一、 实验要求：

抓取 https://www.ccamlr.org/en/organisation/ccamlr-news 网站上的所有新闻的新闻数据（包括翻页），并将每一篇文章的链接、标题、时间和正文保存到 MYSQL 数据库中。

二、 实验环境：

Python 3.5.3，macOS Sierra 10.12.5，MySQL 14.14

用到的第三方库：requests (2.17.3), beautifulsoup4 (4.6.0), PyMySQL (0.7.11)

三、 问题分析：

1. 该网站的新闻分为多页，需要翻页：

   分析翻页时网址的不同，可以发现翻页操作就是将请求参数 page 加 1，因此可以用一个循环来翻页。

   https://www.ccamlr.org/en/organisation/ccamlr-news

   https://www.ccamlr.org/en/organisation/ccamlr-news?page=1

   https://www.ccamlr.org/en/organisation/ccamlr-news?page=2

   ...

   当在某一页无法获得新闻链接地址时，则循环结束。

2. 是否使用 Scrapy 框架？

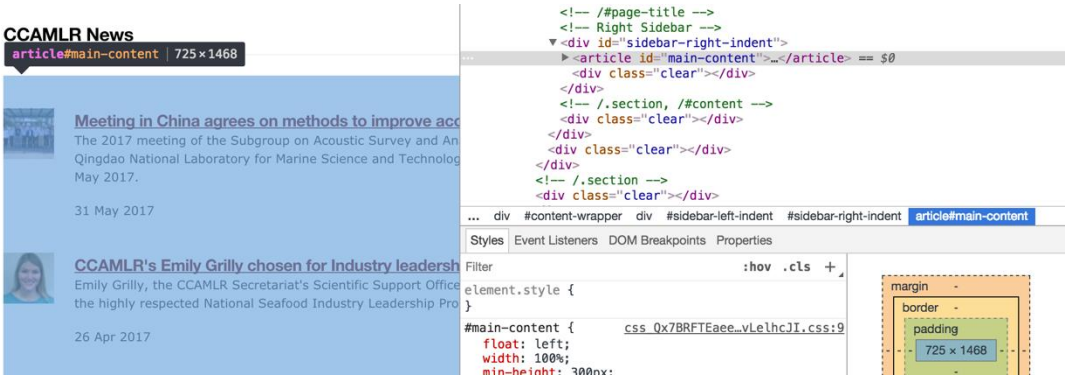
   Requests 库和 Scrapy 都可以进行页面请求和爬取，是 Python 爬虫的两个重要技术路线。Scrapy 是一个成熟的 Python 爬虫框架，它的并发性好，吞吐量高，是网站级爬虫的首选。Requests 是一个主要用于网页请求的优秀的第三方 Python 库，功能没一个框架强大，也无法相提并论，但是入门简单，容易定制，适合网页级别的信息抓取。

   通过对爬取网页的分析，发现所爬取的信息并非网站级别，顶多是确定的有限的网页个数而已。同时一个网页的信息也不多，效率上可以不考虑并发性。因此没必要因此使用复杂的 Scrapy 框架。再者，如果需要提高性能，可以自行对多线程编码提高并发性。

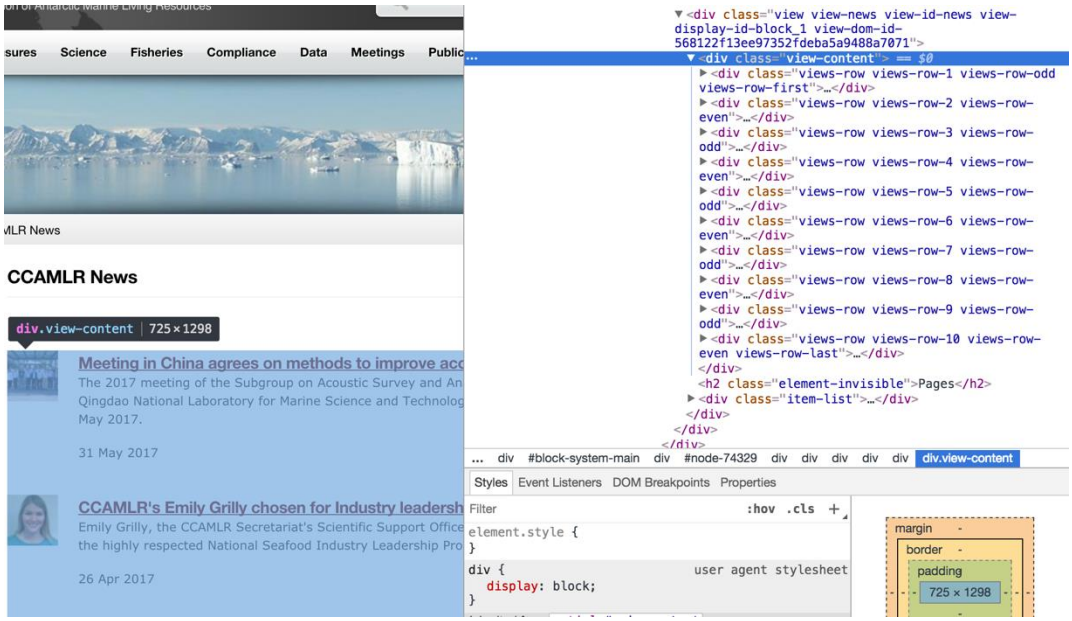
3. 如何定位内容所对应的标签？

   借助比较主流的谷歌浏览器，打开网页后通过谷歌开发者选项打开控制台，就可以通过鼠标指向内容区域，控制台自动显示对应标签的开头。

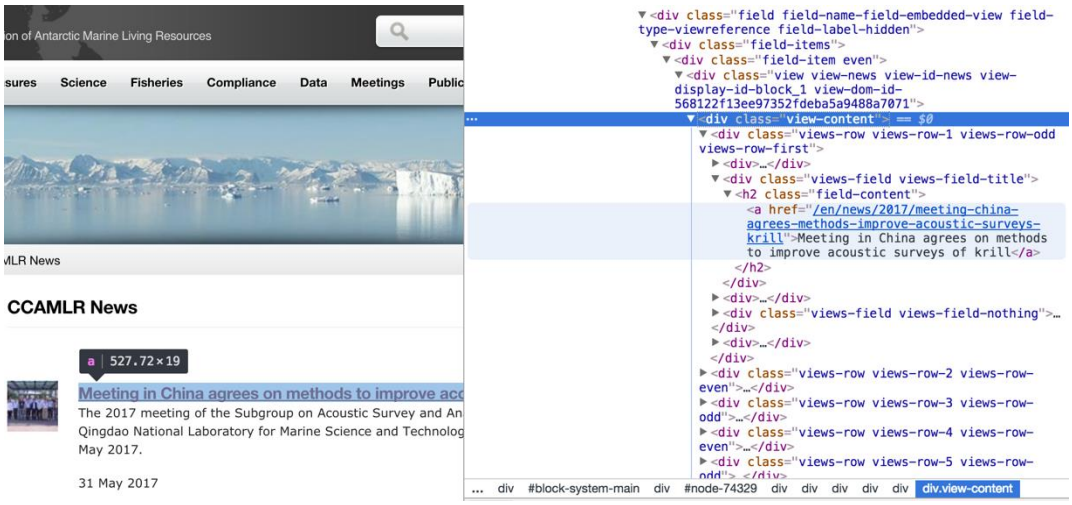
   每页的新闻列表在标签<article id=”main-content”>和</article>之间，而且这两个标签是唯一的，截取它们之间的内容。



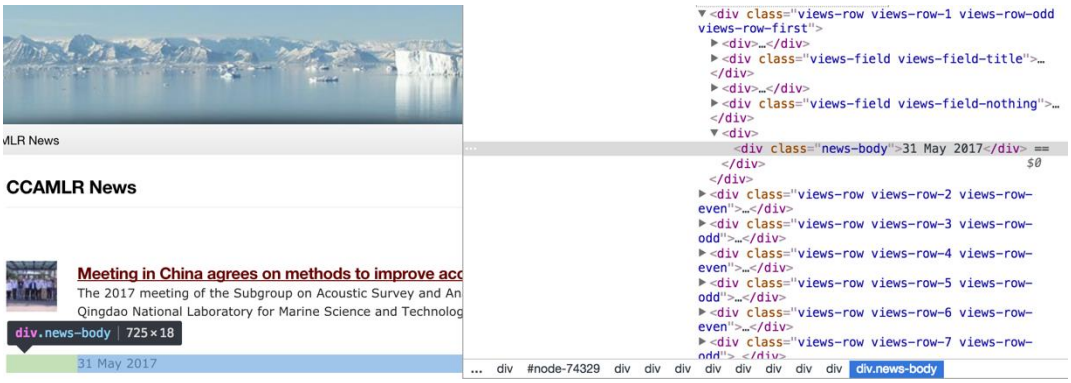
进一步来说是在标签 和 之间。每一条新闻的标签类名都是以 views-row 开始的，而且在列表段中只有新闻使用这种类名，根据类名分割列表获得每一条新闻。



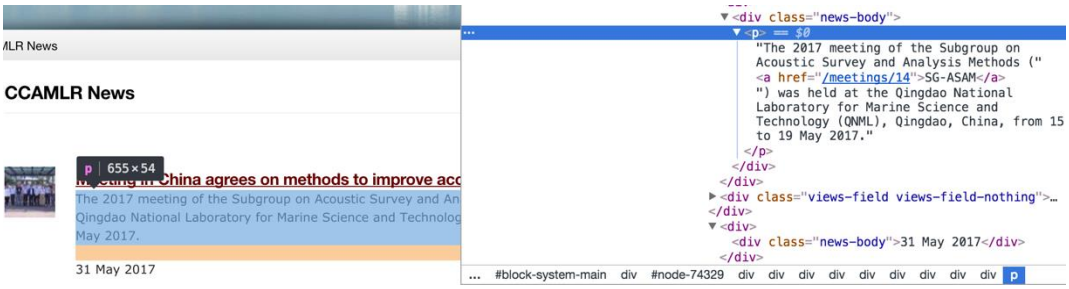
提取标题，要保证正则表达式匹配的唯一性。



提取时间，要保证正则表达式匹配的唯一性。



提取新闻正文。



4. 如何匹配标签？

正则表达式实验中我们使用正则表达式匹配标签，那个实验所用的到 html 文档结构相对简单，而且实验目的也简单，因此使用正则表达式可以轻松完成任务。然而这次实验所涉及的网页标签很多，结构也相对复杂。这时需要找到如问题 3 所提到的浏览器控制台一样的对 html 标签进行解析的库。在 Python 中第三方库中 beautifulsoup4 库就是一个实现解析 html 网页的库。除此之外它还提供各种标签遍历的方式，十分方面。

四、 实验过程、步骤及原始记录：

1. 程序流程：网页请求→解析网页→提取有用内容→写入数据库。
2. 网页请求：使用 requests 库中的 get 方法。

```
def getHTMLText(url, kv):
        try:
                r = requests.get(url, params=kv)
                r.raise_for_status() # 收集状态码，如果不是200则抛出异常
                r.encoding = r.apparent_encoding # 保证编码一致
                return r.text
        except:
                return ""
```

3. 解析网页→提取有用信息：思路和上述问题分析中的第三点一致。

```
def getNewsData(url, kv, db, succCount):
        html = getHTMLText(url, kv)
        soup = BeautifulSoup(html, "html.parser")

        # 查找新闻列表
        articlesoup = soup.find('article', {'id':'main-content'})
        pattern = re.compile(r'views-row.*')
        viewsoup = articlesoup.find('div', {'class':'view-content'})

        #遍历一个网页中的新闻
        for view in viewsoup.find_all('div', {'class':pattern}):
                try:
                        h2 = view.find('h2', {'class':'field-content'})# 先缩小范围
                        a = h2.find('a')
                        href = a.attrs['href'] # 找到新闻链接
                        title = a.string # 找到新闻标题
                        # 找到新闻时间
                        datetag = view.find_all('div', {'class':'news-body'})[-1]
                        date = datetag.string
                        #print(title + '\n' + href + '\n' + date,end='\n')
                        #succCount += 1
                        succCount = writeInDB(db, title, href, date, succCount)
                except:
                        continue
        return succCount
```

4. 获取网页数量：为了提高程序可用性。由于网页数量可能会增加，实验时只有 9 页，不代表一直都只有 9 页，为此写一个获取网页数量的函数：

```
def getPageCount(url, kv):
        html = getHTMLText(url, kv)
        soup = BeautifulSoup(html, "html.parser")
        pagecount = soup.find_all('li', {'class':'pager-item'})
        return len(pagecount)+1
```

5. 写出数据库：这部分相对简单，因为函数式编程要求分工明确，连接数据库的工作并非由这个函数完成，因此这个函数的核心就是一个 sql 语句。

```
def writeInDB(db, title, href, date, succCount):
        cx = db.cursor()
        #SET SQL_SAFE_UPDATES = 0
        #delete from OceanNews
        #http://bbs.csdn.net/topics/70045444
        #title = title.replace("'","''")

        #href = href.replace("'","''")
        insertSql = 'insert into OceanNews (newsTitle, newsURL, newsDate) values("{0}", "{1}", '
        #insertSql = insertSql.encode('utf-8').decode('utf-8')
        #print(str(succCount) + '        ' + insertSql + '\n')
        cx.execute(insertSql)
        succCount += 1
        db.commit()
        cx.close()

        return succCount
```

6. 主函数：主要是数据库的连接与关闭，记录程序允许时间，初始化 url 链接，并调用爬出函数。

```
def main():
        timestart = time.time()
        host = 'https://www.ccamlr.org'
        url = host + '/en/organisation/ccamlr-news'
        kv = {}
        succCount = 0 # 用于计数成功插入数据库的记录数量
        pageCount = getPageCount(url, kv)

        # 一定要指定编码方式，Python3默认是utf8，但mysql的mac版并不是
        db = pymysql.connect('127.0.0.1', 'root', 'abc', 'CourseDesign', use_unicode=True, charset="utf8")
        for i in range(pageCount): # 循环，一个一个网页顺序抓取
                kv['page'] = i
                succCount = getNewsData(url, kv, db, succCount)
        db.close()
        timeend = time.time()
        print("successd update {0} records".format(str(succCount)))
        print('last time: {0} s.'.format(timeend-timestart))
```

五、 实验结果：

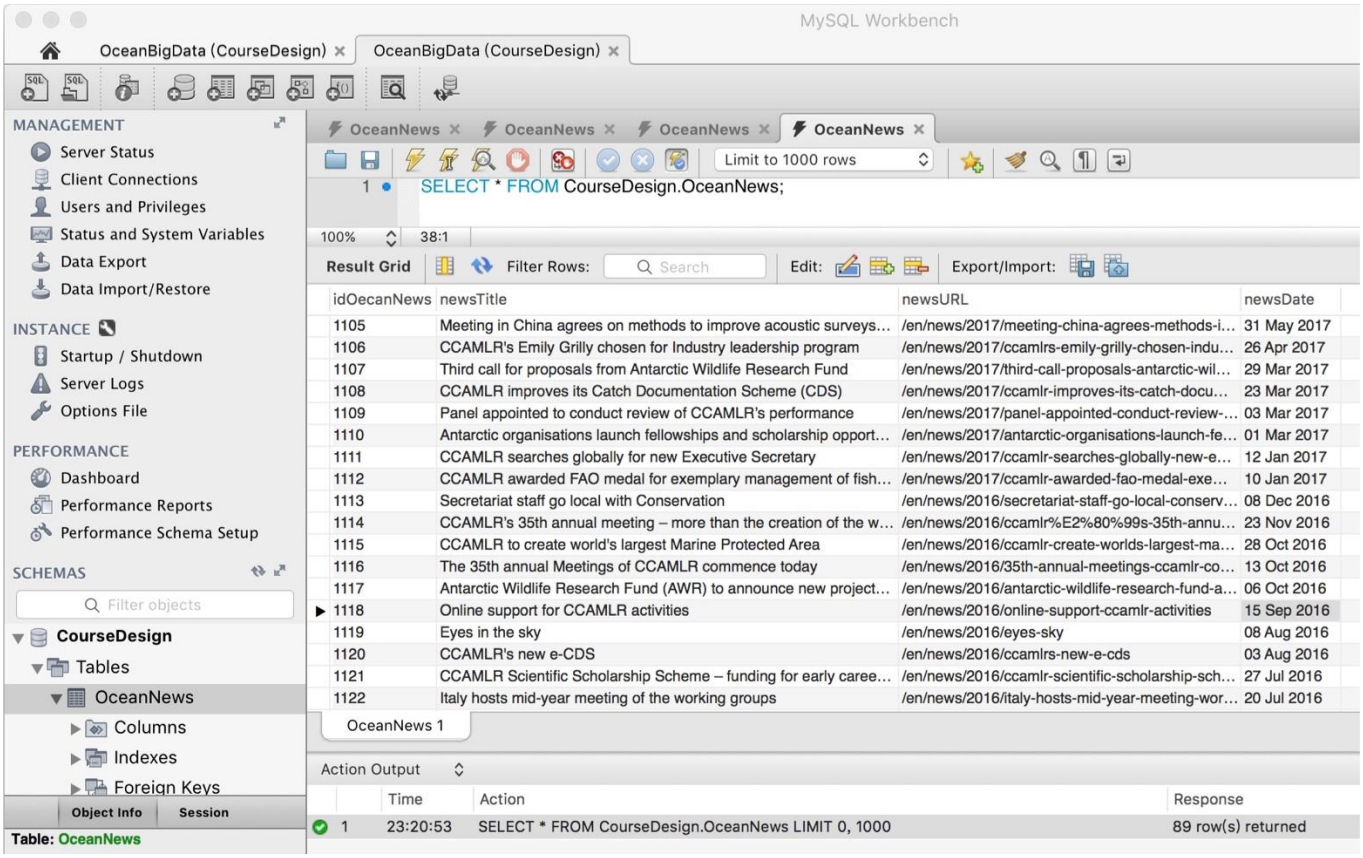
```
Python 3.5.3 Shell
Python 3.5.3 (v3.5.3:1880cb95a742, Jan 16 2017, 08:49:46)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>> WARNING: The version of Tcl/Tk (8.5.9) in use may be unstable.
Visit http://www.python.org/download/mac/tcltk/ for current information.

======== RESTART: /Users/littlesec/BetterMe/涉海保密综合实务2017年春季学期/实验三.py ========
successd update 89 records
last time: 31.311532974243164 s.
>>>
```

控制台

说明：

a) 控制台只显示成功插入数据库的记录条数，考虑到新闻会随着时间而更新，而若有更新则只插入更新的新闻即可，这也是实验所要求的。通过手工核对，初始网页中 9 个网站共 89 条新闻。

b) 运行时间与网络环境有关，也和电脑配置有关。如果不考虑存入数据库，仅仅是把内容打印到控制台上，我对比过同样网络环境下，其他电脑和本实验电脑所需要的时间也是不一样的。



数据库中的记录

说明：

a) idOecanNews 字段已设置成自增长，因此没有参看意义。

b) 图片下方的 89 row(s) returned 也证实了程序运行的正确性。

六、 实验中遇到的问题及解决方法：

1. 编码问题：

安装 MySQL 之后 MySQL 的默认编码是 Latin1，不支持中文，而 Python3 的默认编码是 utf-8，所爬取的网页通过头部也知道是 utf-8 编码。一开始也没考虑到这个问题。因为网页看起来都是英文，所以无论何种编码，不应该存在不支持或不兼容的问题。然而抽次爬虫后得到的结果并不是 89 条记录，而是 70 左右的记录，奇怪的是即使是在写入数据库函数不会报错，加入 try-catch 语句块尝试捕获常见的异常也没有捕获成功。我的解决方法是把提取到的信息打印到控制台看一下，打印的话确实能打印 89 条记录。

进一步的解决方法是给这些记录编号，成功插入数据库的编号自加 1，而不成功的则不变。这时发现了不成功的记录了，我把这些记录通过在 DBMS 中的 sql 语句编辑窗口执行，发现是能成功插入的，这说明之前是没有这条记录的（因为设置了标题和链接唯一）。通过编辑器的高亮发现，原来这些标题中有些单引号是中文字符的单引号（Python Shell 的输出是没有高亮设置的），因此我确定是编码问题了。

通过查阅资料，最快的解决方法就是在连接数据库时指定编码去连接。当然也可以修改MySQL 的配置文件，然而 macOS 版的 MySQL 在路径配置上仍然有漏洞，修改起来有难度。因此用前者方法。