字符串常用操作(课外阅读)

<1>capitalize

把字符串的第一个字符大写

```
mystr.capitalize()
```

```
>>> mystr = 'hello world itcast and itcastcpp'
>>> mystr.capitalize()
'Hello world itcast and itcastcpp'
>>>
```

<2>title

把字符串的每个单词首字母大写

```
>>> a = "hello itcast"
>>> a.title()
'Hello Itcast'
```

<3>startswith

检查字符串是否是以 hello 开头, 是则返回 True，否则返回 False

```
mystr.startswith(hello)
```

```
>>> mystr = 'hello world itcast and itcastcpp'
>>> mystr.startswith("hello")
True
>>> mystr.startswith("Hello")
False
>>>
```

<4>endswith

检查字符串是否以obj结束，如果是返回True,否则返回False.

```
mystr.endswith(obj)
```

```
>>> mystr = 'hello world itcast and itcastcpp'
>>> mystr.endswith('cpp')
True
>>> mystr.endswith('app')
False
>>>
```

<5>lower

转换 mystr 中所有大写字符为小写

```
mystr.lower()
```

```
>>> mystr = 'HELLO world itcast and itcastcpp'
>>> mystr.lower()
'hello world itcast and itcastcpp'
>>>
```

<6>upper

转换 mystr 中的小写字母为大写

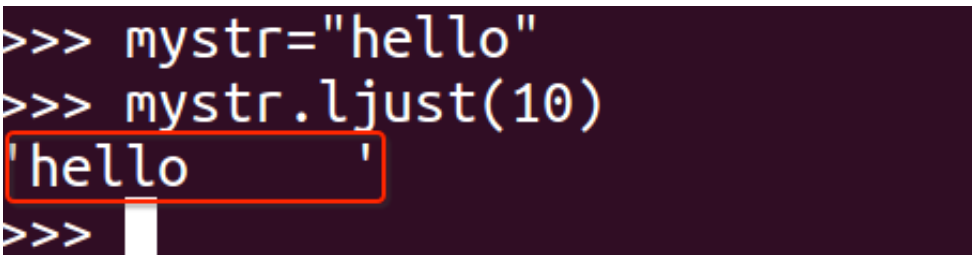
```
mystr.upper()
```

```
>>> mystr = 'HELLO world itcast and itcastcpp'
>>> mystr.upper()
'HELLO WORLD ITCAST AND ITCASTCPP'
>>>
```

<7>ljust

返回一个原字符串左对齐,并使用空格填充至长度 width 的新字符串

```
mystr.ljust(width)
```



<8>rjust

返回一个原字符串右对齐,并使用空格填充至长度 width 的新字符串

```
mystr.rjust(width)
```

```
>>> mystr="hello"
>>> mystr.rjust(10)
'      hello'
>>>
```

<9>center

返回一个原字符串居中,并使用空格填充至长度 width 的新字符串

```
mystr.center(width)
```

```
>>> mystr = 'hello world itcast and itcastcpp'
>>> mystr.center(50)
'         hello world itcast and itcastcpp          '
>>>
```

<10>lstrip

删除 mystr 左边的空白字符

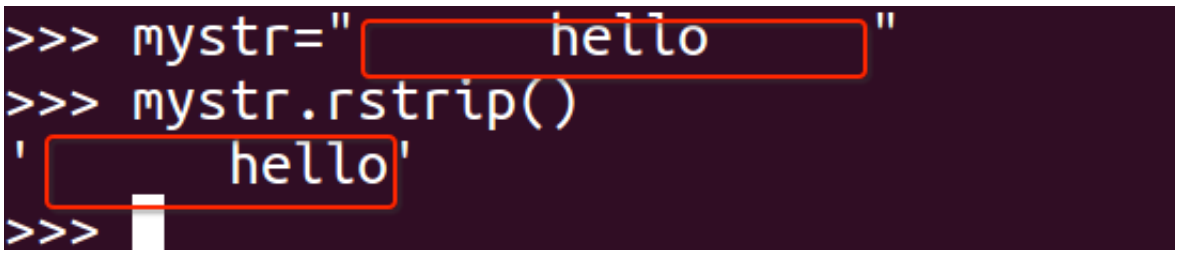
```
mystr.lstrip()
```

```
>>> mystr="      hello"
>>> mystr.lstrip()
'hello'
>>> mystr="      hello      "
>>> mystr.lstrip()
'hello      '
>>>
```

<11>rstrip

删除 mystr 字符串末尾的空白字符

```
mystr.rstrip()
```



<12>strip

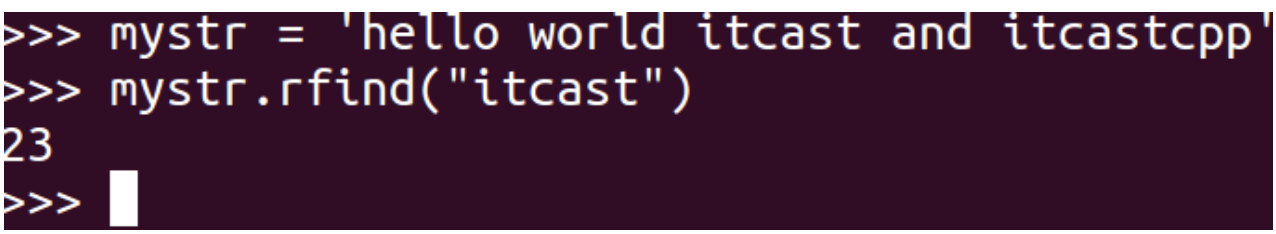
删除mystr字符串两端的空白字符

```
>>> a = "\n\t itcast \t\n"
>>> a.strip()
'itcast'
```

<13>rfind

类似于 find()函数，不过是从右边开始查找.

```
mystr.rfind(str, start=0,end=len(mystr) )
```



<14>rindex

类似于 index()，不过是从右边开始.

```
mystr.rindex( str, start=0,end=len(mystr))
```

```
>>> mystr = 'hello world itcast and itcastcpp'
>>> mystr.rindex("IT")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: substring not found
>>>
```

<15>partition

把mystr以str分割成三部分,str前，str和str后

```
mystr.partition(str)
```

```
>>> mystr = 'hello world itcast and itcastcpp'
>>> mystr.partition("itcast")
('hello world ', 'itcast', ' and itcastcpp')
>>>
```

<16>rpartition

类似于 partition()函数,不过是从右边开始.

```
mystr.rpartition(str)
```

```
>>> mystr = 'hello world itcast and itcastcpp'
>>> mystr.partition("itcast")
('hello world ', 'itcast', ' and itcastcpp')
>>> mystr.rpartition("itcast")
('hello world itcast and ', 'itcast', 'cpp')
>>>
```

<17>splitlines

按照行分隔，返回一个包含各行作为元素的列表

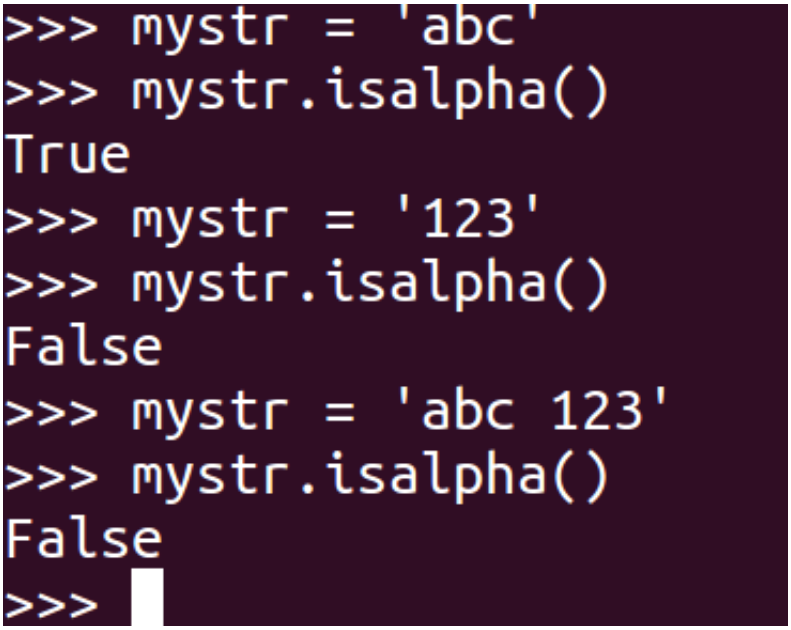
```
mystr.splitlines()
```

```
>>> mystr="hello\nworld"
>>> print mystr
hello
world
>>> mystr.splitlines()
['hello', 'world']
>>>
```

<18>isalpha

如果 mystr 所有字符都是字母 则返回 True,否则返回 False

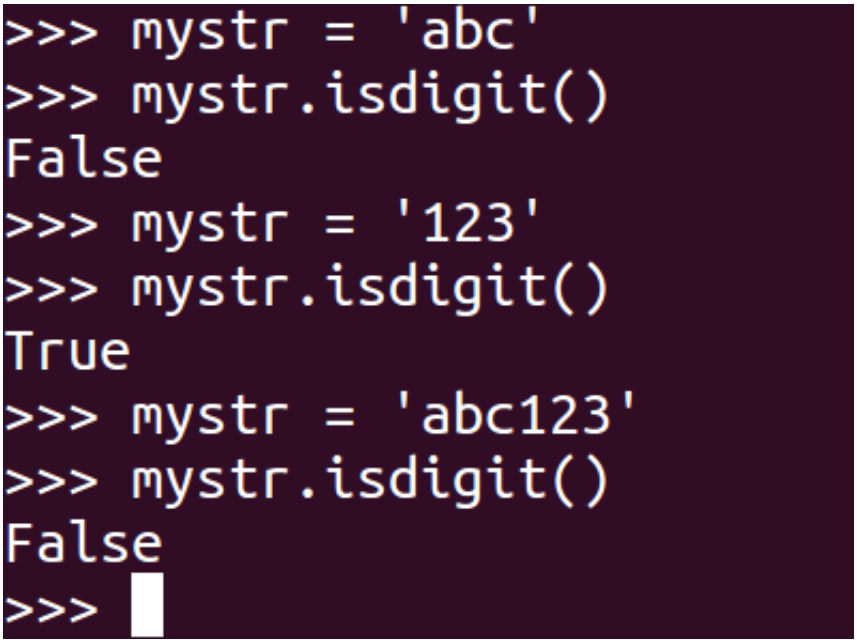
```
mystr.isalpha()
```



<19>isdigit

如果 mystr 只包含数字则返回 True 否则返回 False.

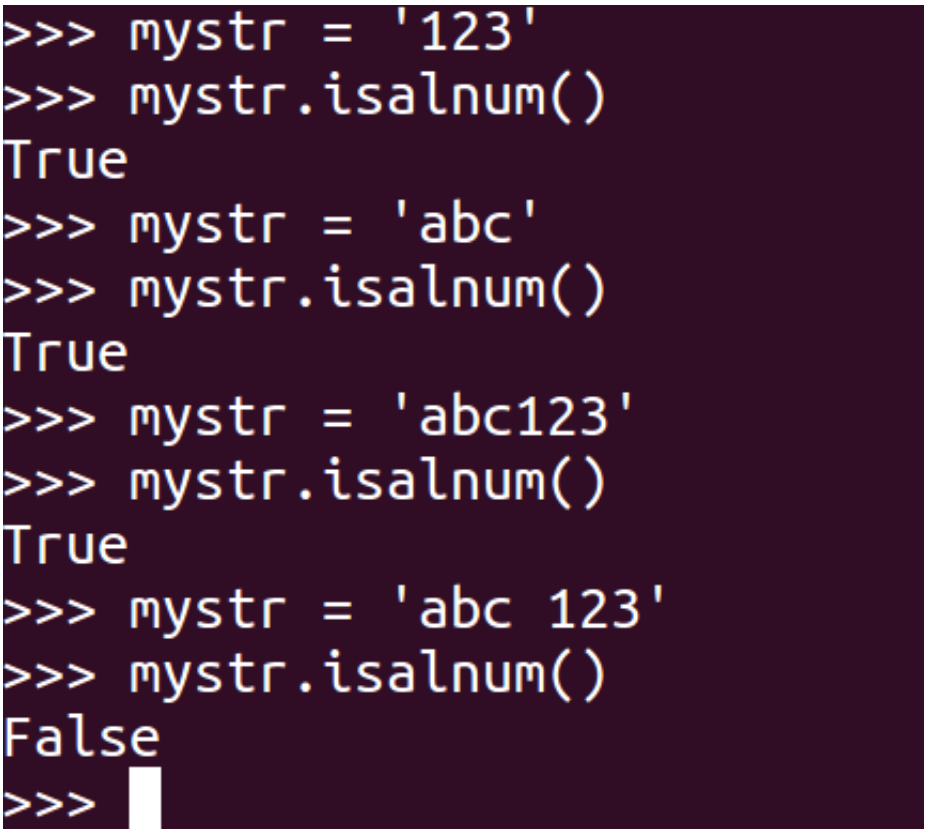
```
mystr.isdigit()
```



<20>isalnum

如果 mystr 所有字符都是字母或数字则返回 True,否则返回 False

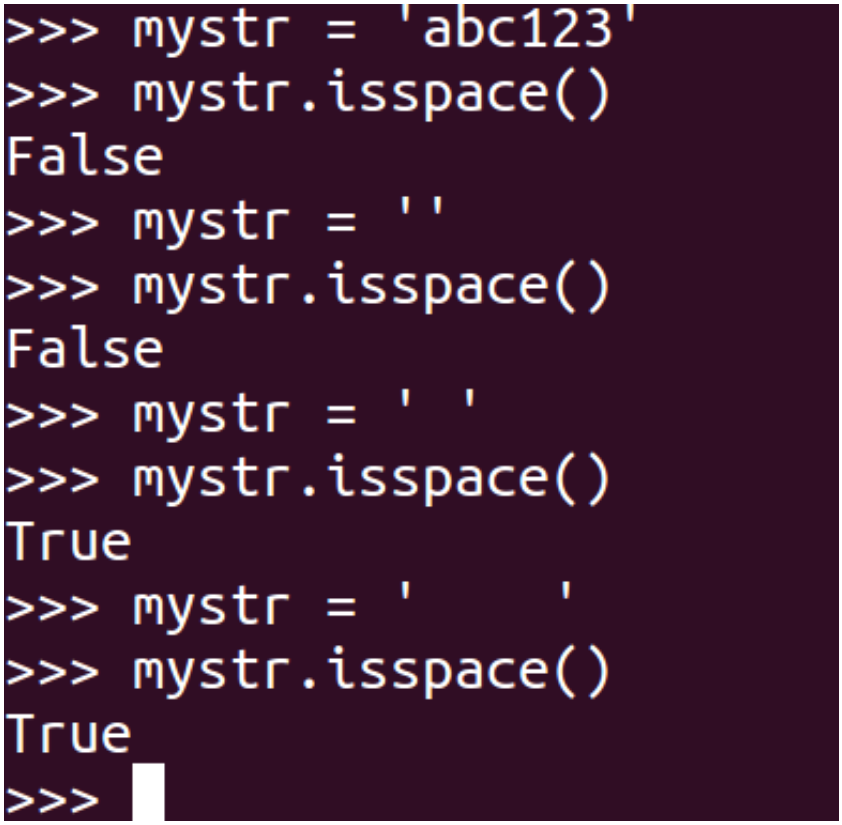
```
mystr.isalnum()
```



<21>isspace

如果 mystr 中只包含空格，则返回 True，否则返回 False.

```
mystr.isspace()
```


>>> mystr = 'abc123'
>>> mystr.isspace()
False
>>> mystr = ''
>>> mystr.isspace()
False
>>> mystr = ' '
>>> mystr.isspace()
True
>>> mystr = '   '
>>> mystr.isspace()
True
>>>