# Python Cheat Sheet-3. 数据结构 （Data Structure）-string

?> PCS-3（    ）：3.1 字符串（String）与文本文件；3.2 文件的打开与读写——用open()\_python内置函数（built-in functions）；3.3 常用字符串操作方法；3.4 字符串格式化；3.5 re,regular expression（正在表达式）

知识点

描述

代码段

运算结果

备注

#### **3.1** 字符串（String）与文本文件

数据处理时必然会遇到包含字符串的文件，或者数值以文本方式存储，读取后再转换为数值型。不管是从文本中提取数据，还是图表中的文字表述，及对文本内容的分析，都需要知道如何处理字符串。字符串处理的方法途径异常繁多，各类模式匹配符号组合表述技巧性较强。除了常规用到的处理方式外，不经常用到或从未用过的方式则很难记住，因此字符串处理部分以查阅为主，当遇到要处理的字符串时，可以根据要处理要求从PCS-3或相关文件，尤其搜索引擎中找到处理方法的答案。当然，对于经常用到的字符串处理方法，需要有意识的练习记忆，避免每次重复搜索。

互联网是对字符串处理最为频繁的领域，城市空间数据分析和数字化设计领域相对较少。

字符串处理除了Python内置方法外，主要使用 [re, Regular expression operations（正则表达式）](https://docs.python.org/3/library/re.html)①。

很多时，用文本文件存储数据，因文本存储的格式不同会表述为不同的文件格式，例如没有格式限制的TXT文件（通常按行记录数据，逗号、分号或空格分隔字段)，后缀名.txt；逗号分隔的[CSV, comma-separated values](https://en.wikipedia.org/wiki/Comma-separated_values)②文件格式（每行为一组数据，逗号隔离字段），后缀名.csv；[JSON, JavaScript Object Notation](https://en.wikipedia.org/wiki/JSON)③，开放的标准文件格式和数据交换格式，以属性-值对和数组的形式记录，后缀名.json。存储数据的方式很多，也可以自定义存储格式和后缀名，不过以常规标准的格式存储数据方便数据交换，因为常用的格式通常已有大量写好的读写代码，例如[pandas.read\_csv()](https://pandas.pydata.org/docs/reference/api/pandas.read_csv.html)，[pandas.DataFrame.to\_csv](https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.to_csv.html)等④，又或者[CSV库](https://docs.python.org/3/library/csv.html)⑤，通过import csv调入库读写方法等，这都极大方便的增加了书写代码的效率，同时也尽量避免了读写错误。

快速查看文本文件（也常用于查看代码）推荐使用[notepad++](https://notepad-plus-plus.org/downloads/)⑥工具。

#### **3.2** 文件的打开与读写——用open()\_python内置函数（built-in functions）

open(file, mode='r', buffering=-1, encoding=None, errors=None, newline=None, closefd=True, opener=None)，在open()函数的参数配置上通常只使用3个参数，file，mode和encoding。其中file为文件存储位置路径；mode为打开文件的模式，包括基本模式和组合模式，基本模式的字符含义有：

| 字符（character） | 含义（meaning） |
| --- | --- |
| ‘r’ | 以只读方式打开文件，读取文件内容的指针位于文件的开始。为默认模式 |
| ‘w’ | 以只写模式打开文件， 如果文件存在，则会清空文件中已有内容；如果文件不存在，则创建新文件 |
| ‘x’ | 建立一个新文件，并以写模式打开。如果文件存在，则报错 |
| ‘a’ | 写模式，如果文件中存在内容，则在其后追加新内容 |
| ‘b’ | 二进制模式，需要’r’、‘w’、’a’等字符模式配合使用。为以二进制格式，读写文件，通常用于非文本文件，例如影音图像，或以二进制存储的数据等 |
| ‘t’ | 文本模式，为默认。如果要以二进制读写，加符号’b’ |
| ‘+’ | 打开一个文件进行更新，可读可写 |

在基本模式之上，可以组合为’rb’、‘r+’、‘rb+’、‘wb’、‘w+’、‘wb+’、‘ab’、‘a+’、’ab+’等多种组合模式，组合后含义为单独字符模式含义的组合。

如果不配置encoding参数，文件打开时报错（经常出现在含有中文字符的文件中），则需要指定该参数值，为打开该文件所使用的编码格式。编码格式通常配置为utf-8（有时也写为utf8），对于utf-8无法识别含有中文的文件，通常可以尝试配置为GBK, Chinese Internal Code Specification（汉字内码扩展规范）。

f=open()打开文件的返回值是一个文件对象句柄（help(open)给出的解释是 Open file and return a stream），并将其赋给自定义变量f，通过句柄（该变量）对文件进行操作。此时f包含操作文件内容的多个属性和方法：

* f属性

f.name， 返回文件名称；

f.mode， 返回文件打开时的文件打开模式；

f.encoding， 返回文件打开时使用的编码格式；

f.closed， 判断文件是否已经关闭。

这里例举了实际研究项目的数据为从百度地图应用中检索下载的POI，其中第一行为1,0101000020897F000008D599CB26E312418530CECF16EB4C41,美香源,34.23709808337344,108.93100212046282,美食;中餐厅,0,,309449.6988290106,3790381.623479905，按逗号分割，各个字段名为序号，ID，名称，维度，经度，一级行业分类；二级行业分类，评分，均价，地理坐标投影后y值，地理坐标投影后X值。（这是经过处理了的数据，并非下载的原数据）

xian\_poi\_fn='./data/xian\_poi.csv' # 存储有西安POI, point of interesting兴趣点数据  
xian\_poi\_f=open(xian\_poi\_fn,'r', encoding="utf-8") # 以只读方式打开文件  
print(xian\_poi\_f)  
  
print('\_'\*50)  
print(xian\_poi\_f.name)  
print(xian\_poi\_f.mode)  
print(xian\_poi\_f.encoding)  
print(xian\_poi\_f.closed)

<\_io.TextIOWrapper name='./data/xian\_poi.csv' mode='r' encoding='utf-8'>  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
./data/xian\_poi.csv  
r  
utf-8  
False

* f 读取方法

f.read(size=-1, /)， 参数size未指定或为负值时返回整个文件，否则到指定字符长度位置或到文件末尾（EOF，end of file）；

f.readline(size=-1, /)， 读取一行，即读到换行符或者EOF。如果给定size，则按长度读取；

f.readlines()，返回所有行的一个列表；

f.tell()， 返回当前读取文件的位置；

f.clost()，关闭文件。

类定义时，如果以f.attribute方式返回值，则为属性，例如f.name； 如果以f.function()方式，即调用类的方法，例如f.read()。

print(xian\_poi\_f.read(57))  
print("\_"\*10,xian\_poi\_f.tell())  
print(xian\_poi\_f.readline()) # 从上一语句57处继续读，读到该行结束  
print("\_"\*10,xian\_poi\_f.tell())  
print(xian\_poi\_f.readline()) # 继续读写一行文本内容  
print('\_'\*50)  
print(xian\_poi\_f.readlines()[:5]) # 这里只打印了返回列表的前5行  
print("\_"\*10,xian\_poi\_f.tell())  
  
print('\_'\*50)  
xian\_poi\_f.close()  
print(xian\_poi\_f.closed)

1,0101000020897F000008D599CB26E312418530CECF16EB4C41,美香源,  
\_\_\_\_\_\_\_\_\_\_ 63  
34.23709808337344,108.93100212046282,美食;中餐厅,0,,309449.6988290106,3790381.623479905  
  
\_\_\_\_\_\_\_\_\_\_ 156  
2,0101000020897F0000B038F4CF9BE31241389AD606BFEC4C41,雷记澄城水盆羊肉(红樱路店),34.244750060429915,108.93113243785623,美食;中餐厅,3.9,20.0,309478.95308006834,3791230.053424146  
  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
['3,0101000020897F00005C24B17F97E3124160F86A81E5EC4C41,段府农家菠菜面(红缨路店),34.245443456204875,108.93110375582032,美食;中餐厅,4,,309477.8746991807,3791307.011076972\n', '4,0101000020897F000040F7FD9433E412412CD771E043EC4C41,平价餐厅(友谊西路店),34.24253719175486,108.93159854262964,美食;中餐厅,4.2,26.0,309516.8955000527,3790983.753474137\n', '5,0101000020897F0000EFFE229C56E61241ABEE21C2D7EC4C41,山妹川菜,34.24522784875664,108.93301750888804,美食;中餐厅,3.5,22.0,309653.6524772485,3791279.5166605315\n', '6,0101000020897F0000522963D2BFE312419E6ED4B833EC4C41,老三澄合羊肉水盆,34.24224069599151,108.93129159808142,美食;中餐厅,3,,309487.95545639575,3790951.443982913\n', '7,0101000020897F00008C6ACABF13E412416B0C7D7B09ED4C41,湘村菜馆(红缨路店),34.24609764121232,108.9314250012306,美食;中餐厅,4.2,20.0,309508.937295594,3791378.9647536776\n']  
\_\_\_\_\_\_\_\_\_\_ 2470810  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
True

* f写入方法

f.write()，将字符串写入到文本，如果是数值等数据，需要将其转换为字符串后再写入；

f.writelines()，将字符串列表逐行写入到文件；

f.flush()， 将数据刷至硬盘。通常在f.close()文件关闭时，会自动一次性刷至硬盘，除非特殊需求，否则不用执行f.flush();

seek(cookie, whence=0, /)， 更改当前读写位置，为字节偏移量（byte offset）。whence为0时（默认值），代表从文件开始定位算起；为1时，以当前位置定位算起；为2时，以文件末尾定位算起。

在下述的示例中，poi\_1PieceOFdata变量只存储了一行数据；而poi\_2PiecesOFdata变量存储了两行数据，行之间用换行符\n完成换行动作。poi\_piecesOFdata.flush()会将先写入的一行数据刷至硬盘文件中，因为使用了w+模式，因此可以用poi\_piecesOFdata.flush()方法定位到文本开始，再用poi\_piecesOFdata.read()方法查看，否则返回内容为空。也可以用外部notepad++等工具打开查看内容。而后将包含两行数据的poi\_2PiecesOFdata变量写入，并调用poi\_piecesOFdata.close()方法，将后续写入的数据刷至硬盘文件中。

poi\_1PieceOFdata='2,0101000020897F0000B038F4CF9BE31241389AD606BFEC4C41,雷记澄城水盆羊肉(红樱路店),34.244750060429915,108.93113243785623,美食;中餐厅,3.9,20.0,309478.95308006834,3791230.05342414'  
  
poi\_2PiecesOFdata='\n3,0101000020897F00005C24B17F97E3124160F86A81E5EC4C41,段府农家菠菜面(红缨路店),34.245443456204875,108.93110375582032,美食;中餐厅,4,,309477.8746991807,3791307.011076972,\n4,0101000020897F000040F7FD9433E412412CD771E043EC4C41,平价餐厅(友谊西路店),34.24253719175486,108.93159854262964,美食;中餐厅,4.2,26.0,309516.8955000527,3790983.753474137'  
poi\_piecesOFdata\_fn='./data/poi\_piecesOFdata.csv'  
poi\_piecesOFdata=open(poi\_piecesOFdata\_fn,'w+',encoding='utf-8')  
poi\_piecesOFdata.write(poi\_1PieceOFdata)  
poi\_piecesOFdata.flush()

注意，写入文本内容后，读写位置位于文件末尾，不通过f.seek()指定开始位置，读取的内容会为空。

poi\_piecesOFdata.seek(0)  
print(poi\_piecesOFdata.read())

2,0101000020897F0000B038F4CF9BE31241389AD606BFEC4C41,雷记澄城水盆羊肉(红樱路店),34.244750060429915,108.93113243785623,美食;中餐厅,3.9,20.0,309478.95308006834,3791230.05342414

不管读或者写，当完成读写动作后，需要调用f.close()关闭文件。

poi\_piecesOFdata.write(poi\_2PiecesOFdata)  
  
poi\_listOFdata=['5,0101000020897F0000EFFE229C56E61241ABEE21C2D7EC4C41,山妹川菜,34.24522784875664,108.93301750888804,美食;中餐厅,3.5,22.0,309653.6524772485,3791279.5166605315\n',   
 '6,0101000020897F0000522963D2BFE312419E6ED4B833EC4C41,老三澄合羊肉水盆,34.24224069599151,108.93129159808142,美食;中餐厅,3,,309487.95545639575,3790951.443982913\n',   
 '7,0101000020897F00008C6ACABF13E412416B0C7D7B09ED4C41,湘村菜馆(红缨路店),34.24609764121232,108.9314250012306,美食;中餐厅,4.2,20.0,309508.937295594,3791378.9647536776\n']  
poi\_piecesOFdata.write('\n') # 因为写入两行时，末尾为写入'\n'换行符。因此单独写入，避免后续写入内容未起新行  
poi\_piecesOFdata.writelines(poi\_listOFdata)  
poi\_piecesOFdata.close()

用with open(fn, mode) as f:上下文管理的方式打开文件，则不需要调用f.close()的方式关闭文件，也可以避免文件读写时可能产生的IOError。

这里将读取的CSV格式数据转换为字典格式，格式样式为{ID:{'name':name,'coordi':{'lat':lat,'lon':lon}}}其中有3层嵌套，同时将字符串格式的经纬度使用float()方法转换为浮点型。具体方法是应用字符串处理中的S.split()将字符串切分为字段列表后循环提取需要的数据内容。注意，这里提前应用了非常好用的匿名函数（lambda）及列表推导式（comprehensions）。

同样，可以将poi\_info\_dict={S\_split(S)[0]:{'name':S\_split(S)[2],'coordi':{'lat':float(S\_split(S)[3]),'lon':float(S\_split(S)[4])}} for S in poi\_lst}，这个语句用for循环的方式拆分处理。

poi\_piecesOFdata\_fn='./data/poi\_piecesOFdata.csv'  
with open(poi\_piecesOFdata\_fn, 'r',encoding='utf-8') as f:  
 poi\_lst=f.readlines()  
print(poi\_lst)  
   
S\_split=lambda S:S.split(",") #为  
poi\_info\_dict={S\_split(S)[0]:{'name':S\_split(S)[2],'coordi':{'lat':float(S\_split(S)[3]),'lon':float(S\_split(S)[4])}} for S in poi\_lst}  
print("\_"\*50)  
print(poi\_info\_dict)

['2,0101000020897F0000B038F4CF9BE31241389AD606BFEC4C41,雷记澄城水盆羊肉(红樱路店),34.244750060429915,108.93113243785623,美食;中餐厅,3.9,20.0,309478.95308006834,3791230.05342414\n', '3,0101000020897F00005C24B17F97E3124160F86A81E5EC4C41,段府农家菠菜面(红缨路店),34.245443456204875,108.93110375582032,美食;中餐厅,4,,309477.8746991807,3791307.011076972,\n', '4,0101000020897F000040F7FD9433E412412CD771E043EC4C41,平价餐厅(友谊西路店),34.24253719175486,108.93159854262964,美食;中餐厅,4.2,26.0,309516.8955000527,3790983.753474137\n', '5,0101000020897F0000EFFE229C56E61241ABEE21C2D7EC4C41,山妹川菜,34.24522784875664,108.93301750888804,美食;中餐厅,3.5,22.0,309653.6524772485,3791279.5166605315\n', '6,0101000020897F0000522963D2BFE312419E6ED4B833EC4C41,老三澄合羊肉水盆,34.24224069599151,108.93129159808142,美食;中餐厅,3,,309487.95545639575,3790951.443982913\n', '7,0101000020897F00008C6ACABF13E412416B0C7D7B09ED4C41,湘村菜馆(红缨路店),34.24609764121232,108.9314250012306,美食;中餐厅,4.2,20.0,309508.937295594,3791378.9647536776\n']  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
{'2': {'name': '雷记澄城水盆羊肉(红樱路店)', 'coordi': {'lat': 34.244750060429915, 'lon': 108.93113243785623}}, '3': {'name': '段府农家菠菜面(红缨路店)', 'coordi': {'lat': 34.245443456204875, 'lon': 108.93110375582032}}, '4': {'name': '平价餐厅(友谊西路店)', 'coordi': {'lat': 34.24253719175486, 'lon': 108.93159854262964}}, '5': {'name': '山妹川菜', 'coordi': {'lat': 34.24522784875664, 'lon': 108.93301750888804}}, '6': {'name': '老三澄合羊肉水盆', 'coordi': {'lat': 34.24224069599151, 'lon': 108.93129159808142}}, '7': {'name': '湘村菜馆(红缨路店)', 'coordi': {'lat': 34.24609764121232, 'lon': 108.9314250012306}}}

#### **3.3** 常用字符串操作方法

下表中融合了字符串常用操作的方法，这包括字符串的运算、函数和方法。

| 操作 | 解释 |
| --- | --- |
| S='' | 建立空字符 |
| "''" | 双引号与单引号嵌套使用 |
| bool('') | 可以用于检查是否为空字符 |
| \t \n | 转义字符（escape, string backslash characters）中常用到的字符，制表符（Horiozntal tab）和换行符（Newline/linefeed） |
| S1+S2 | 合并字符串 |
| S\*n | 复制字符串 |
| S[idx] | 按索引（字符位置）提取字符 |
| S[start:end] | 分片方式提取字符串 |
| len(S) | 计算字符串长度 |
| r'string' | 原始字符串（无转义） |
| S.split(sep=None, maxsplit=-1) | 按分隔符（delimiter）切分字符串为字段列表 |
| '%s'%String | %形式格式字符串 |
| '{}'.format(value) | format()方法格式字符串 |
| S.find(sub[, start[, end]]) | 寻找给定字段的开始索引值 |
| S.strip() | 移除字符串中前后的空白（空格） |
| S.lstrip() | 移除字符串中左端的空白 |
| S.rstrip() | 移除字符串中右端的空白 |
| S.isdigit() | 判断字符串是否为整数字符串 |
| S.lower() | 将字符串小写 |
| S.upper() | 将字符串大写 |
| S.endswith(suffix[, start[, end]]) | 判断字符串末尾字符，返回布尔值 |
| S.encode(encoding='utf-8', errors='strict') | 字符串编码 |
| S.decode() | 字符串解码 |
| str in S | 成员运算符，给定字符或字段是否在字符串中，返回布尔值 |
| str not in S | 成员运算符，给定字符或字段是否不在字符串中，返回布尔值 |
| map(ord,S) | 返回给定字符在Unicode中的码值 |
| [s for s in S] | 用列表推导式循环拆解字符串为单个字符 |
| 's'.join(iterable, /) | 给定分隔符，合并字段列表为一个字符串 |

字符串的方法还有很多，罗列如下方便查询，或查看[Python String Methods](https://www.w3schools.com/python/python_ref_string.asp)⑦等在线文件：

| 1 | 2 | 3 | 4 | 5 | 6 |
| --- | --- | --- | --- | --- | --- |
| S.capitalize() | S.ljust(width [, fill]) | S.casefold() | S.lower() | S.center(width [, fill]) | S.lstrip([chars]) |
| S.count(sub [, start [, end]]) | S.maketrans(x[, y[, z]]) | S.encode([encoding [,errors]]) | S.partition(sep) | S.endswith(suffix [, start [, end]]) | S.replace(old, new [, count]) |
| S.expandtabs([tabsize]) | S.rfind(sub [,start [,end]]) | S.find(sub [, start [, end]]) | S.rindex(sub [, start [, end]]) | S.format(fmtstr, \*args, \*\*kwargs) | S.rjust(width [, fill]) |
| S.index(sub [, start [, end]]) | S.rpartition(sep) | S.isalnum() | S.rsplit([sep[, maxsplit]]) | S.isalpha() | S.rstrip([chars]) |
| S.isdecimal() | S.split([sep [,maxsplit]]) | S.isdigit() | S.splitlines([keepends]) | S.isidentifier() | S.startswith(prefix [, start [, end]]) |
| S.islower() | S.strip([chars]) | S.isnumeric() | S.swapcase() | S.isprintable() | S.title() |
| S.isspace() | S.translate(map) | S.istitle() | S.upper() | S.isupper() | S.zfill(width) |
| S.join(iterable) |  |  |  |  |  |

S=''  
print(bool(S))  
print(S)  
  
print("\_"\*50)  
S="coordi:'34.244750060429915,108.93113243785623'"  
print(S)  
print(bool(S))  
  
print("\_"\*50)  
S='ID:2,\tname:restaurant\tscore:5\nID:3,\tname:hotel\tscore:3'  
print(S)  
  
S="""\_\_\_triple-quoted block strings\_\_\_"""  
print(S)  
  
print("\_"\*50)  
S='\ID\name'  
print(S)  
print("\_"\*25)  
S=r'\ID\name'  
print(S)  
  
print("\_"\*50)  
S1='category:'  
S2='restaurant'  
print(S1+S2)  
  
S='name,'\*3  
print(S)  
S\_split\_lst=S.split(",")  
print(S\_split\_lst)  
  
print("\_"\*50)  
S\_poi='2,雷记澄城水盆羊肉(红樱路店),34.244750060429915,108.93113243785623,美食;中餐厅,3.9,20.0,309478.95308006834,3791230.05342414\n'  
print(S\_poi[2])  
print(S\_poi[2:10])  
print('string length={}'.format(len(S\_poi)))  
print('name=%s'%S\_poi[2:10])  
lat\_start\_position=S\_poi.find('34.244750060429915')  
lat\_end\_position=S\_poi.find('108.93113243785623')-1  
print(lat\_start\_position)  
print(S\_poi[lat\_start\_position:lat\_end\_position])  
  
print("\_"\*50)  
S\_rstrip=" 34.244 ".strip()  
print("{1}={0};".format(S\_rstrip,'lat'))  
  
S\_rstrip=" 34.244 ".lstrip()  
print("{1}={0};".format(S\_rstrip,'lat'))  
  
S\_rstrip=" 34.244 ".rstrip()  
print("{1}={0};".format(S\_rstrip,'lat'))  
  
print("\_"\*50)  
print('name:108.931'.replace('name','lon'))  
print('108.931'.isdigit())  
print('108'.isdigit())  
  
print("\_"\*50)  
print('code'.upper())  
print('CODE'.lower())  
  
S\_poi\_lst=S\_poi.split(",")  
print(S\_poi\_lst)  
print('\_'.join(S\_poi\_lst))  
  
S='美食;中餐厅'  
encode\_S=S.encode('GBK')  
print(encode\_S)  
decode\_S=encode\_S.decode('GBK')  
print(decode\_S)  
  
ord\_s=map(ord,['S','a'])  
print(list(ord\_s))  
  
print("\_"\*50)  
print('p' in 'python')  
print('j' in 'python')  
print('j' not in 'python')  
print([s for s in 'python'])  
print('python'.endswith('on'))

False  
  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
coordi:'34.244750060429915,108.93113243785623'  
True  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
ID:2, name:restaurant score:5  
ID:3, name:hotel score:3  
\_\_\_triple-quoted block strings\_\_\_  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
\ID  
ame  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
\ID\name  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
category:restaurant  
name,name,name,  
['name', 'name', 'name', '']  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
雷  
雷记澄城水盆羊肉  
string length=107  
name=雷记澄城水盆羊肉  
17  
34.244750060429915  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
lat=34.244;  
lat=34.244 ;  
lat= 34.244;  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
lon:108.931  
False  
True  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
CODE  
code  
['2', '雷记澄城水盆羊肉(红樱路店)', '34.244750060429915', '108.93113243785623', '美食;中餐厅', '3.9', '20.0', '309478.95308006834', '3791230.05342414\n']  
2\_雷记澄城水盆羊肉(红樱路店)\_34.244750060429915\_108.93113243785623\_美食;中餐厅\_3.9\_20.0\_309478.95308006834\_3791230.05342414  
  
b'\xc3\xc0\xca\xb3;\xd6\xd0\xb2\xcd\xcc\xfc'  
美食;中餐厅  
[83, 97]  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
True  
False  
True  
['p', 'y', 't', 'h', 'o', 'n']  
True

help(ord)

Help on built-in function ord in module builtins:  
  
ord(c, /)  
 Return the Unicode code point for a one-character string.

* \转义字符（String backslash characters）

转义字符\n可以转义很多字符，例如\n表示换行，\t表示制表符等。字符\本身也需要转义，用\\表示。如果字符串里有很多字符需要转义，则直接使用无转义的原始字符串r""达到目的，这在表述文件路径时经常使用，例如r'.\data\xian\_poi.csv'（也可使用做斜杠'./data/poi\_piecesOFdata.csv'，则不用原始字符串）。而如果字符串中有很多换行，为了避免每次末尾敲入\n，可以使用"""line1,line2,...,lineN"""表述。

| 转义字符（escape character） | 意义 |
| --- | --- |
| \a | 响铃 Bell |
| \b | 退格，将当前位置移到前一列 Backspace |
| \f | 换页，将当前位置移到下页开头 Formfeed |
| \n | 换行，将当前位置移到下一行开头 Newline（linefeed） |
| \r | 回车，将当前位置移到本行开头 Carriage return |
| \t | 水平制表符 Horizontal tap |
| \v | 垂直制表符 Vertical tap |
| \\ | 代表一个反斜线字符\ Backslash |
| \' | 代表一个单引号 Single quote |
| \" | 代表一个双引号 Double quote |
| \0 | 空字符 Null:binary 0 character（doesn’t end string） |
| \xhh | 十六进制所代表的任意字符 Character with hex value hh（exactly 2 digits） |
| \newline | 忽略（续行） Ignored（continuation line） |

S="""  
line1,  
line2,  
line2  
"""  
print(S)  
  
print("\_"\*50)  
  
# 打印转义字符对应的Unicode码值  
  
print(list(map(ord,['\a','\b','\f','\n','\r','\t','\v','\\','\'','\"',])))

line1,  
line2,  
line2  
  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
[7, 8, 12, 10, 13, 9, 11, 92, 39, 34]

#### **3.4** 字符串格式化

字符串格式化在数据分析领域可以用于以文本方式存储格式化后的数据，方便后续数据读取分析；更经常用于图表中的文字表达，这也包括动态交互内容；也用于代码调试过程中print()打印字符，标识打印变量名，格式化数值，方便查看，或者用于交流。

* % 的方式

'string'%value/(values)/{Ks:Vs}的格式化语句语法为%[(keyname)][flags][width][.precision]typecode， 如果格式化右侧提供的数据结构为字典形式，则keyname为字典键名索引；如果提供的为列表，则按顺序索引；也可以为单个值。flags标记包括，-：在指定字符宽度时，当字符位数小于宽度则字符左对齐，末尾空格；+：在数值前添加整数或负数符号；0：在指定字符宽度时，当字符位数小于宽度则在字符前用0填充；如果为空格，则在前添加空格符号位。width为字符宽度。.precision为数值精度（保留小数点位数）。typecode为转换类型代码（conversion type codes），如表：

| 代码（code） | 含义 |
| --- | --- |
| s | 字符串，或将非字符类型对象用str()转换为字符串 |
| r | 同s，不过用repr()函数转换非字符型对象为字符串 |
| c | 参数为单个字符或者字符的Unicode码时，将Unicode码转换为对应的字符 |
| d | 参数为数值时，转换为带有符号的十进制整数 |
| i | 同d转换数值为整数 |
| u | 同d转换数值为整数 |
| o | 参数为数值时，转换为带有符号的八进制整数 |
| x | 参数为数值时，转换为带有符号的十六进制整数，字母小写 |
| X | 参数为数值时，转换为带有符号的十六进制整数，字母大写 |
| e | 将数值转换为科学计数法格式，字母小写 |
| E | 将数值转换为科学计数法格式，字母大写 |
| f | 将数值转换为十进制浮点数 |
| F | 同f，将数值转换为十进制浮点数 |
| g | 浮点格式。如果指数小于-1或不小于精度（默认为6）使用指数格式，否则使用十进制格式 |
| G | 同g |
| % | %%即为字符% |

如果是使用Python官网提供的[IDLE Shell](https://www.python.org/downloads/)⑧，下述示例中的from scipy.stats import norm，需要安装[SciPy](https://scipy.org/)⑨库，对于windows系统，在Command Prompt下敲入py -3 -m pip install scipy进行安装。另，IDLE Shell可能无法输入中文。推荐使用[anaconda](https://www.anaconda.com/)⑩这一专门用于数据分析，科学计算的（数据科学，data science）解释器。

在数据分析时，会涉及到很多计算结果显示查看，尤其用于交流的代码。下述是正态分布（normal distribution/Gaussian distribution）的概率计算，调用了[SciPy](https://scipy.org/)库的norm.sf(x,loc,scale)，norm.cdf()和norm.ppf()的方法，计算给定值(x)，给定正态分布均值（loc）和标准差（scale），求取大于等于（sf）或小于等于（cdf）给定值的概率；反之，求取满足概率的值（ppf）。

from scipy.stats import norm  
  
print("用.sf计算值大于或等于0.7待概率为：%s",norm.sf(0.7,0,1))   
print("用.cdf计算值小于或等于0.7的概率为：%f"%norm.cdf(0.7,0,1)) #  
print("可以观察到.cdf（<=0.7）概率结果+.sf(>=0.7)概率结果为：%.3f"%(norm.cdf(113,0,1)+norm.sf(113,0,1)))  
print("用.ppf找到给定概率值为0.758036(约75.80%%)的数值为：%e"%norm.ppf(0.758036,0,1))

用.sf计算值大于或等于0.7待概率为：%s 0.24196365222307303  
用.cdf计算值小于或等于0.7的概率为：0.758036  
可以观察到.cdf（<=0.7）概率结果+.sf(>=0.7)概率结果为：1.000  
用.ppf找到给定概率值为0.758036(约75.80%)的数值为：6.999989e-01

数据分析必不可少的表达方式是图表，python可以调用的各类图表扩展库不少，其中最为基础和常用的是[Matplotlib](https://matplotlib.org/)⑪。对于此类库通常不必记忆，一般是在需要图表表述数据分析过程、结果，传达研究发现时，查看各类图表库的示例，或者网络分享的示例，直接复用该代码，加以调整，替换数据，进一步调整表达风格，例如颜色、字体、线型、图样等，完成对自身数据分析的表达。下述表述正态分布的图表表达就是复用Matplotlib曲线示例部分代码，替换数据，调整形式而成。对于Matplotlib中常用的语句和参数，如果经常用到则会被记住，不常用的，只要搜索找到可复用的代码即可。

下述图表除了表达均值为0，标准差为1的正态分布曲线，同时增加了数值0.7的位置表述垂直虚线，并增加了注释。图表文字的代码则是使用了%的字符串格式化方式，如图例部分增加了均值和标准差的显示，注释上增加了小于等于0.7的概率值说明。

图表会在后续的各类数据分析中必不可少的加以应用，以便直观表述各类数据分析，佐证研究成果。不同的分析内容和表述目的会比较选择适合的图表表述方式。

from scipy.stats import norm  
import matplotlib.pyplot as plt  
import matplotlib  
import numpy as np  
matplotlib.rcParams['font.family'] = ['SimSun'] # 解决中文乱字符  
  
fig, ax=plt.subplots(1, 1)  
mean, var, skew, kurt = norm.stats(moments='mvsk')   
print('mean=%s, var=%s, skew=%s, kurt=%s\n'%(mean, var, skew, kurt)) # 验证符合标准正态分布的相关统计量  
x=np.linspace(norm.ppf(0.01),norm.ppf(0.99), 100) # norm.ppf 百分比点函数 - Percent point function (inverse of cdf — percentiles)  
ax.plot(x, norm.pdf(x), 'r-', lw=5, alpha=0.6, label='norm pdf\_%s-%s'%(mean,var)) # norm.pdf为概率密度函数  
ax.legend(loc='best', frameon=False)  
ax.axvline(x=0.7,ymin=0.05,color='k', linestyle='--')  
bbox = dict(boxstyle ="round", fc ="0.8")  
ax.annotate("≤%s的概率为%.3f"%(0.7,norm.cdf(0.7,0,1)),(0.78,0.05),bbox=bbox)  
plt.show()

mean=0.0, var=1.0, skew=0.0, kurt=0.0  
  
  
  
C:\Users\richi\anaconda3\envs\AoT\lib\site-packages\IPython\core\pylabtools.py:151: UserWarning: Glyph 8722 (\N{MINUS SIGN}) missing from current font.  
 fig.canvas.print\_figure(bytes\_io, \*\*kw)

用字符串格式化的方式组织数据，并写入文件。这里第一行写入的为字段名，其它每一行为一组数据，对应字段名使用制表符\t格式化数据，并在每一行末增加\n换行符。因为这里用制表符分割字符串，并没有使用逗号等分隔符，因此格式化字符串连在一起，阅读起来需要仔细分析字符、转义字符和格式化字符，及各类标示符。

poi\_info\_dict={'2': {'name': '雷记澄城水盆羊肉(红樱路店)', 'coordi': {'lat': 34.244750060429915, 'lon': 108.93113243785623}}, '3': {'name': '段府农家菠菜面(红缨路店)', 'coordi': {'lat': 34.245443456204875, 'lon': 108.93110375582032}}, '4': {'name': '平价餐厅(友谊西路店)', 'coordi': {'lat': 34.24253719175486, 'lon': 108.93159854262964}}, '5': {'name': '山妹川菜', 'coordi': {'lat': 34.24522784875664, 'lon': 108.93301750888804}}, '6': {'name': '老三澄合羊肉水盆', 'coordi': {'lat': 34.24224069599151, 'lon': 108.93129159808142}}, '7': {'name': '湘村菜馆(红缨路店)', 'coordi': {'lat': 34.24609764121232, 'lon': 108.9314250012306}}}  
poi\_info\_lst=['%s\t%s\t%.5f\t%.5f\t\n'%(k,v['name'],v['coordi']['lat'],v['coordi']['lon']) for k,v in poi\_info\_dict.items()]  
poi\_info\_lst\_fn='./data/poi\_info\_dict.txt'  
with open(poi\_info\_lst\_fn,'w',encoding='utf8') as f:  
 f.write('%s\t%s\t%s\t%s\t\n'%('ID','name','lat','lon'))  
 f.writelines(poi\_info\_lst)  
with open(poi\_info\_lst\_fn,'r',encoding='utf8') as f:  
 print(f.read())

ID name lat lon   
2 雷记澄城水盆羊肉(红樱路店) 34.24475 108.93113   
3 段府农家菠菜面(红缨路店) 34.24544 108.93110   
4 平价餐厅(友谊西路店) 34.24254 108.93160   
5 山妹川菜 34.24523 108.93302   
6 老三澄合羊肉水盆 34.24224 108.93129   
7 湘村菜馆(红缨路店) 34.24610 108.93143

import datetime  
today=datetime.datetime.now()  
  
print('%s'%today)  
print('%r'%today)  
  
print(ord('a'))  
print('%c'%97)  
print('%c'%'a')  
  
print('%d'%99.35)  
print('%i'%99.35)  
print('%u'%99.35)  
  
print('%o'%109)  
print('%x'%109)  
print('%X'%109)  
  
import math  
print('%e'%(math.pi\*10\*\*6))  
print('%E'%(math.pi\*10\*\*6))  
  
print('%f'%math.pi)  
print('%F'%math.pi)  
print('%f'%0x6D)  
print('%f'%0o155)  
  
print('%g'%(3.30\*10\*\*10))  
print('%g'%(3.30\*10\*\*5))  
print('%G'%(3.30\*10\*\*5))  
  
print('%.3f%%'%(3.0/11.0\*100))

2022-07-22 17:21:38.978794  
datetime.datetime(2022, 7, 22, 17, 21, 38, 978794)  
97  
a  
a  
99  
99  
99  
155  
6d  
6D  
3.141593e+06  
3.141593E+06  
3.141593  
3.141593  
109.000000  
109.000000  
3.3e+10  
330000  
330000  
27.273%

print('name:%s,category:%s,score:%s'%('湘村菜馆','美食\_中餐厅',4))  
info\_dict={'name':'湘村菜馆','category':'美食\_中餐厅','score':4}  
print('name:%(name)s,category:%(category)s,score:%(score)s'%info\_dict)  
  
print('\_'\*50)  
import math  
print('%+-10.3f:)'%-math.pi)  
print('%+-10.3f:)'%math.pi)  
print('%+-10.\*f:)'%(3,math.pi))  
print('%010.3f:)'%math.pi)

name:湘村菜馆,category:美食\_中餐厅,score:4  
name:湘村菜馆,category:美食\_中餐厅,score:4  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
-3.142 :)  
+3.142 :)  
+3.142 :)  
000003.142:)

* format()的方式

format()支持位置索引和关键字，且可以自由搭配进行格式化，从而形成多种格式化方式。对format()格式化的字符串配置宽度和数值精度，一般语法为{idx/keyname:witdh/.precision}，中间由:分割，右侧配置相关参数。

template='name:{0},category:{1},score:{2}'  
print(template.format('湘村菜馆','美食\_中餐厅',4))  
  
template='name:{},category:{},score:{}'  
print(template.format('湘村菜馆','美食\_中餐厅',4))  
  
template='name:{name},category:{category},score:{score}'  
print(template.format(name='湘村菜馆',category='美食\_中餐厅',score=4))  
  
info\_dict={'name':'湘村菜馆','category':'美食\_中餐厅','score':4}  
template='name:{0[name]},category:{0[category]},score:{0[score]}'  
print(template.format(info\_dict))  
  
template='name:%(name)s,category:%(category)s,score:%(score)s'  
print(template%dict(name='湘村菜馆',category='美食\_中餐厅',score=4))  
  
template='name:{0},category:{category},score:{score}'  
print(template.format('湘村菜馆',category='美食\_中餐厅',score=4))  
  
import sys  
print('My {1[name]} runs {0.platform}.'.format(sys,{'name':'Omen'}))  
  
info\_lst=['湘村菜馆','美食\_中餐厅']  
print('name:{0[0]},category:{0[1]},score:{1}'.format(info\_lst,4))

name:湘村菜馆,category:美食\_中餐厅,score:4  
name:湘村菜馆,category:美食\_中餐厅,score:4  
name:湘村菜馆,category:美食\_中餐厅,score:4  
name:湘村菜馆,category:美食\_中餐厅,score:4  
name:湘村菜馆,category:美食\_中餐厅,score:4  
name:湘村菜馆,category:美食\_中餐厅,score:4  
My Omen runs win32.  
name:湘村菜馆,category:美食\_中餐厅,score:4

import math  
print('{0:10}={1:5}'.format('pi',math.pi))  
print('{0:>10}={1:5}'.format('pi',math.pi))  
print('{0:<10}={1:5}'.format('pi',math.pi))  
print('{0}={1:.3f}'.format('pi',math.pi))

pi =3.141592653589793  
 pi=3.141592653589793  
pi =3.141592653589793  
pi=3.142

* f"{}"（f-string）的方式

f-string的{}中为变量名，直接进行格式化。也可以为Python的表达式或者函数（包括lambda匿名函数），及方法等。

number=2  
where="sea"  
  
print(f"{number} of us are gone to {where}.")  
  
print("-"\*50)  
print(f"{17-2}") # {}之内可以放置任何有效的Python表达式  
  
print("-"\*50)  
def to\_uppercase(words):  
 return words.upper()  
print(f"{number} of us are gone to {to\_uppercase(where)}.") # {}之内可以调用函数  
print(f"{number} of us are gone to {where.upper()}.") # {}之内可以直接调用方法  
print(f"{(lambda a,b:a+b)(1,1)} of us are gone to {where}.") # {}之内可以使用lambda匿名函数

2 of us are gone to sea.  
  
--------------------------------------------------  
  
15  
  
--------------------------------------------------  
  
2 of us are gone to SEA.  
2 of us are gone to SEA.  
2 of us are gone to sea.

f-string自定义格式：对齐、宽度、符号、补零、精度和进位制等于%和format()方式基本同，其基本格式为，{content:format}，配置宽度和精度时为{content:width.precision}。

import math  
pi=math.pi  
print(f"The number pi approximately equal to {pi:.3f}.") # 配置数值精度  
print(f"approximately equal to {pi:12.3f}") # 同时配置宽度和精度  
print(f"approximately equal to {pi:012.3f}") # 以0填充宽度  
print(f"approximately equal to {pi:.3e}") # 用科学计数法  
print(f"approximately equal to {pi:.3%}") # 百分比形式  
print(f"approximately equal to {pi:.3g}") # 为有效位数（小数点前位数+小数点后位数）  
  
s="world"  
print(f"Hello {s:10s}!") # 字符串形式

The number pi approximately equal to 3.142.   
approximately equal to 3.142  
approximately equal to 00000003.142  
approximately equal to 3.142e+00  
approximately equal to 314.159%  
approximately equal to 3.14  
Hello world !

格式化方法可以使用时间格式化方式。

import datetime  
dt=datetime.datetime.today()  
print(dt)  
print(f'The time is {dt:%Y-%m-%d (%a) %H:%M:%S}')

2022-11-02 08:31:21.267394  
The time is 2022-11-02 (Wed) 08:31:21

#### **3.5** 正则表达式（[regular expression，re](https://docs.python.org/3/library/re.html)）①

字符串处理常用到标准库模块中的re，regular expression（正则表达式），re非常强大，可以处理更复杂的字符串，本质是可以匹配文本片断的模式。最简单的re是普通字符串，即大多数字母和字符一般都会和自身匹配，例如’python’可以匹配字符串’python’。

* 字符匹配-模式语法

re可以使用特殊字符的方式匹配一个或者多于一个的字符串，例如使用点号.，可以匹配除了换行符之外的任何字符，但是.只匹配一个字母，多于一个或者零个都不会匹配。点号特殊字符只匹配一个字符，如果希望匹配多个可以使用\*星号，匹配前面表达式的0个或者多个副本， 并匹配尽可能多的副本；而+加号则匹配至少1个或者多个副本；？问号也是匹配0个或者多个副本。如果想确定具体匹配的数量区间，可以使用{m,n}的方式，即匹配前面表达式的第m到n各副本，如果省略了m则默认值为0，如果省略了n，则默认设置为无穷大。

在使用\*、+、?、{m,n}时，如果模式为r'Hello Py\*thon!'，则\*星号只对星号之前的一个字符y进行匹配，如果希望同时对P也进行匹配，则需要使用[]中括号字符集把Py括起来即[Py]，完整的模式为r'Hell [Py]\*thon!'。还可以应用于更加广泛的范围，例如[a-z]能够匹配a到z的任意一个字符，甚至[a-zA-Z0-9]的使用方式可以匹配任意大小写字母和数字。同时可以配合使用^字符放置于字符集的开头反转字符集，例如[^abc]则是匹配除了a、b、c之外的字符。

在建立re表达式时，希望能够选择性的匹配几种不同的情况，例如即匹配字符’python’又匹配’grasshopper’ ，为同时匹配’python’和’grasshopper’，那么就需要使用|管道符号，re表达式可以写为'python|grasshopper'。如果仅是对部分模式使用管道符号即选择符，可以用圆括号括起需要的部分，例如'p(ython|erl)'。

在匹配字符串时，有时仅需要在开头或者结尾处匹配，这时可以使用脱字符^标记开始，使用美元符号$标记结尾。

主要使用的re特殊字符列表如下：

| 字符 | 描述 |
| --- | --- |
| . | 匹配除换行符外任何字符串 |
| ^ | 匹配字符串的开始标志 |
| $ | 匹配字符串的结束标志 |
| \* | 匹配前面表达式的0个或多个副本，匹配尽可能多的副本。例如ab\*会匹配a，ab，或者abb，abbb等尽可能多（任何数量）的跟随a后b的副本 |
| + | 匹配前面表达式的1个或多个副本，匹配尽可能多的副本。例如ab\*会匹配除了a外的ab，或者abb，abbb等尽可能多（任何数量）的跟随a后b的副本 |
| ? | 匹配前面表达式的0个或多个副本，例如ab?将匹配a或者ab |
| \*? | 匹配前面表达式的0个或多个副本，匹配尽可能少的副本 |
| +? | 匹配前面表达式的1个或多个副本，匹配尽可能少的副本 |
| ?? | 匹配前面表达式的0个或1个副本，匹配尽可能少的副本 |
| {m} | 准确匹配前面表达式的m个副本。例如a{6}会精确匹配6个a，而不是5个或其它 |
| {m,n} | 匹配前面表达式的第m到n个副本，匹配尽可能多的副本。如果省略了m，则默认为0；如果省略了n，默认为无穷大。例如a{3,5}会匹配3-5个a字符。而a{4,}b会匹配aaaab，甚至无以计数前置b的a字符，但不会匹配aaab，因为a 的数量少于了4 |
| {m,n}? | 匹配前面表达式的第m到n个副本， 匹配尽可能少的副本。例如a{3,5}会匹配5个a字符，但是，a{3,5}?，只会匹配3个a字符 |
| [...] | 匹配一组字符，如'[abcdef]',或'[a-zA-Z]'。特殊字符，例如\*在字符集中将失去特殊字符意义， 例如[(+\*)]会匹配(，+，\*或)。 |
| [^...] | 匹配集合中未包含的字符，例如[^5]将匹配除了5之外的所有字符 |
| A\|B | 匹配A或B |
| (...) | 匹配圆括号中的re表达式（圆括号中的内容为一个分组），并保存匹配的子字符串。在匹配时，分组中的内容可以使用所获取的MatchObject对象的group()方法获取 |
| (?aiLmsux) | 扩展标记法，以?符号开头，其后第一个字符决定采用什么样的语法。其中，a只匹配 ASCII 字符re.A(re.ASCII)；i忽略大小写re.I(re.IGNORECASE)；L 由当前语言区域决定re.L (locale dependent),；m多行模式re.M(re.MULTILINE)；s为.匹配全部字符re.S(re.DOTALL)；u Unicode匹配，Python3默认开启这个模式re.U；x冗长模式re.X(re.VERBOSE) |
| (?:...) | 常规括号的非捕获版本（A non-capturing version of regular parentheses.），匹配括号内的任何re表达式，但是分组所匹配的子字符串不能再执行匹配后获取或是在之后的模式种被引用 |
| (?P<name>...) | 类似于常规括号，但组匹配的子字符串可通过符号组名成访问。组名必须是有效的Python标示符，并且每个组名只能在re表达式中定义一次。 |
| (?P=name) | 对命名组的反向引用。它匹配与早先名为name的组匹配到的任何文本（字符串） |
| (?#...) | 注释信息，里面的内容会被忽略 |
| (?=...) | 只有在括号中的模式匹配时，才匹配前面的表达式，为a lookahead assertion，前视断言 |
| (?!...) | 只有在括号中的模式不匹配时，才匹配前面的表达式，为a negative lookahead assertion， 前视取反 |
| (?<=...) | 如果括号后面的表达式前面的值与括号中的模式匹配，则匹配该表达式，为a positive lookbehand assersion |
| (?<!...) | 如果括号后面的表达式前面的值与括号中的模式不匹配，则匹配该表达式，为a negative lookbehand assersion |
| (?(id/name)yes-pattern\|no-pattern) | 检查id或name标识的re表达式组是否存在。如果存在，则匹配re表达式的yes-pattern ；否则，匹配可选的表达式no-pattern |

一些用 \开始的特殊字符所表示的预定义字符集通常是很有用的，例如数字集，字母集，或其它非空字符集，列表如下：

| 字符 | 描述 |
| --- | --- |
| \number | 匹配相同组编号的组内容。 组编号范围为1-99，从左侧开始 |
| \A | 仅匹配字符串的开始标志 |
| \b | 匹配空字符串，但只匹配单词的开头和结尾。例如r'\bfoo\b'匹配'foo'，'foo.'，'bar foo baz'， 但是不会匹配'foobar'，或者'foo3' |
| \B | 匹配空字符串，但仅当它不在单词的开头或结尾时。例如r'py\B'匹配'python'，'py3'，'py2'，但是不会匹配'py'，'py.'或者'py!' |
| \d | 匹配任何Unicode中的十进制数，等同于r'[0-9]' |
| \D | 匹配任何非十进制数的字符，等同于r'[^0-9]' |
| \s | 匹配Unicode空白字符，包括['\t','\n','\r','\f','\v']及许多其它字符 |
| \S | 匹配任何非空格字符 |
| \w | 匹配Unicode单词字符，这包括可以成为任何语言中单词部分的大多数字符，及数字和下划线。如果使用ASCII标志，仅匹配'[a-zA-Z0-9]' |
| \W | 匹配\w 中定义集合中不包含的字符 |
| \z | 仅匹配字符串的末尾 |
| \\ | 匹配反斜杠本身 |

import re  
  
kml\_description='<description>线路开始时间：2017-07-20 08:14:41,结束时间：2017-07-20 20:53:03,线路长度：197801。由GPS工具箱导出。</description>'  
pattern='description'  
print(re.findall(pattern,kml\_description)) # 使用re.findall()方法以列表形式返回给定模式的所有匹配项  
  
# .  
pattern='.description'  
print(re.findall(pattern,kml\_description))   
  
# ? +  
pattern=r'w?cadesign\.cn, w+\.cadesign\.cn' # 用转义字符使用点号，而不是用作特殊字符  
text='cadesign.cn, www.cadesign.cn'  
print(re.findall(pattern,text)) # ？号可以匹配0个 或者多个字符，因此即使不存在字符'w'，也会匹配'cadesign.cn；+号需要匹配至少一个，并尽可能多的匹配， 因此可以提取出'www.cadesign.cn'  
  
# {m}  
pattern=r'w{2}\.cadesign\.cn'   
print(re.findall(pattern,text))   
  
# [...]  
pattern='[Py]\*thon!'   
textA='Hello Python!'  
textB='Hello Pthon!'  
textC='Hello ython!'   
textD='Hello thon!'  
print(re.findall(pattern,textA))  
print(re.findall(pattern,textB))  
print(re.findall(pattern,textC))  
print(re.findall(pattern,textD))  
  
# A|B  
pattern='<description>|</description>'  
print(re.findall(pattern,kml\_description))

['description', 'description']  
['<description', '/description']  
['cadesign.cn, www.cadesign.cn']  
['ww.cadesign.cn']  
['Python!']  
['Pthon!']  
['ython!']  
['thon!']  
['<description>', '</description>']

# (?aiLmsux)  
print(re.findall('(?i)ab', 'Ab')) # i-为忽略大小写  
print(re.findall('(?si)ab.', 'Ab\n')) # s为.匹配了全部字符，包括换行符，i忽略大小写。连用了s和i  
  
print(re.findall('^a.', 'ab\nac'))  
print(re.findall('(?m)^a.', 'ab\nac')) # m为多行模式  
  
print(re.findall('(?x)\d+\.\d\*', '3.1415926nan')) # x为冗长模式

['Ab']  
['Ab\n']  
['ab']  
['ab', 'ac']  
['3.1415926']

# (?:...)  
print(re.findall('(abc){2}', 'abcabc')) # 常规捕获版本，捕获到()分组内的匹配字符  
print(re.findall('(?:abc)', 'abcabc')) # 非捕获版本，将()分组视为一个整体  
  
print(re.findall('(a(bc))cbs', 'abccbs'))  
print(re.findall('(a(?:bc))cbs', 'abccbs')) # 嵌套捕获模式  
  
print(re.findall('(abc)|cbs', 'abccbs'))  
print(re.findall('(abc)|cbs', 'cbs'))  
print(re.findall('(?:abc)|cbs', 'cbs'))

['abc']  
['abc', 'abc']  
[('abc', 'bc')]  
['abc']  
['abc', '']  
['']  
['cbs']

# (?P<name>...)与(?P=name)，和(?#...)  
print(re.findall('(?P<name>abc)\\1', 'abcabc'))  
print(re.findall('(?P<gname>abc)\d+(?P=gname)(?#后面的gname匹配到前面的匹配到的字符abc)', 'abc996abc'))

['abc']  
['abc']

# (?=...) 与(?!...)  
print(re.findall('Isaac (?=Asimov)', 'Isaac Asimov'))  
print(re.findall('Isaac (?=Asimov)', 'Isaac Asi'))  
print(re.findall('Isaac (?!Asimov)', 'Isaac Asi'))

['Isaac ']  
[]  
['Isaac ']

# (?<=...)与(?<!...)  
m=re.search('(?<=abc)def', 'abcdef')  
print(m.group(0))  
  
m=re.search(r'(?<=-)\w+', 'spam-egg')  
print(m.group(0))

def  
egg

* re方法

正则表达式的模式需要配合正则表达式的方法使用，主要方法的解释如下：

re.findall(pattern, string)， 以列表形式返回给定模式的所有匹配项；

re.search(pattern,string)， 会在给定字符串中寻找第一个匹配给定正则表达式的子字符串，返回匹配对象（Match Object）。如果字符串中没有位置与模式匹配，则返回 None；

re.match(pattern,string)， 会在给定字符串的开头匹配正则表达式，返回匹配对象。如果字符串中没有位置与模式匹配，则返回 None；

re.fullmatch(pattern,string)，如果整个字符串与正则表达式模式匹配，则返回相应的匹配对象。 如果字符串与模式不匹配，则返回 None；

re.split(pattern,string[,maxsplit=0])，按出现的模式拆分字符串。 如果在模式中使用捕获括号，则模式中所有组的文本也会作为结果列表的一部分返回。 如果 maxsplit 不为零，则最多发生maxsplit个拆分，并将字符串的其余部分作为列表的最后一个元素返回;

re.sub(pattern,repl,string), 使用给定的替换内容repl将匹配模式pattern的子字符串替换掉。如果未找到该模式，则字符串原样返回。repl可以是字符串或函数;

re.subn(pattern, repl, string, count=0,flags=0)， 同re.sub()，只是返回一个元组为(new\_string, number\_of\_subs\_made);

re.escape(string)， 可以对字符串中所有可能被解释为正则运算符的字符进行转义，避免输入较多的反斜杠；

re.compile(pattern)， 可以将以字符串书写的正则表达式转换为模式对象，例如转换为模式对象后可以直接使用pattern.search(string)的方法，这与re.search(pattern,string)方式一样。因为使用re模块的方法时，不管是re.search()还是re.math()都会在内部将字符串表示的正则表达式转换为正则表达式模式对象，因此re.compile()的方法可以避免每次使用模式时都得 从新转化的过程；

re.purge()， 清除正则表达式缓存。

import re  
print(re.findall(r'\bf[a-z]\*', 'which foot or hand fell fastest'))  
print(re.findall(r'(\w+)=(\d+)', 'set width=20 and height=10'))  
  
pattern='[a-z]+'  
text='<coordinates>120.130095,30.21169,20.5</coordinates>'  
print(re.findall(pattern,text))  
pattern=r'(?x)\d+\.\d\*'  
print(re.findall(pattern,text))  
  
  
pattern='description'  
text='<description>GPS工具箱导出数据</description>'  
print(re.search(pattern,text))  
print(re.search(pattern,text).group())  
  
print(re.match(pattern,text))  
  
if re.search(pattern,text):  
 print('found a match')  
else:  
 print('no match')   
   
pattern='g...s'  
text='geeks'  
print(re.fullmatch(pattern,text))

['foot', 'fell', 'fastest']  
[('width', '20'), ('height', '10')]  
['coordinates', 'coordinates']  
['120.130095', '30.21169', '20.5']  
<re.Match object; span=(1, 12), match='description'>  
description  
None  
found a match  
<re.Match object; span=(0, 5), match='geeks'>

re.search()和re.match()返回的MatchObject实例对象包含关于分组内容的信息，和匹配值的位置数据。组就是放置在圆括号内的子模式。可以通过m.group()返回组，m.start()获取组的开始索引值，m.end()则获取结束位置索引值，m.span()返回区间值。

m=re.match(r'www\.(.\*)\..{3}','www.python.org') # 对字符串进行模式匹配，返回MatchObject对象  
print( m.group(1))  
print(m.start(1))  
print(m.end(1))  
print(m.span(1))

python  
4  
10  
(4, 10)

下述应用了手机APP记录调研路径，存储为KML数据格式，在Python中读取，提取需要数据内容的简化示例。对于KML格式数据，其后缀名通常为.kml，与KMZ一样是[Google Earth](https://www.google.com/earth/versions/)所使用的点、线和面的地标文件格式。记录的数据均由类似<coordinates>...</coordinates>，<description>...</description>，<name>...</name>等<identifier>text</identifier>模式构成，这有助于数据的提取。

pattern\_coordi=re.compile('<coordinates>(.\*?)</coordinates>')   
pattern\_description=re.compile('<description>(.\*?)</description>')  
  
coordi\_text='<coordinates>120.132007,30.300508,9.7</coordinates>'  
description\_text='<description>线路开始时间：2017-07-20 08:14:41,结束时间：2017-07-20 20:53:03,线路长度：197801。由GPS工具箱导出。</description>'  
  
print(pattern\_coordi.findall(coordi\_text))  
print(pattern\_description.findall(description\_text))

['120.132007,30.300508,9.7']  
['线路开始时间：2017-07-20 08:14:41,结束时间：2017-07-20 20:53:03,线路长度：197801。由GPS工具箱导出。']

print(re.split(r'\W+', 'Words, words, words.'))  
print(re.split(r'(\W+)', 'Words, words, words.')) # 匹配分组文本作为列表一部分返回  
print(re.split(r'\W+', 'Words, words, words.', 1)) # 只拆分了一次，余下部分作为列表一部分返回  
print(re.split('[a-f]+', '0a3B9', flags=re.IGNORECASE))  
print(re.split('[a-f]+', '0a3B9'))

['Words', 'words', 'words', '']  
['Words', ', ', 'words', ', ', 'words', '.', '']  
['Words', 'words, words.']  
['0', '3', '9']  
['0', '3B9']

如果分隔符中有捕获组并且它在字符串的开头匹配，则结果将以空字符串开头。 这同样适用于字符串的结尾。

模式的空匹配仅在与先前的空匹配不相邻时才拆分字符串。

print(re.split(r'(\W+)', '...words, words...'))  
print("\_"\*50)  
  
print(re.split(r'\b', 'Words, words, words.'))  
print(re.split(r'\W+', '...words...'))  
print(re.split(r'\W\*', '...words...')) # \*匹配0个或多个，+匹配1个或多个   
print(re.split(r'(\W\*)', '...words...'))

['', '...', 'words', ', ', 'words', '...', '']  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
['', 'Words', ', ', 'words', ', ', 'words', '.']  
['', 'words', '']  
['', '', 'w', 'o', 'r', 'd', 's', '', '']  
['', '...', '', '', 'w', '', 'o', '', 'r', '', 'd', '', 's', '...', '', '', '']

如果repl是一个函数，则每次出现不重叠的模式时都会调用它。 该函数采用单个匹配对象参数，并返回替换字符串。

print(re.sub(r'def\s+([a-zA-Z\_][a-zA-Z\_0-9]\*)\s\*\(\s\*\):',  
 r'static PyObject\*\npy\_\1(void)\n{',  
 'def myfunc():'))  
  
# 这里用到了一个自定义函数  
def dashrepl(matchobj):  
 if matchobj.group(0) == '-': return ' '  
 else: return '-'  
print(re.sub('-{1,2}', dashrepl, 'pro----gram-files'))  
  
print(re.sub(r'\sAND\s', ' & ', 'Baked Beans And Spam', flags=re.IGNORECASE))

static PyObject\*  
py\_myfunc(void)  
{  
pro--gram files  
Baked Beans & Spam

转义模式中的特殊字符。 例如匹配文本中可能包含正则表达式元字符的任意文字字符串。

import string  
print(re.escape('https://www.python.org'))  
legal\_chars = string.ascii\_lowercase + string.digits + "!#$%&'\*+-.^\_`|~:"  
print('[%s]+' % re.escape(legal\_chars))  
operators = ['+', '-', '\*', '/', '\*\*']  
print('|'.join(map(re.escape, sorted(operators, reverse=True))))

https://www\.python\.org  
[abcdefghijklmnopqrstuvwxyz0123456789!\#\$%\&'\\*\+\-\.\^\_`\|\~:]+  
/|\-|\+|\\*\\*|\\*

text='<coordinates>120.130095,30.21169,20.5</coordinates>'  
pattern=re.compile(r'(?x)\d+\.\d\*')  
pattern.findall(text)

['120.130095', '30.21169', '20.5']

注释（Notes）：

① re, Regular expression operations（正则表达式），用于匹配和查找字符串（<https://docs.python.org/3/library/re.html>）。

② CSV, comma-separated values。是一个以逗号分隔的文本文件<https://en.wikipedia.org/wiki/Comma-separated_values>）。

③ JSON, JavaScript Object Notation，是一种开放的标准文件格式和数据交换格式，使用人类可读的文本来存储和传输由属性-值对和数组（或其他可序列化的值）组成的数据对象。它是一种常见的数据格式，在电子数据交换中具有多种用途，包括网络应用程序与服务器的数据交换（<https://en.wikipedia.org/wiki/JSON>）。

④ pandas.read\_csv()，（<https://pandas.pydata.org/docs/reference/api/pandas.read_csv.html>）；pandas.DataFrame.to\_csv，（<https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.to_csv.html>）。

⑤ CSV库，实现了读取和写入CSV格式的表格数据的类（<https://docs.python.org/3/library/csv.html>）。

⑥ notepad++，是一个自由的源代码编辑器和记事本的替代品，支持多种语言（<https://notepad-plus-plus.org/>）。

⑦ Python String Methods，Python有一组内置的方法，可以在字符串上使用（<https://www.w3schools.com/python/python_ref_string.asp>）。

⑧ IDLE Shell，在Python官网下载解释器（<https://www.python.org/downloads/>）。

⑨ SciPy，Python中科学计算的基本算法库（<https://scipy.org/>）。

⑩ Anaconda，包含了Python的环境管理、代码编辑器、包管理等软件平台（<https://www.anaconda.com/>）。

⑪ Matplotlib，Python图表库，课创建静态、动画和交互式数据可视化（<https://matplotlib.org/>）。

PC3-ipynb download