**实验10 正则表达式与网络爬虫**

**姓名： 徐超信 学号：1935010102 班级：计科1902 成绩：**

# 1. 实验内容与目标

本次实验旨在了解正则表达式相关基本概念以及在网络爬虫中的应用，掌握Python正则表达式模块re的常见用法，初步学会运用基本的爬虫模块urllib，以及第三方库requests和Beautiful Soup，进行web数据的自动抓取实践。需掌握的相关知识点主要包括：

**1) 正则表达式：**特殊字符的用法

**2) re模块：**

(a)正则表达式对象、匹配对象等概念

(b)核心函数与方法：compile、findall、finditer、match、search、group、sub、split

**3) 网络爬虫：**

(a) url的概念（包括http协议、ftp协议等）、爬虫的工作流程

(b) urllib基本爬虫库：掌握urllib.request模块的基本用法，结合正则表达式学会web数据的自动抓取方法。

(c)简单入门requests和Beautiful Soup等第三方爬虫库

“软件1902-1912190001-张三-实验10\_正则表达式与网络爬虫.docx”

最后，请在规定时间内完成本次实验，并将报告的word文档(非pdf版本)以附件形式提交至学习通平台。

# 2. 实验练习题

**答题要求**：将正确运行的**源代码书写在题目下方**，并紧跟着贴上一份**源代码的截图**和运行结果的**截图**。注：如题目要求保存结果数据到文件，则该相关数据也视作运行结果，需截图粘贴。但提交报告时无需提供数据文件。

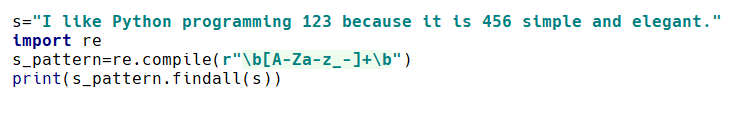
## 2.1 运用re.findall函数查找所有匹配。

### (1) 设计一个正则表达式，提取出字符串中的全部单词(数字和标点符号都不是单词部分)。这个字符串为s="I like Python programming 123 because it is 456 simple and elegant."

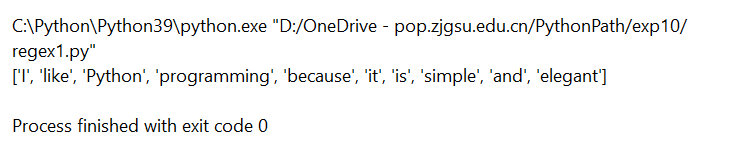
源代码：

s=**"I like Python programming 123 because it is 456 simple and elegant."  
import** re  
s\_pattern=re.compile(**r"\b[A-Za-z\_-]+\b"**)  
print(s\_pattern.findall(s))

源代码截图：



运行结果截图：



### (2) 提取下列文本每行中完整的年月日和时间字段：

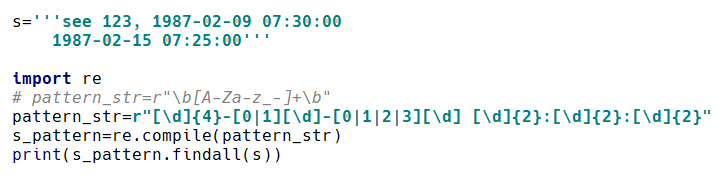
s='''see 123, 1987-02-09 07:30:00

1987-02-15 07:25:00'''

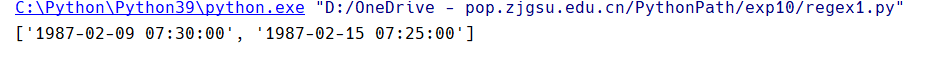
源代码：

s=**'''see 123, 1987-02-09 07:30:00  
 1987-02-15 07:25:00'''  
  
import** re  
*# pattern\_str=r"\b[A-Za-z\_-]+\b"*pattern\_str=**r"[\d]{4}-[0|1][\d]-[0|1|2|3][\d] [\d]{2}:[\d]{2}:[\d]{2}"**s\_pattern=re.compile(pattern\_str)  
print(s\_pattern.findall(s))

源代码截图：



运行结果截图：



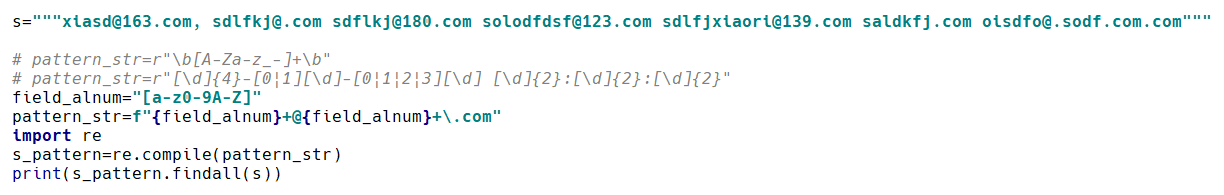
### (3) 使用正则表达式找出下列文本中全体合法的邮件地址：

s="""xiasd@163.com, sdlfkj@.com sdflkj@180.com solodfdsf@123.com sdlfjxiaori@139.com saldkfj.com oisdfo@.sodf.com.com"""

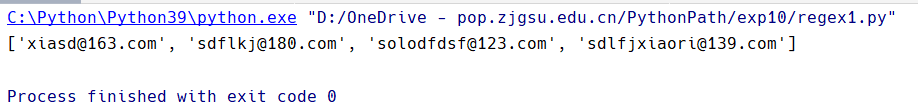
源代码：

s=**"""xiasd@163.com, sdlfkj@.com sdflkj@180.com solodfdsf@123.com sdlfjxiaori@139.com saldkfj.com oisdfo@.sodf.com.com"""**field\_alnum=**"[a-z0-9A-Z]"**pattern\_str=**f"{**field\_alnum**}+@{**field\_alnum**}+\.com"  
import** re  
s\_pattern=re.compile(pattern\_str)  
print(s\_pattern.findall(s))

源代码截图：



运行结果截图：



### (4) 令s="see 123, 1987-02-09 07:30:00, 1986-02-15 07:25:00"

执行下面代码：

fa1=re.findall("\d+-\d+-\d+",s)

fa2=re.findall("(\d+)-\d+-\d+",s)

fa3=re.findall("(\d+)-(\d+)-(\d+)",s)

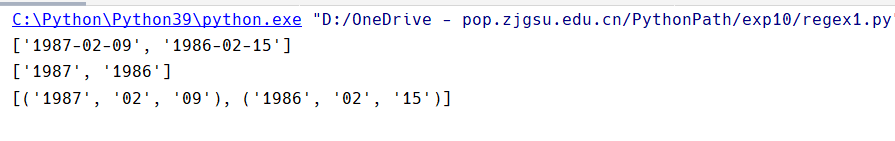
print(fa1)

print(fa2)

print(fa3)

程序输出结果是什么？通过考察结果，请学习掌握findall在正则式有否分组情形下输出结果的不同格式。

运行结果截图：



findall不同情形说明：

*# ['1987-02-09', '1986-02-15'];整个字符处匹配,返回完整的被匹配值*  
*#['1987', '1986'];可以匹配到未分组时同样的字符串,但是返回值是返回的括号内(分组)的所匹配到的那部分内容(截取),这里只有一个组,所以每个返回值只有一个子串*  
*#[('1987', '02', '09'), ('1986', '02', '15')];可以匹配到未分组时同样的字符串,但是此处分了三组,且仅截取其个括号组匹配到的内容返回  
# 元组的个数有被完全匹配的串决定(但内容由分组情况分别截取构成)*

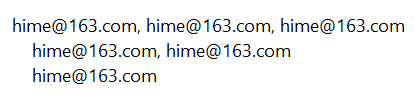
## 2.2 利用re.sub函数进行字符串的替换任务。

### (1) 将下列文本中的电子邮件地址替换为你自己的电子邮件地址([hime@163.com](mailto:hime@163.com)):

s="""hi69@qq.com, werksdf@163.com, sdf@sina.com

    sfjsdf@139.com, soifsdfj@134.com

[pwoeir23@126.com](mailto:pwoeir23@126.com)"""



### (2) 利用re.sub函数，将上述文本s中的数字加上212，重新输出。

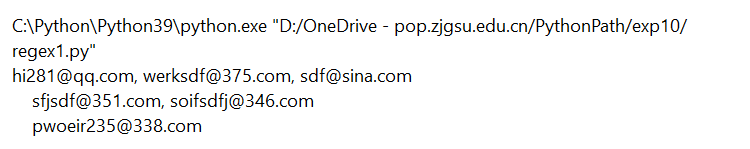
源代码：

s = **"""hi69@qq.com, werksdf@163.com, sdf@sina.com  
 sfjsdf@139.com, soifsdfj@134.com  
 pwoeir23@126.com"""  
  
import** re  
  
  
**def** add\_212\_method(match):  
 *"""* **:param** *match: Match* **:return***: sum\_result Str  
 """* int\_str = match.group(**"number"**)  
 value\_int = int(int\_str)  
 sum\_result = value\_int + 212  
 **return** str(sum\_result)  
  
  
field\_alnum = **"[a-z0-9A-Z]"***# pattern\_str=f"{field\_alnum}+@{field\_alnum}+\.com"*pattern\_str = **"(?P<number>\d+)"**s\_pattern = re.compile(pattern\_str)  
  
substitute\_result = s\_pattern.sub(repl=add\_212\_method, string=s)  
*# substitute\_result=re.sub("(?P<number>\d+)",add\_212\_method,s)*print(substitute\_result)

源代码截图：



运行结果截图：



## 2.3 简单网络爬虫实践。

按下列两种方式, 将网页(http://www.jituwang.com/tuku/biology/)中的图片分别爬取保存至本地文件目录“image1/”和“image2/”中（请自行先创建好文件夹）。

(1) 仅利用urllib模块完成图片爬取任务，可参考“picture\_crawler.py”。

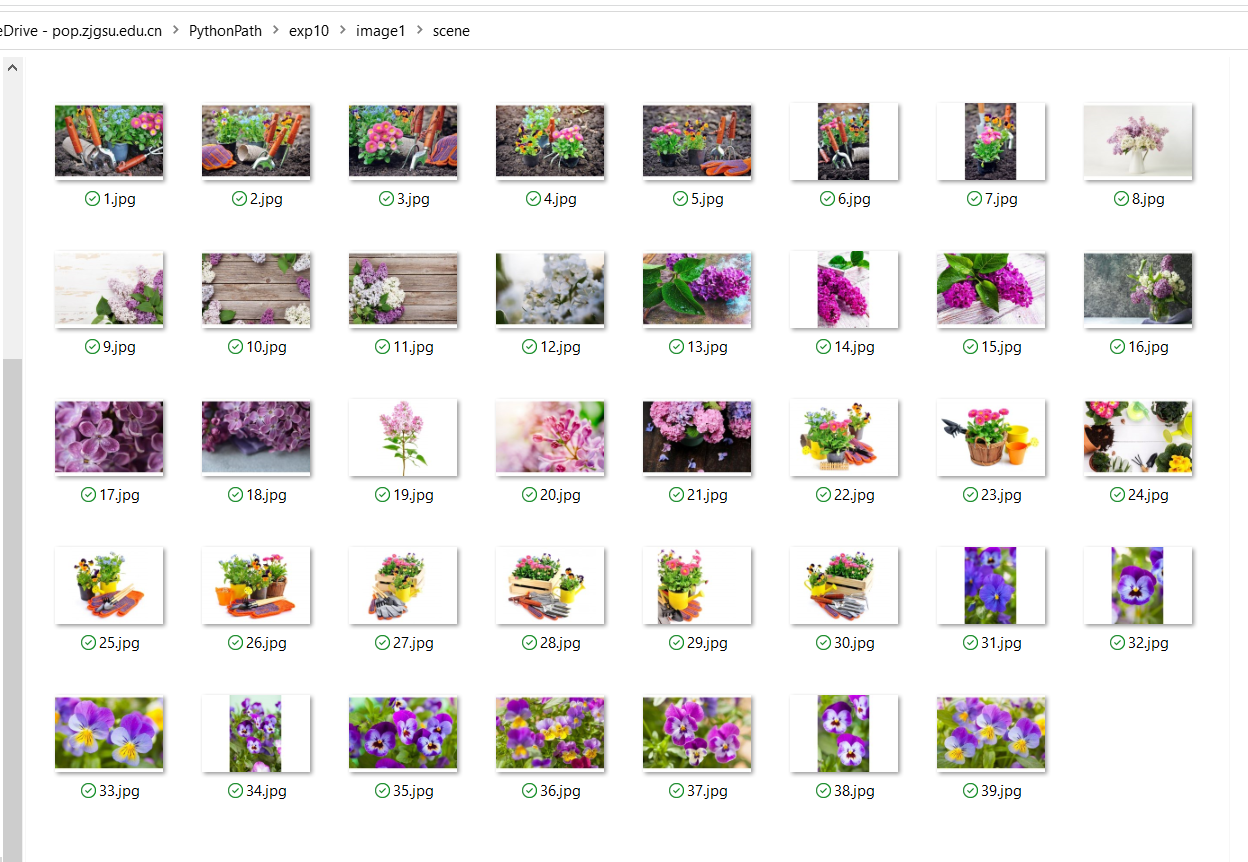
源代码：

*# -\*- coding: utf-8 -\*-***import** os  
**import** re  
**import** urllib.error *# 用于错误处理***import** urllib.request *# 主要用于打开和阅读url*prefix\_path = **r"image1"  
  
  
def** picCraw(url, topic, img\_pattern\_str):  
 count = 1  
 file\_name = os.path.join(prefix\_path, topic + **".html"**)  
 print(**"正在保存:"** + file\_name)  
 read\_result\_bytes = urllib.request.urlopen(url).read()  
 *# set save path* to\_save\_path = os.path.join(prefix\_path, topic)  
 **if not** os.path.isdir(to\_save\_path): os.makedirs(to\_save\_path) *# 创建目录保存图片  
 #从字节码解码成文本(您应当注意,如果decode()方法不加参数,可能会导致ascii编码相关报错.* page\_data\_str = read\_result\_bytes.decode(**"utf8"**,**"ignore"**)  
  
 MatchedImage\_link\_list = re.findall(img\_pattern\_str, page\_data\_str) *# 找出所有匹配* print(**"MatchedImages:"**, MatchedImage\_link\_list)  
 **for** image **in** MatchedImage\_link\_list: *# 用正则表达式匹配所有的图片* pattern = re.compile(**r'//.\*\.jpg$'**) *# 匹配jpg格式的文件* **if** pattern.search(image): *# 如果匹配成功，则获取图片信息；若不成功继续下一个* **try**:  
 **if "http" not in** image: image = **"http:"** + image  
 image\_data\_bytes = urllib.request.urlopen(image).read() *# 获取图片信息* image\_path = os.path.join(prefix\_path, topic, str(count) + **".jpg"**) *# 给图片命名* count += 1  
 **with** open(image\_path, **"wb"**) **as** image\_file:  
 image\_file.write(image\_data\_bytes) *# 将图片写入jpg文件* **except** urllib.error.URLError **as** e:  
 print(**"Download failed"**)  
 **with** open(file\_name, **"wb"**) **as** file: *# 将页面写入文件* file.write(read\_result\_bytes)  
  
  
**if** \_\_name\_\_ == **"\_\_main\_\_"**:  
 *# https://sc.chinaz.com/tupian/fengjingtupian.html* url = **'http://www.jituwang.com/tuku/biology/'** *# 匹配图片的pattern,可通过查看网页源代码获悉  
 #通过findall(),该会返回匹配该分组的匹配部分所构成的集合* img\_pattern\_str = **r'<img .\*src=\"(.+\.jpg)\"'** picCraw(url, **"scene"**, img\_pattern\_str)

源代码截图：



图片数据截图：

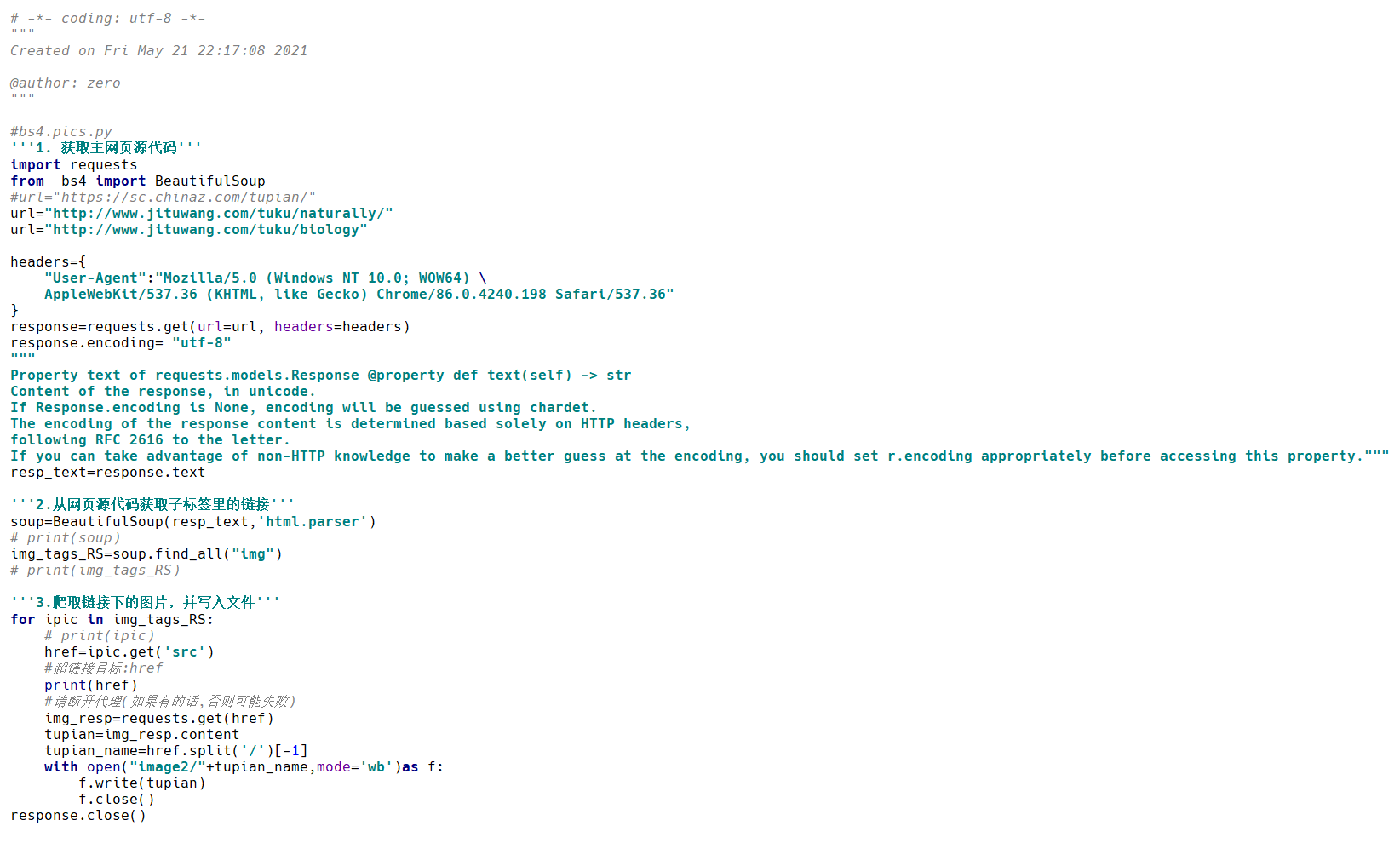


(2) 结合第三方库Beautiful Soup和requests完成图片爬取任务，可参考“bs4\_pics.py”。

源代码：

*# -\*- coding: utf-8 -\*-  
"""  
Created on Fri May 21 22:17:08 2021  
  
@author: zero  
"""  
  
#bs4.pics.py***'''1. 获取主网页源代码'''  
import** requests  
**from** bs4 **import** BeautifulSoup  
*#url="https://sc.chinaz.com/tupian/"*url=**"http://www.jituwang.com/tuku/naturally/"**url=**"http://www.jituwang.com/tuku/biology"**headers={  
 **"User-Agent"**:**"Mozilla/5.0 (Windows NT 10.0; WOW64) \  
 AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.198 Safari/537.36"**}  
response=requests.get(url=url, headers=headers)  
response.encoding= **"utf-8"  
"""  
Property text of requests.models.Response @property def text(self) -> str  
Content of the response, in unicode.  
If Response.encoding is None, encoding will be guessed using chardet.  
The encoding of the response content is determined based solely on HTTP headers, following RFC 2616 to the letter. If you can take advantage of non-HTTP knowledge to make a better guess at the encoding, you should set r.encoding appropriately before accessing this property."""**resp\_text=response.text  
  
**'''2.从网页源代码获取子标签里的链接'''**soup=BeautifulSoup(resp\_text,**'html.parser'**)  
*# print(soup)*img\_tags\_RS=soup.find\_all(**"img"**)  
*# print(img\_tags\_RS)***'''3.爬取链接下的图片，并写入文件'''  
for** ipic **in** img\_tags\_RS:  
 *# print(ipic)* href=ipic.get(**'src'**)  
 *#超链接目标:href* print(href)  
 *#请断开代理(如果有的话,否则可能失败)* img\_resp=requests.get(href)  
 tupian=img\_resp.content  
 tupian\_name=href.split(**'/'**)[-1]  
 **with** open(**"image2/"**+tupian\_name,mode=**'wb'**)**as** f:  
 f.write(tupian)  
 f.close()  
response.close()

源代码截图：



图片数据截图：

