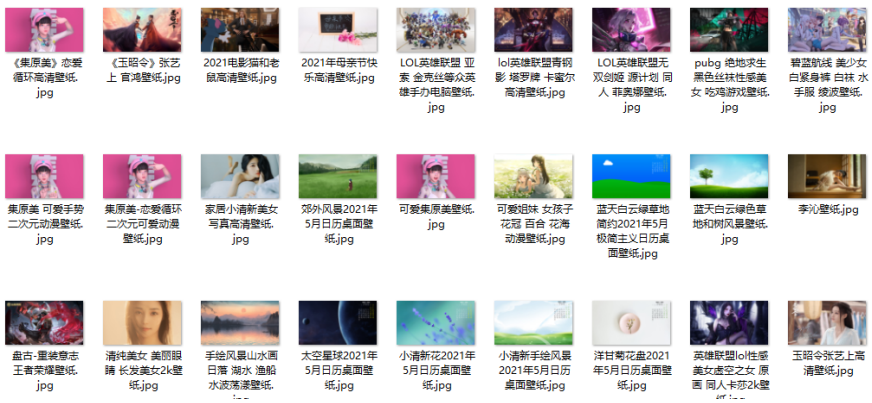
网络爬虫

项目导读：

1、图片批量下载

网站地址：http://www.netbian.com/

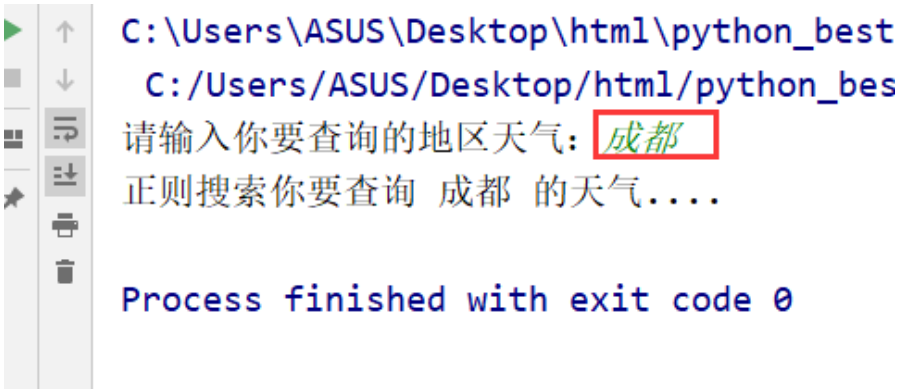
运行结果：

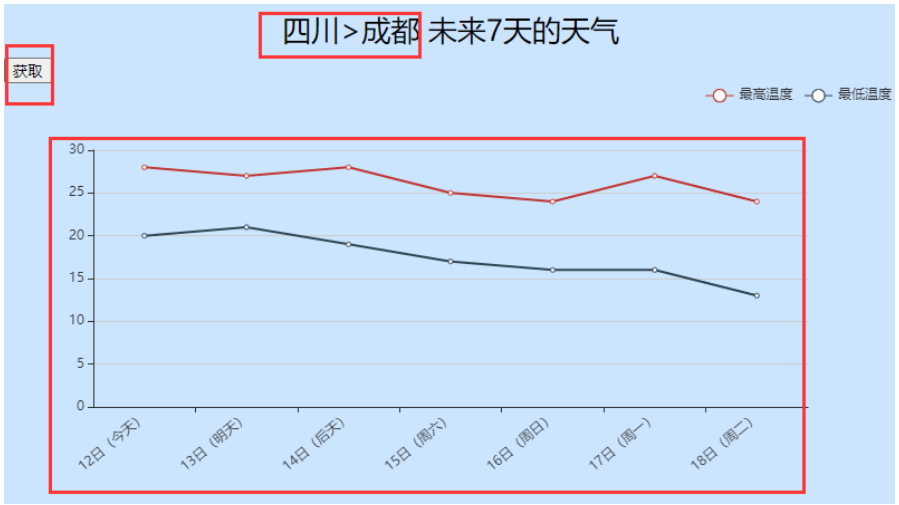


2、全国各城市未来七日温度可视化

网站地址：
http://www.weather.com.cn/weather/101270101.shtml

运行结果：





项目实施步骤

1.认识爬虫 2.模求拟浏览器发送请 3.提取网页内容技术（xpath） 4.实现数据可视化技术

项目涉及知识

1.爬虫语法

1.1认识爬虫

网络爬虫，又被称为网页蜘蛛，网络机器人，在FOAF社区中间，更经常的称为网页追逐者，是一种按照一定的规则，自动地抓取万维网信息的程序或者脚本，另外一些不常使用的名字还有蚂蚁、自动索引、模拟程序或者蠕虫。

简单来讲，爬虫就是一个探测机器，它的基本操作就是模拟人的行为去各个网站溜达，点点按钮，查查数据，或者把看到的信息背回来。就像一只虫子在一幢楼里不知疲倦地爬来爬去。每天放出无数爬虫到各个网站，把他们的信息抓回来,并保存

1.2爬虫的使用场景

1. 个人信息检索系统
2. 特定信息收集系统
3. 自动填写调查问卷
4. 电话号码收集系统
5. 爬虫分析热度排行
6. **爬虫进行股票分析**
7. 爬虫网站定向数据
8. 视频网站视频批量下载
9. 购物网站比价系统
10. 文章批量下载
11. 飞机票比价系统（以一个地方到另外一个地方飞机票）
12. 招聘公司爬虫招聘信息
13. 爬虫房产网站做房产分析
14. 财务报表下载助手
15. 排行分析之指数分析
16. 畅销书排行分析
17. 验证码破解
18. **用户拓展关系分析**
19. 模拟登陆系统
20. 文件下载助手开发
21. 抢票软件
22. 构建代理IP池
23. 分析网页结构
24. 房产数据分析
25. 音乐网站批量下载
26. 城市旅游数据分析
27. **购物网站数据挖掘分析**
28. V企业信息、分类信息、房地产信息、 电商信息

29. 论坛发帖、问答推广、效果回访、网上购物、宝贝收藏、投票点赞
30. 爬取微信公众号，进行数据分析
31. 头条自动发文章，赚点广告费
32. 网站薅羊毛
33. 淘宝关键词展位排名
34. 新闻数据分析
35. **为人工智能，科学分析提供大量的数据**

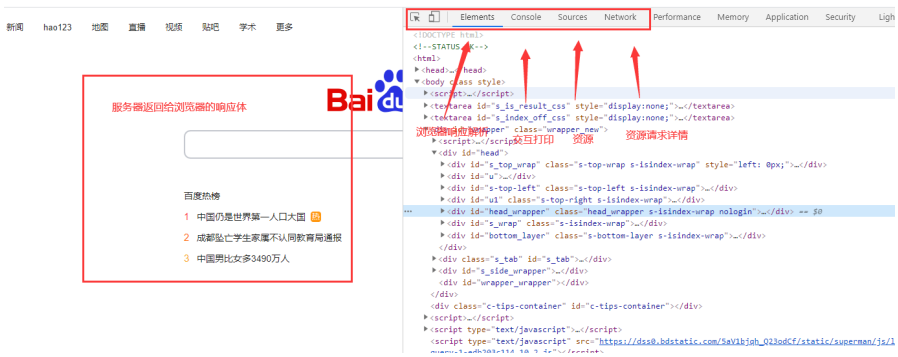
1.3 HTTP请求

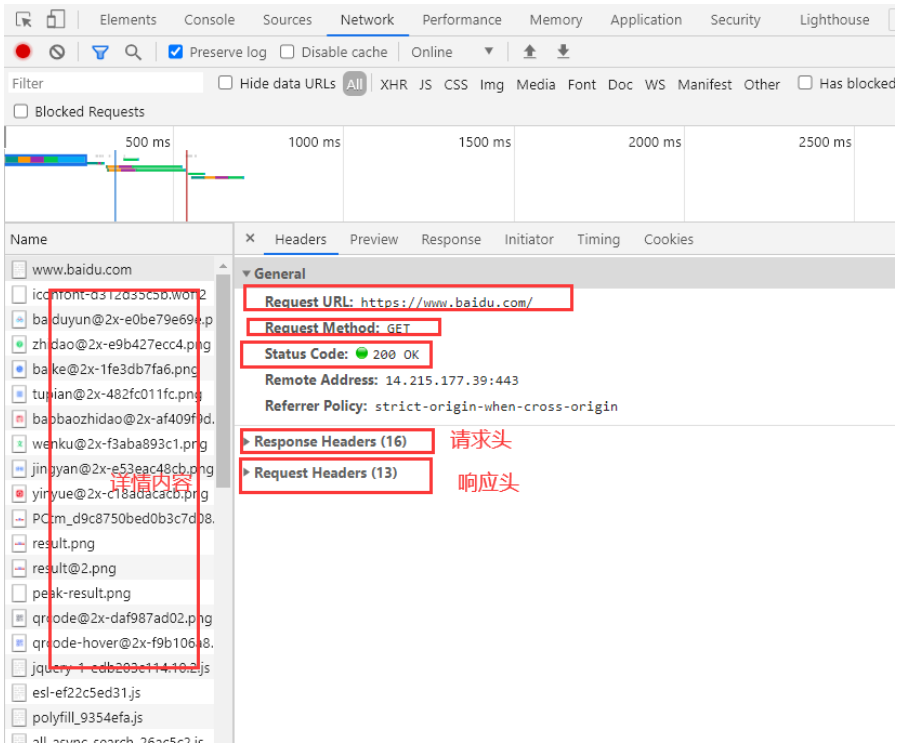
客户端：浏览器（谷歌）

服务端：服务器（百度）

以百度网站为案例：

http://www.baidu.com 网页请求地址





1.3 Requets安装

作用：模拟浏览器，发送网络请求，并接收服务器返回的数据

```
pip intall requests
```

1.4 发起响应，获取网页

```
# 网络请求的第三方库
import requests
# 百度地址
url = "http://www.baidu.com"
# 模拟浏览器发起请求
response = requests.get(url)
# 获取浏览器状态码：200成功
print(response) # <Response [200]>
# 解析服务器返回内容
print(response.content.decode()) #<!--STATUS OK--><html> <head><meta
http-equiv=content-type
```

1.5 反爬

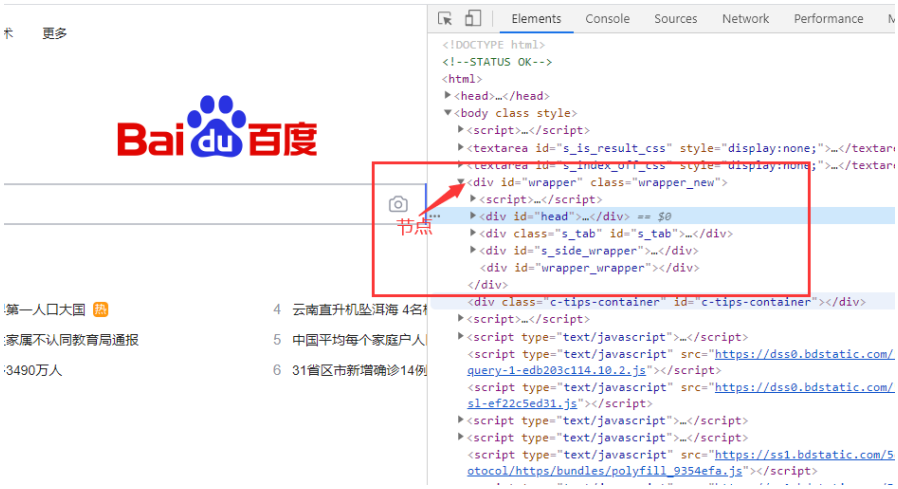
```
import requests
url = "http://www.baidu.com"
header = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.212
Safari/537.36'}
response = requests.get(url,headers=header)
print(response.content.decode())
```

1.5 XPath语法，提取数据

安装Python第三方库：lxml，提取数据

```
pip install lxml
```

1.5.1 XPath语术--节点



1.5.2 XPath语法

以百度贴吧为例：https://tieba.baidu.com/f?kw=%CC%F9%B0%C9&fr=ala0&tpl=5

选取节点

NODENAME	选取此节点的所有子节点。
/	从根节点选取。
//	从匹配选择的当前节点选择文档中的节点，而不考虑它们的位置。
.	选取当前节点。
..	选取当前节点的父节点。
@	选取属性。

```
import requests
from lxml import etree
url = "https://tieba.baidu.com/f?kw=%CC%F9%B0%C9&fr=ala0&tpl=5"
response = requests.get(url)
# print(response)
# print(response.content.decode())
# 获取网页内容
content = response.content.decode()
# 将网页内容通过通过lxml梳理
html_str = etree.HTML(content)
# 使用xpath提取网页内容
p_obj = html_str.xpath("//p")  # 全文搜索p标签
print(p_obj)  # [<Element p at 0x22eb3e0e888>],p所在的对象
# 找到class为card_slogan的p标签
p_class = html_str.xpath("//p[@class='card_slogan']")
print(p_class)  # [<Element p at 0x1f325c4af88>],p标签的位置
# 找到class为card_slogan的p标签下的文字
p_text = html_str.xpath("//p[@class='card_slogan']/text()")
print(p_text) # ['百度贴吧吧，吧友交流的家园']
```

案例作业：


```
import requests
from lxml import etree
url = "https://tieba.baidu.com/f?kw=%CC%F9%B0%C9&fr=ala0&tpl=5"
response = requests.get(url)
# print(response)
# print(response.content.decode())
# 获取网页内容
content = response.content.decode()
# 将网页内容通过通过lxml梳理
html_str = etree.HTML(content)
#获取"看帖"文字
```

```
#获取"看帖"文字
a_text= html_str.xpath("//a[@id='tab_forumname']/text()")
print(a_text)  # ['看贴']
# 获取"看帖","图片","精品"， "视频" 文字
li = html_str.xpath("//li[@class=' j_tbnav_tab ']") # 属性相等
print(li) # [<Element li at 0x1d4c5e5cd88>, <Element li at 0x1d4c615adc8>,
<Element li at 0x1d4c615ad08>]
for i in li:
    a_txt = i.xpath("./a/text()") # 当前li标签下的a标签下的文字
    print(a_txt)  # ['图片'] ['精品'] ['视频']
```

选取节点谓语

/BOOKSTORE/BOOK[1]	**选取属于 BOOKSTORE 子元素的第一个 BOOK 元素。**
/bookstore/book[last()]	选取属于 bookstore 子元素的最后一个 book 元素。
/bookstore/book[last()-1]	选取属于 bookstore 子元素的倒数第二个 book 元素。
//title[@lang='eng']	选取所有 title 元素，且这些元素拥有值为 eng 的 lang 属性。

```
import requests
from lxml import etree
url = "https://tieba.baidu.com/f?kw=%CC%F9%B0%C9&fr=ala0&tpl=5"
response = requests.get(url)
# print(response)
# print(response.content.decode())
# 获取网页内容
content = response.content.decode()
# 将网页内容通过通过lxml梳理
html_str = etree.HTML(content)

# 获取"看帖"href属性
a_href = html_str.xpath("//a[@id='tab_forumname']/@href")
print(a_href) # ['/f?kw=%E8%B4%B4%E5%90%A7&ie=utf-8&tab=main']

# 找到精品属性
# 方法1：
a_list = html_str.xpath("//a[@class='j_nav_local_tab_link j_tbnav_tab_a']")
print(a_list) # [<Element a at 0x181fe4fca08>, <Element a at 0x181fe6cadc8>,
<Element a at 0x181fe507f08>, <Element a at 0x181fe6ce808>]
a_list2_text = a_list[2].xpath("./@href")
print(a_list2_text) # ['/f?kw=%E8%B4%B4%E5%90%A7&ie=utf-8&tab=good']
# 方法2：
a_list3 = html_str.xpath("//ul[@class='nav_list j_nav_list']/li[3]/a/@href")
print(a_list3) # ['/f?kw=%E8%B4%B4%E5%90%A7&ie=utf-8&tab=good']
```

```
print(a_list3) # ['/f?kw=%E8%B4%B4%E5%90%A7&ie=utf-8&tab=good']
```

选取若干路径

径表达式	结果
//book/title | //book/price	选取 book 元素的所有 title 和 price 元素。
//title | //price	选取文档中的所有 title 和 price 元素。
/bookstore/book/title | //price	选取属于 bookstore 元素的 book 元素的所有 title 元素，以及文档中所有的 price 元素。


```
import requests
from lxml import etree
url = "https://tieba.baidu.com/f?kw=%CC%F9%B0%C9&fr=ala0&tpl=5"
response = requests.get(url)
# print(response)
# print(response.content.decode())
# 获取网页内容
content = response.content.decode()
# 将网页内容通过通过lxml梳理
html_str = etree.HTML(content)

# 提取多项内容
a_text = html_str.xpath("//ul[@class='forum_dir_info bottom_list
clearfix']//a[@rel='noreferrer']/text()")
print(a_text) # ['贴吧活动与交流']
a_list_text = html_str.xpath("//a[@class='j_nav_local_tab_link
j_tbnav_tab_a']/text()")
print(a_list_text) # ['看贴', '图片', '精品', '视频']
# 合并提取
a_list = html_str.xpath("//ul[@class='forum_dir_info bottom_list
clearfix']//a[@rel='noreferrer']/text() | //a[@class='j_nav_local_tab_link
j_tbnav_tab_a']/text()")
print(a_list) # ['贴吧活动与交流', '看贴', '图片', '精品', '视频']
```

1.6 项目实训--彼岸桌面壁纸下载

项目需求：

根据彼岸网页地址，下载桌面图片，并保存到本地img文件夹中

实施步骤：

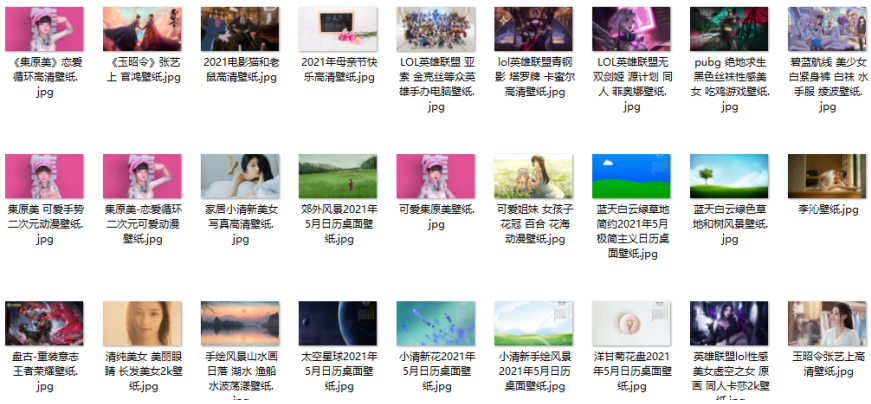
1. 安装requests库：pip install requests
2. 安装xpath库：pip install lxml
3. 获取彼岸网站url地址
4. 发起requests响应，获取服务器内容
5. 使用Xpath提取图片地址和名字
6. 根据图片地址提取图片内容
7. 下载图片，并保存

项目代码

```
import requests
from  lxml import etree
import time
import os
# 彼岸桌面
# 1.获取url链接地址
# def get_url():
#     url = "http://www.netbian.com/"
#     return url
def get_url():
    url_list = ["http://www.netbian.com/"]
    for i in range(2,100):
        url_list.append("http://www.netbian.com/index_{}.htm".format(i))
    return url_list
# 2.发起响应
def get_response(url):
    time.sleep(0.3)
    response = requests.get(url)
    content = response.content
    return content
# 3.提取数据列表
def get_content(content):
    html_str = etree.HTML(content.decode("gbk"))
    content_list = []
    li_list = html_str.xpath(".//div[@class='list']/ul/li")
    # print(li_list)
    # 4.提取图片的url地址和图片名
    for li in li_list:
        item = {}
```

```
        item = {}
        item["name"] = li.xpath("./a/b/text()")
        item["img_href"]  = li.xpath("./a/img/@src")
        if item["name"]:
            content_list.append(item)
    return content_list

#4下载并保存图片
def download_img(content_list):
    for content in content_list:
        img_name = content["name"][0]
        path_img = "./img"
        if os.path.exists(path_img):
            pass
        else:
            os.mkdir(path_img)
        with open(path_img+"/{}.jpg".format(img_name),"wb") as f:
            f.write(get_response(content["img_href"][0]))
            print("{}.....下载成功".format(img_name))

# 运行函数
def run():
    # url = get_url()
    url_list = get_url()
    for url in url_list:
        content = get_response(url)

        content_list = get_content(content)

        download_img(content_list)

if __name__ == '__main__':
    run()
```

运行结果：



项目分析:

此项目是对Xpath语法的应用和总结，需要有扎实的Python语言基础.重点在爬虫实施步骤和Xpath语法规则

2 Echarts图形化显示

2.1 Echarts认识

ECharts 是一个使用 JavaScript 实现的开源可视化库，涵盖各行业图表，满足各种需求。

ECharts 遵循 Apache-2.0 开源协议，免费商用。

ECharts 兼容当前绝大部分浏览器（IE8/9/10/11，Chrome，Firefox，Safari 等）及兼容多种设备，可随时随地任性展示。

2.2 Echarts 使用

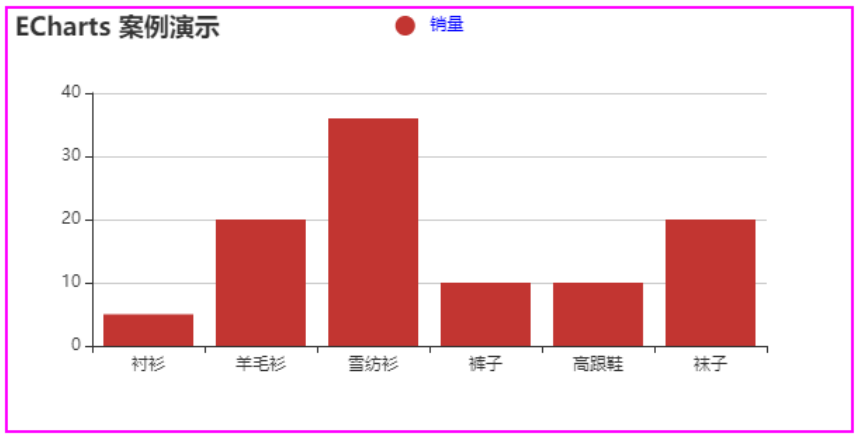
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <script src="echarts.min.js"></script>
</head>
<body>

</body>
</html>
```

2.3 Echarts语法--柱状图

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
  <script src="echarts.min.js"></script>
  <style>
    #echar{
      /*盒子宽度*/
      width: 600px;
      /*盒子高度*/
      height: 300px;
      /*盒子边框*/
```

```
        /*盒子边框*/
        border: 2px solid magenta;
    }
  </style>
</head>
<body>
<div id="echar"></div>
</body>
<script>
// 基于准备好的dom，初始化echarts实例
var myChart = echarts.init(document.getElementById('echar'));
// 绘制图表
myChart.setOption({
  //为图表配置标题：
  title: {
    text: 'ECharts 案例演示'
  },
  // 图例组件，图例组件展现了不同系列的标记(symbol)，颜色和名字。
  legend: {
        data: [{
          name: '销量',
          // 强制设置图形为圆。
          icon: 'circle',
          // 设置文本为；蓝色
          textStyle: {
            color: 'blue'
          },
          left:"right"
        }]
    },
  //X 轴  配置要在 X 轴显示的项：
  xAxis: {
    data: ['衬衫', '羊毛衫', '雪纺衫', '裤子', '高跟鞋', '袜子']
  },
  //Y 轴  配置要在 Y 轴显示的项。
  yAxis: {
  },
  //系列列表:每个系列通过 type 决定自己的图表类型：
  series: [{
    name: '销量',
    type: 'bar',
    data: [5, 20, 36, 10, 10, 20]
  }]
})
</script>
</html>
```

运行结果：



修改柱状图颜色：

```
//系列列表:每个系列通过 type 决定自己的图表类型:
series: [{
   name: '销量',
   type: 'bar',
   // 修改颜色
   itemStyle:{
      normal:{
         color:"green"
      }
   },
   data: [5, 20, 36, 10, 10, 20]
}]
```

修改图形：

- type: 'bar'：柱状/条形图
- type: 'line'：折线/面积图
- type: 'pie'：饼图
- type: 'scatter'：散点（气泡）图
- type: 'effectScatter'：带有涟漪特效动画的散点（气泡）
- type: 'radar'：雷达图
- type: 'tree'：树型图
- type: 'treemap'：树型图
- type: 'sunburst'：旭日图
- type: 'boxplot'：箱形图
- type: 'candlestick'：K线图
- type: 'heatmap'：热力图
- type: 'map'：地图
- type: 'parallel'：平行坐标系的系列
- type: 'lines'：线图
- type: 'graph'：关系图
- type: 'sankey'：桑基图
- type: 'funnel'：漏斗图

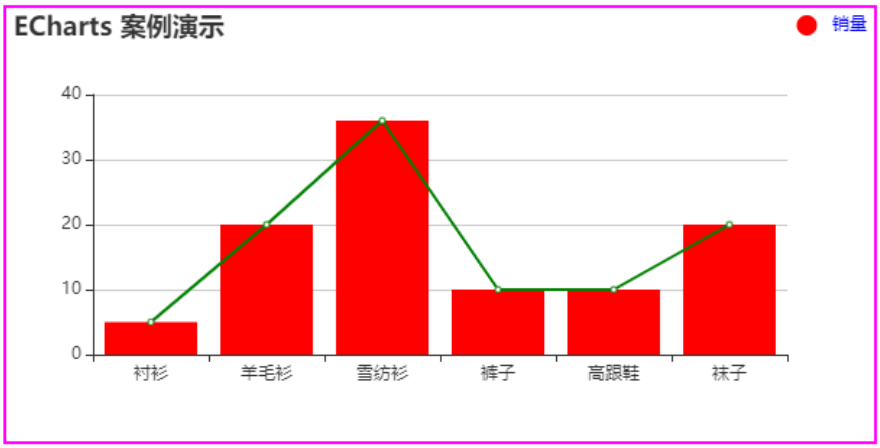
- type: 'gauge'：仪表盘
- type: 'pictorialBar'：象形柱图
- type: 'themeRiver'：主题河流

多图绘制

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>Title</title>
   <script src="echarts.min.js"></script>
   <style>
      #echar{
         /*盒子宽度*/
         width: 600px;
         /*盒子高度*/
         height: 300px;
         /*盒子边框*/
         border: 2px solid magenta;
      }
   </style>
</head>
<body>
<div id="echar"></div>
</body>
<script>
// 基于准备好的dom，初始化echarts实例
var myChart = echarts.init(document.getElementById('echar'));
// 绘制图表
myChart.setOption({
   //为图表配置标题:
   title: {
      text: 'ECharts 案例演示'
   },
   // 图例组件，图例组件展现了不同系列的标记(symbol)，颜色和名字。
   legend: {
         data: [{
            name: '销量',
            // 强制设置图形为圆。
            icon: 'circle',
            // 设置文本为；蓝色
            textStyle: {
               color: 'blue'
            }
         }],
      // 图例位置

      left:"right"
```

```
        left: right
        },
    //X 轴  配置要在 X 轴显示的项:
    xAxis: {
        data: ['衬衫', '羊毛衫', '雪纺衫', '裤子', '高跟鞋', '袜子']
    },
    //Y 轴  配置要在 Y 轴显示的项。
    yAxis: {
    },
    //系列列表:每个系列通过 type 决定自己的图表类型:
    series: [{
        name: '销量',
        type: 'bar',
        // 修改颜色
        itemStyle:{
            normal:{
                color:"red"
            }
        },
        data: [5, 20, 36, 10, 10, 20]
    },{
        name: '销量',
        type: 'line',
        // 修改颜色
        itemStyle:{
            normal:{
                color:"green"
            }
        },
        data: [5, 20, 36, 10, 10, 20]
    }
    ]
})
</script>
</html>
```

运行结果



2.4Python自带的简单服务器

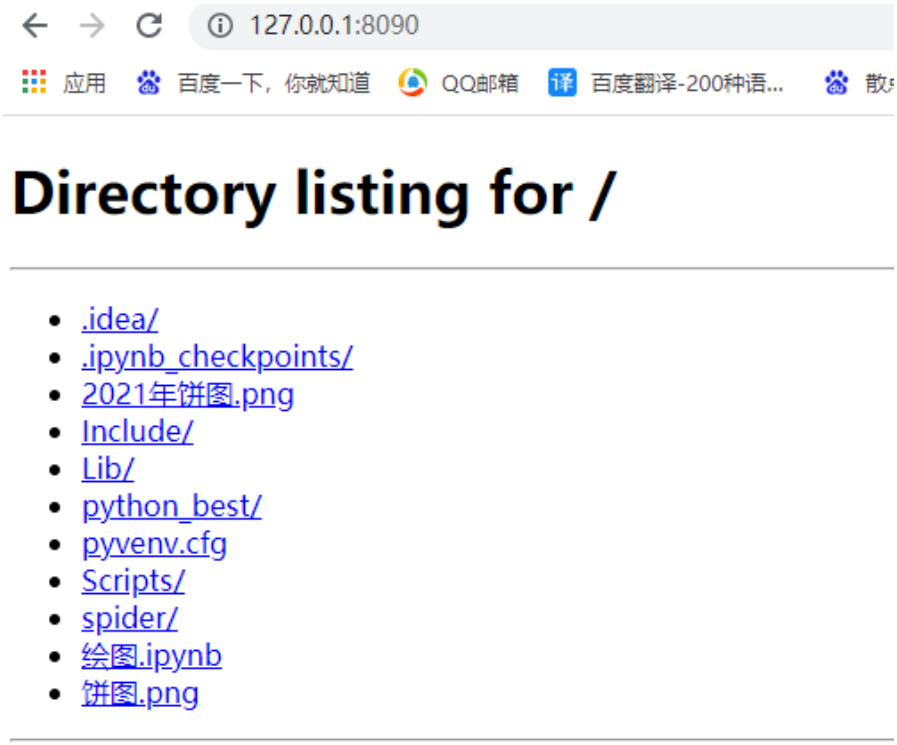
开启服务器语法：

```
python -m http.server 8090
```

```
Microsoft Windows [版本 10.0.19042.928]
(c) Microsoft Corporation。保留所有权利。

(python_best) C:\Users\ASUS\Desktop\html\python_best>python -m http.server 8090
Serving HTTP on 0.0.0.0 port 8090 (http://0.0.0.0:8090/) ...
```

浏览器访问语法：

```
http://127.0.0.1:8090
```



2.5 提取JSON文件并显示

demo.json文件

```
[
  {"value":235, "name":"视频广告"},
  {"value":274, "name":"联盟广告"},
  {"value":310, "name":"邮件营销"},
  {"value":335, "name":"直接访问"},
  {"value":400, "name":"搜索引擎"}
  ]
```

$.get，提取并显示：

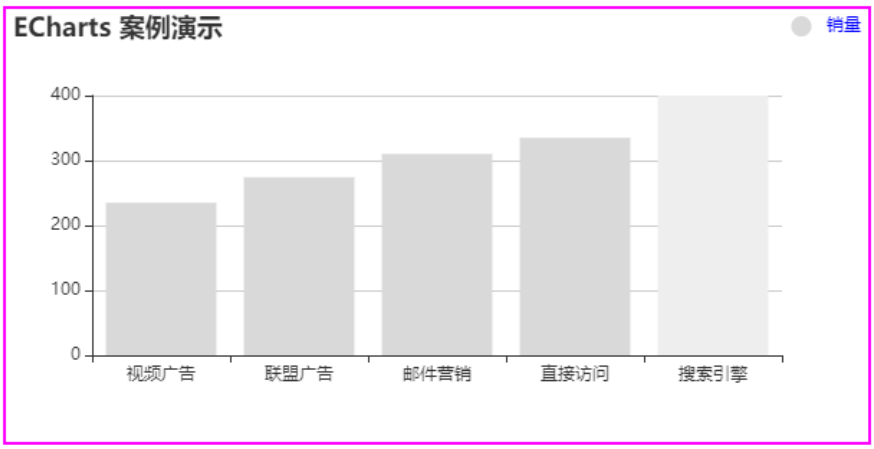
注意：使用$get，需要加载Jqury,并且需要开启Python服务器

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
  <script src="jquery.min.js"></script>
  <script src="echarts.min.js"></script>
  <style>
    #echar{
      /*盒子宽度*/
      width: 600px;
```

```
        width: 600px;
        /*盒子高度*/
        height: 300px;
        /*盒子边框*/
        border: 2px solid magenta;
    }
  </style>
</head>
<body>
<div id="echar"></div>
</body>
<script>
// 基于准备好的dom，初始化echarts实例
var myChart = echarts.init(document.getElementById('echar'));

$.get("./demo.json",function (data) {
  // console.log(data); # 获取的data数据
  var x_list = [];
  var y_list = [];
  for (var i=0;i<data.length;i++){
    x_list.push(data[i].name);
    y_list.push(data[i].value);
  }
  // 绘制图表
  myChart.setOption({
    //为图表配置标题：
    title: {
      text: 'ECharts 案例演示'
    },
    // 图例组件，图例组件展现了不同系列的标记(symbol)，颜色和名字。
    legend: {
        data: [{
          name: '销量',
          // 强制设置图形为圆。
          icon: 'circle',
          // 设置文本为；蓝色
          textStyle: {
            color: 'blue'
          }
        }],
      // 图里位置
      left:"right"
      },
    //X 轴 配置要在 X 轴显示的项：
    xAxis: {
      data: x_list
    },
    //Y 轴 配置要在 Y 轴显示的项。

    yAxis: {
```

```
    yAxis: {
    },
    //系列列表:每个系列通过 type 决定自己的图表类型:
    series: [{
        name: '销量',
        type: 'bar',
        // 修改颜色
        itemStyle:{
            normal:{
                // 十六进制颜色表示方式
                color:"#d9d9d9"
            }
        },
        data: y_list
    }
    ]
  })
});

</script>
</html>
```

显示效果：



2.6 项目实训--全国各个城市未来七日温度变化图

项目需求：

需要使用到Python中爬虫技术，根据输入的城市名，获取到全国各个城市的天气数据，将数据保存成JSON文件，并使用Echarts进行可视化显示

实施步骤：

1. 获取全国各个成绩的JS表
2. 根据输入城市名，判断是否，在JS表中

3. 在JS提取不同城市的ID
4. 根据ID获取七日气温变化图url地址
5. 发起响应，获取数据
6. 提取未来七日温度数据
7. 保存成JSON文件
8. 开始Python服务器
9. 创建HTML文件
10. 使用$.get获取天器的JSON数据
11. 使用Echarts绘制最高温度和最低温度，未来七日温度折线图

项目代码

weather_spider.py文件:

```
import requests
import json
from lxml import etree
import os
# 输入天气绘制成表
# 城市地址
def get_city_list():
    city_url = "https://j.i8tq.com/weather2020/search/city.js"
    response = requests.get(city_url)
    city_js = response.content.decode()
    citys = city_js.split("=")[-1]
    # print(citys)
    city_dic = json.loads(citys)
    # print(city_dic.items())
    city_list = []
    for k,v in city_dic.items():
        # print(k)
        # print(v)
        for k1,v1 in v.items():
            # print(v1)
            for k2, v2 in v1.items():
                # print(v2)
                city_list.append(v2)
    return city_list

# city_list 所有天气城市
def get_city_url(city_list):
    input_city = input("请输入你要查询的地区天气：")
    status = False
    for city in city_list:
        if input_city == city["NAMECN"]:
            status = True


            city_urls =
```

```
        city_urls =
"http://www.weather.com.cn/weather/{}.shtml".format(city["AREAID"])
        print("正则搜索你要查询 {} 的天气....".format(input_city))
        return city_urls

# 发起响应
def get_response(url):
    response = requests.get(url)
    content = response.content.decode()
    return content

# 提取数据列表
def get_content_list(content):
    html_str = etree.HTML(content)
    # 地址
    city = html_str.xpath("//div[@class='crumbs fl']/a[2]/text()")[0]
    town = html_str.xpath("//div[@class='crumbs fl']/a[3]/text()")[0]
    # print(city,town)

    li_list = html_str.xpath("//ul[@class='t clearfix']/li")
    weather_list = []
    for li in li_list:
        item = {}
        item["address"] = city+">"+town
        item["week_date"] = li.xpath("./h1/text()")[0]
        item["max_temp"]=li.xpath("./p[@class='tem']/span/text()")[0]
        item["min_temp"]=li.xpath("./p[@class='tem']/i/text()")[0].split("℃")[0]
        item["wea"] = li.xpath("./p[@class='wea']/text()")[0]
        # print(item)
        weather_list.append(item)
    return weather_list

#保存成json文件
def save_json(weather_list):
    file_path = "./source"
    if os.path.exists(file_path):
        with open(file_path+"/weather.json","w",encoding="utf-8") as f:
          json.dump(weather_list,f,ensure_ascii=False,indent=2)
    else:
        os.mkdir(file_path)
        with open(file_path + "/weather.json", "w", encoding="utf-8") as f:
            json.dump(weather_list,f, ensure_ascii=False, indent=2)

def run():
    city_list = get_city_list()
    url = get_city_url(city_list)
    # url = "http://www.weather.com.cn/weather/101010200.shtml"
    if url:

        #发起响应
```

```
        #发起响应
        content = get_response(url)
        #提取数据
        weather_list = get_content_list(content)
        #报存
        save_json(weather_list)

    else:
        print("你输入的城市地区名有误，未获取到需要的天气.....")

if __name__ == '__main__':
    run()
```

服务器运行：

```
(python_best) C:\Users\ASUS\Desktop\html\python_best>python -m http.server 8090
Serving HTTP on 0.0.0.0 port 8090 (http://0.0.0.0:8090/) ...
127.0.0.1 - - [12/May/2021 10:51:31] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [12/May/2021 10:51:31] code 404, message File not found
127.0.0.1 - - [12/May/2021 10:51:31] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [12/May/2021 11:02:07] "GET /spider HTTP/1.1" 301 -
127.0.0.1 - - [12/May/2021 11:02:07] "GET /spider/ HTTP/1.1" 200 -
```

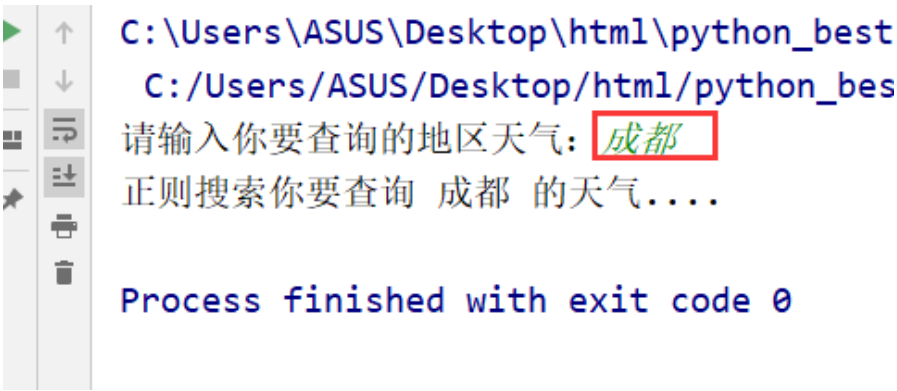
weather_echarts.html

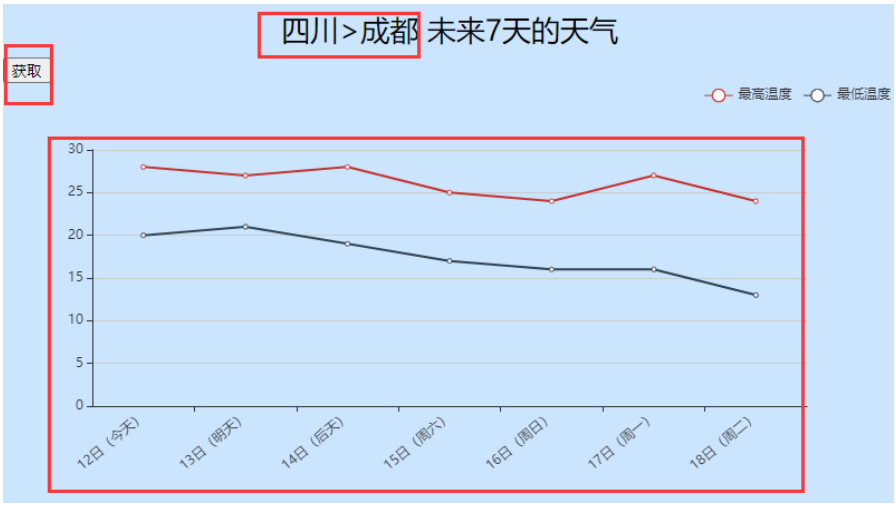
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>weather</title>
    <style>
        .wea-box{
            width: 800px;
            height: 450px;
            margin: 100px auto;
            background-color: rgba(153,204,255,0.5);
        }
        .wea-tit{
            font-size: 26px;
            text-align: center;
            line-height: 50px;
        }
        .wea-map{
            height: 350px;
            /*border: 2px solid red;*/
        }
    </style>
</head>
<body>

<div class="wea-box">
```

```
<div class="wea-box">
<!--  地址-->
  <div class="wea-tit" >
    <span id="wea_tit"></span> 未来7天的天气
  </div>
  <input type="button" value="获取" id="btn">
<!-- 可视化-->
  <div class="wea-map" id="wea_map"></div>
</div>
<script src="jquery.min.js"></script>
<script src="echarts.min.js"></script>
<script>
  $("#btn").click(function(){
    var wea_map = document.getElementById("wea_map");
    var my_map = echarts.init(wea_map);
    var wea_date = [];
    var max_temp = [];
    var min_temp = [];
  $.get("./source/weather.json",function (weather_list) {
    // console.log(weather_list);
    $("#wea_tit").html(weather_list[0]["address"]);
    for (var i=0;i<weather_list.length;i++){
      wea_date.push(weather_list[i]["week_date"]);
      max_temp.push(weather_list[i]["max_temp"]);
      min_temp.push(weather_list[i]["min_temp"])
    }
    console.log(wea_date);
    console.log(max_temp);
    var option1 = {
    xAxis:{
      data:wea_date,
    //  斜体显示
      axisLabel:{
        rotate:40
      }
    },
    //  图例
    legend:{
      left:"right"
    },
    yAxis:{

    },
    series:[{
      name:"最高温度",
      type:"line",
      data:max_temp
    },

      {
```

```
        {
            name:"最低温度",
            type:"line",
            data: min_temp

        }]
    };

    my_map.setOption(option1);
});
})


</script>
</body>
</html>
```

运行结果：





项目分析：

重点是在：爬虫--》服务器--》可视化，实现流程提别重要，需要能熟练使用XPath语句，网页分析和Echars绘图语法。

3.习题

编程题：

使用Python爬虫技术，获取网站：https://www.runoob.com/try/xml/books.xml，中class=en的title节点文字和price节点中的文字，并保存成JSON文件，开启服务器，使用Echarts绘制柱状图