字符串常见操作

字符串的常见操作包括：

- 获取长度:len
- 查找内容:find,index,rfind,rindex
- 判断:startswith,endswith,isalpha,isdigit,isalnum,isspace
- 计算出现次数:count
- 替换内容:replace
- 切割字符串:split,rsplit,splitlines,partition,rpartition
- 修改大小写:capitalize,title,upper,lower
- 空格处理:ljust,rjust,center,lstrip,rstrip,strip
- 字符串拼接:join

注意：在Python中，字符串是不可变的！所有的字符串相关方法，都不会改变原有的字符串，都是返回一个结果，在这个新的返回值里，保留了执行后的结果！

一、 len

len函数可以获取字符串的长度。

```
mystr = '今天天气好晴朗，处处好风光呀好风光'
print(len(mystr))  # 17 获取字符串的长度
```

二、 查找

查找相关的方法，使用方式大致相同，但是略有区别。

1. find

查找指定内容在字符串中是否存在，如果存在就返回该内容在字符串中第一次出现的开始位置索引值，如果不存在，则返回-1.

语法格式：

```
S.find(sub[, start[, end]]) -> int
```

示例：

```
mystr = '今天天气好晴朗，处处好风光呀好风光'
print(mystr.find('好风光'))  # 10 '好风光'第一次出现时，'好'所在的位置
print(mystr.find('你好'))  # -1  '你好'不存在，返回 -1
print(mystr.find('风', 12))  # 15 从下标12开始查找'风',找到风所在的位置试15
print(mystr.find('风光',1,10)) # -1 从下标1开始到12查找"风光",未找到，返回 -1
```

2. rfind

类似于 find()函数，不过是从右边开始查找。

```
mystr = '今天天气好晴朗，处处好风光呀好风光'
print(mystr.rfind('好')) # 14
```

3.index

跟find()方法一样，只不过，find方法未找到时，返回-1,而str未找到时，会报一个异常。

语法格式：

```
S.index(sub[, start[, end]]) -> int
```

4.rindex

类似于 index()，不过是从右边开始。

三、判断

python提供了非常丰富的方法，可以用来对一个字符串进行判断。

1. startswith

判断字符串是否以指定内容开始。语法格式:

```
S.startswith(prefix[, start[, end]]) -> bool
```

示例：

```
mystr = '今天天气好晴朗，处处好风光呀好风光'
print(mystr.startswith('今'))  # True
print(mystr.startswith('今日')) # False
```

2. endswith

判断字符串是否以指定内容结束。

```
mystr = '今天天气好晴朗，处处好风光呀好风光'
print(mystr.endswith('好风光')) #True
print(mystr.endswith('好日子')) #False
```

3. isalpha

判断字符串是否是纯字母。

```
mystr = 'hello'
print(mystr.isalpha())  # True
mystr = 'hello world'
print(mystr.isalpha()) # False 因为中间有空格
```

4. isdigit

判断一个字符串是否是纯数字，只要出现非0~9的数字，结果就是False.

```
mystr = '1234'
print(mystr.isdigit()) # True
mystr = '123.4'
print(mystr.isdigit()) # False
mystr = '-1234'
print(mystr.isdigit()) # False
```

5. isalnum

判断是否由数字和字母组成。只要出现了非数字和字母，就返回False.

```
mystr = 'abcd'
print(mystr.isalnum())  # True
mystr = '1234'
print(mystr.isalnum()) # True
mystr = 'abcd1234'
print(mystr.isalnum()) # True
mystr = 'abcd1234_'
print(mystr.isalnum()) # False
```

6. isspace

如果 mystr 中只包含空格，则返回 True，否则返回 False.

```
mystr = ''
print(mystr.isspace()) # False mystr是一个空字符串
mystr = '  '
print(mystr.isspace()) # True 只有空格
mystr = ' d'
print(mystr.isspace()) # False 除了空格外还有其他内容
```

四、count

返回 str在start和end之间 在 mystr里面出现的次数。

语法格式:

```
S.count(sub[, start[, end]]) -> int
```

示例:

```
mystr = '今天天气好晴朗，处处好风光呀好风光'
print(mystr.count('好'))  # 3. '好'字出现三次
```

五、替换

替换字符串中指定的内容，如果指定次数count，则替换不会超过count次。

```
mystr = '今天天气好晴朗，处处好风光呀好风光'
newstr = mystr.replace('好', '坏')
print(mystr)  # 今天天气好晴朗，处处好风光呀好风光  原字符串未改变!
print(newstr)  # 今天天气坏晴朗，处处坏风光呀坏风光 得到的新字符串里，'好'被修改成了'坏'

newstr = mystr.replace('好','坏',2)  # 指定了替换的次数
print(newstr) # 今天天气坏晴朗，处处坏风光呀好风光 只有两处的'好'被替换成了'坏'
```

六、内容分隔

内容分隔主要涉及到split,splitlines,partition和rpartition四个方法。

split

以指定字符串为分隔符切片，如果 maxsplit有指定值，则仅分隔 maxsplit+1 个子字符串。**返回的结果是一个列表。**

```
mystr = '今天天气好晴朗，处处好风光呀好风光'
result = mystr.split() # 没有指定分隔符，默认使用空格，换行等空白字符进行分隔
print(result) #['今天天气好晴朗，处处好风光呀好风光'] 没有空白字符，所以，字符串未被分隔

result = mystr.split('好')  # 以 '好' 为分隔符
print(result) # ['今天天气', '晴朗，处处','风光呀,'风光']

result = mystr.split("好",2) # 以 '好' 为分隔符，最多切割成3份
print(result) # ['今天天气', '晴朗，处处', '风光呀好风光']
```

rsplit

用法和split基本一致，只不过是从右往左分隔。

```
mystr = '今天天气好晴朗，处处好风光呀好风光'
print(mystr.rsplit('好',1))  #['今天天气好晴朗，处处好风光呀', '风光']
```

splitlines

按照行分隔，返回一个包含各行作为元素的列表。

```
mystr = 'hello \nworld'
print(mystr.splitlines())
```

partition

把mystr以str分割成三部分,str前，str和str后，三部分组成一个元组

```
mystr = '今天天气好晴朗，处处好风光呀好风光'
print(mystr.partition('好'))  # ('今天天气', '好', '晴朗，处处好风光呀好风光')
```

rpartition

类似于 partition()函数,不过是从右边开始.

```
mystr = '今天天气好晴朗，处处好风光呀好风光'
print(mystr.rpartition('好'))   # ('今天天气好晴朗，处处好风光呀', '好', '风光')
```

七、修改大小写

修改大小写的功能只对英文有效，主要包括，首字母大写capitalize,每个单词的首字母大写title,全小写lower,全大写upper.

capitalize

第一个单词的首字母大写。

```
mystr = 'hello world'
print(mystr.capitalize()) # Hello world
```

title

每个单词的首字母大写。

```
mystr = 'hello world'
print(mystr.title()) # Hello World
```

lower

所有都变成小写。

```
mystr = 'hElLo WorLD'
print(mystr.lower()) # hello world
```

upper

所有都变成大写。

```
mystr = 'hello world'
print(mystr.upper())  #HELLO WORLD
```

八、空格处理

Python为我们提供了各种操作字符串里表格的方法。

1. ljust

返回指定长度的字符串，并在右侧使用空白字符补全(左对齐)。

```
str = 'hello'
print(str.ljust(10))  # hello     在右边补了五个空格
```

2. rjust

返回指定长度的字符串，并在左侧使用空白字符补全(右对齐)。

```
str = 'hello'
print(str.rjust(10))  #       hello在左边补了五个空格
```

3. center

返回指定长度的字符串，并在两端使用空白字符补全(居中对齐)

```
str = 'hello'
print(str.center(10))  #  hello    两端加空格，让内容居中
```

4. lstrip

删除 mystr 左边的空白字符。

```
mystr = '    he   llo      '
print(str.lstrip())  #he   llo       只去掉了左边的空格，中间和右边的空格被保留
```

5. rstrip

删除 mystr 右边的空白字符。

```
mystr = '    he   llo      '
print(str.rstrip())  #    he   llo右边的空格被删除
```

6. strip

删除两断的空白字符。

```
str = '    he   llo      '
print(str.strip())  #he   llo
```

字符串拼接

把参数进行遍历，取出参数里的每一项，然后再在后面加上mystr

语法格式:

```
S.join(iterable)
```

示例:

```
mystr = 'a'
print(mystr.join('hxmdq'))  #haxamadaq  把hxmd一个个取出，并在后面添加字符a. 最后的 q 保留，没有加 a
print(mystr.join(['hi','hello','good']))  #hiahelloagood
```

作用：可以把列表或者元组快速的转变成为字符串，并且以指定的字符分隔。

```
txt = '_'
print(txt.join(['hi','hello','good'])) #hi_hello_good
print(txt.join(('good','hi','hello'))) #good_hi_hello
```

字符串运算符

1. 字符串和字符串之间能够使用加法运算符，作用是将两个字符串拼接成为一个字符串。例如: `'hello' + 'world'` 的结果是 `'helloworld'`
2. 字符串和数字之间可以做乘法运算，结果是将指定的字符串重复多次。例如: `'hello'*2` 的结果是 `hellohello`
3. 字符串和字符串之间，如果使用比较运算符进行计算，会获取字符对应的编码，然后进行比较。
4. 除上述几种运算符以外，字符串默认不支持其他运算符。