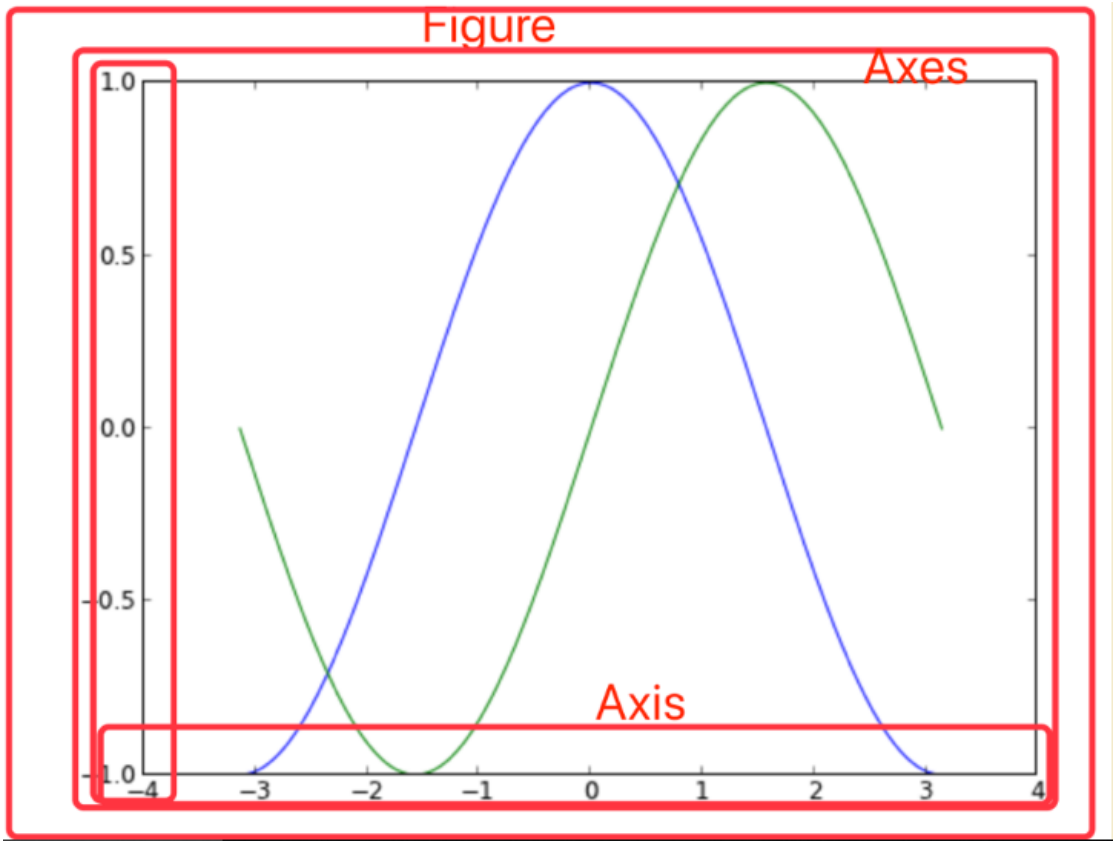

总结:

- 图像保存【知道】
  - plt.savefig("路径")
- 添加x,y轴刻度【知道】
  - plt.xticks()
  - plt.yticks()
  - **注意:在传递进去的第一个参数必须是数字,不能是字符串,如果是字符串吗,需要进行替换操作**
- 添加网格显示【知道】
  - plt.grid(linestyle="--", alpha=0.5)
- 添加描述信息【知道】
  - plt.xlabel()
  - plt.ylabel()
  - plt.title()
- 多次plot【了解】
  - 直接进行添加就OK
- 显示图例【知道】
  - plt.legend(loc="best")
  - **注意:一定要在plt.plot()里面设置一个label,如果不设置,没法显示**
- 多个坐标系显示【了解】
  - plt.subplots(nrows=, ncols=)
- 折线图的应用【知道】
  - 1.应用于观察数据的变化
  - 2.可是画出一些数学函数图像

1-简介

- 是专门用于开发2D图表(包括3D图表)
- 使用起来及其简单
- 以渐进、交互式方式实现数据可视化

2-三层结构



容器层

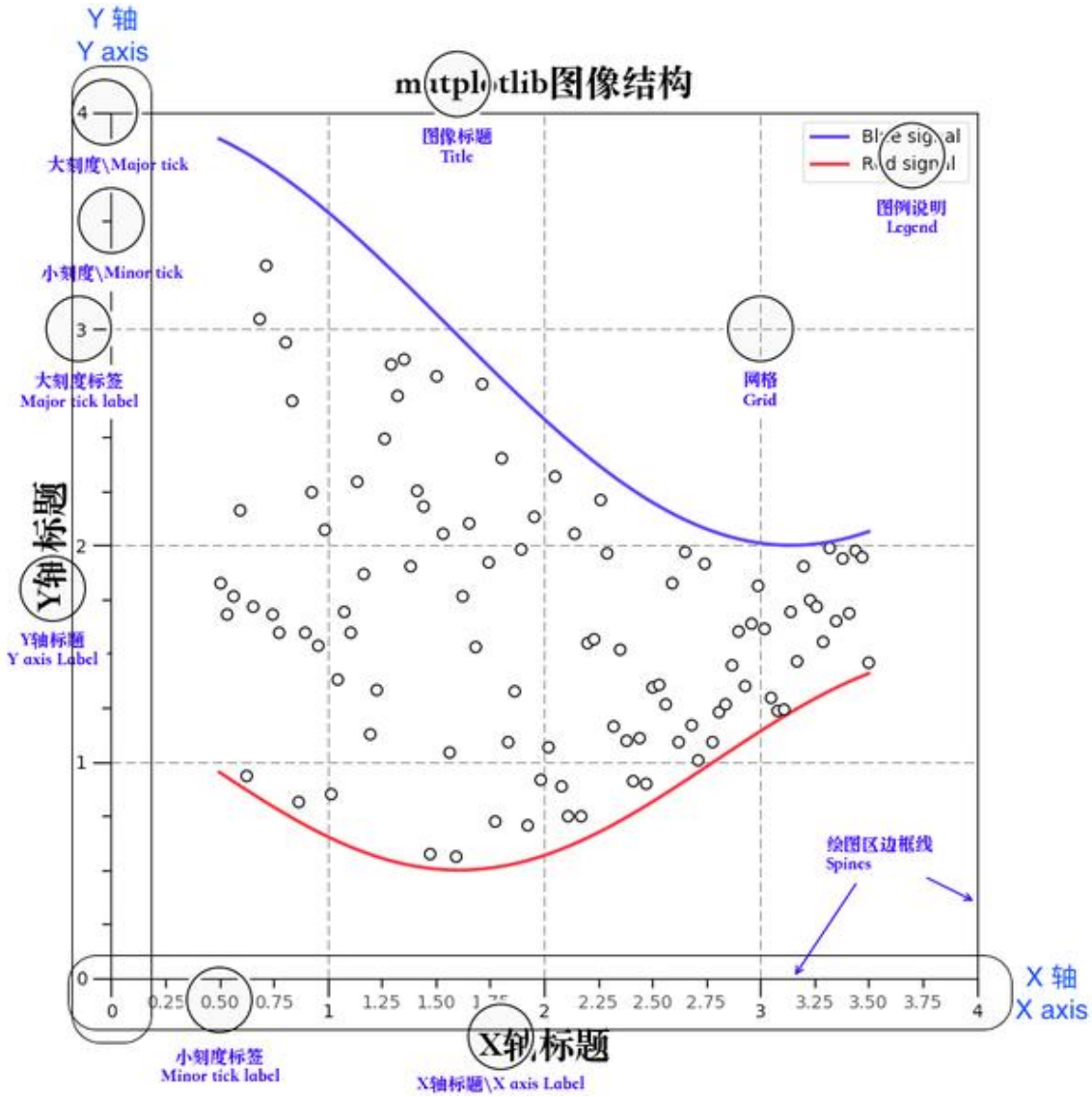
容器层主要由Canvas、Figure、Axes组成。

画板Canvas是位于最底层的系统层，在绘图的过程中充当画板的角色，即放置画布(Figure)的工具。

画布Figure是Canvas上方的第一层，也是需要用户来操作的应用层的第一层，在绘图的过程中充当画布的角色。

坐标系Axes是应用层的第二层，在绘图的过程中相当于画布上的绘图区的角色。

- Figure:指整个图形(可以通过plt.figure()设置画布的大小和分辨率等)
- Axes(**坐标系**):数据的绘图区域
- Axis(**坐标轴**)：坐标系中的一条轴，包含大小限制、刻度和刻度标签



辅助显示层

辅助显示层为Axes(绘图区)内的除了根据数据绘制出的图像以外的内容，主要包括Axes外观(facecolor)、边框线(spines)、坐标轴(axis)、坐标轴名称(axis label)、坐标轴刻度(tick)、坐标轴刻度标签(tick label)、网格线(grid)、图例(legend)、标题(title)等内容。

图像层

图像层指Axes内通过plot、scatter、bar、histogram、pie等函数根据数据绘制出的图像。

总结:

- Canvas（画板）位于最底层，用户一般接触不到
- Figure（画布）建立在Canvas之上
- Axes（绘图区）建立在Figure之上
- 坐标轴（axis）、图例（legend）等辅助显示层以及图像层都是建立在Axes之上

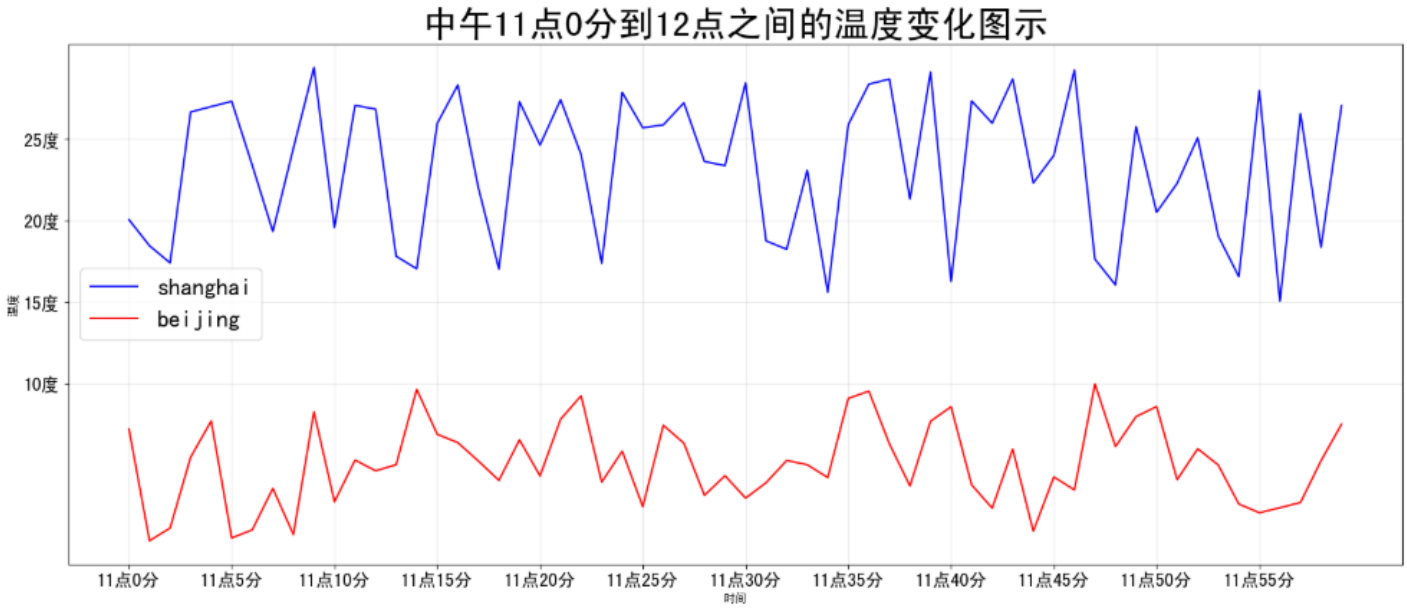
3-画图基本过程

1. 导入包

a. import matplotlib.pyplot as plt

2. 建立画图
   a. plt.figsure(figsize(30,8),dip=100)
3. 设置数据
   a. x,y=([1,2,3],[2,3,4])
4. 绘制图像
   a. plt.plot(x,y)
5. 添加标签刻度
   a. 第一个参数为x刻度值，必须为标量，第二个参数默认与标量一致，可以自定义刻度标签
   b. plt.xticks(x, **kwargs)
   c. plt.yticks(x, **kwargs)
6. 绘制网格
   a. plt.grid(True, linestyle='-', alpha=0.3)
7. 添加标签
   a. plt.xlabel("时间")
   b. plt.ylabel("温度")
   c. plt.title("中午",fontsize=30)
8. 保存图像
   a. plt.savefig('保存路径')
9. 展示图像
   a. plt.show()

4-实例-温度变化折线图




```
1 import matplotlib.pyplot as plt
2 import random
3 # 准备画布
```

```python
plt.figure(figsize=(20,8),dpi=100)

# 准备数据
x = range(60)
y_shanghai = [random.uniform(25,40) for i in x]
y_beijing = [random.uniform(10,20) for i in x]
# 多次绘制,要想显示图例需要添加label参数
plt.plot(x,y_shanghai,color='b',label="shanghai")
plt.plot(x,y_beijing,color='r',label="beijing")

# 显示图例
plt.legend(loc="best",fontsize=20)

# 添加刻度
x_ticks_label = ["11点{}分".format(i) for i in x]

y_ticks = range(40)
y_ticks_label = ["{}度".format(i) for i in y_ticks]
# 修改刻度显示

    # 第一个参数为：刻度，第二个参数为刻度标签
plt.xticks(x[::5], x_ticks_label[::5],fontsize=15)

    # 刻度标签没有就跟y一致
plt.yticks(y_ticks[20:40:5],y_ticks_label[10:40:5],fontsize=15)

# 添加网格
plt.grid(True, linestyle='-', alpha=0.3)

# 添加标签
plt.xlabel("时间")
plt.ylabel("温度")
plt.title("中午11点0分到12点之间的温度变化图示",fontsize=30)

plt.show()
```

5-多个坐标系显示-plt.subplots(面向对象的画图方法)

- API: plt.subplots(nrows=1, ncols=1, **kwagrs)
- 参数:
  - nrows: 行数

    - ncols: 列数
    - figsize: 画布大小
    - dpi: 清晰度
- 返回:
    - 画布fig, 坐标系列表axes

坐标系功能与plt相同, 有些方法不同，使用对象绘图

- axes[0].set_xticks()
- axes[0].set_yticks()
- axes[0].set_xticklabels()
- axes[0].set_xlabel()
- axes[0].set_ylabel()
- axes[0].set_title()

```
import matplotlib.pyplot as plt

import random
# 需求：画出某城市11点到12点1小时内每分钟的温度变化折线图，温度范围在15度~18度

# 创建画布，允许多个
fig , axes = plt.subplots(nrows=1,ncols=2,figsize=(20,8),dpi=100)

# 构造数据
x = range(60)
y_guangzhou = [random.uniform(30,40) for i in x]
y_beijing = [random.uniform(10,20) for i in x]

# 图像绘制
axes[0].plot(x,y_guangzhou,label="广州")
axes[1].plot(x,y_beijing,label="北京")

# 编写坐标
y_ticks = range(40)
x_ticks_label = ["11点{}".format(i) for i in x]

axes[0].set_xticks(x[::5])
axes[0].set_yticks(y_ticks[::5])
axes[0].set_xticklabels(x_ticks_label[::5])

axes[1].set_xticks(x[::5])
```

```
axes[1].set_yticks(y_ticks[::5])
axes[1].set_xticklabels(x_ticks_label[::5])

# 2.2 添加网格显示
# plt.grid(True, linestyle="--", alpha=0.5)
axes[0].grid(True, linestyle="--", alpha=0.5)
axes[1].grid(True, linestyle="--", alpha=0.5)

# 2.3 添加x,y轴描述和标题
# plt.xlabel("时间")
# plt.ylabel("温度")
# plt.title("中午11点--12点温度变化图", fontsize=25)
axes[0].set_xlabel("时间")
axes[0].set_ylabel("温度")
axes[0].set_title("广州中午11点--12点温度变化图", fontsize=25)
axes[1].set_xlabel("时间")
axes[1].set_ylabel("温度")
axes[1].set_title("背北京中午11点--12点温度变化图", fontsize=25)

# 2.4 显示图例
# plt.legend(loc=0)
axes[0].legend(loc=0)
axes[1].legend(loc=0)

# 3.显示
plt.show()
```

6-拓展：画各种数学函数图像

- 注意：plt.plot()除了可以画折线图，也可以用于画各种数学函数图像
- 可以配合Numpy

```
import numpy as np
# 0.准备数据
x = np.linspace(-10, 10, 1000) # -10 到10，去1000个值，间隔一致
y = np.sin(x)    # 求每个x对应sin值

# 1.创建画布
plt.figure(figsize=(20, 8), dpi=100)

```

```
# 2.绘制函数图像
plt.plot(x, y)
# 2.1 添加网格显示
plt.grid()

# 3.显示图像
plt.show()
```

7-常见图像绘制[*]

1. API与含义:

- **折线图:**
  - 概念: 用于展示数据的变化情况的
  - API: plt.plot(x, y)
- **散点图**: 用于分析两个变量的规律, 展示离散点分布情况
  - API: plt.scatter(x, y)
- **柱状图**: 统计/对比
  - 比较数据之间的差别。(统计/对比)
  - API: plt.bar(x, height, width, color)
- **直方图**: 展示连续数据的分布情况
  - API: plt.hist(x, bins) bins 整数, 序列 可选
- **饼状图**: 占比
  - API: plt.pie(x, labels, autopct, colors)

2. 案例

柱状图

```
import matplotlib.pyplot as plt
# 0.准备数据
# 电影名字
movie_name = ['雷神3：诸神黄昏','正义联盟','东方快车谋杀案','寻梦环游记','全球风暴','降魔传','追捕','七十
# 横坐标
x = range(len(movie_name))
# 票房数据
y = [73853,57767,22354,15969,14839,8725,8716,8318,7916,6764,52222]

# 建立画布
plt.figure(figsize=(40,13),dpi=100)

```

```
# 柱状图
plt.bar(x, y, 0.5, color=["r","b","y"])


# 设置x标签
plt.xticks(x,movie_name,fontsize=30)

# 2.2 添加网格显示
plt.grid(linestyle="--", alpha=0.5)

# 2.3 添加标题
plt.title("电影票房收入对比")

plt.show()

```

**修改字体

```
1. 下载黑体
2. 拷贝字体文件
  ~/.virtualenvs/ai/local/lib/python3.5/site-packages/matplotlib/mpl-data/fonts/ttf
  # 字体文件
  # 打印matplotlib配置文件的位置
  import matplotlib
  print(matplotlib.matplotlib_fname())

3. 编辑配置文件
  #cd  ~/.virtualenvs/ai/local/lib/python3.5/site-packages/matplotlib/mpl-data
  #vim matplotlibrc

  # 添加下面内容到配置文件中
  font.family         : sans-serif
  font.sans-serif         : SimHei
  axes.unicode_minus  : False

4. 删除缓存文件
```

```
  rm -r ~/.cache/matplotlib/*
  # 缓存文件位置
  matplotlib.get_cachedir()->获取字体cache位置的方法

5. 重新运行 jupyter notebook
```