- 日产的Python教学
  - Python简介
  - 安装Python
  - 第一个日产的Python程序
  - 变量和数据类型
  - 字符串
  - 列表和元组
  - 分支结构
  - 字典和集合
  - 内置函数
  - 模块
  - 面向对象编程
  - 异常处理和调试程序
  - 文件和目录操作
  - 多线程
  - 多进程
  - 网络编程
  - 正则表达式
  - 爬虫开发基础
  - requests
  - bs4
  - selenium
  - scrapy
  - 数据分析
  - numpy
  - pandas
  - matplotlib
  - 机器学习
  - sklearn
  - 深度学习
  - keras
  - tensorflow
  - 数据库开发
  - mysql
  - redis
  - mongodb
  - 网页开发
  - flask
  - 日产的Web开发
  - 数据库工程师教程
  - sql server
  - 数学教程
  - 微信编程
  - 小程序开发
  - 小红书编程教程
  - Linux开发
  - 运维
  - FAQ
  - 更新日志
  - 关于我

文件目录

DYNA ▼ (1.0.1 [2022])

Python 入门

SSL

[ssl_error: 0x1416f086: error:1416f086:SSL routines:tls_process_server_certificate:certificate verify failed:ssl/t1_lib.c:2580:]

DNS 查询失败时出现错误，如何确保程序能正确处理？正式环境使用每日任务，每日一次检查下载内容的MD5。

DOM vs SAX

解析大型文件时，DOM 会消耗 GB 级内存，而 SAX 流式解析可保证内存占用恒定。解析时，建议采用流式解析器（如 SAX 或 iterparse），边读边解析，边写入目标文件，解析完一条，处理完一条，内存占用保持恒定。

正常情况下，文件体积大，解析时间长，因为内存交换会拖慢速度。

但 iterparse 和 SAX 解析速度通常比 DOM 更快，因为省去了构建内存对象树的开销。iterparse 和 iwalk 的区别：iterparse 默认只监听 start 和 end 事件，而 iwalk 可以通过参数监听更多事件（如 start-ns、end-ns 等）。

正常情况下，文件体积大，解析时间长，因为内存交换会拖慢速度。

但 iterparse 和 SAX 解析速度通常比 DOM 更快，因为省去了构建内存对象树的开销。iterparse 默认只监听 start 和 end 事件，而 iwalk 可以通过参数监听更多事件（如 start-ns、end-ns 等）。

```
for event, elem in iwalk(f, tag='item'):
    if event == 'start':
        print('开始处理 item')
    elif event == 'end':
        print('结束处理 item')
    elem.clear()
```

解析文件时，可以通过参数监听更多事件：

```
for event, elem in iterparse(f, events=('start', 'end', 'start-ns', 'end-ns')):
    if event == 'start-ns':
        print('开始处理命名空间')
    elif event == 'end-ns':
        print('结束处理命名空间')
    elif event == 'start':
        print('开始处理 element')
    elif event == 'end':
        print('结束处理 element')
    elem.clear()
```

如果需要处理多种事件，建议用 iterparse 或 iwalk 配合多个事件监听，比如监听 start 和 end 来统计嵌套深度，或监听 start-ns 和 end-ns 来处理命名空间。

小结

解析大文件，流式解析是唯一可行方案。iterparse 和 iwalk 的区别在于默认监听的事件类型，iwalk 更灵活，可以监听更多事件。

练习

请用 iterparse 或 iwalk 解析 weather.xml，统计出现了多少个城市（tag 为 city）。

提示：

- 默认情况下，iterparse 只监听 start 和 end 事件，城市信息在 start 事件时就已经可用，不需要等到 end
- 使用 elem.clear() 及时释放内存，即使只统计数量，也要保持内存占用恒定

```
from xml.parsing.sax.sax import ParseException
from urllib.request import urlopen
from io import StringIO

# 模拟 weather.xml 的内容
weather_data = StringIO('''<?xml version="1.0"?>
<weather_report>
    <city_list>
        <city_info_1>
            <name_cn_1>北京</name_cn_1>
            <temperature_1>15</temperature_1>
            <weather_1>晴</weather_1>
            <wind_1>东风 3级</wind_1>
            <humidity_1>45%</humidity_1>
            <aqi_1>50</aqi_1>
            <pm25_1>35</pm25_1>
            <suggestion_1>适宜户外活动</suggestion_1>
        </city_info_1>
        <city_info_2>
            <name_cn_2>上海</name_cn_2>
            <temperature_2>18</temperature_2>
            <weather_2>多云</weather_2>
            <wind_2>西风 2级</wind_2>
            <humidity_2>60%</humidity_2>
            <aqi_2>80</aqi_2>
            <pm25_2>55</pm25_2>
            <suggestion_2>注意防晒</suggestion_2>
        </city_info_2>
        <city_info_3>
            <name_cn_3>广州</name_cn_3>
            <temperature_3>22</temperature_3>
            <weather_3>雾</weather_3>
            <wind_3>北风 1级</wind_3>
            <humidity_3>85%</humidity_3>
            <aqi_3>120</aqi_3>
            <pm25_3>75</pm25_3>
            <suggestion_3>建议佩戴口罩</suggestion_3>
        </city_info_3>
    </city_list>
</weather_report>''')

# 解析文件，统计城市数量
for event, elem in iterparse(weather_data, events=('start', 'end')):
    if event == 'start' and elem.tag == 'city':
        print('发现一个城市')
    elif event == 'end' and elem.tag == 'city':
        print('城市信息解析完成')
    elem.clear()
```

小结

iterparse 是处理大文件的利器，内存占用恒定，适合生产环境。

练习

请用 iterparse