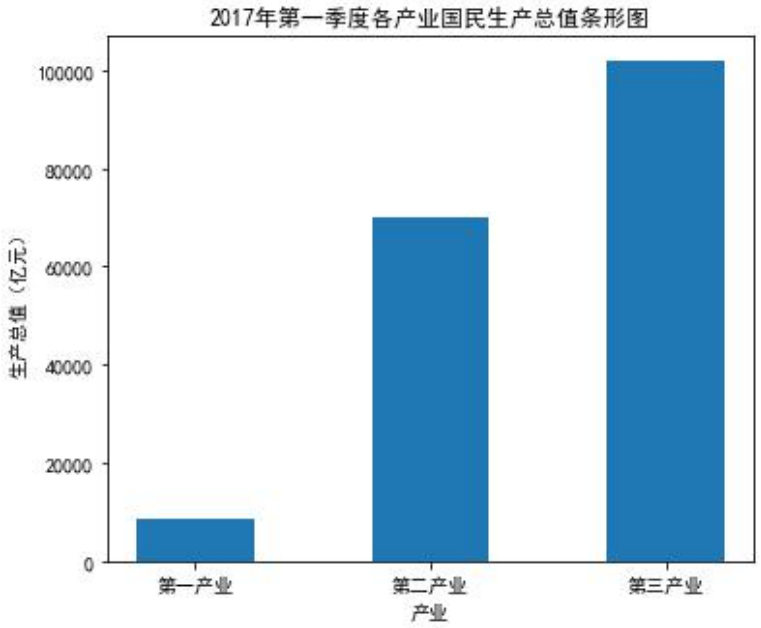


目录

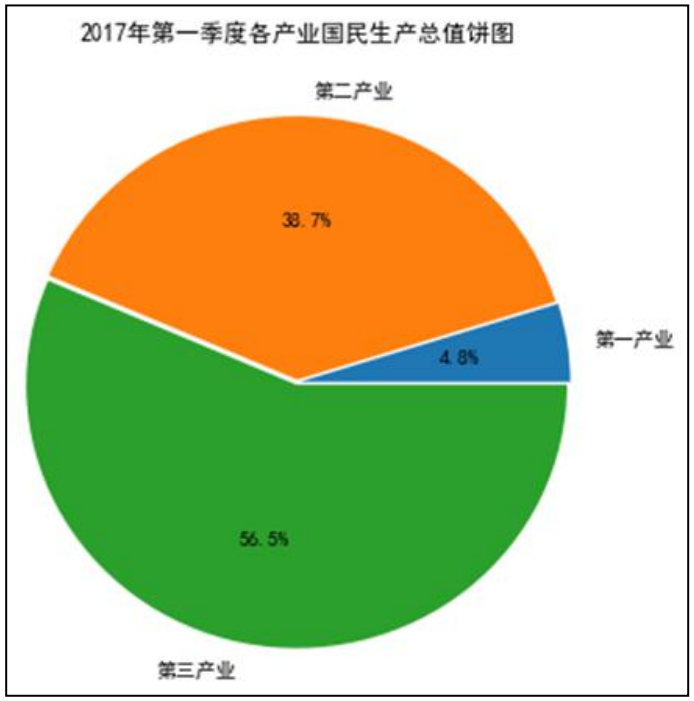


分析特征内部数据分布与分散状况

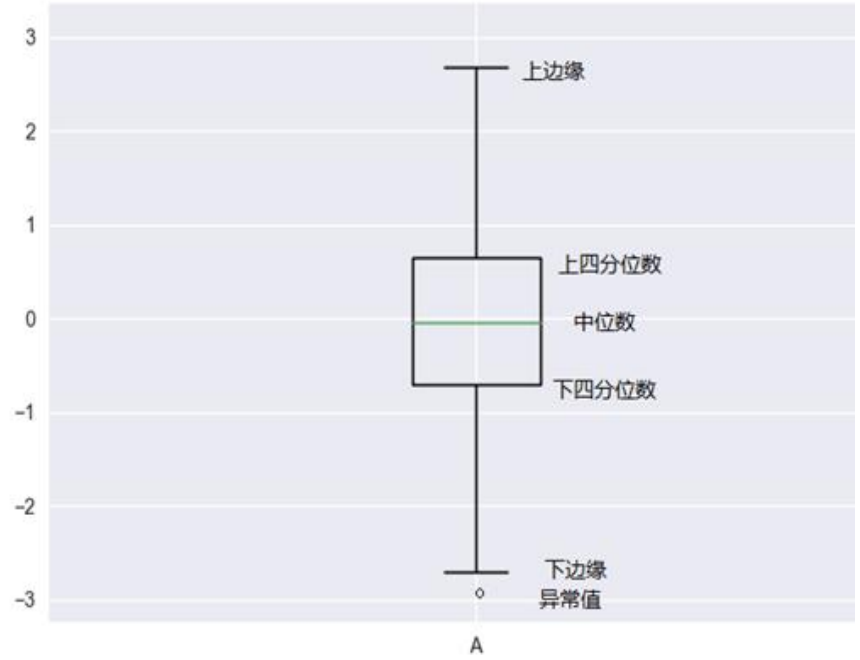
条形图



饼图



箱线图

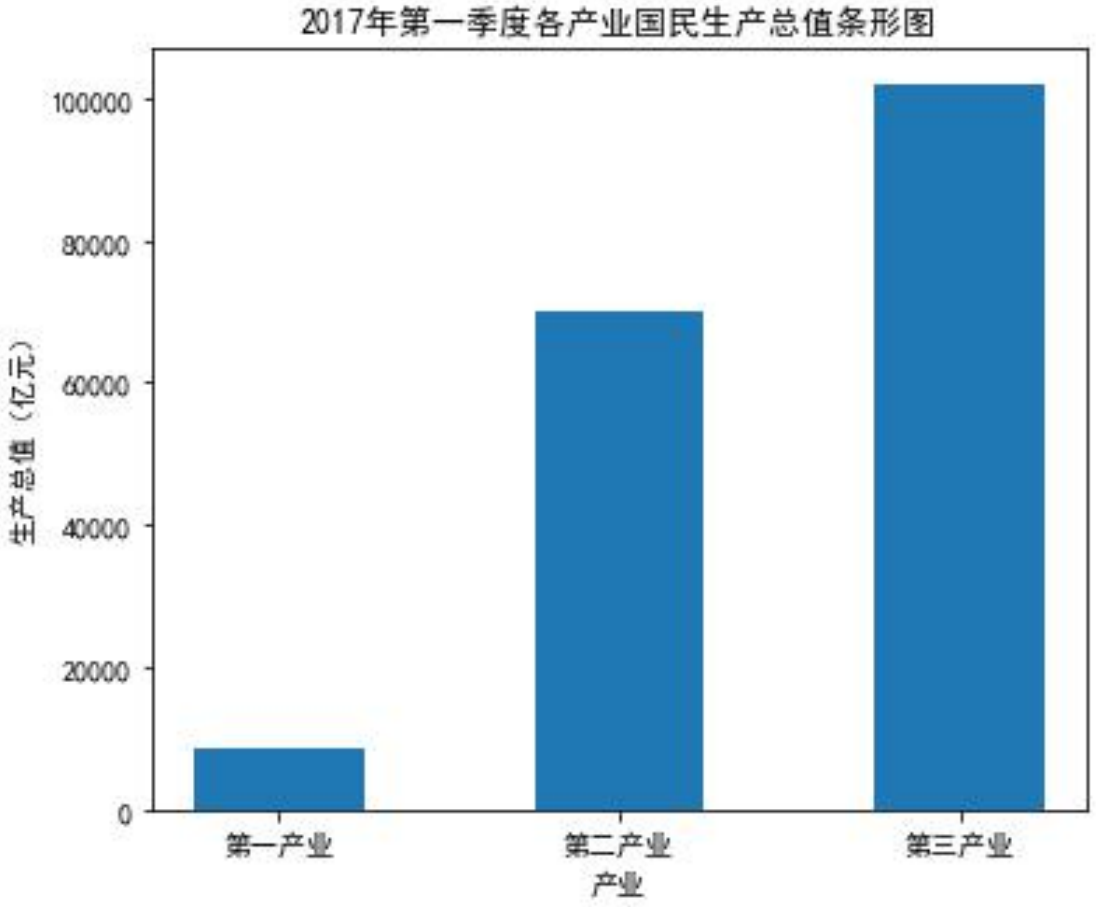


- 条形图、饼图、箱线图是另外3种数据分析常用的图形，主要用于分析数据内部分布状态和分散状态。

绘制条形图

条形图

- 条形图（bar chart），是用宽度相同的条形的高度或长短来表示数据多少的图形。
- 条形图可以横置或纵置，纵置时也称为柱形图（column chart）。
- 此外，条形图有简单条形图、复式条形图等形式。



绘制条形图

bar函数

*matplotlib.pyplot.bar(left，height，width = 0.8，bottom = None，hold = None，data = None，** kwargs)*

- 常用参数及说明如下表所示：

参数名称	说明
left	接收array。表示x轴数据。无默认。
height	接收array。表示x轴所代表数据的数量。无默认。
width	接收0-1之间的float。指定条形图宽度。默认为0.8。
color	接收特定string或者包含颜色字符串的array。表示条形图颜色。默认为None。

绘制条形图

示例：绘制2017年第一季度各产业国民生产总值条形图（柱状图）

```
import numpy as np
import matplotlib.pyplot as plt
plt.rcParams['font.sans-serif'] = 'SimHei'              # 设置中文显示
plt.rcParams['axes.unicode_minus'] = False              # 设置负号显示
data = np.load('data/国民经济核算季度数据.npz')          # 读取数据
name = data['columns']                  # 提取其中的columns数组，视为数据的标签
values = data['values']                 # 提取其中的values数组，数据的存在位置
plt.figure(figsize=(6,5))               # 设置画布
```

绘制条形图

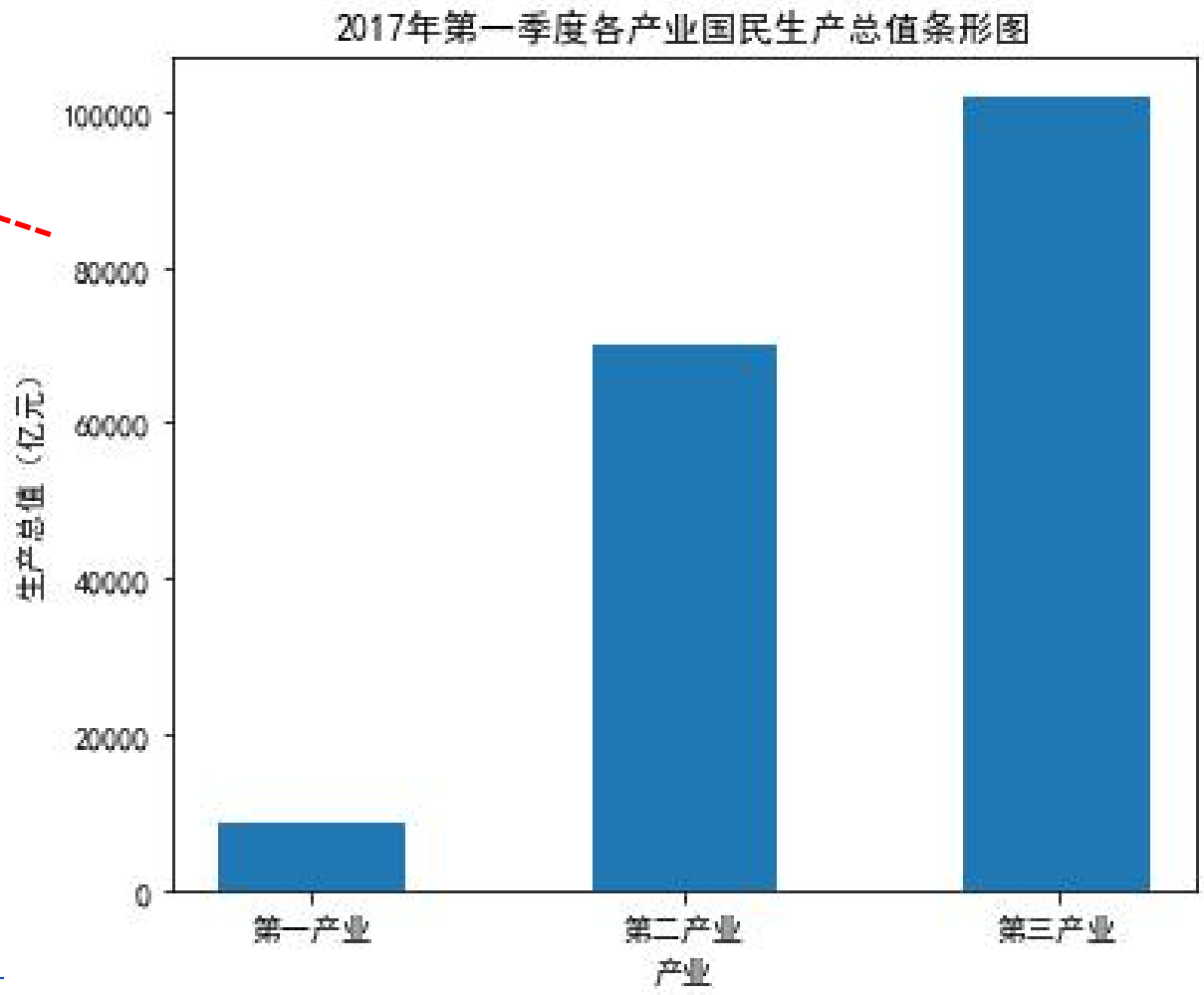
示例：绘制2017年第一季度各产业国民生产总值条形图（柱状图）

```
plt.bar(range(3),values[-1,3:6],width = 0.5)          # 绘制柱状图
plt.xlabel('产业')                                    # 添加x轴标签
plt.ylabel('生产总值（亿元）')                        # 添加y轴名称
label = ['第一产业','第二产业','第三产业']            # 刻度标签
plt.xticks(range(3),label)                            # 添加x轴刻度数目与取值
plt.title('2017年第一季度各产业国民生产总值柱状图')   # 添加图表标题
plt.savefig('../tmp/2017年第一季度各产业国民生产总值柱状图.png')
plt.show()
```

绘制条形图

示例：绘制2017年第一季度各产业国民生产总值条形图（柱状图）

- 从条形图看出，2017年第一季度的第一产业生产总值不到第二产业的六分之一，基本与第三产业的十分之一持平。
- 第二产业生产总值和第三产业生产总值相差大约三分之一。



绘制条形图

示例：绘制2017年第一季度各行业国民生产总值条形图

```
import numpy as np
import matplotlib.pyplot as plt
plt.rcParams['font.sans-serif'] = 'SimHei'              # 设置中文显示
plt.rcParams['axes.unicode_minus'] = False              # 设置负号显示
data = np.load('data/国民经济核算季度数据.npz')          # 读取数据
name = data['columns']                  # 提取其中的columns数组，视为数据的标签
values = data['values']                 # 提取其中的values数组，数据的存在位置
plt.figure(figsize=(6,5))               # 设置画布
```

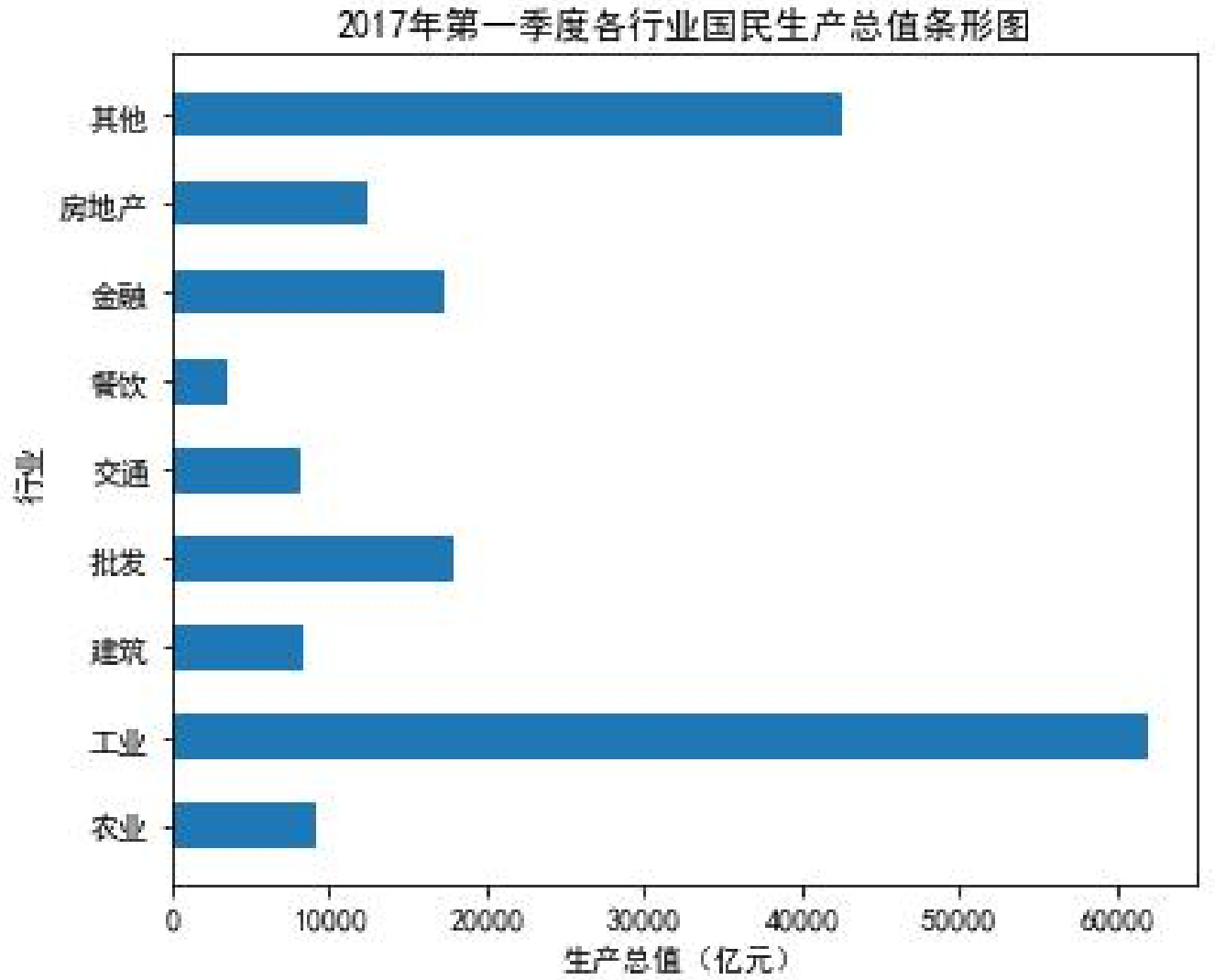
绘制条形图

示例：绘制2017年第一季度各行业国民生产总值条形图

```
plt.barh(range(9),values[-1,6:15],height = 0.5)      # 绘制条形图
plt.ylabel('行业')                                    # 添加y轴标签
plt.xlabel('生产总值（亿元）')                         # 添加x轴名称
label = ['农业','工业','建筑','批发','交通','餐饮','金融','房地产','其他']      # 刻度标签
plt.yticks(range(9),label)                           # 添加y轴刻度数目与取值
plt.title('2017年第一季度各行业国民生产总值条形图')  # 添加图表标题
plt.savefig('../tmp/2017年第一季度各行业国民生产总值条形图.png')
plt.show()
```

绘制条形图

示例：绘制2017年第一季度各行业国民生产总值条形图



绘制条形图

示例：绘制2016-2017年第一季度各行业国民生产总值对比柱状图

```
import numpy as np
import matplotlib.pyplot as plt
plt.rcParams['font.sans-serif'] = 'SimHei'              # 设置中文显示
plt.rcParams['axes.unicode_minus'] = False              # 设置负号显示
data = np.load('data/国民经济核算季度数据.npz')          # 读取数据
name = data['columns']                  # 提取其中的columns数组，视为数据的标签
values = data['values']                 # 提取其中的values数组，数据的存在位置
plt.figure(figsize=(6,5))               # 设置画布
```

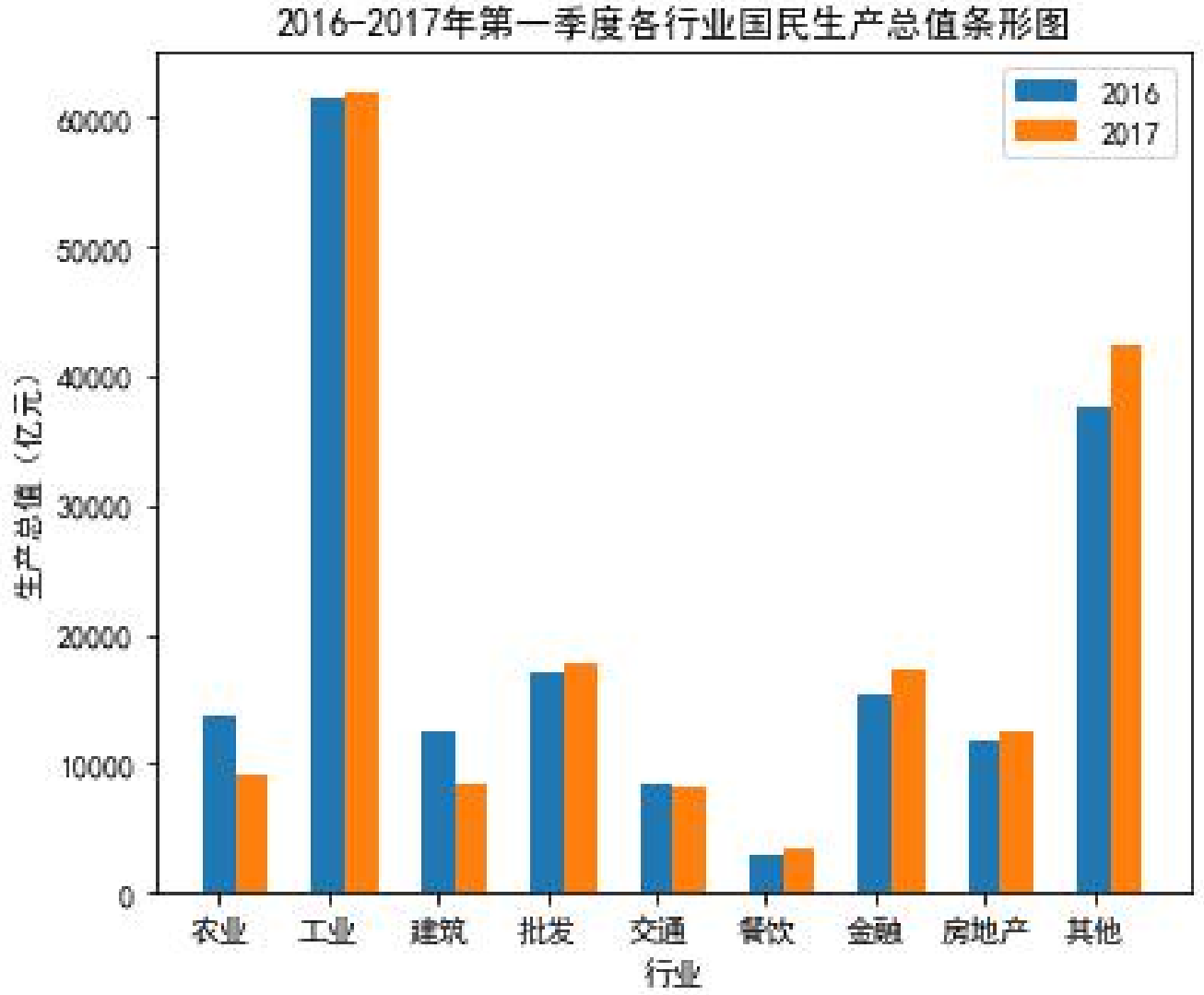
绘制条形图

示例：绘制2016-2017年第一季度各行业国民生产总值对比柱状图

```
x = range(9)
plt.bar(left=x,height=values[-4,6:15],width=0.3,label='2016')              # 绘制柱状图
plt.bar(left=[i+0.3 for i in x],height=values[-1,6:15],width=0.3,label='2017')  # 绘制柱状图
plt.xlabel('行业')                                                          # 添加x轴标签
plt.ylabel('生产总值（亿元）')                                               # 添加y轴名称
label = ['农业','工业','建筑','批发','交通','餐饮','金融','房地产','其他']   # 刻度标签
plt.xticks(range(9),label)                                                 # 添加x轴刻度数目与取值
plt.title('2016-2017年第一季度各行业国民生产总值对比柱状图')                  # 添加图表标题
plt.legend()                                                               # 添加图例
plt.savefig('2016-2017年第一季度各行业国民生产总值对比柱状图.png')
plt.show()
```

绘制条形图

示例：绘制2016-2017年第一季度各行业国民生产总值对比柱状图



绘制条形图

示例：绘制2016-2017年第一季度各行业国民生产总值堆叠柱状图

```
import numpy as np
import matplotlib.pyplot as plt
plt.rcParams['font.sans-serif'] = 'SimHei'              # 设置中文显示
plt.rcParams['axes.unicode_minus'] = False              # 设置负号显示
data = np.load('data/国民经济核算季度数据.npz')          # 读取数据
name = data['columns']                  # 提取其中的columns数组，视为数据的标签
values = data['values']                 # 提取其中的values数组，数据的存在位置
plt.figure(figsize=(6,5))               # 设置画布
```

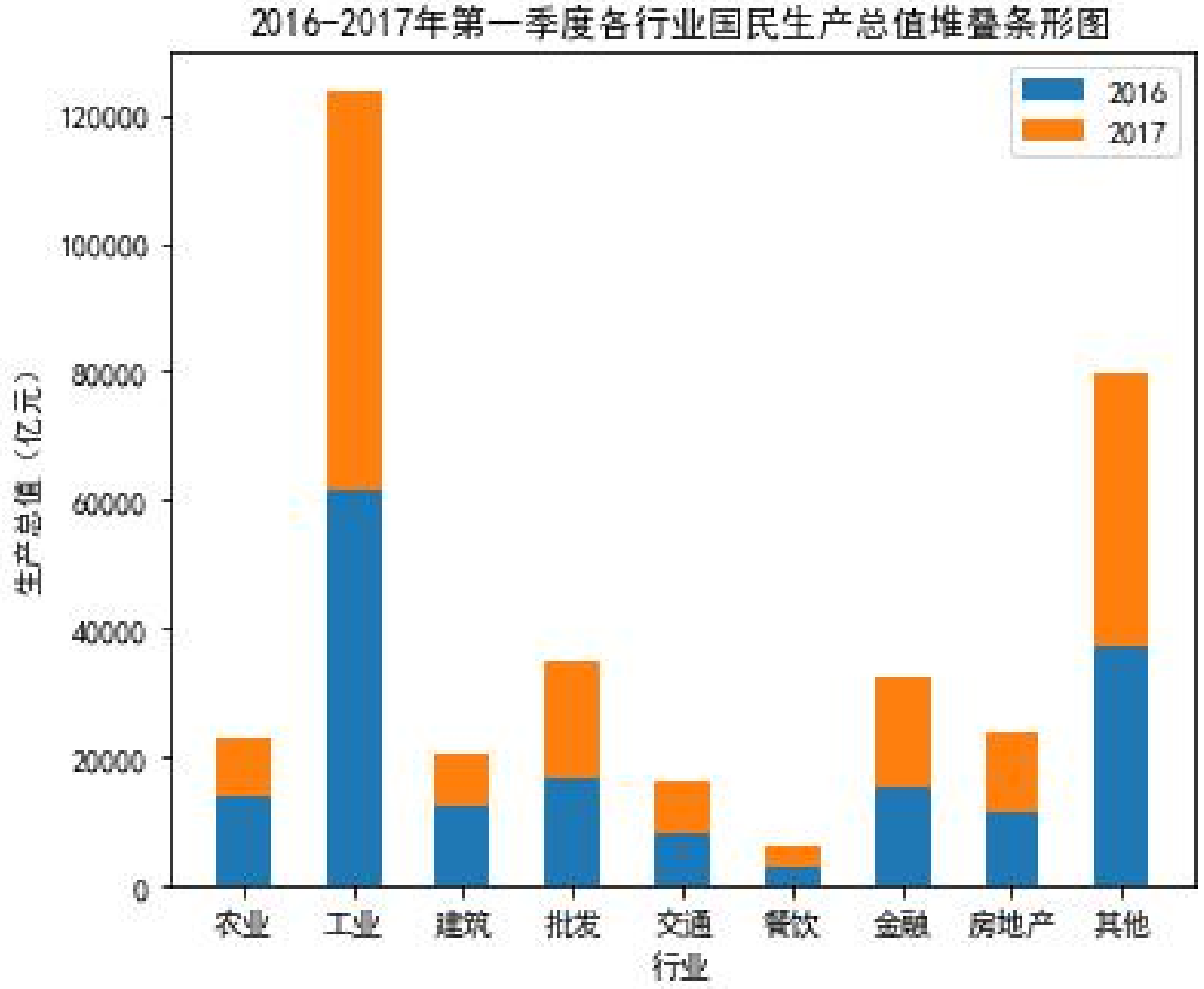
绘制条形图

示例：绘制2016-2017年第一季度各行业国民生产总值堆叠柱状图

```
x = range(9)
plt.bar(left=x,height=values[-4,6:15],width=0.3,label='2016')          # 绘制柱状图
plt.bar(left=x,height=values[-1,6:15],width=0.3,label='2017',bottom=values[-4,6:15])
plt.xlabel('行业')                                                      # 添加x轴标签
plt.ylabel('生产总值（亿元）')                                          # 添加y轴名称
label = ['农业','工业','建筑','批发','交通','餐饮','金融','房地产','其他']  # 刻度标签
plt.xticks(range(9),label)                                              # 添加x轴刻度数目与取值
plt.title('2016-2017年第一季度各行业国民生产总值堆叠柱状图')            # 添加图表标题
plt.legend()                                                            # 添加图例
plt.savefig('2016-2017年第一季度各行业国民生产总值堆叠柱状图.png')
plt.show()
```

绘制条形图

示例：绘制2016-2017年第一季度各行业国民生产总值堆叠柱状图

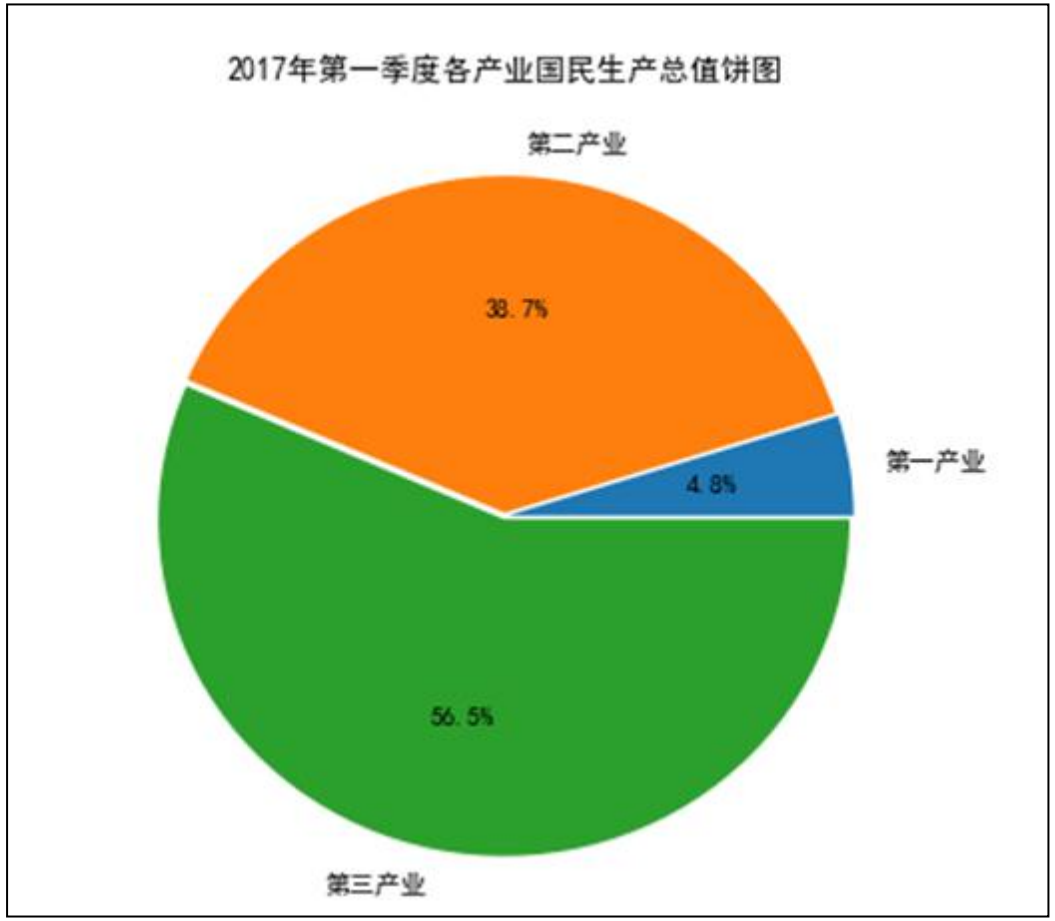


绘制饼图

饼图

- 饼图（Pie Graph）是将各项的大小与各项总和的比例显示在一张“饼”中，以“饼”的大小来确定每一项的占比。

- 饼图可以比较清楚地反映出部分与部分、部分与整体之间的比例关系，易于显示每组数据相对于总数的大小，而且显现方式直观。



绘制饼图

pie函数

matplotlib.pyplot.pie(x, explode=None, labels=None, colors=None, autopct=None, pctdistance=0.6, shadow=False, labeldistance=1.1, startangle=None, radius=None, …)

- 常用参数及说明如下表所示：

参数名称	说明	参数名称	说明
x	接收array。表示用于绘制饼图的数据。无默认。	autopct	接收特定string。指定数值的显示方式。默认为None。
explode	接收array。表示指定项离饼图圆心为n个半径。默认为None。	pctdistance	接收float。指定每一项的比例和距离饼图圆心n个半径。默认为0.6。
labels	接收array。指定每一项的名称。默认为None。	labeldistance	接收float。指定每一项的名称和距离饼图圆心多少个半径。默认为1.1。
color	接收特定string或者包含颜色字符串的array。表示饼图颜色。默认为None。	radius	接收float。表示饼图的半径。默认为1。

绘制饼图

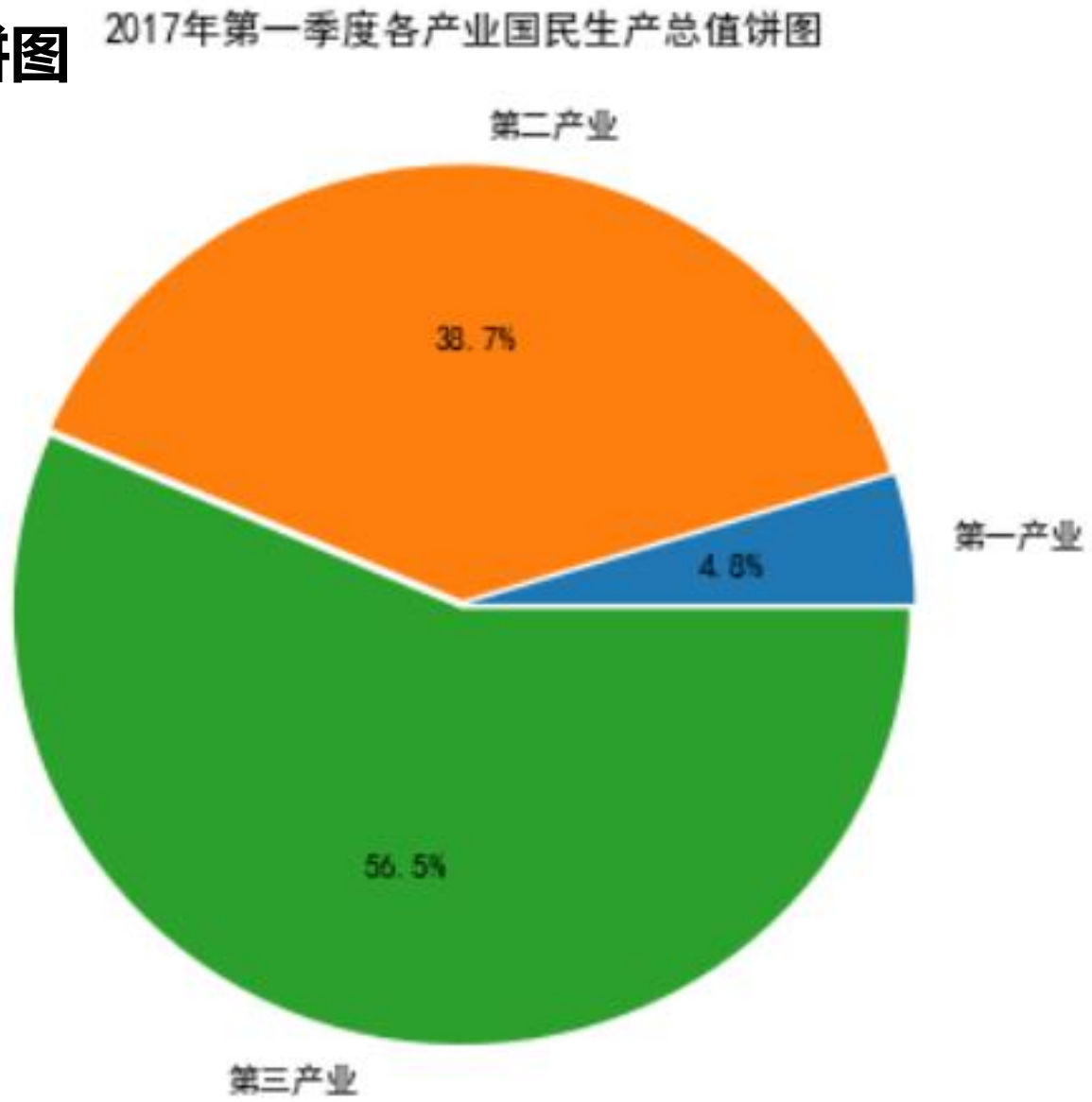
绘制2017年第一季度各产业国民生产总值饼图

```
plt.figure(figsize=(6,6))                          # 将画布设定为正方形，则绘制的饼图是正圆
label= ['第一产业','第二产业','第三产业']            # 定义饼状图的标签，标签是列表
explode = [0.01,0.01,0.01]                         # 设定各项离心n个半径
## 绘制饼图
plt.pie(values[-1,3:6],explode=explode,labels=label, autopct='%1.1f%%')
plt.title('2017年第一季度各产业国民生产总值饼图')
plt.savefig('../tmp/2017年第一季度各产业生产总值占比饼图')
plt.show()
```

绘制饼图

绘制2017年第一季度各产业国民生产总值饼图

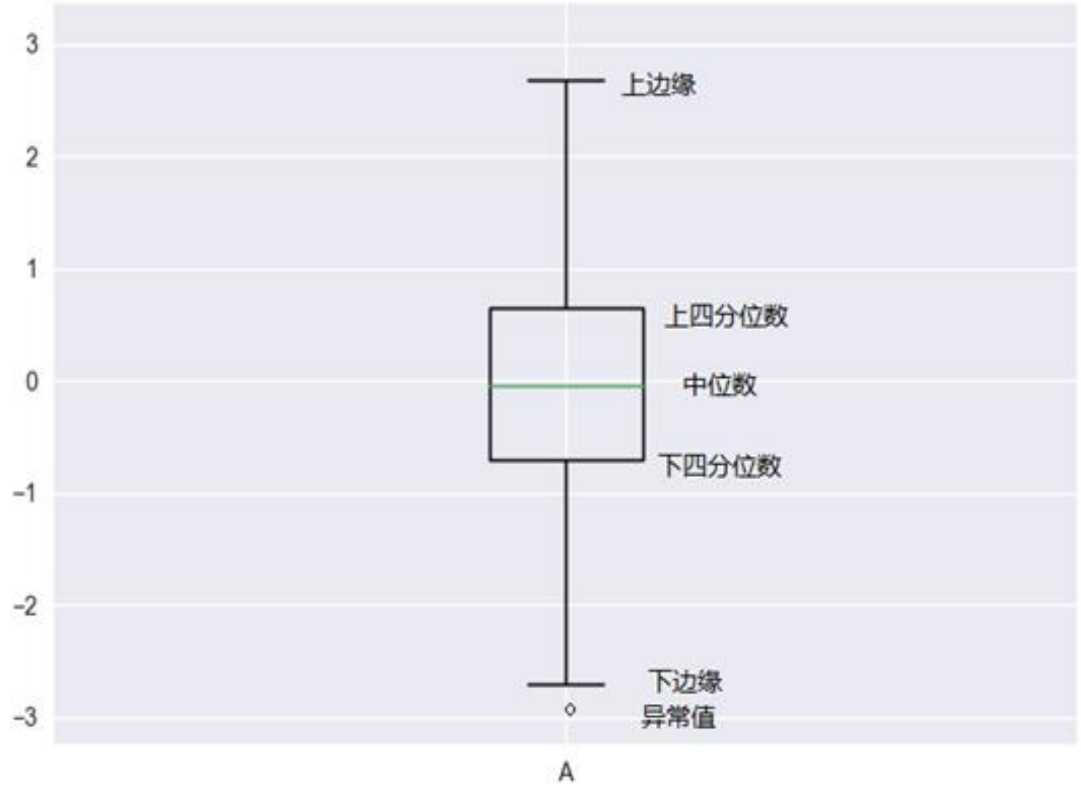
- 通过饼图可以明确看出3个产业在整个国民生产总值中的占比。
- 第一产业不到5%，第三产业超过50%
- 说明现阶段我国经济的主要贡献产业为第三产业。



绘制箱线图

箱线图

- 箱线图（boxplot）也称箱须图，其绘制需使用常用的统计量，能提供有关数据位置和分散情况的关键信息，尤其在比较不同特征时，更可表现其分散程度差异。
- 箱线图利用数据中的五个统计量（最小值、下四分位数、中位数、上四分位数和最大值）来描述数据
- 可以粗略地看出数据是否具有对称性、分布的分散程度等信息，特别可以用于对几个样本的比较。



绘制箱线图

boxplot函数

matplotlib.pyplot.boxplot(x, notch=None, sym=None, vert=None, whis=None, positions=None, widths=None, patch_artist=None,meanline=None, labels=None, …)

- 常用参数及说明如下表所示：

参数名称	说明	参数名称	说明
x	接收array。表示用于绘制箱线图的数据。无默认。	positions	接收array。表示图形位置。默认为None。
notch	接收boolean。表示中间箱体是否有缺口。默认为None。	widths	接收scalar或者array。表示每个箱体的宽度。默认为None。
sym	接收特定sting。指定异常点形状。默认为None。	labels	接收array。指定每一个箱线图的标签。默认为None。
vert	接收boolean。表示图形是纵向或者横向。默认为None。	meanline	接收boolean。表示是否显示均值线。默认为False。

绘制箱线图

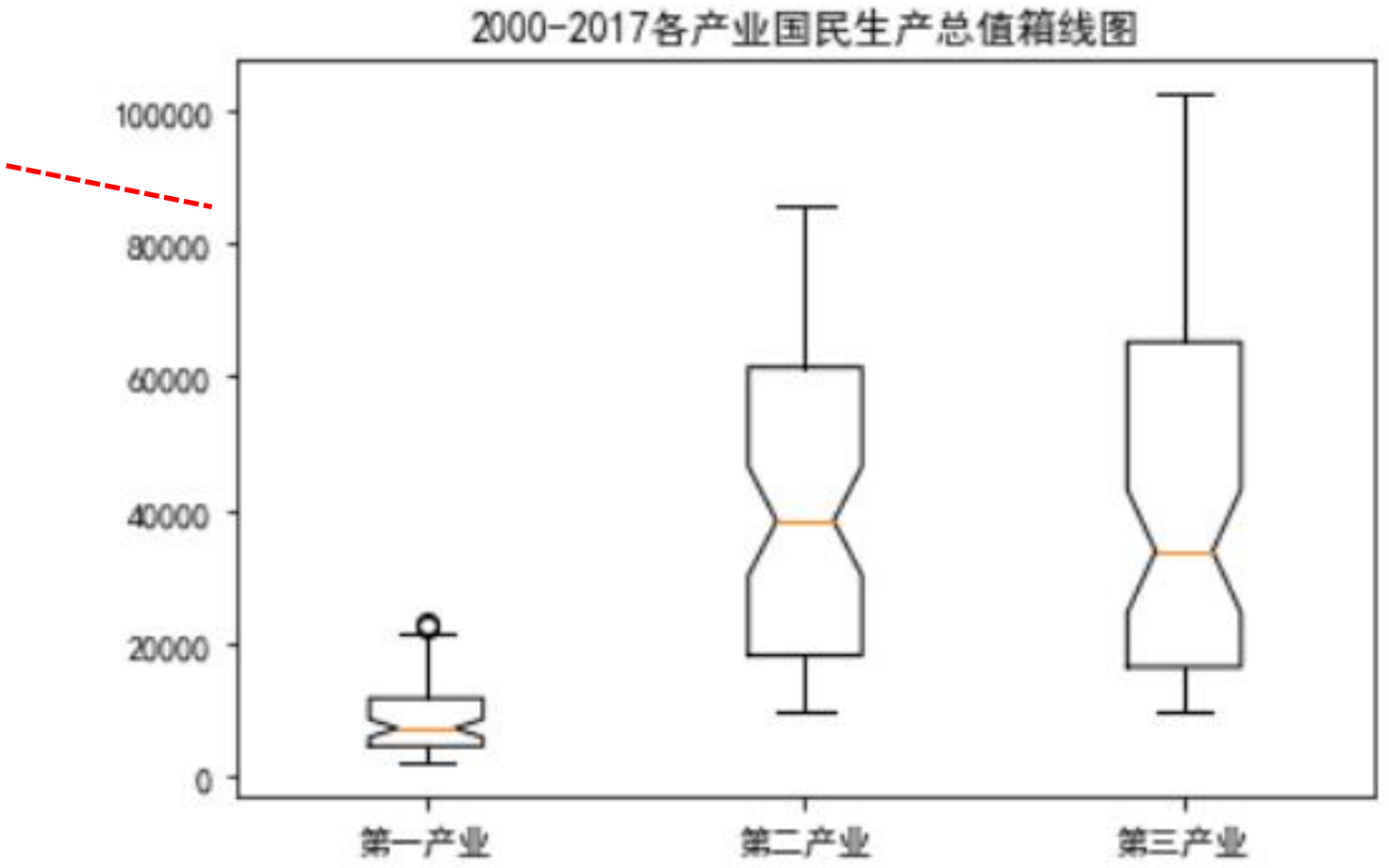
绘制2000-2017年各产业国民生产总值箱线图

```
label= ['第一产业','第二产业','第三产业']                                    # 定义标签
gdp = (list(values[:,3]),list(values[:,4]),list(values[:,5]))              # 确定数据
plt.figure(figsize=(6,4))
plt.boxplot(gdp,notch=True,labels = label, meanline=True)                  # 绘制箱线图
plt.title('2000-2017各产业国民生产总值箱线图')
plt.savefig('../tmp/2000-2017各产业国民生产总值箱线图.png')
plt.show()
```

绘制箱线图

绘制2000-2017年各产业国民生产总值箱线图

- 通过这张箱线图看出，在2000-2017年，第一产业在某一年的某个季度具有一个异常值。
- 第三产业整体增速变大，导致了第三产业数据前半部分相对密集，而后半部分相对的分散。



任务实现

1. 绘制国民生产总值构成分布柱状图

- 通过柱状图分析2000年第一季度和2017年第一季度之间的三大产业的国民生产总值，可以发现各产业绝对数值之间的关系，并通过对比发现产业结构的变化。

- 同理可以得出行业间的绝对数值关系以及17年来行业发展状况。

任务实现

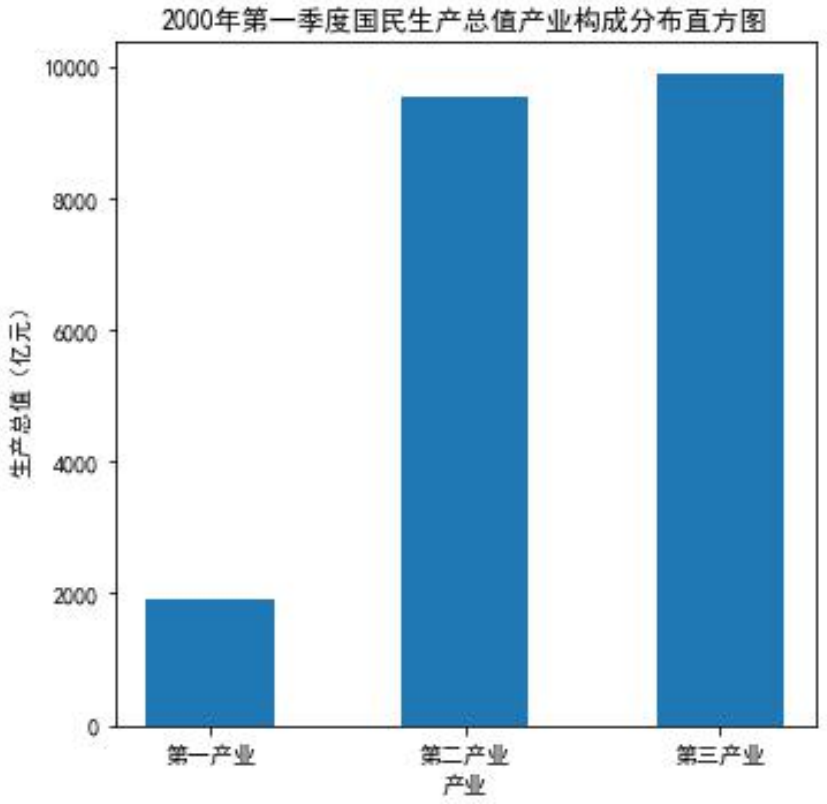
1. 绘制国民生产总值构成分布柱状图

```
import numpy as np
import matplotlib.pyplot as plt
data = np.load('data/国民经济核算季度数据.npz')
name = data['columns']                          # 提取其中的columns数组，视为数据的标签
values = data['values']                         # 提取其中的values数组，数据的存在位置
plt.rcParams['font.sans-serif'] = 'SimHei'      # 设置中文显示
plt.rcParams['axes.unicode_minus'] = False
label1 = ['第一产业','第二产业','第三产业']       # 刻度标签1
label2 = ['农业','工业','建筑','批发','交通','餐饮','金融','房地产','其他']          # 刻度标签2
p = plt.figure(figsize=(12,12))
```

任务实现

1. 绘制国民生产总值构成分布柱状图

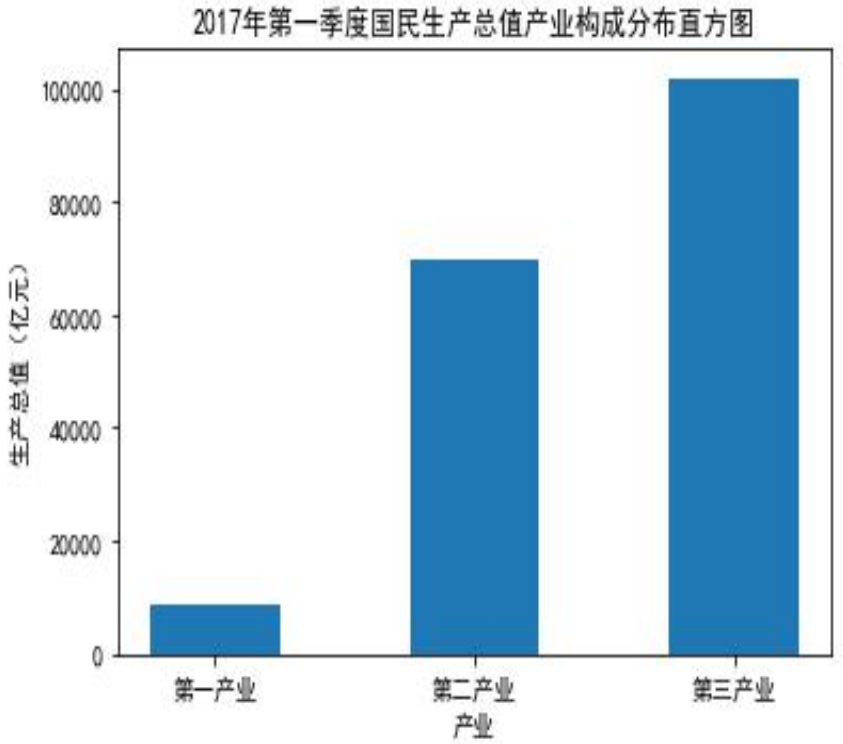
```
## 子图1
ax1 = p.add_subplot(2,2,1)
plt.bar(range(3),values[0,3:6],width = 0.5)  # 绘制柱状图
plt.xlabel('产业')                           # 添加横轴标签
plt.ylabel('生产总值（亿元）')               # 添加y轴名称
plt.xticks(range(3),label1)
## 添加图表标题
plt.title('2000年第一季度国民生产总值产业构成分布柱状图')
```



任务实现

1. 绘制国民生产总值构成分布柱状图

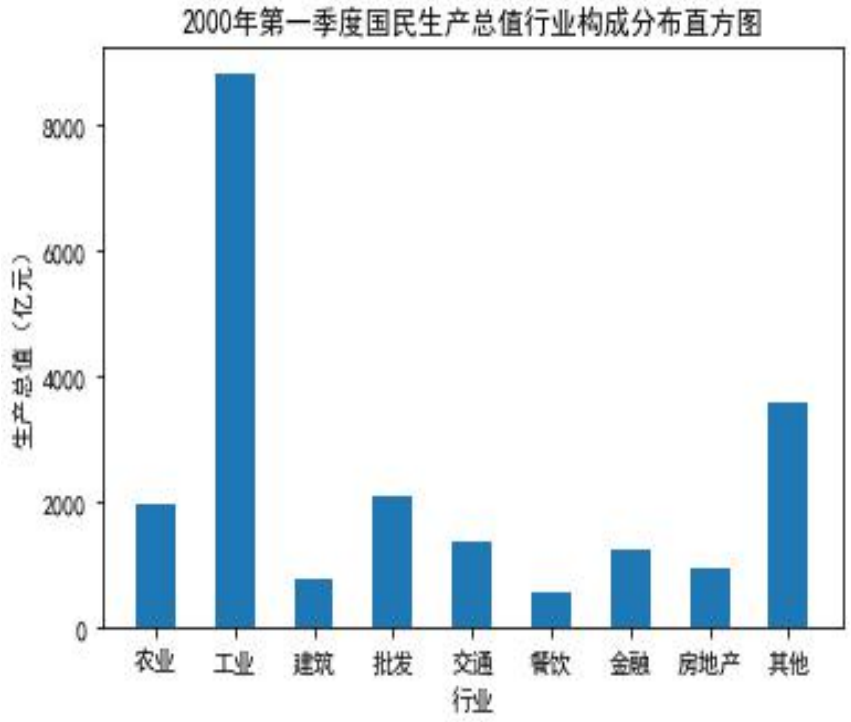
```
## 子图2
ax2 = p.add_subplot(2,2,2)
plt.bar(range(3),values[-1,3:6],width = 0.5)# 绘制柱状图
plt.xlabel('产业')                          # 添加x轴标签
plt.ylabel('生产总值（亿元）')               # 添加y轴名称
plt.xticks(range(3),label1)
## 添加图表标题
plt.title('2017年第一季度国民生产总值产业构成分布柱状图')
```



任务实现

1. 绘制国民生产总值构成分布柱状图

```
## 子图3
ax3 = p.add_subplot(2,2,3)
plt.bar(range(9),values[0,6:],width = 0.5)    # 绘制柱状图
plt.xlabel('行业')                             # 添加x轴标签
plt.ylabel('生产总值（亿元）')                  # 添加y轴名称
plt.xticks(range(9),label2)
## 添加图表标题
plt.title('2000年第一季度国民生产总值行业构成分布柱状图')
```

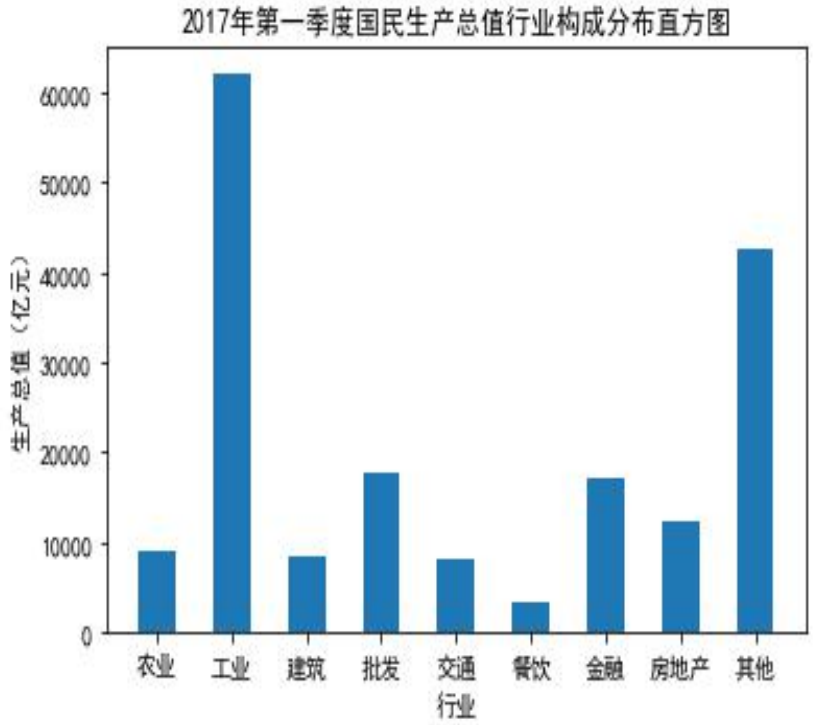


任务实现

1. 绘制国民生产总值构成分布柱状图

```
## 子图4
ax4 = p.add_subplot(2,2,4)
plt.bar(range(9),values[-1,6:],width = 0.5)  # 绘制柱状图
plt.xlabel('行业')                           # 添加x轴标签
plt.ylabel('生产总值（亿元）')                 # 添加y轴名称
plt.xticks(range(9),label2)
## 添加图表标题
plt.title('2017年第一季度国民生产总值行业构成分布柱状图')

## 保存并显示图形
plt.savefig('../tmp/国民生产总值构成分布柱状图.png')
plt.show()
```

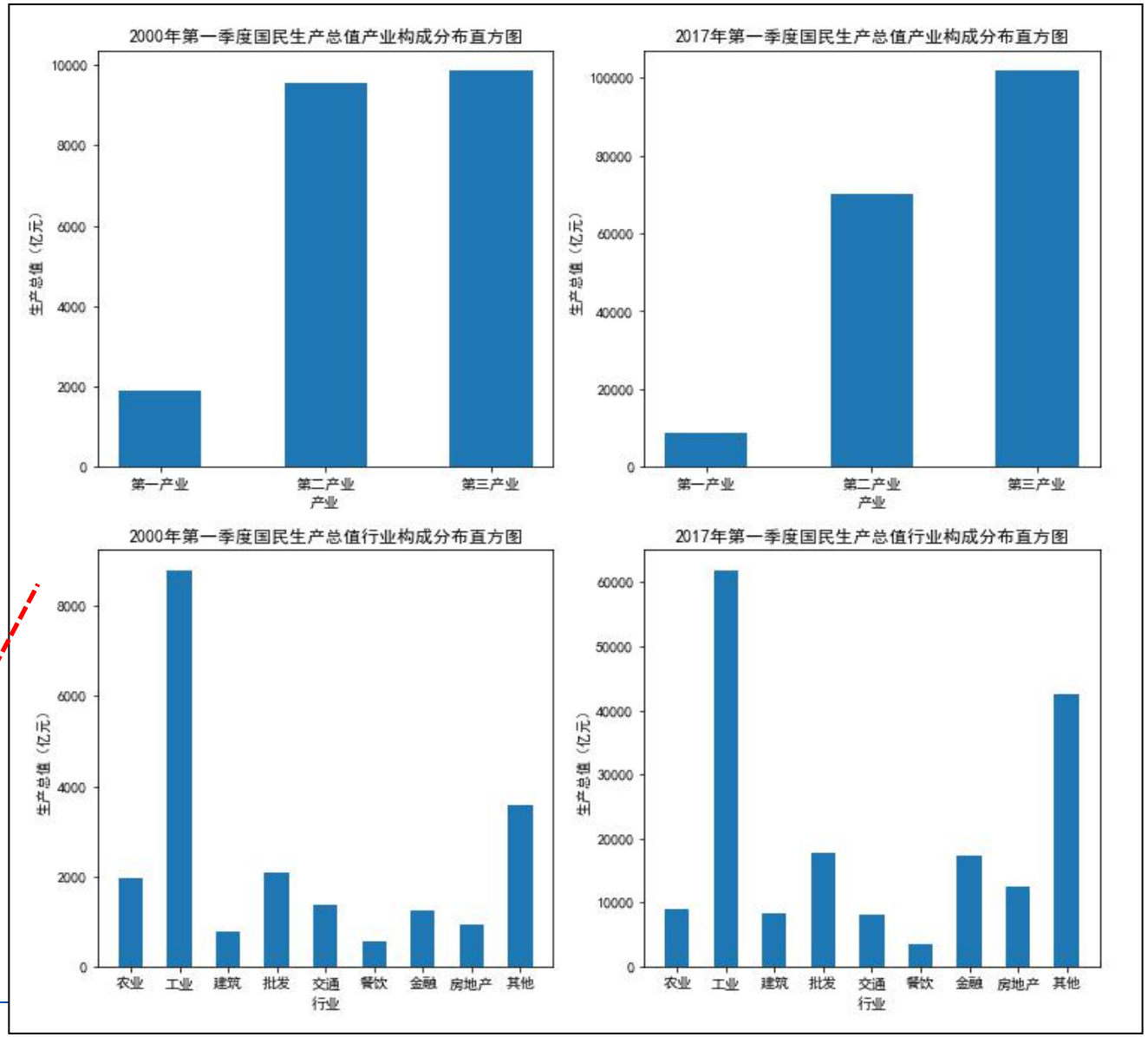


任务实现

1. 绘制国民生产总值构成分布柱状图

- 通过柱状图分析2000年第一季度和2017年第一季度之间的三大产业的国民生产总值，可以发现各产业绝对数值之间的关系，并通过对比发现产业结构的变化。
- 同理可以得出行业间的绝对数值关系以及17年来行业发展状况。

- 通过这几张柱状图对比看出，第一产业与第二产业和第三产业的国民生产总值差距愈发巨大。
- 根据坐标轴变化可以发现，国民生产总值增长近10倍。2010-2017年，金融行业与其他行业增长幅度相比较较为明显。



2. 绘制国民生产总值构成分布饼图

- 通过分析2000年与2017年不同的产业和行业在国民生产总值中的占比，可以发现我国产业结构变化和行业变迁。

任务实现

2. 绘制国民生产总值构成分布饼图

```
import numpy as np
import matplotlib.pyplot as plt
data = np.load('data/国民经济核算季度数据.npz')
name = data['columns']                          # 提取其中的columns数组，视为数据的标签
values = data['values']                         # 提取其中的values数组，数据的存在位置
plt.rcParams['font.sans-serif'] = 'SimHei'      # 设置中文显示
plt.rcParams['axes.unicode_minus'] = False
label1 = ['第一产业','第二产业','第三产业']       # 标签1
label2 = ['农业','工业','建筑','批发','交通','餐饮','金融','房地产','其他'] # 标签2
explode1 = [0.01,0.01,0.01]
explode2 = [0.01,0.01,0.01,0.01,0.2,0.01,0.01,0.01,0.01]    #设定交通子块离圆心远一些
p = plt.figure(figsize=(12,12))
```

任务实现

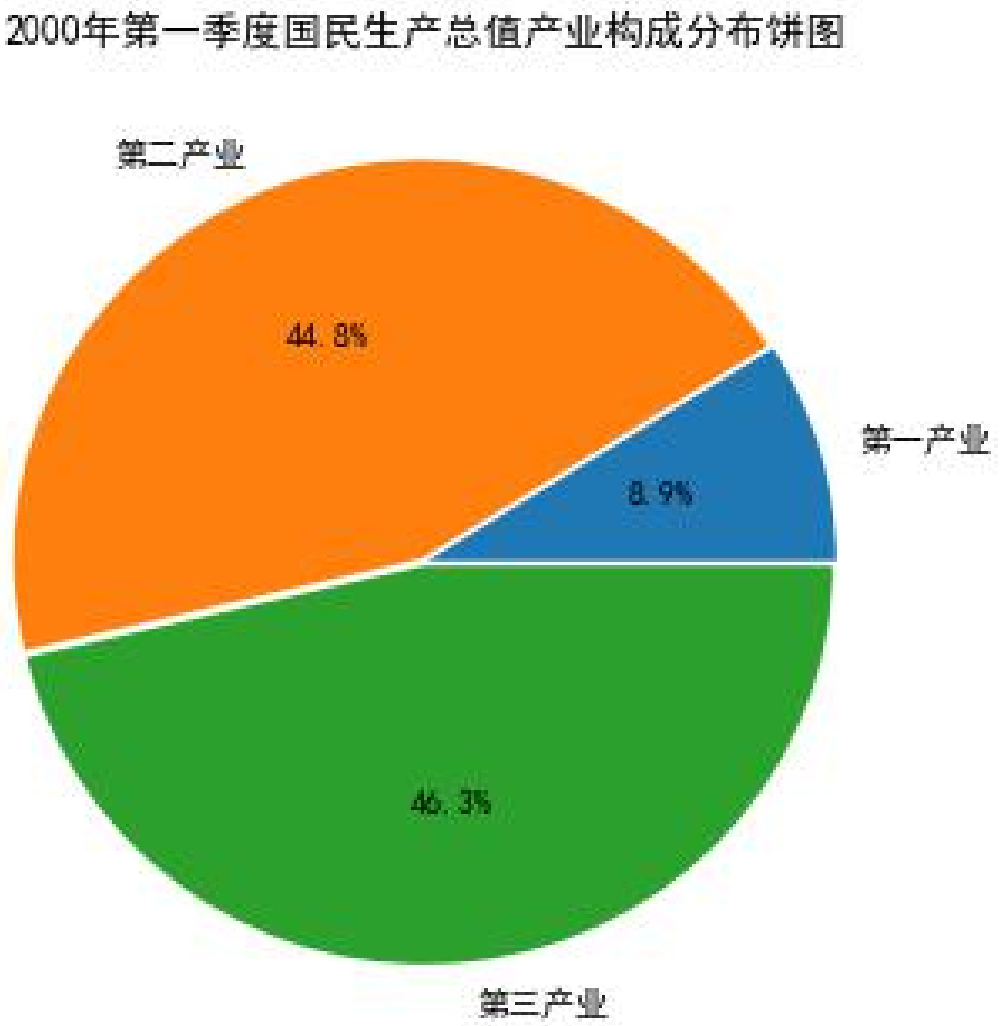
2. 绘制国民生产总值构成分布饼图

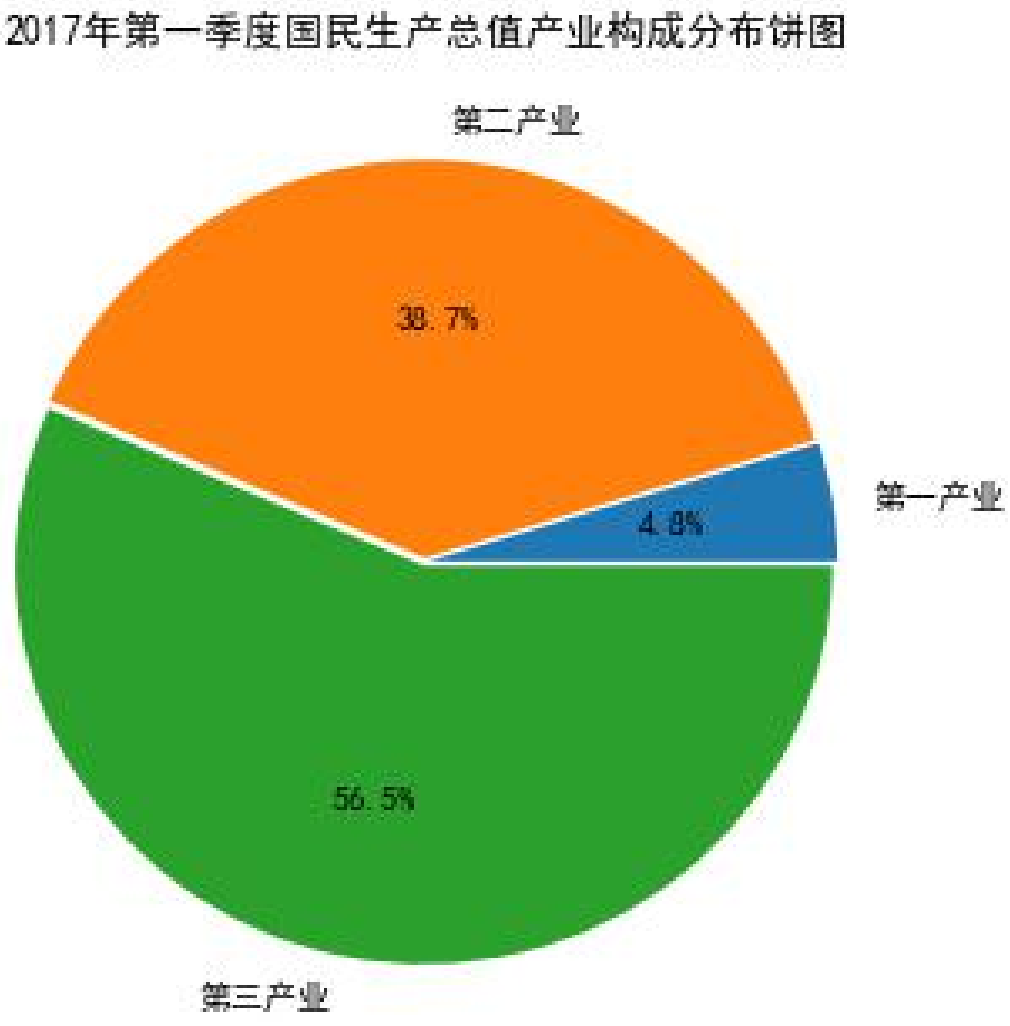
```
## 子图1
ax1 = p.add_subplot(2,2,1)
plt.pie(values[0,3:6],explode=explode1,labels=label1,autopct='%1.1f%%')      # 绘制饼图
plt.title('2000年第一季度国民生产总值产业构成分布饼图')

## 子图2
ax2 = p.add_subplot(2,2,2)
plt.pie(values[-1,3:6],explode=explode1,labels=label1,autopct='%1.1f%%')     # 绘制饼图
plt.title('2017年第一季度国民生产总值产业构成分布饼图')
```

任务实现

2. 绘制国民生产总值构成分布饼图





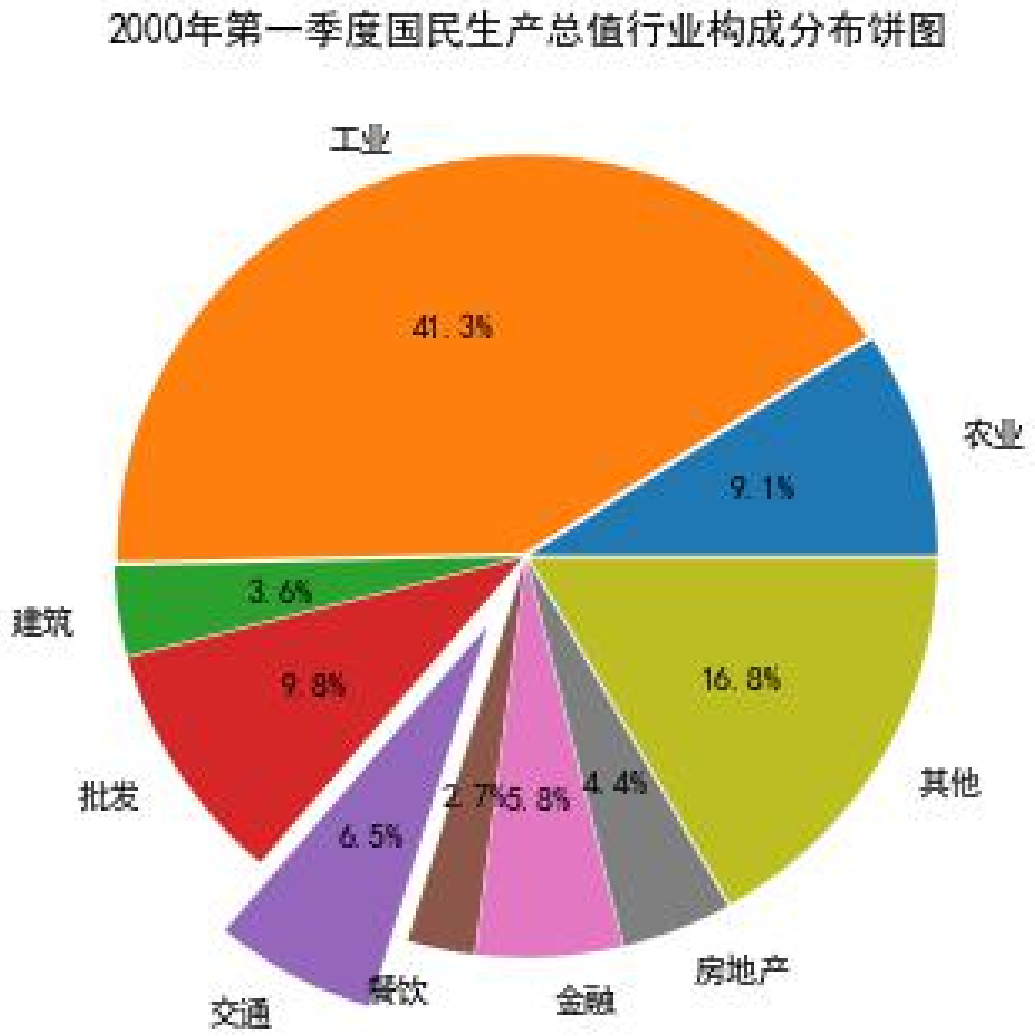
任务实现

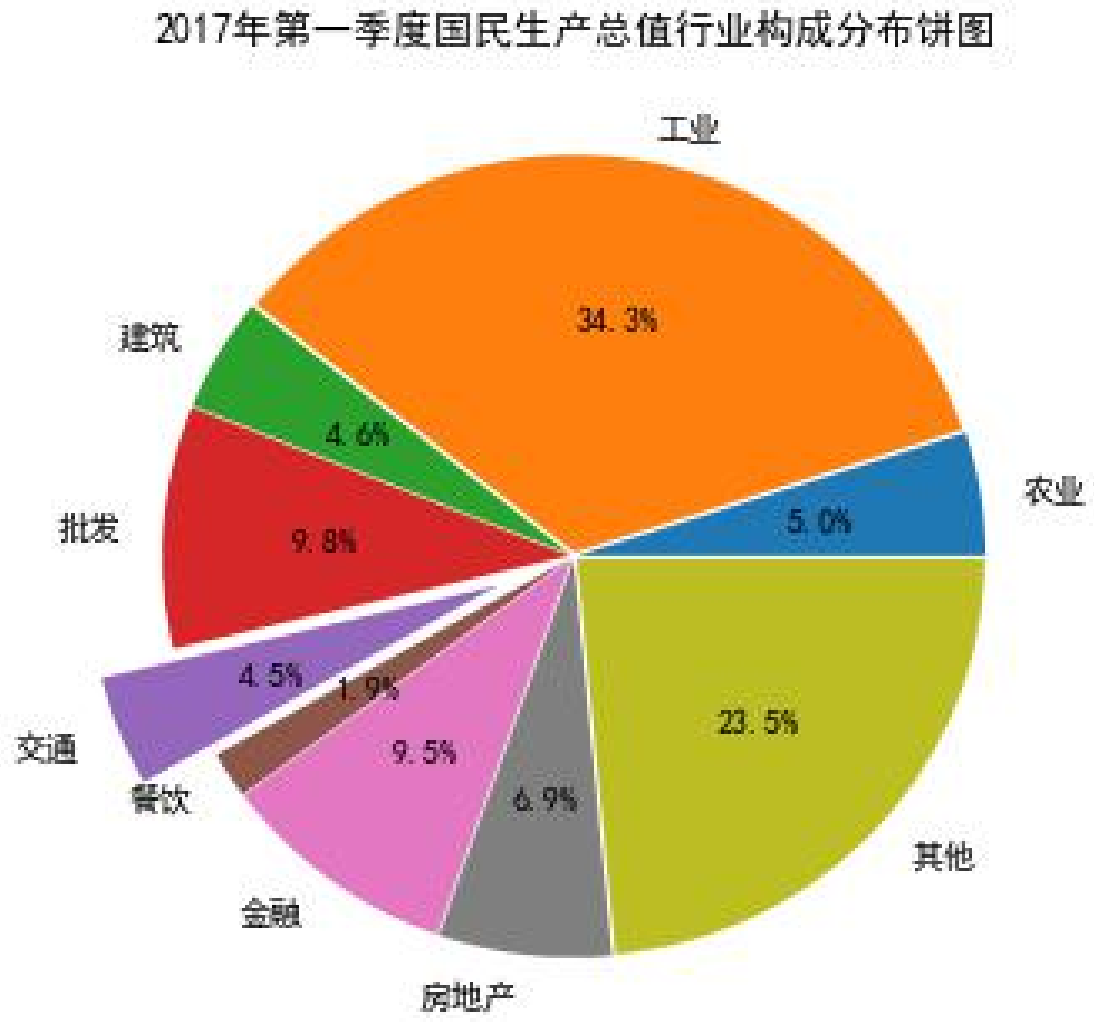
2. 绘制国民生产总值构成分布饼图

```
## 子图3
ax3 = p.add_subplot(2,2,3)
plt.pie(values[0,6:],explode=explode2,labels=label2,autopct='%1.1f%%')    # 绘制饼图
plt.title('2000年第一季度国民生产总值行业构成分布饼图’)                    # 添加图表标题
## 子图4
ax4 = p.add_subplot(2,2,4)
plt.pie(values[-1,6:],explode=explode2,labels=label2,autopct='%1.1f%%')    # 绘制饼图
plt.title('2017年第一季度国民生产总值行业构成分布饼图’)## 添加图表标题
## 保存并显示图形
plt.savefig('../tmp/国民生产总值构成分布饼图.png')
plt.show()
```

任务实现

2. 绘制国民生产总值构成分布饼图





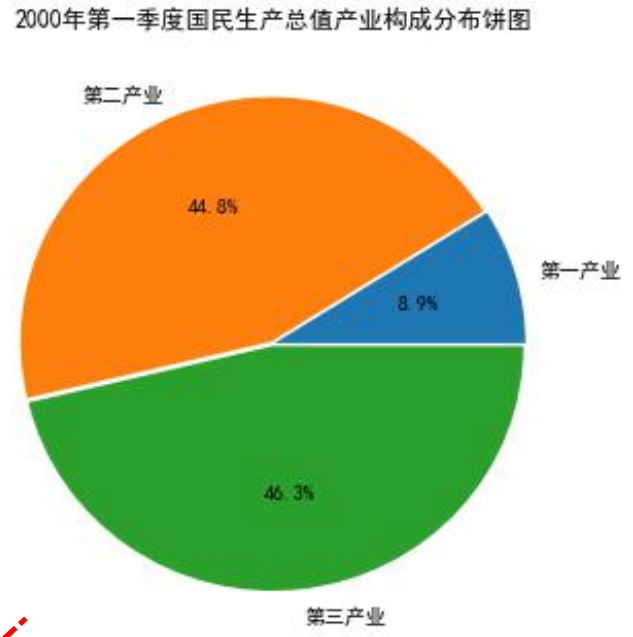
任务实现

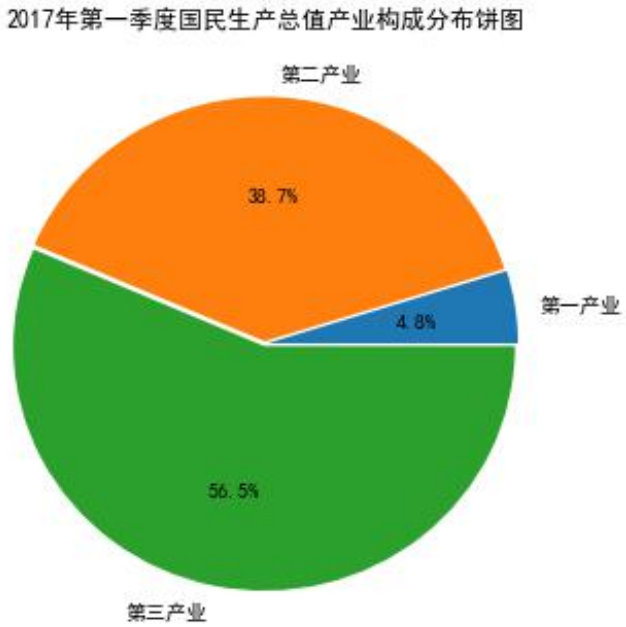
2. 绘制国民生产总值构成分布饼图

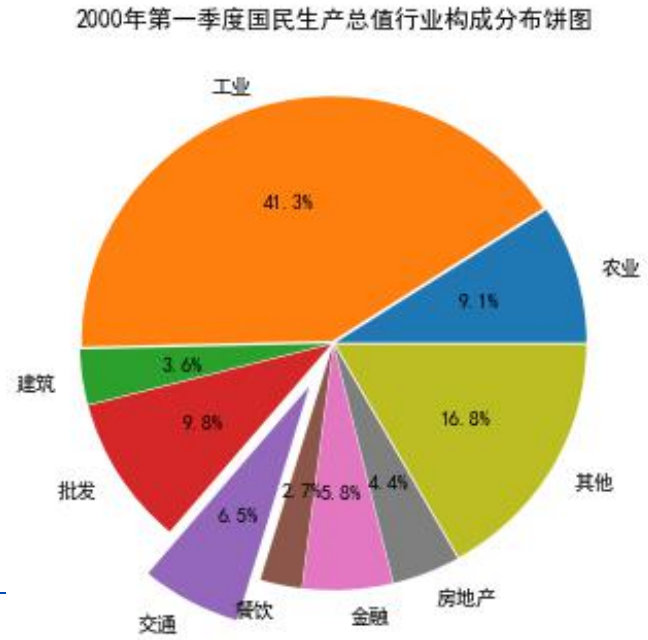
- 通过分析2000年与2017年不同的产业和行业在国民生产总值中的占比，可以发现我国产业结构变化和行业变迁。

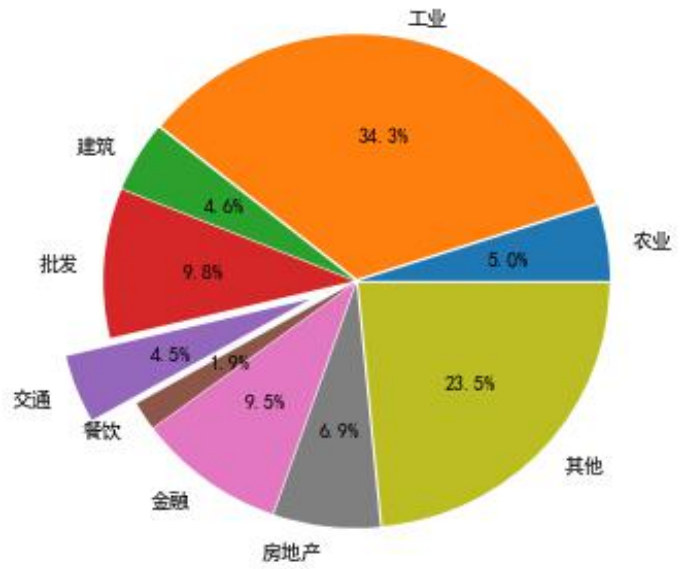
通过这个饼图可以看出

- 2000-2017年，第三产业在整个国民生产总值中的占比约提高了10%。
- 第一产业和第二产业在国民生产总值中的占比分别下降了约4%和6%。
- 工业在整个国民生产总值中的比例下降了7%，
- 其他行业与金融行业则分别提升了6.7%和3.7%









3. 绘制国民生产总值分散情况箱线图

- 通过分析2000年至2017年不同的产业和行业在国民生产总值中的分散情况，可以发现整体分散情况，从而判断整体增速是否加快。

任务实现

3. 绘制国民生产总值分散情况箱线图

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

data = np.load('data/国民经济核算季度数据.npz')

name = data['columns']                          # 提取其中的columns数组，视为数据的标签
values = data['values']                         # 提取其中的values数组，数据的存在位置
plt.rcParams['font.sans-serif'] = 'SimHei'      # 设置中文显示
plt.rcParams['axes.unicode_minus'] = False

p = plt.figure(figsize=(8,8))
```

任务实现

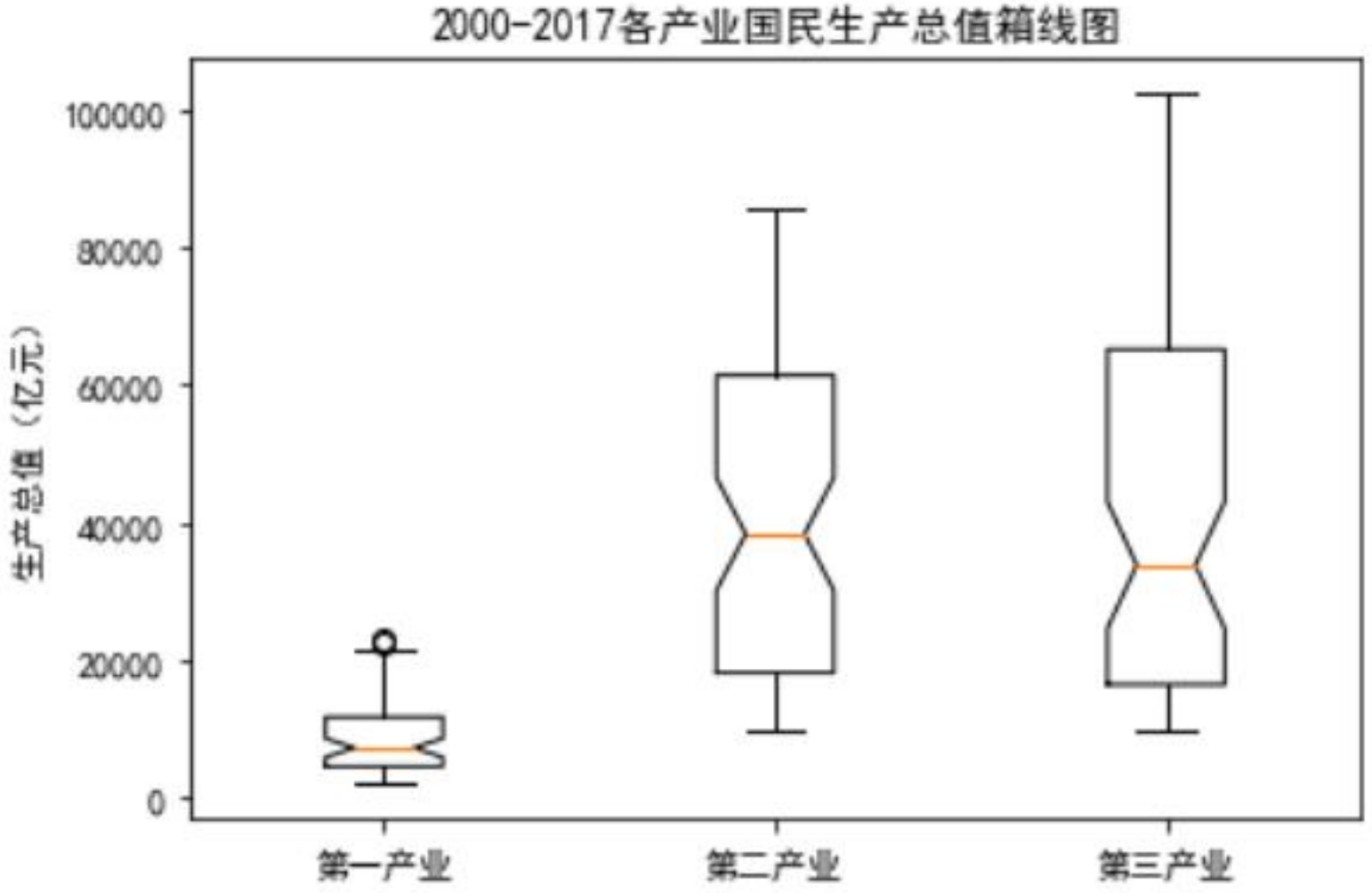
3. 绘制国民生产总值分散情况箱线图

```
## 子图1
ax1 = p.add_subplot(2,1,1)
gdp1 = (list(values[:,3]),list(values[:,4]),list(values[:,5]))
label1 = ['第一产业','第二产业','第三产业']                    # 标签1

## 绘制箱线图
plt.boxplot(gdp1,notch=True,labels = label1, meanline=True)
plt.title('2000-2017各产业国民生产总值箱线图')
plt.ylabel('生产总值（亿元）')                                  # 添加y轴名称
```

任务实现

3. 绘制国民生产总值分散情况箱线图



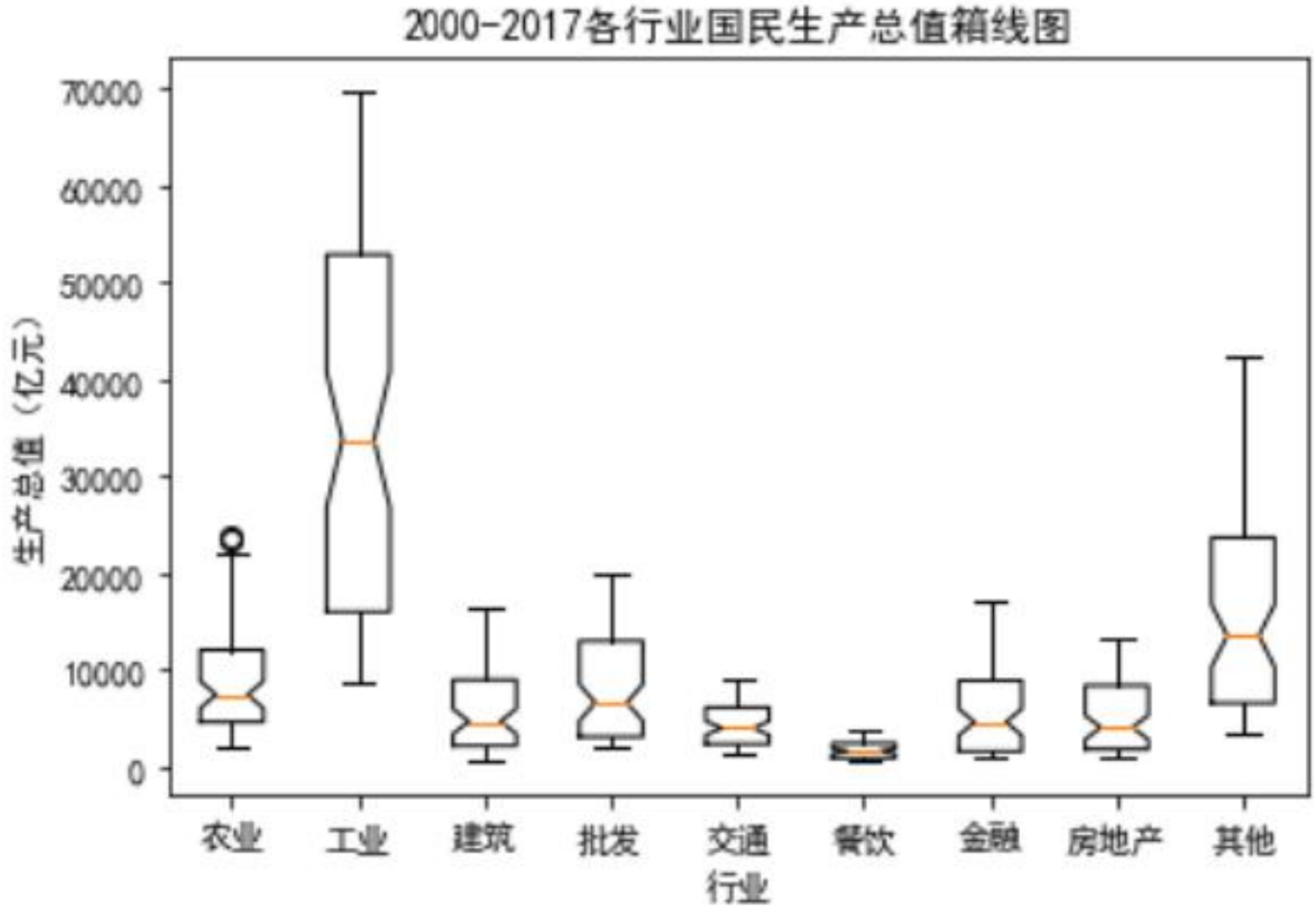
任务实现

3. 绘制国民生产总值分散情况箱线图

```
## 子图2
ax2 = p.add_subplot(2,1,2)
gdp2 = ([list(values[:,i]) for i in range(6,15)])
label2 = ['农业','工业','建筑','批发','交通 ','餐饮','金融','房地产','其他']          # 标签2
## 绘制箱线图
plt.boxplot(gdp2,notch=True,labels = label2, meanline=True)
plt.title('2000-2017各行业国民生产总值箱线图')
plt.xlabel('行业')                                                    # 添加x轴标签
plt.ylabel('生产总值（亿元）')                                          # 添加y轴名称
## 保存并显示图形
plt.savefig('../tmp/国民生产总值分散情况箱线图.png')
plt.show()
```

任务实现

3. 绘制国民生产总值分散情况箱线图

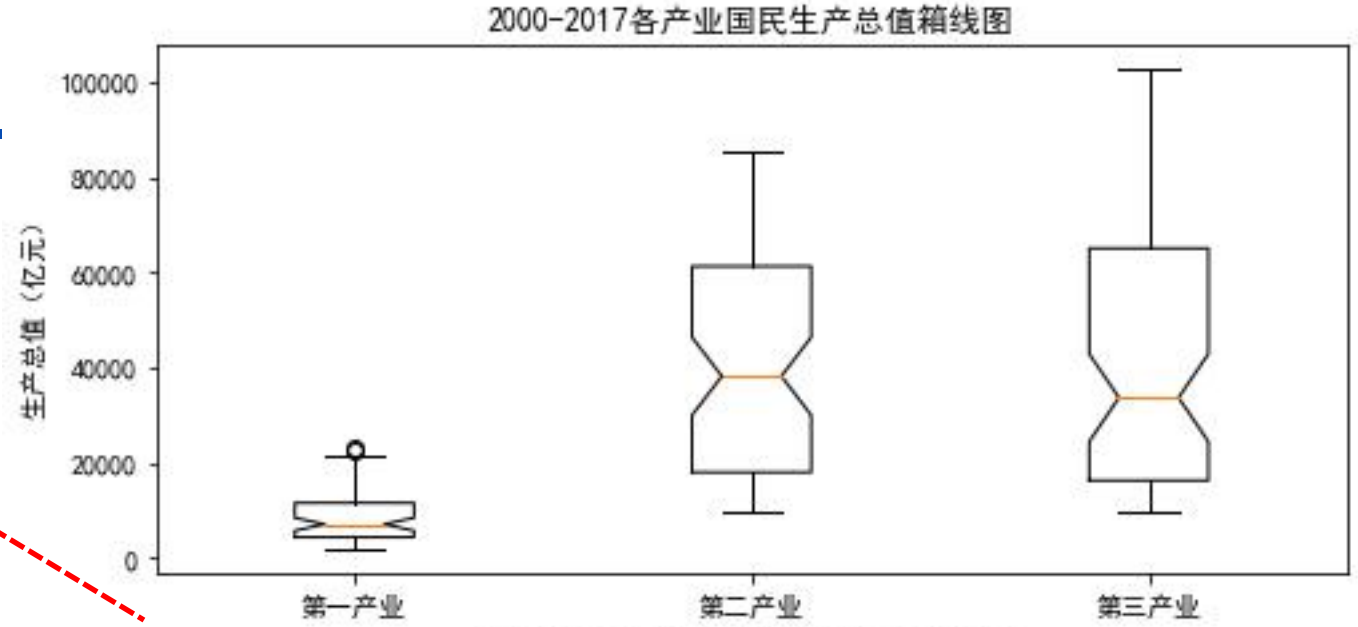


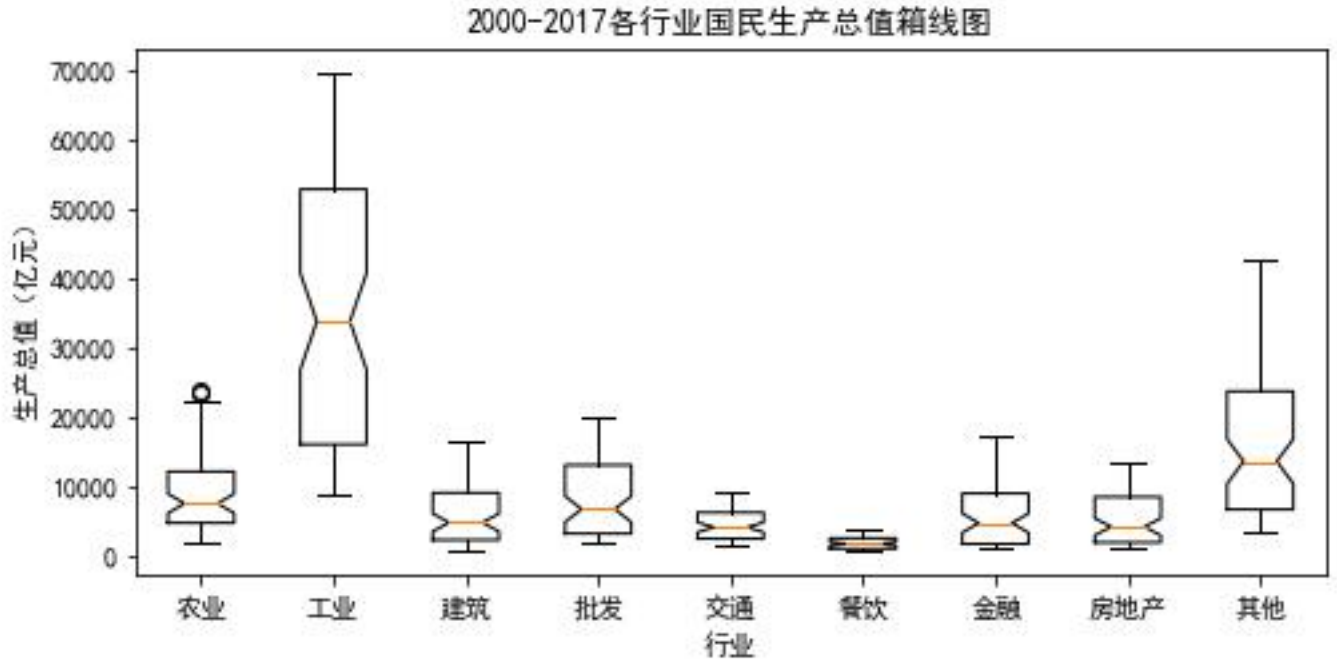
任务实现

3. 绘制国民生产总值分散情况箱线图

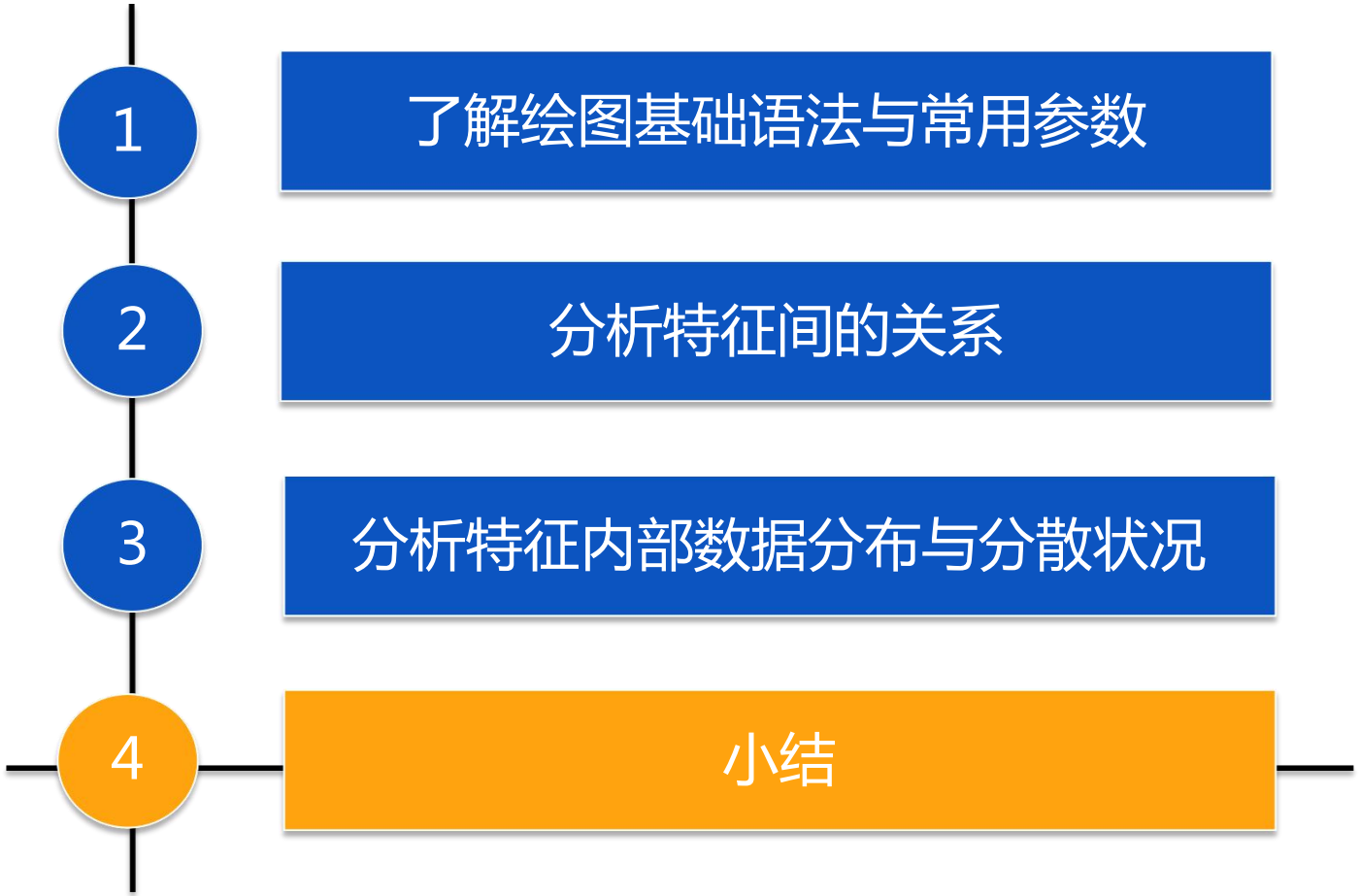
通过箱线图可以看出：

- 整体经济趋势是上升的。
- 产业中的第二产业增长平缓。
- 行业中的工业与餐饮业的增长比较平缓，其他行业、批发行业、建筑行业、金融行业和房地产行业增速均有所加快





目录



小结

本章以2000至2017年各季度国民生产总值数据为例，介绍了pyplot绘图的基本语法、常用参数。

- 介绍了分析特征间相关关系的**散点图**。
- 分析特征间趋势关系的**折线图**。
- 分析特征内部数据分布的**条形图**和**饼状图**。
- 以及分析特征内部数据分散情况的**箱线图**。

为读者后续深入学习Matplotlib数据可视化打下了深厚的基础。

Thank you!