1. a==b判断a和b的值是否相等，而a is b判断a和b的引用地址是否相等。Python中的变量都是作为对象存在的，其中如果字符串已经被创建，则后续新的内容相同的字符串变量对应的是相同对象；一定范围内的整数也只在缓存中保存一份，后续不再创建新的对象。

我额外编写了一段代码来验证这一特性：

# 整型常量池范围测试，[-5,256]

list1 = list(range(-1, -255, -1))

for i in range(-1, -255, -1):

# print(i is list1[-i-1])

if not i is list1[-i-1]:

print("小于等于"+str(i)+"后为false")

break

list2 = list(range(0, 512, 1))

for i in range(0, 512, 1):

# print(i is list1[i])

if not i is list2[i]:

print("大于等于"+str(i)+"后为false")

break

# 直接初始化的两个相同字符串会被编译器设为同一对象

str1 = "abcdef"

str2 = "abcdef"

# print(id(str1))

# print(id(str2))

print(id(str1)==id(str2))

# 通过类型转换生成的字符串会成为新的对象

a = 12345

str3 = "12345"

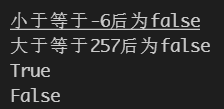
str4 = str(a)

# print(id(str3))

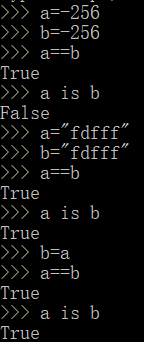
# print(id(str4))

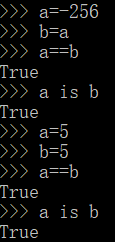
print(id(str3)==id(str4))

在VS Code + Pylance环境下运行结果如下：



原题目在Python3.7 idle环境下运行结果如下：





2.

（1）散列表

代码：

# 散列表：字典

# 初始化

dict = {'key1': 'value1'}

# 增加新的键值对或更新已有键值对

dict['key2'] = 'newValue'

dict['key1'] = 'newValue'

# 取出键值对

print(dict.pop('key1'))

print('key1' in dict) # 观察键key1是否在dict中

运行结果：

newValue

False

（2）队列：使用python内置库queue

代码：

import queue

# 初始化队列（长度为maxsize）

q = queue.Queue(maxsize=4)

# 顺序入队1，2，3，4

for i in range(4):

q.put(i)

# 观察出队顺序

for i in range(4):

print("出队列元素{}".format(q.get()))

运行结果：

出队列元素0

出队列元素1

出队列元素2

出队列元素3

（3）栈：只需要满足“后入先出”即LIFO原则，因此仍可使用queue库

代码：

# 初始化栈（后入先出“队列”）

stack = queue.LifoQueue(maxsize=4)

# 顺序入队1，2，3，4

for i in range(4):

stack.put(i)

# 观察出栈顺序

for i in range(4):

print("出栈元素{}".format(stack.get()))

运行结果：

出栈元素3

出栈元素2

出栈元素1

出栈元素0

3. 运行结果

17

23

-60

-6.666666666666667

-1

0.000125

-7

3

10

3.14

(20-3j)

(20+3j)

(-7, -1)

4. 代码

n = 9

for i in range(1, n+1): # range(start, end), 区间左开右闭

for j in range(1, i+1):

# 直接print

# print(str(j)+"\*"+str(i)+"="+str(i\*j), end=" ")

# .format写法

print("{}\*{}={}".format(j,i,j\*i), end=' ')

print()

运行结果

1\*1=1

1\*2=2 2\*2=4

1\*3=3 2\*3=6 3\*3=9

1\*4=4 2\*4=8 3\*4=12 4\*4=16

1\*5=5 2\*5=10 3\*5=15 4\*5=20 5\*5=25

1\*6=6 2\*6=12 3\*6=18 4\*6=24 5\*6=30 6\*6=36

1\*7=7 2\*7=14 3\*7=21 4\*7=28 5\*7=35 6\*7=42 7\*7=49

1\*8=8 2\*8=16 3\*8=24 4\*8=32 5\*8=40 6\*8=48 7\*8=56 8\*8=64

1\*9=9 2\*9=18 3\*9=27 4\*9=36 5\*9=45 6\*9=54 7\*9=63 8\*9=72 9\*9=81

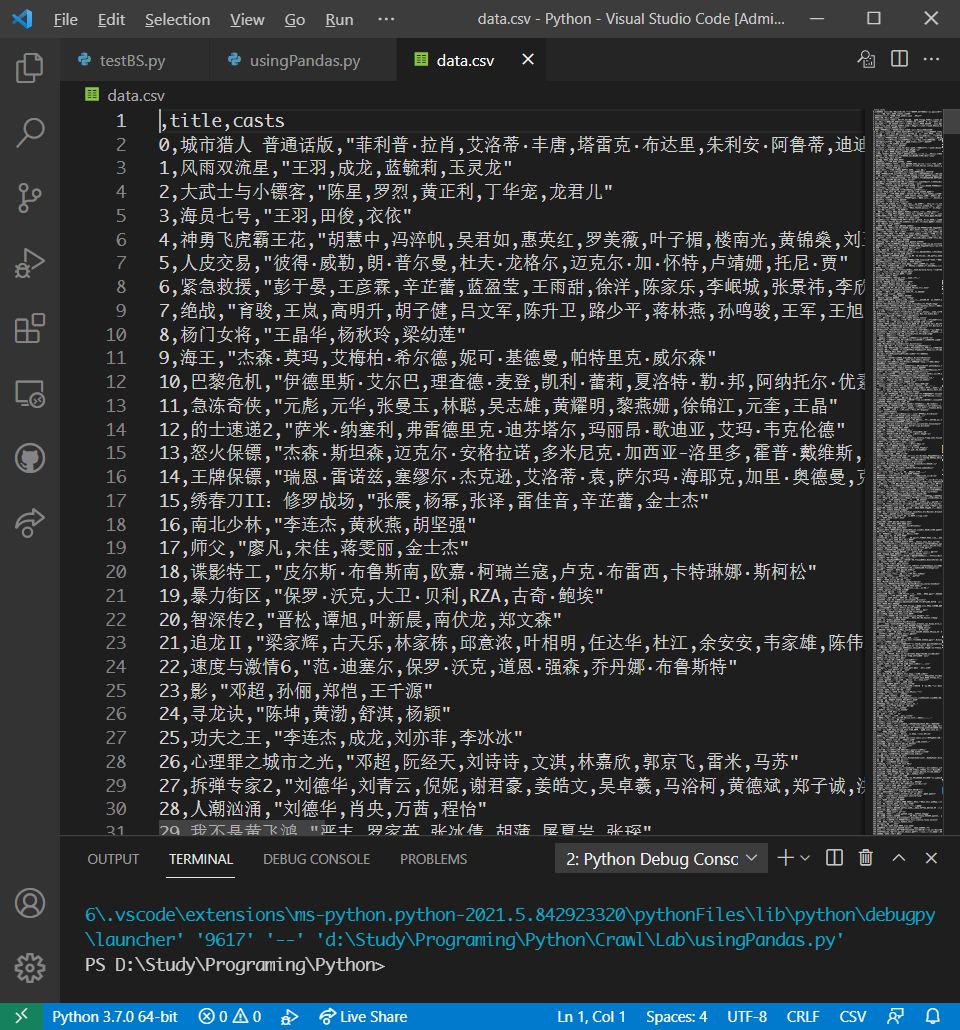
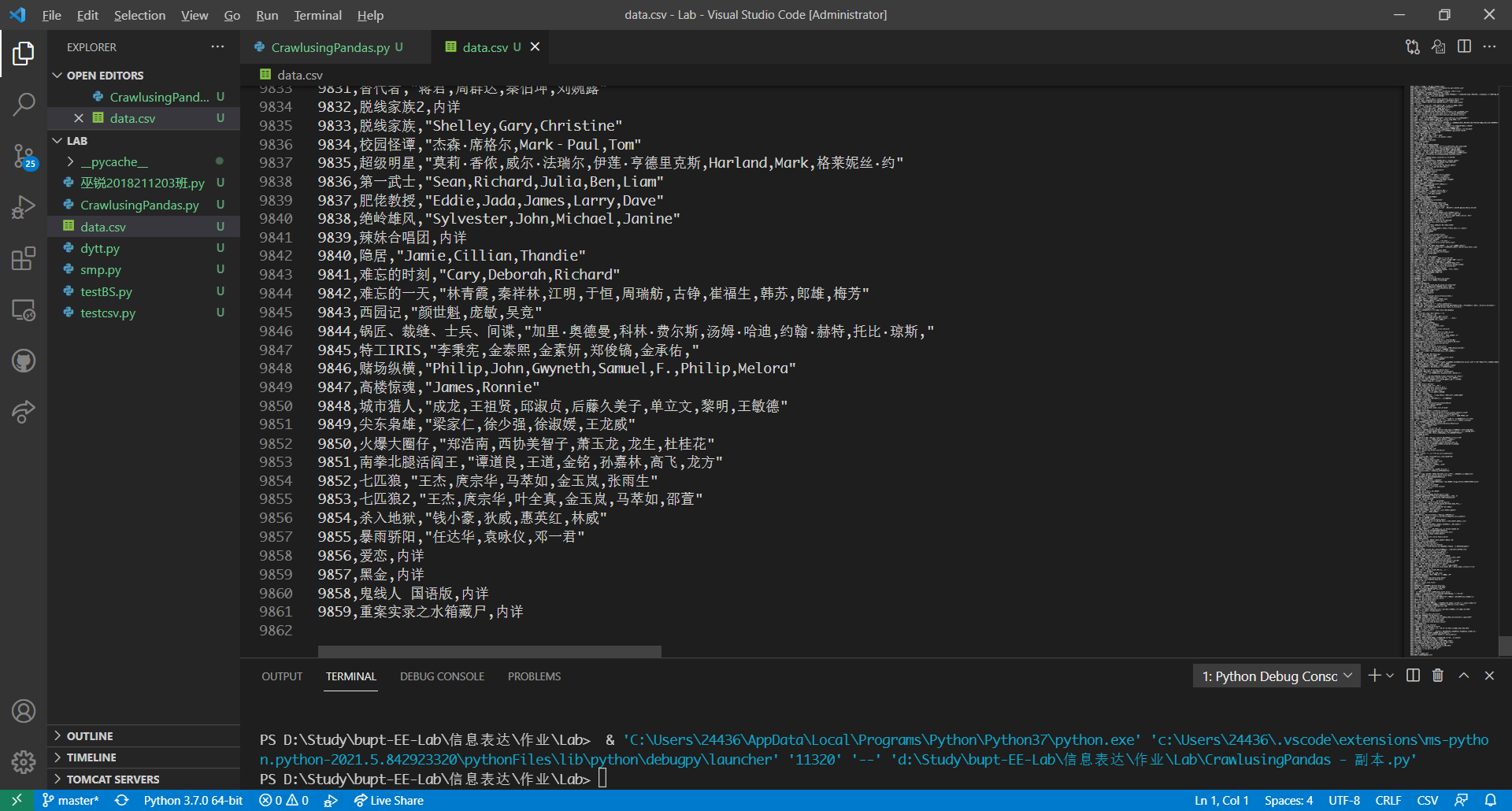
5. 网络爬虫实践题

代码功能简要说明：

爬虫程序由五个子程序和主函数组成，子程序功能依次为：

1. 获取第1~end页的url ：getAllUrl；
2. 通过requests库请求获得目标网页的HTML页面内容：getHTMLText；
3. 解析html页面，初始化BS对象：analyze；
4. 在bs对象内检索电影详情内容，并使用正则表达式去除html标签，最后将获得的电影名、信息存在data.csv中：getDetails
5. 使用pandas库保存获取的内容到工作目录下的data.csv：saveContents

子程序由主函数依次调用，爬取到的信息保存在爬虫程序工作目录下的data.csv中，设置爬取网站当前最后一页（274页）时文件首尾截图如下：（共9860条数据）

由于该网站数据量不大，csv文件只有950kb，因此爬取到的数据一并提交了。

备注：由于python和Windows10的中文编码格式不一致（utf-8和gbk），所以生成的csv文件在excel中会显示为乱码，可以使用记事本打开。

关键代码说明：

1. 分析得到不同页码的url格式后，可以自动化生成所需页数的url，保存在列表中，且页数便于修改。

for i in range(1, end):

url.append("https://www.dytt8.net.cn/index.php/vod/show/id/6/page/"+str(i)+".html")

1. 使用正则表达式删除html标签

regl = re.compile("<[^>]\*>")

item1 = regl.sub('', str(item))

1. 研究网页源码后我发现，有且仅有电影标题和演员表属于类“text-overflow”，且二者交替出现，因此只需一次beautifulsoup选择，再将获取的内容交替保存至name和detail列表中，即可在保证数据正确性的同时提高效率。