lab1 html Parser

1.实验概览

本次实验使用python自带的urllib库实现通过自动程序对html网站进行访问并返回相应结果，并使用beautifulsoap库对返回的结果进行解析和筛选，以得到期望的结果，其中可以使用正则表达式来对解析的结果进行更为精确的筛选。

2.实验环境

本实验基于 Docker 中的 DEV-container--SJTUEE208 具体使用到的环境为：

1. Ubuntu 20.04.5 LTS
2. Vscode python的urllib库, beautiful soap库以及re库

3.练习

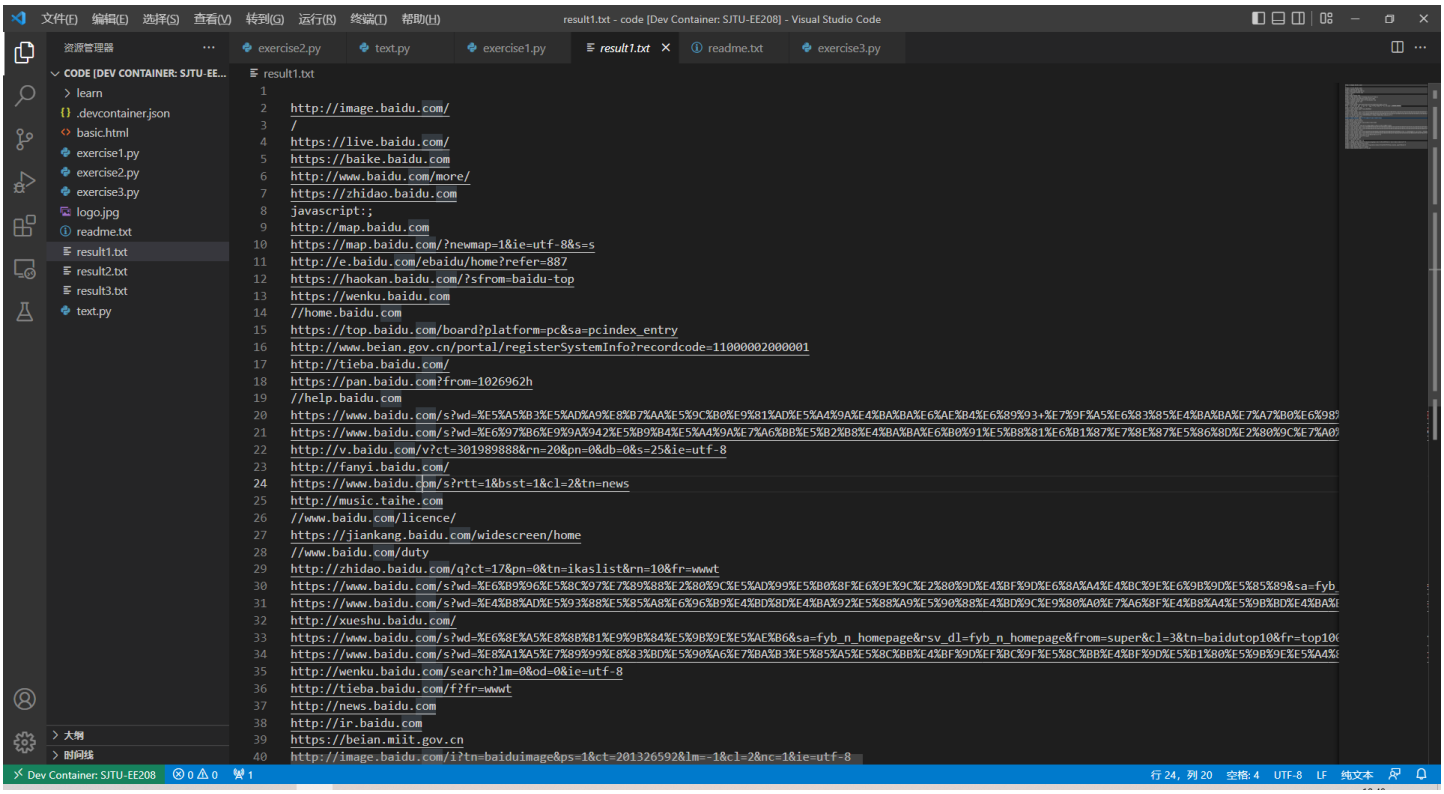
3.1 练习1

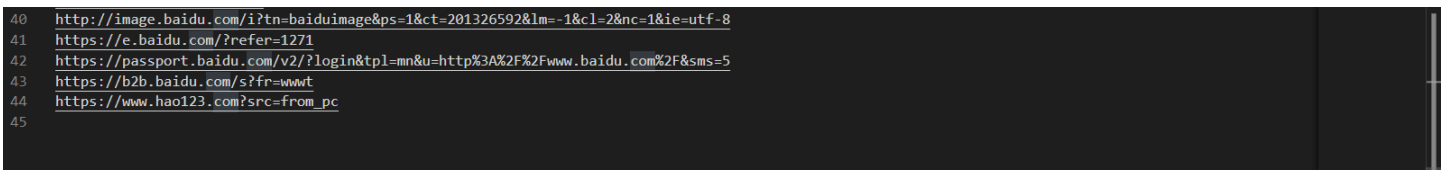
解决思路：通过urllib.urlopen方法对网页进行访问并返回相应结果，使用beautiful soap对返回结果进行解析，在parseURL函数中使用findAll方法和get方法来获得原网页中的所有超链接的url，并将其加入urlset中，最后通过write_outputs函数将所得超链接写入result1.txt中。

核心代码展示如下：

```
def parseURL(content):
    urlset = set()
    soup = BeautifulSoup(content, 'html.parser')
    for i in soup.findAll('a'):
        urlset.add(i.get('href',''))
    return urlset
```

运行结果如下：





3.2 练习2

解决思路：用户输入想要访问的网站的http地址，访问方式和解析方式基本同练习1，唯一不同的是通过urllib.request.Request.add_header() 方法添加了请求标头，以通过部分网站对爬虫程序的反爬。

同时，对于一些目前暂时无法通过的反爬技术，我参考知乎：python爬虫入门：HTTPError异常处理加入了一个catch

结果写入在result2中

核心代码展示如下：

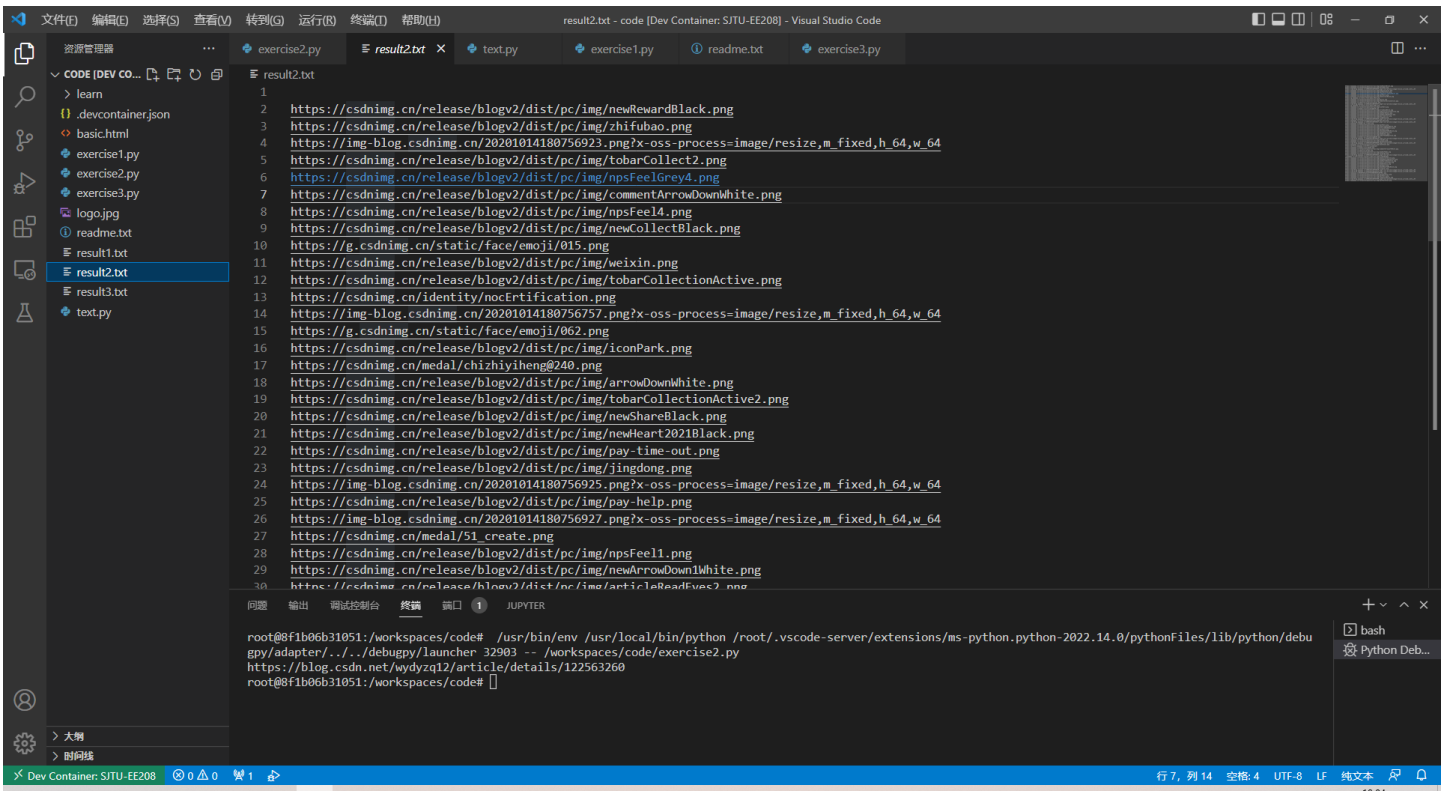
```
def parseURL(content):
    urlset = set()
    soup = BeautifulSoup(content, 'html.parser')
    for i in soup.findAll('img'):
        toAdd = i.get('src', '')
        urlset.add(toAdd)
    return urlset
```

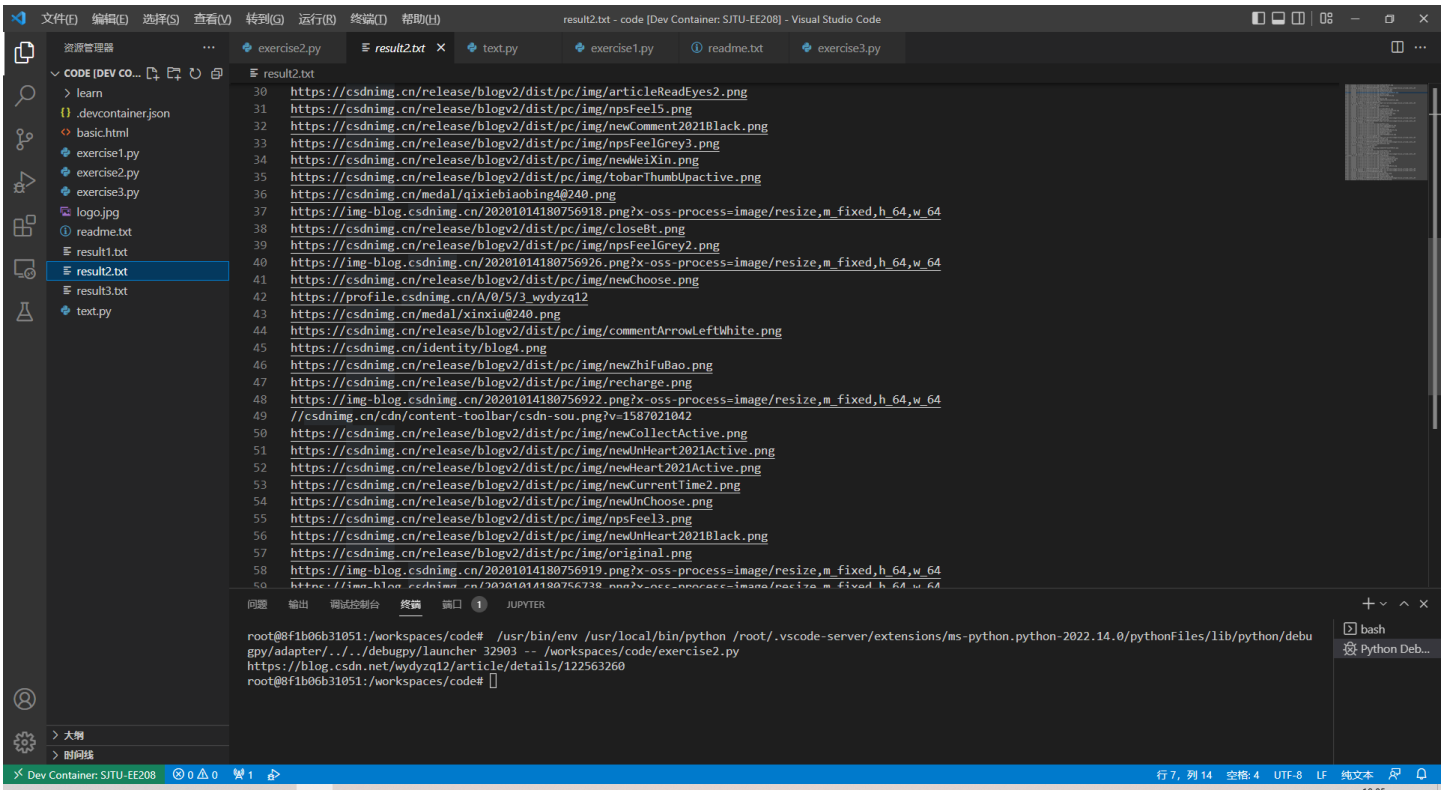
```
def main():
```

```
    url = input()
    req = urllib.request.Request(url)
    req.add_header('User-Agent','Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleW
    try:
        content = urllib.request.urlopen(req).read()
        urlSet = parseURL(content)
        write_outputs(urlSet, "result2.txt")
    except urllib.error.HTTPError:
        print('本次访问失败')
```

运行结果如下：

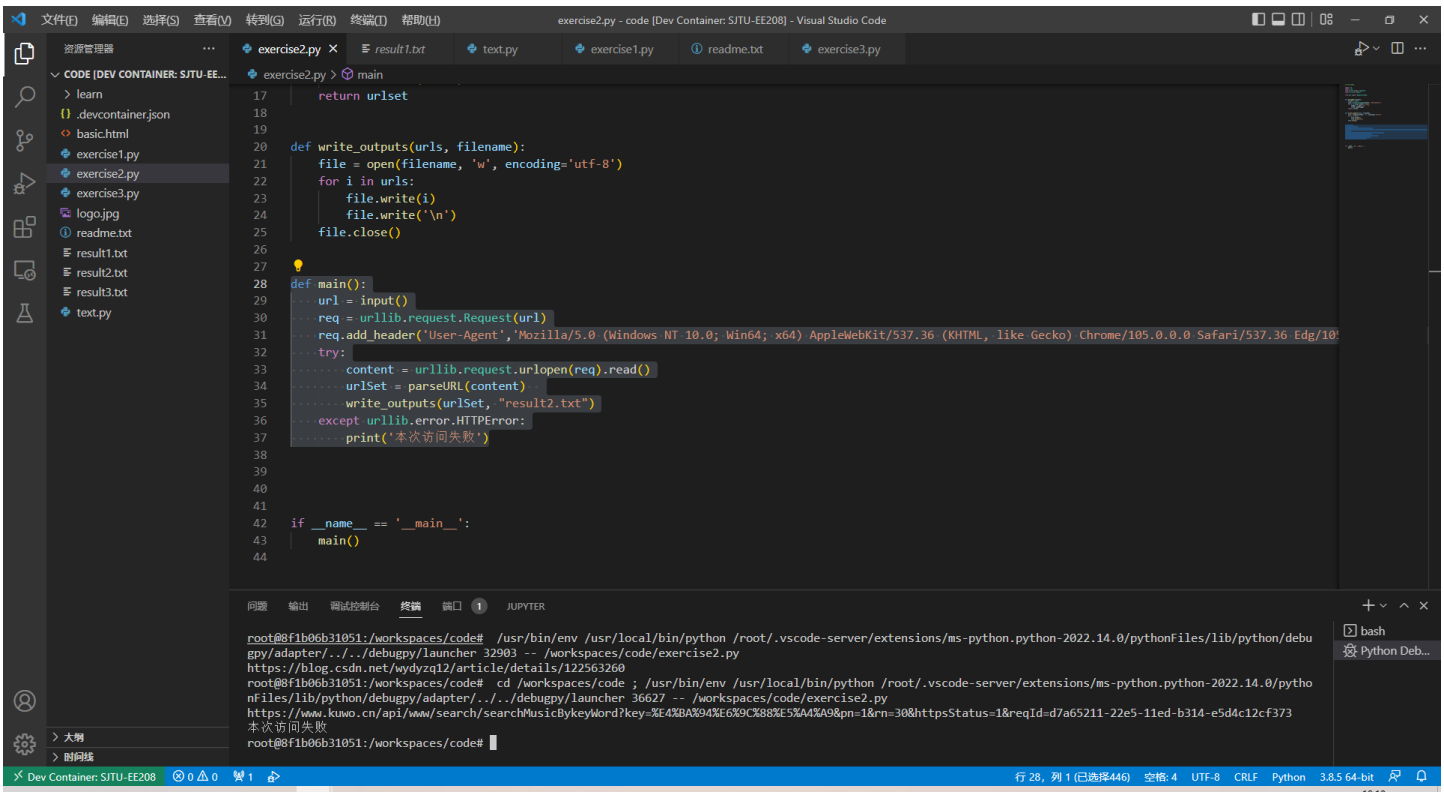
访问成功时：





（注：这里使用的链接是Python爬虫：测试网址是否请求成功）

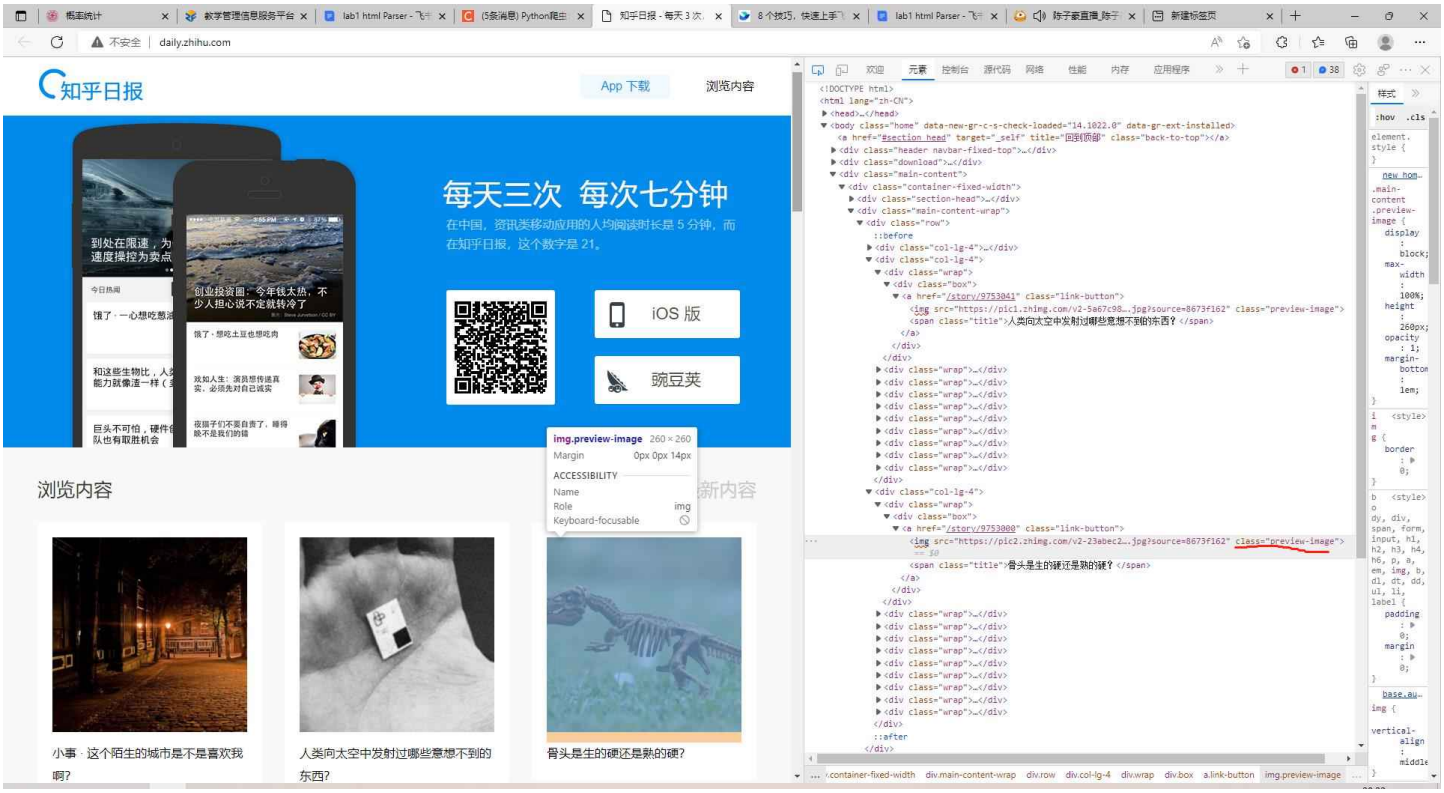
访问失败时：



（注：这里使用的链接是https://www.kuwo.cn/api/www/search/searchMusicBykeyWord?key=%E4%BA%94%E6%9C%88%E5%A4%A9&pn=1&rn=30&httpsStatus=1&reqId=d7a65211-22e5-11ed-b314-e5d4c12cf373）

3.3 练习3

解决思路：首先进入知乎日报页面，打开F12审查我们要获得的链接的源代码



首先我们发现需要获取的图片地址所对应的html代码中，所属的类是‘preview-image’类，进而发现图片所链接的网页地址存放在图片链接的父节点中，同时所需要提取的文本存放在图片链接的姐妹节点中，于是我的方案是先通过findAll找出所有类名为‘preview-image’的链接，再通过访问父节点得到图片的超链接地址，最后通过访问父节点的contents[1].string来获得图片对应的文本。

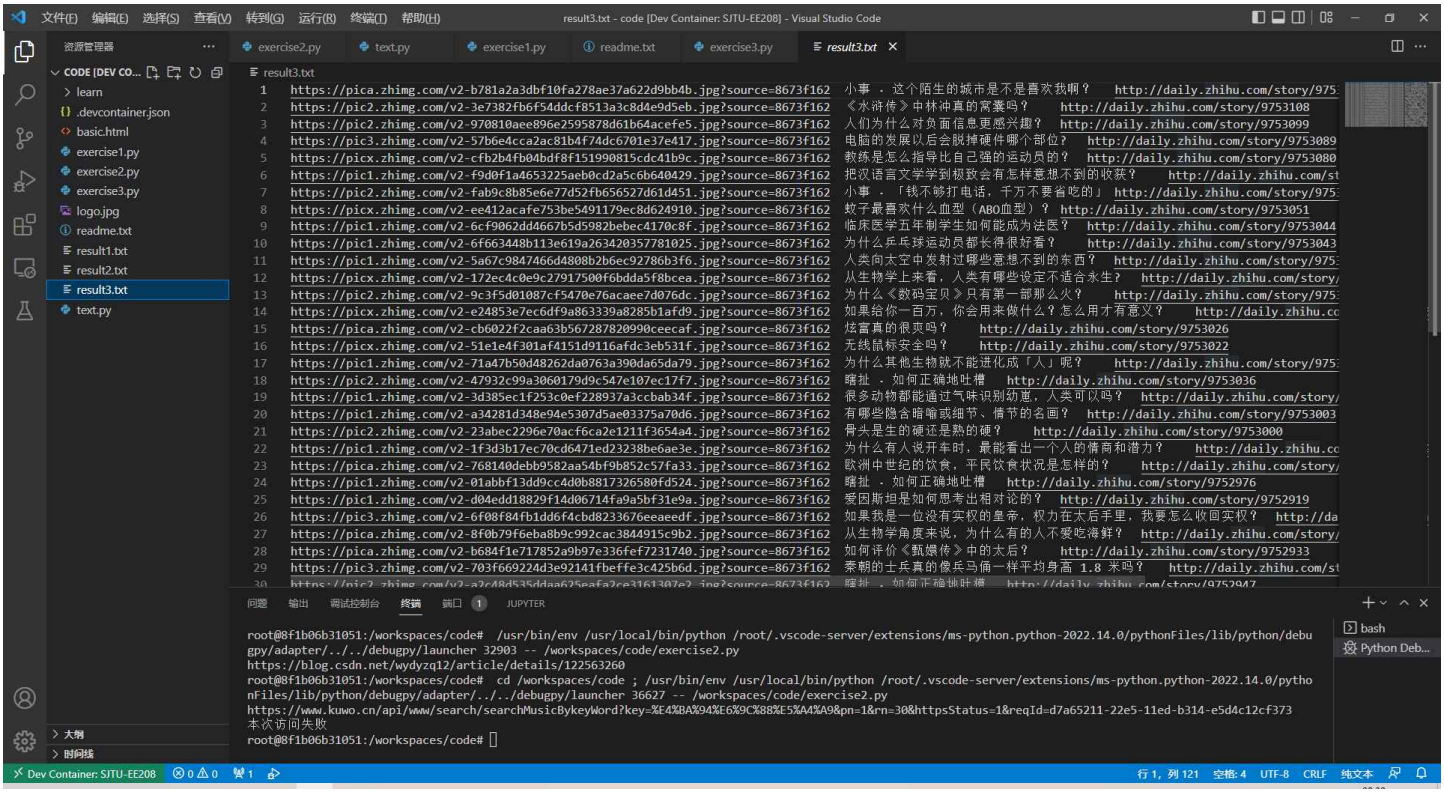
这里为了防止爬到一些同样属于‘preview-image’类却没有文本的图片，于是增加了一条判断语句。

最后将结果按格式写入在result3中

核心代码展示如下：

```
def parseZhihuDaily(content, url):
    zhihulist = list()
    soup = BeautifulSoup(content, 'html.parser')
    for i in soup.findAll('img', {'class' : 'preview-image'}):
        src = i.get('src','')
        templinkpage = i.parent.get('href', '')
        temptitle = i.parent.contents
        if (len(temptitle) >= 2 and templinkpage):
            title = temptitle[1]
            linkpage = urllib.parse.urljoin(url, templinkpage)
            zhihulist.append([src, title.string, linkpage])
    return zhihulist
```

运行结果展示如下

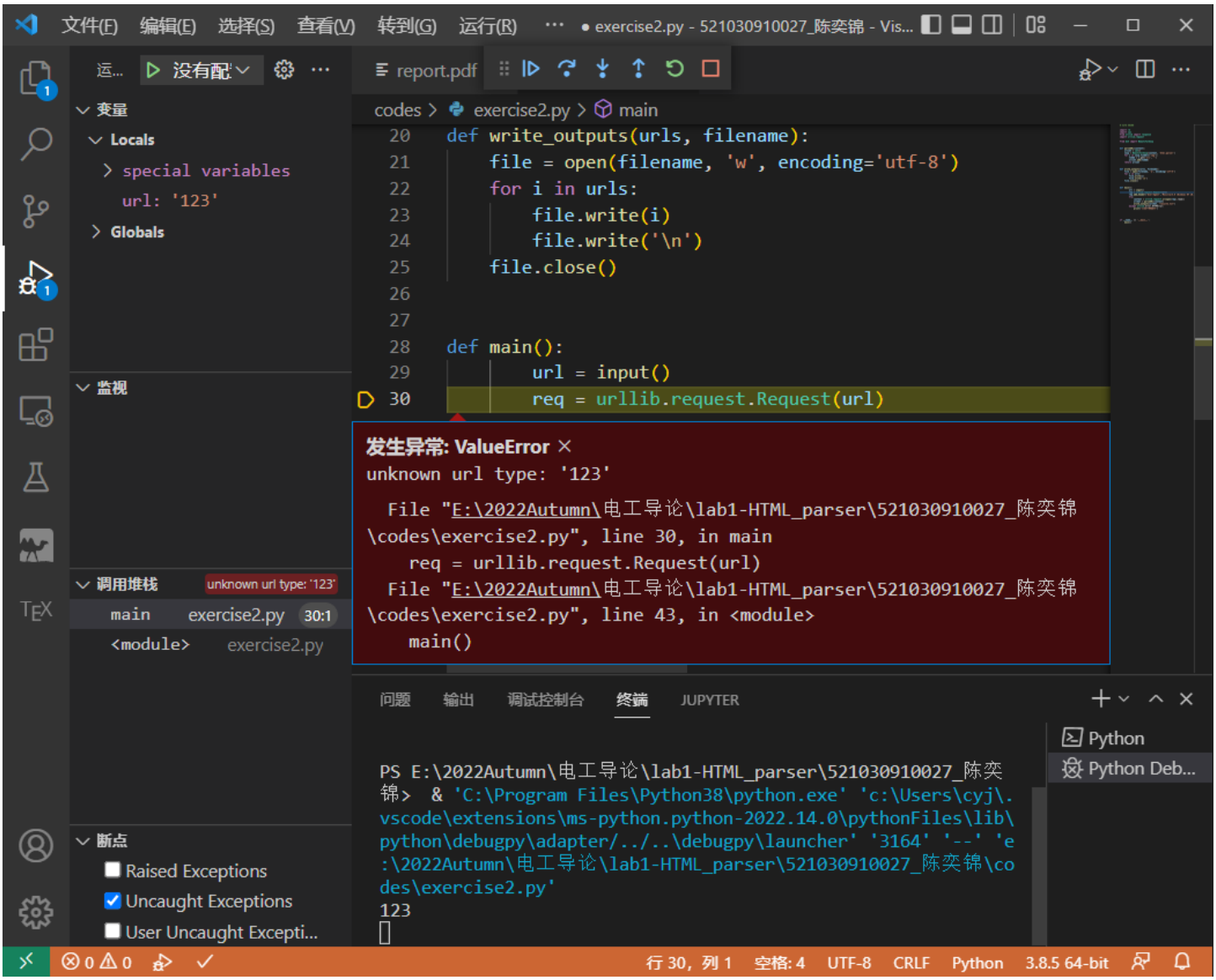


4.自己的分析与思考

1.爬取到的rhef有几种形式?

- E1：http://image.baidu.com/
- E2：https://live.baidu.com/
- E3: //help.baidu.com
- E4: javascript:;
- 其中E1，E2为可以正常访问的链接
- E3为相对于原网页的相对路径，如果想要正常访问的话需要在前面添加http:或者https:
- E4应该为链接到javascript的链接，用于进行一些按钮之类的交互行为，截取网页源码如下

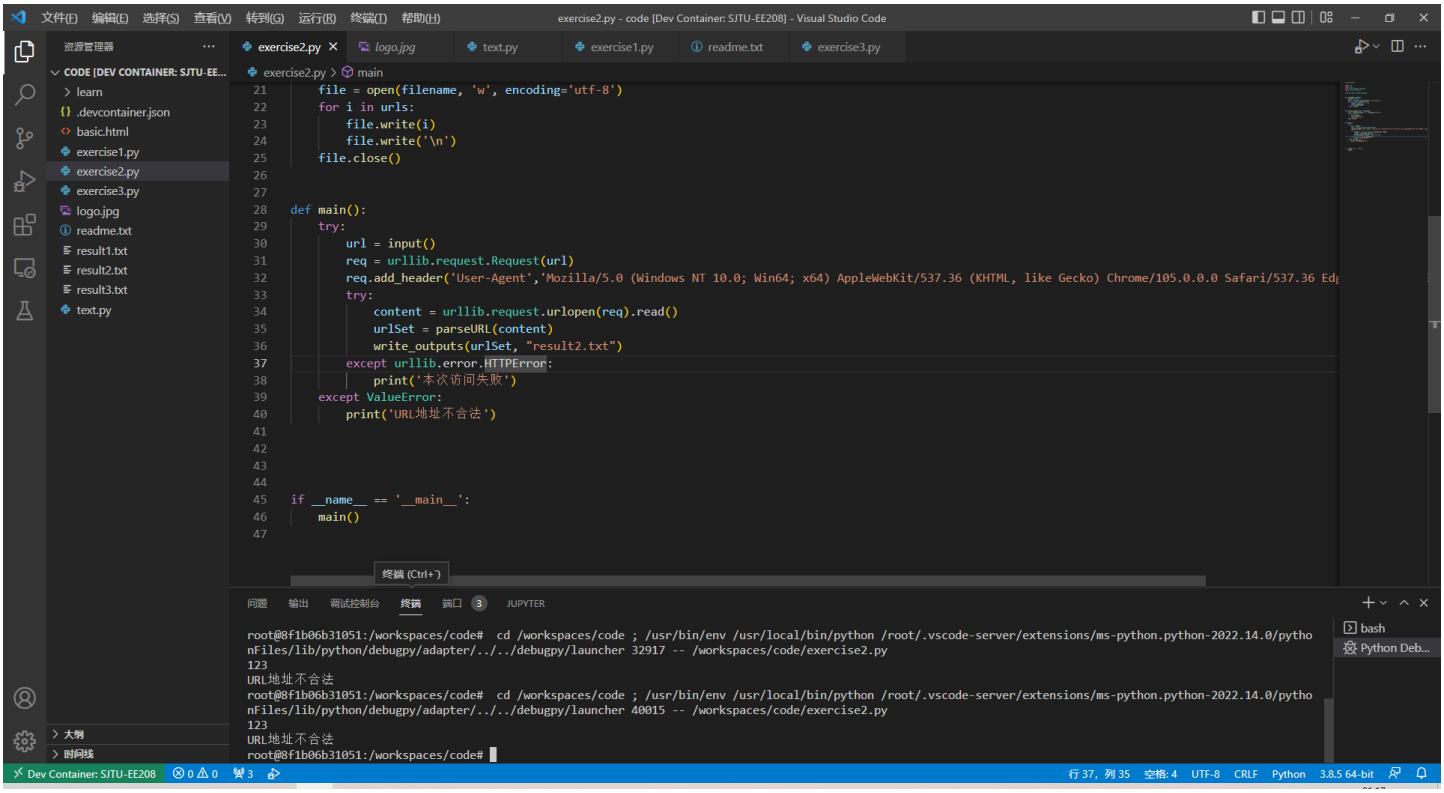
2.关于练习2中写的catch的问题，它好像只能处理HTTP地址合法但是网页无法访问的情况，如果输入一些不合法的HTTP地址就会出现如下图的情况



所以我就又加了一层try和catch，好像就有点复杂了

```python
def main():
    try:
        url = input()
        req = urllib.request.Request(url)
        req.add_header('User-Agent','Mozilla/5.0 (Windows NT 10.0; Win64; x64) Ap
        try:
            content = urllib.request.urlopen(req).read()
            urlSet = parseURL(content)
            write_outputs(urlSet, "result2.txt")
        except urllib.error.HTTPError:
            print('本次访问失败')
    except ValueError:
        print('URL地址不合法')
```

运行结果如下：



虽然但是我觉得也许用requests库然后直接判断它的statuscode是否等于两百应该比这个要稍微好一点，但是由于requests库不是python自带的于是我就没有把代码改成使用requests库的形式了。

所以说这算不算提一个建议就是用requests库会方便一点呢