Python Bootcamp & Masterclass

string methods 1

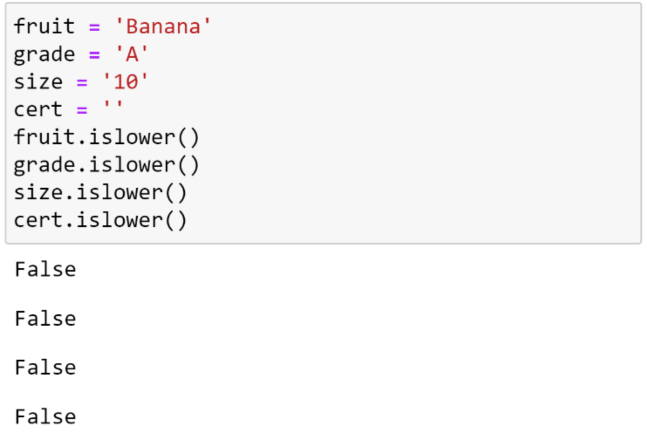
gknxt

.islower( ) and .isupper( )

Since strings are objects their methods need to use `object.method` notation.

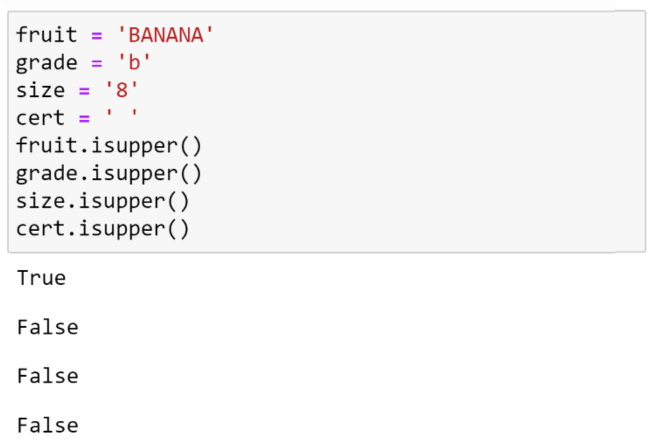
The `.islower( )` method returns `True` if all cased characters in the string are lowercase and there is at least one cased character, `False` otherwise

The `.isupper( )` method returns `True` if all cased characters in the string are uppercase and there is at least one cased character, `False` otherwise



```python
fruit = 'Banana'
grade = 'A'
size = '10'
cert = ''
fruit.islower()
grade.islower()
size.islower()
cert.islower()
```

```
False

False

False

False
```



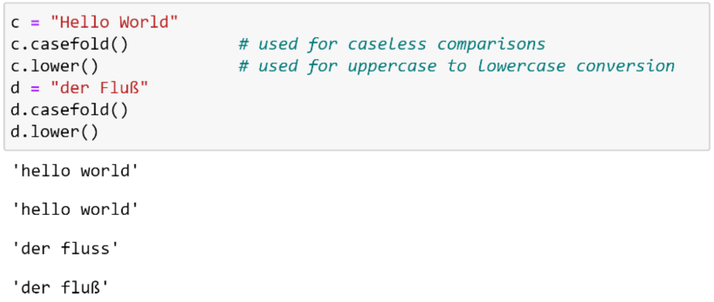
```python
fruit = 'BANANA'
grade = 'b'
size = '8'
cert = ' '
fruit.isupper()
grade.isupper()
size.isupper()
cert.isupper()
```

```
True

False

False

False
```

.lower( ) and .casefold( )

The `.lower( )` method returns a copy of the string with all the cased characters converted to lowercase

The `.casefold( )` method returns a casefolded copy of the string. Casefolded strings may be used for caseless matching. Casefolding is similar to lowercasing but more aggressive because it is intended to remove all case distinctions in a string. For example, the German lowercase letter 'ß' is equivalent to "ss". Since it is already lowercase, lower() would do nothing to 'ß'; casefold() converts it to "ss".



```
c = "Hello World"
c.casefold()          # used for caseless comparisons
c.lower()             # used for uppercase to lowercase conversion
d = "der Fluß"
d.casefold()
d.lower()
```

```
'hello world'

'hello world'

'der fluss'

'der fluß'
```

gknxt

.upper( )

The **.upper( )** method returns a copy of the string with all the cased characters converted to uppercase

```
city = 'Dallas, Texas, USA'
city.upper()
```

```
'DALLAS, TEXAS, USA'
```

```
city = 'Dallas, Texas, USA'
city.upper().isupper()
```

```
True
```

```
cost = '$246.50'
cost.upper()
cost.upper().isupper()
```

```
'$246.50'

False
```

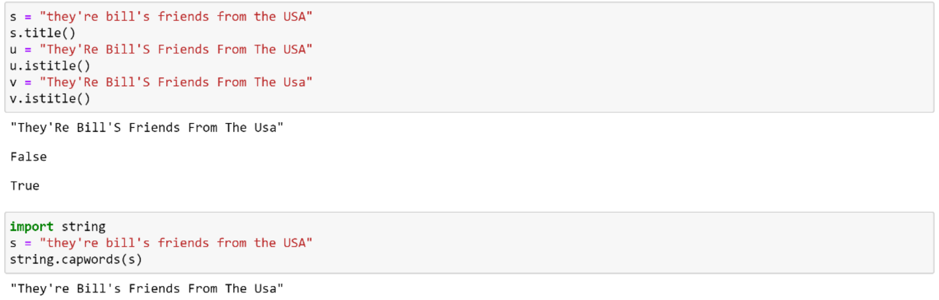
gk nxt

.title( ) and istitle( )

The `.title( )` method returns a title-cased version of the string where words start with an uppercase character and the remaining characters are lowercase. The algorithm uses a simple definition of a word as groups of consecutive letters. Since apostrophes in contractions and possessives form word boundaries, the result may not always be desired.

Using `string.capwords( )` is typically more desirable compared to `.title( )`

The `.istitle( )` method returns **True** if the string is a title-cased string and there is at least one character. Returns **False** otherwise.



```
s = "they're bill's friends from the USA"
s.title()
u = "They'Re Bill'S Friends From The USA"
u.istitle()
v = "They'Re Bill'S Friends From The Usa"
v.istitle()
```

```
"They'Re Bill'S Friends From The Usa"

False

True
```

```
import string
s = "they're bill's friends from the USA"
string.capwords(s)
```

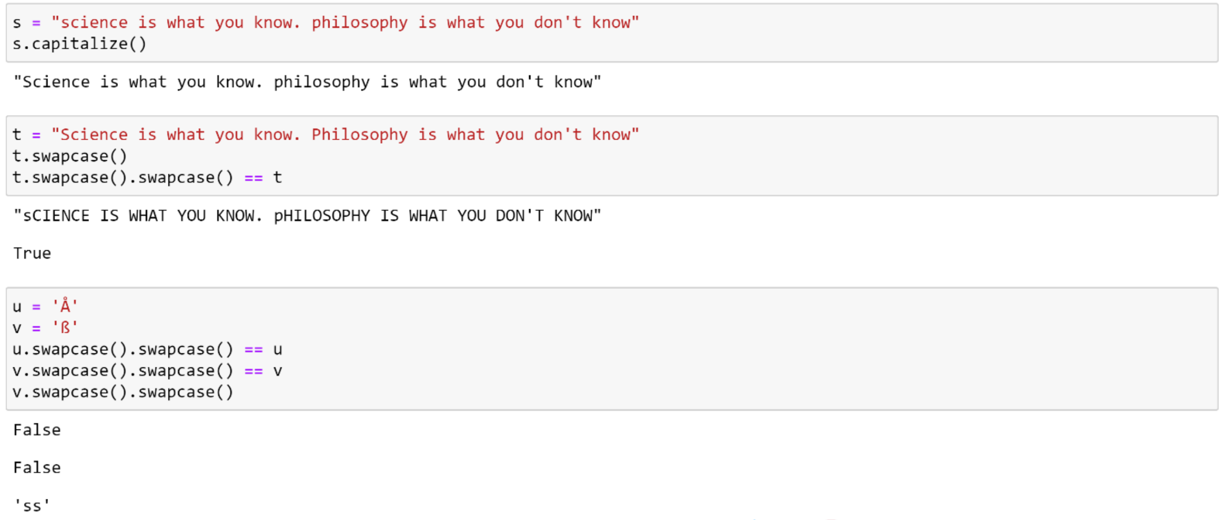
```
"They're Bill's Friends From The Usa"
```

gknxt

.capitalize( ) and swapcase( )

The `.capitalize( )` method returns a copy of the string with its first character capitalized and the rest lowercased.

The `.swapcase( )` method returns a copy of the string with uppercase characters converted to lowercase and vice versa.



```
s = "science is what you know. philosophy is what you don't know"
s.capitalize()
```

```
"Science is what you know. philosophy is what you don't know"
```

```
t = "Science is what you know. Philosophy is what you don't know"
t.swapcase()
t.swapcase().swapcase() == t
```

```
"sCIENCE IS WHAT YOU KNOW. pHILOSOPHY IS WHAT YOU DON'T KNOW"

True
```

```
u = 'Å'
v = 'ß'
u.swapcase().swapcase() == u
v.swapcase().swapcase() == v
v.swapcase().swapcase()
```

```
False

False

'ss'
```

gknxt

.lstrip( )

The `.lstrip( )` method returns a copy of the string with leading characters removed.

A string specifying the set of characters to be removed can be given as the argument.

If called with no argument or `None` as the argument, it defaults to removing leading whitespace(s).

```
s = '   Taj Mahal    '
s.lstrip( )     # removes leading whitespace(s)
s.lstrip(None)  # removes leading whitespace(s)
s.lstrip(' ')   # removes leading whitespace(s)
```

```
'Taj Mahal    '

'Taj Mahal    '

'Taj Mahal    '
```

```
ws = 'www.gknxt.com'
ws.lstrip('w.')      # removes all leading w and . characters
```

```
'gknxt.com'
```

```
t = '\t no parking \n'
t
t.lstrip()
```

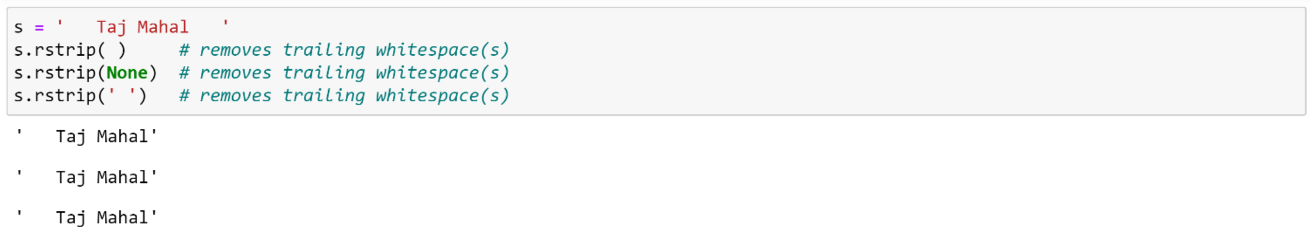
```
'\t no parking \n'

'no parking \n'
```

.rstrip( )

The `.rstrip( )` method returns a copy of the string with trailing characters removed.

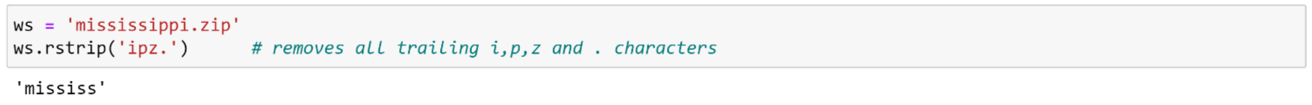
A string specifying the set of characters to be removed can be given as the argument.

If called with no argument or `None` as the argument, it defaults to removing trailing whitespace(s).



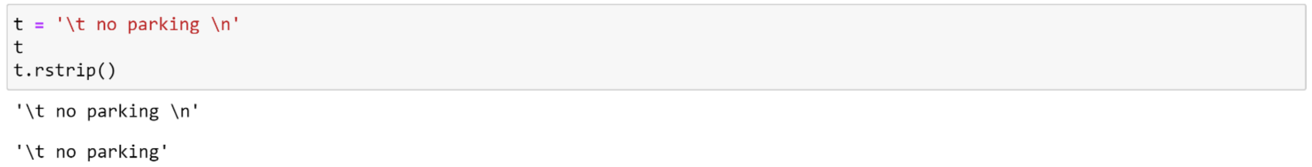
```
s = '   Taj Mahal   '
s.rstrip( )     # removes trailing whitespace(s)
s.rstrip(None)  # removes trailing whitespace(s)
s.rstrip(' ')   # removes trailing whitespace(s)
```

```
'   Taj Mahal'

'   Taj Mahal'

'   Taj Mahal'
```



```
ws = 'mississippi.zip'
ws.rstrip('ipz.')      # removes all trailing i,p,z and . characters
```

```
'mississ'
```



```
t = '\t no parking \n'
t
t.rstrip()
```

```
'\t no parking \n'

'\t no parking'
```

gknxt

.strip( )

The `.strip( )` method returns a copy of the string with leading and trailing characters removed. A string specifying the set of characters to be removed can be given as the argument. If called with no argument or `None` as the argument, it defaults to removing leading and trailing whitespace(s).

```python
s = '   Taj Mahal   '
s.strip( )     # removes leading & trailing whitespace(s)
s.strip(None)  # removes leading & trailing whitespace(s)
s.strip(' ')   # removes leading & trailing whitespace(s)
```

```
'Taj Mahal'

'Taj Mahal'

'Taj Mahal'
```

```python
ws = '#....... Section 3.2.1 Issue #32 .......'
ws.strip('#. ')      # removes all leading & trailing #,  (space), and . characters
```

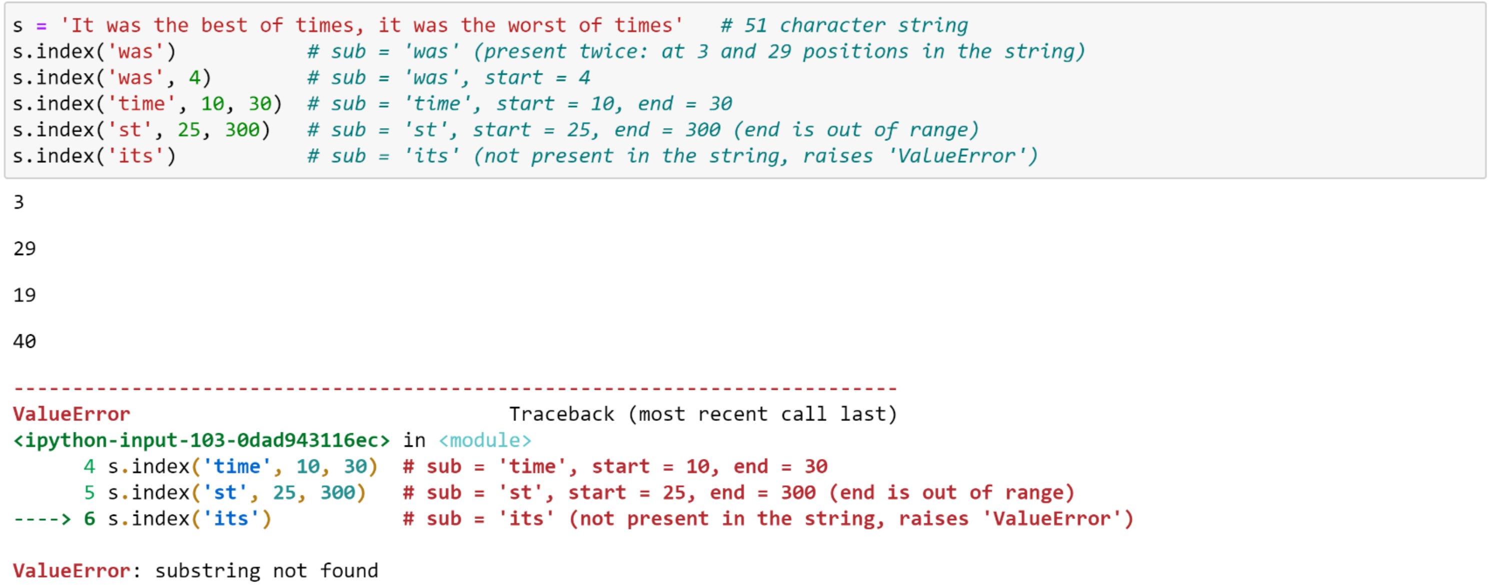
```
'Section 3.2.1 Issue #32'
```

```python
t = '\t no parking \n'
t
t.strip()
```

```
'\t no parking \n'

'no parking'
```

.index( )

The **.index(sub,start,end)** method returns the lowest index in the string where the substring (sub) starting at **start** (optional) upto **end** (optional) is found. Raises **ValueError** if the substring is not found. The **.index( )** method works with all sequences.



```python
s = 'It was the best of times, it was the worst of times'   # 51 character string
s.index('was')            # sub = 'was' (present twice: at 3 and 29 positions in the string)
s.index('was', 4)         # sub = 'was', start = 4
s.index('time', 10, 30)   # sub = 'time', start = 10, end = 30
s.index('st', 25, 300)    # sub = 'st', start = 25, end = 300 (end is out of range)
s.index('its')            # sub = 'its' (not present in the string, raises 'ValueError')
```

```
3

29

19

40

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-103-0dad943116ec> in <module>
      4 s.index('time', 10, 30)   # sub = 'time', start = 10, end = 30
      5 s.index('st', 25, 300)    # sub = 'st', start = 25, end = 300 (end is out of range)
----> 6 s.index('its')            # sub = 'its' (not present in the string, raises 'ValueError')

ValueError: substring not found
```

gknxt

.rindex( )

The `.rindex(sub,start,end)` method returns the highest index in the string where the substring (sub) starting at `start` (optional) upto `end` (optional) is found. Raises `ValueError` if the substring is not found. The `.rindex( )` method works with all sequences.

```
s = 'It was the best of times, it was the worst of times'   # 51 character string
s.rindex('was')           # sub = 'was' (present twice: at 3 and 29 positions in the string)
s.rindex('was', 2, 20)    # sub = 'was', start = 2, end = 20
s.rindex('time', 10, 30)  # sub = 'time', start = 10, end = 30
s.rindex('st', 25, 300)   # sub = 'st', start = 25, end = 300 (end is out of range)
s.rindex('its')           # sub = 'its' (not present in the string, raises 'ValueError')
```

```
29

3

19

40

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-109-76f6e8213a84> in <module>
      4 s.rindex('time', 10, 30)  # sub = 'time', start = 10, end = 30
      5 s.rindex('st', 25, 300)   # sub = 'st', start = 25, end = 300 (end is out of range)
----> 6 s.rindex('its')           # sub = 'its' (not present in the string, raises 'ValueError')

ValueError: substring not found
```

gknxt

Online Resources

For best python resources, please visit:

gknxt.com/python/

gk nxt

Python Bootcamp & Masterclass

Thank You

for your Rating & Review

gk nxt