Finder

Python Strings

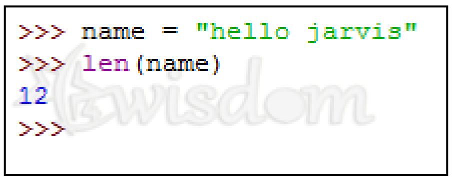
Like 2 Share

in this section we will examine the internal structure of a Python Strings.

String : A Python string is a sequence of zero or more characters.

The string is an immutable data structure. Although you can access the internal data elements but can not modified. Sequences fall into one of two categories:mutable or immutable. Mutable means changeable. Mutable sequence is one that can change. Immutable means unchangeable, means that can not be change. Python strings are immutable sequences, which means that they can't change. So,for example, the string "Satyamev Jayte " will always be the string "Satyamev Jayte " In order to find the lenght of Python string, function **len(string)** is used, see the example below.

Example 1:



```
>>> name = "hello jarvis"
>>> len(name)
12
>>>
```

len()

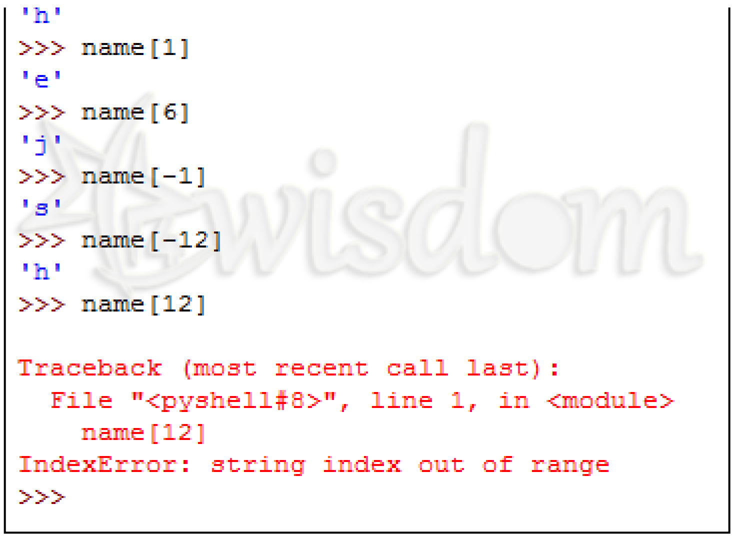
The Subscript Operator

Sometimes user want to examine one character of string at a given position without analysing them all. The subscript operator makes this possible. The syntax and example of subscript operator is given below:

<given string>[<index>]

The given Python string is that you want to examine, The index is an integer means the position of particular character as shown in example below. In the following examples, the subscript operator is used to access characters in the string "Hello Jarvis".

Example 2:

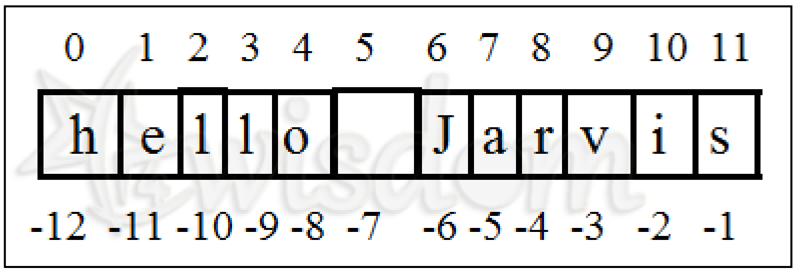


```
'h'
>>> name[1]
'e'
>>> name[6]
'j'
>>> name[-1]
's'
>>> name[-12]
'h'
>>> name[12]

Traceback (most recent call last):
  File "<pyshell#8>", line 1, in <module>
    name[12]
IndexError: string index out of range
>>>
```

subscript operator

I hope you can understand the subscript operator. The "hello jarvis" is 12 character long means [0:11]. The name[0] respresent the character 'h'. name[12] genrate error "out of range", name[-1] respresents the last character of the Python string and name[-12] represents the first character of the string. See the figure below:

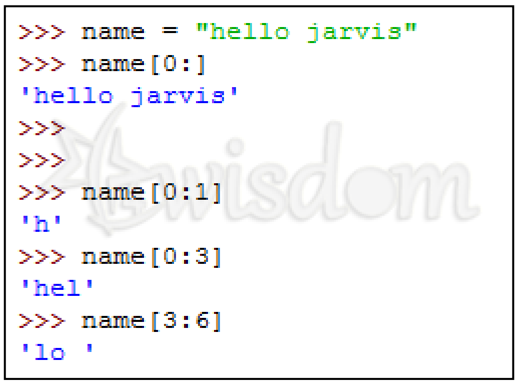


Indexing

Slicing for Substrings

The extracted portions of Python strings called substrings. In many situations, you need some portion of string such as first 2 characters of a strings. Python's subscript operator uses a process called slicing. In slicing colon **:** is used. An integer value will appear on either side of colon. See the example below:

Example 3:



```
>>> name = "hello jarvis"
>>> name[0:]
'hello jarvis'
>>>
>>>
>>> name[0:1]
'h'
>>> name[0:3]
'hel'
>>> name[3:6]
'lo '
```

Python String Operations

There are several Python string operations, will be discussed one by one. In Python string functions string **str** will be used.

center()

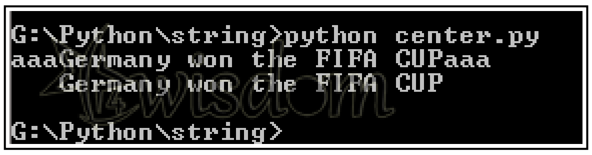
Syntax

str.center(width[, fillchar])

The method **center()** makes str centred by taking *width* parameter into account. Padding is specified by parameter *fillchar*. Default filler is a space.

```
str = "Germany won the FIFA CUP"
str1= str.center(30,'a')
str2= str.center(30)
print str1
print str2
```

Out-put



```
G:\Python\string>python center.py
aaaGermany won the FIFA CUPaaa
   Germany won the FIFA CUP

G:\Python\string>
```

Python String center()

count()

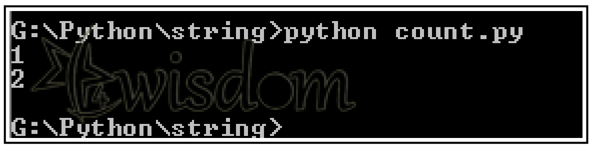
Syntax

str.count(substr [, start [, end]])

The method **str.count(substr [, start [, end]])** returns the number of occurrence of Python string *substr* in string *str*. By using parameter *start* and *end* you can give slice of *str* .

```
str = "Ths is a count example"
sub = "i"
print str.count(sub, 4, len(str))
sub = "a"
print str.count(sub)
```

Out-put



```
G:\Python\string>python count.py
1
2

G:\Python\string>
```

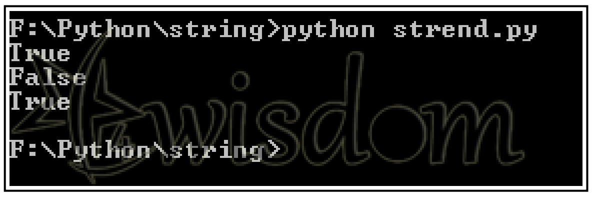
Python string count()

I hope you can understand the above code, in *str.count(sub, 4, len(str))* parameter 4 and *len(str)* represent the slice of Python string *str*

This method returns True if the string ends with the specified substring, otherwise return False **Syntax**
str.endswith(suffix[, start[, end]])
The use of *start* and *end* to generate slice of Python string *str*

```
str ="it is not easy to play another man's game"
sub = "is"
print str.endswith(sub,2,5)
print str.endswith(sub,30)
sub = "game"
print str.endswith(sub)
```

Out-put



```
F:\Python\string>python strend.py
True
False
True

F:\Python\string>
```

endswith()

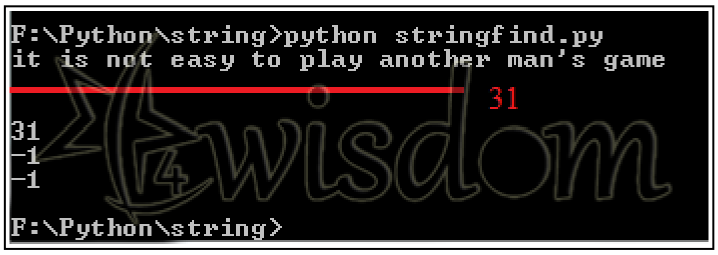
First result is true, because substring *"is"* is examined in slice 2 to 5 position of Python string *str*. Second result is false and third result is true.

find()

Syntax
str.find(str, beg=0 end=len(string))
The **find()** method used to find out whether a string occur in given string or its substrings.

```
str ="it is not easy to play another man's game"
print str
print "\n"
str1="man"
str2= "woman"
print str.find(str1)
print str.find(str1,38)
print str.find(str2)
```

Out-put



```
F:\Python\string>python stringfind.py
it is not easy to play another man's game
                                          31

31
-1
-1

F:\Python\string>
```

find()

returned.

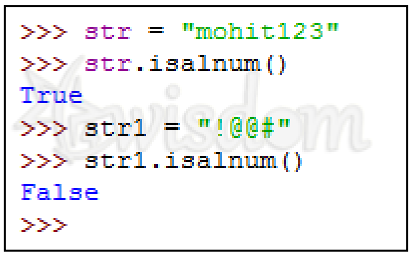
isalnum()

Syntax

str.isalnum()

The method **isalnum()** is used to determine whether the Python string consists of alphanumeric characters,false otherwise.

Out-put



```
>>> str = "mohit123"
>>> str.isalnum()
True
>>> str1 = "!@@#"
>>> str1.isalnum()
False
>>>
```

isalnum()

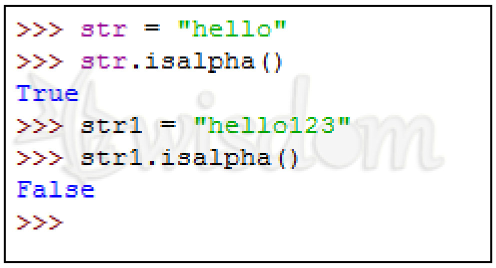
isalpha()

Syntax

str.isalpha()

The method **isalpha()** return true if the Python string contains only alphabetic character(s),false otherwise.

Out-put



```
>>> str = "hello"
>>> str.isalpha()
True
>>> str1 = "hello123"
>>> str1.isalpha()
False
>>>
```

isalpha()

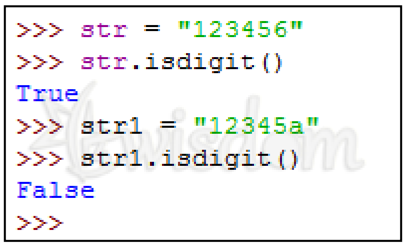
isdigit()

Syntax

str.isdigit()

The method **isdigit()** return true if the Python string contains only digit(s),false otherwise. Example shows rest story

Out-put



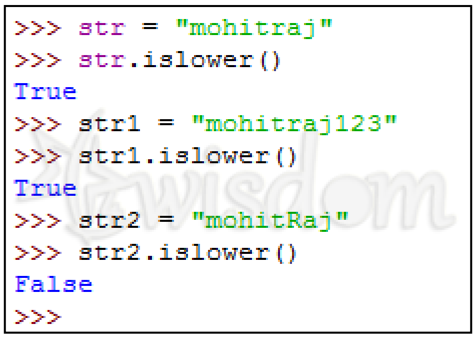
```
>>> str = "123456"
>>> str.isdigit()
True
>>> str1 = "12345a"
>>> str1.isdigit()
False
>>>
```

isdigit()

str.islower()

The method **islower()** return true if the Python string contains only lower cased character(s), false otherwise. Example shows rest story

Out-put



```
>>> str = "mohitraj"
>>> str.islower()
True
>>> str1 = "mohitraj123"
>>> str1.islower()
True
>>> str2 = "mohitRaj"
>>> str2.islower()
False
>>>
```

islower()

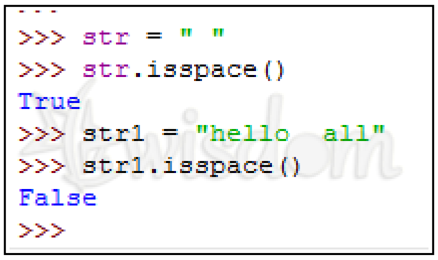
isspace()

Syntax

str.isspace()

The method **isspace()** return true if the Python string contains only white space(s). Example shows rest story

Out-put



```
>>> str = " "
>>> str.isspace()
True
>>> str1 = "hello  all"
>>> str1.isspace()
False
>>>
```

isspace

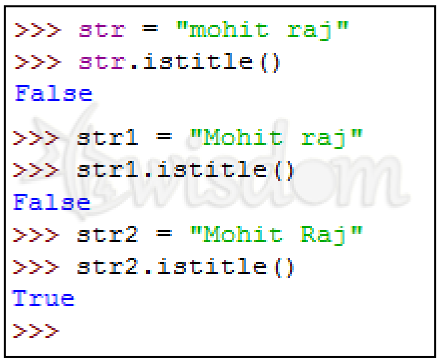
istitle()

Syntax

str.istitle()

The method **istitle()** return true if the string is a titlecased. Example shows rest of the story.

Out-put



```
>>> str = "mohit raj"
>>> str.istitle()
False
>>> str1 = "Mohit raj"
>>> str1.istitle()
False
>>> str2 = "Mohit Raj"
>>> str2.istitle()
True
>>>
```

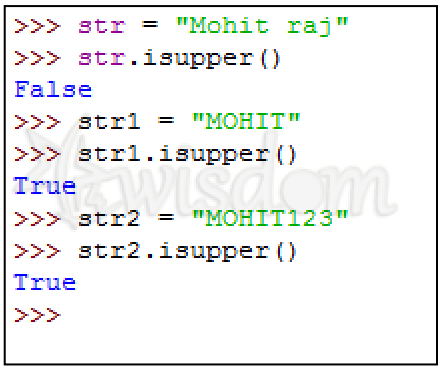
istitle()

Syntax

str.isupper()

The method **isupper()** return true if the string contains only upper cased character(s), false otherwise. . Example shows rest story

Out-put



```
>>> str = "Mohit raj"
>>> str.isupper()
False
>>> str1 = "MOHIT"
>>> str1.isupper()
True
>>> str2 = "MOHIT123"
>>> str2.isupper()
True
>>>
```

isupper()

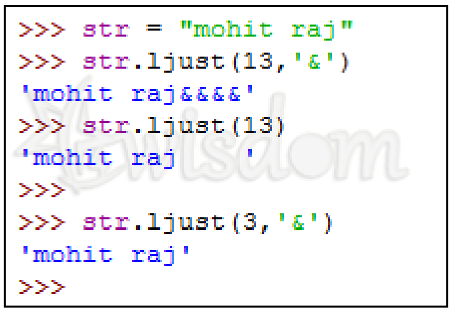
ljust()

Syntax

str.ljust(width[, fillchar])

When you provide the string to the method **ljust()**, it returns the string left justified. Total length of string is defined in first parameter of method *width* . Padding is done as defined in second parameter *fillchar* .(default is space)

Out-put



```
>>> str = "mohit raj"
>>> str.ljust(13,'&')
'mohit raj&&&&'
>>> str.ljust(13)
'mohit raj    '
>>>
>>> str.ljust(3,'&')
'mohit raj'
>>>
```

ljust()

In above example you can see that if you don't define the *fillchar* then the method **ljust()** automatically take space as *fillchar.*

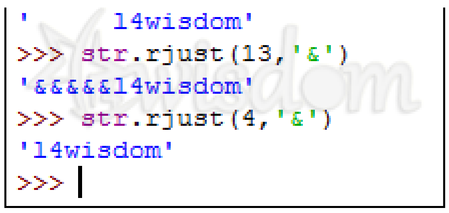
rjust()

Syntax

str.rjust(width[, fillchar])

When you provide the string to the method **rjust()**, it returns the string right justified. Total length of string is defined in first parameter of method *width* . Padding is done as defined in second parameter *fillchar* .(default is space)

Out-put

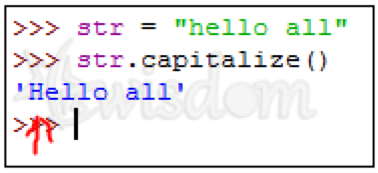


```
'     l4wisdom'
>>> str.rjust(13,'&')
'&&&&&l4wisdom'
>>> str.rjust(4,'&')
'l4wisdom'
>>> |
```

rjust

capitalize()

This function capitalizes first letter of string.



```
>>> str = "hello all"
>>> str.capitalize()
'Hello all'
>>> |
```

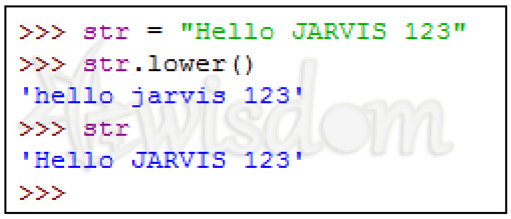
capitalize()

lower()

Syntax

str.lower()

The method **lower()** returns a copy of the string in which all case-based characters have been converted to lower case.



```
>>> str = "Hello JARVIS 123"
>>> str.lower()
'hello jarvis 123'
>>> str
'Hello JARVIS 123'
>>>
```

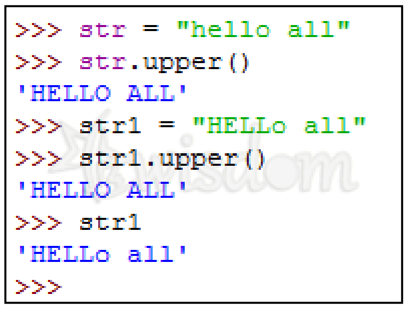
lower()

upper()

Syntax

str.upper()

The method **upper()** returns a copy of the string in which all case-based characters have been converted to upper case.



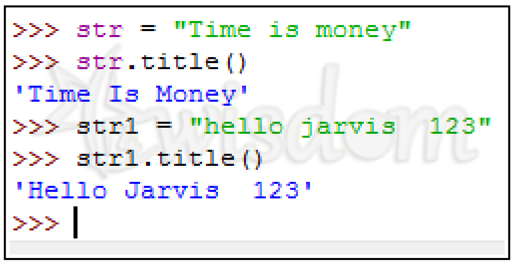
```
>>> str = "hello all"
>>> str.upper()
'HELLO ALL'
>>> str1 = "HELLo all"
>>> str1.upper()
'HELLO ALL'
>>> str1
'HELLo all'
>>>
```

upper()

title()

Syntax

str.title()



```
>>> str = "Time is money"
>>> str.title()
'Time Is Money'
>>> str1 = "hello jarvis  123"
>>> str1.title()
'Hello Jarvis  123'
>>> 
```

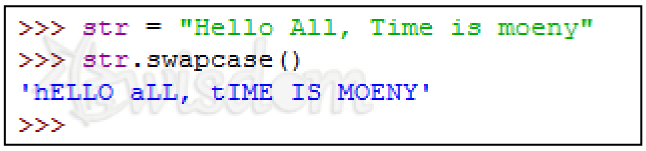
title()

swapcase()

Syntax

str.swapcase()

The method **swapcase()** returns a copy of the string in which all cased based character swap their case .



```
>>> str = "Hello All, Time is moeny"
>>> str.swapcase()
'hELLO aLL, tIME IS MOENY'
>>>
```

swapcase()

join()

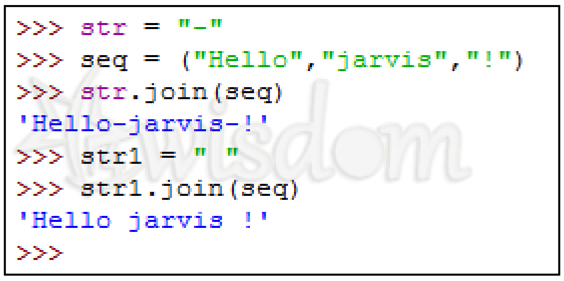
Syntax

str.join(seq)

seq = it contains the seqeunce of separated strings.

str = it is the string which is used to replace the separator of seqcence

This method **join()** returns a string which is the concatenation of given sequence and string as shown in example.



```
>>> str = "-"
>>> seq = ("Hello","jarvis","!")
>>> str.join(seq)
'Hello-jarvis-!'
>>> str1 = " "
>>> str1.join(seq)
'Hello jarvis !'
>>>
```

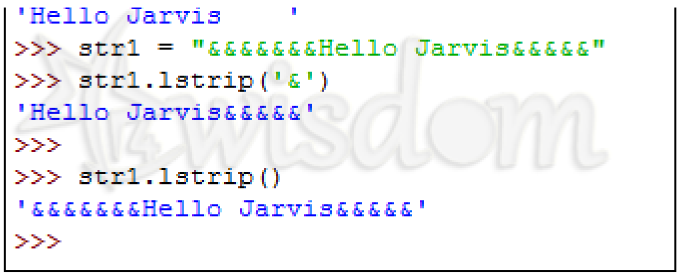
lower()

lstrip()

Syntax

str.lstrip([chars])

The method **lstrip()** returns a copy of the string in which specified char(s) have been stripped from left side of string. If char is not specified then space is taken as default.



```
'Hello Jarvis    '
>>> str1 = "&&&&&&&Hello Jarvis&&&&&"
>>> str1.lstrip('&')
'Hello Jarvis&&&&&'
>>>
>>> str1.lstrip()
'&&&&&&&Hello Jarvis&&&&&'
>>>
```

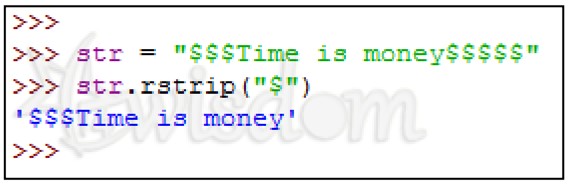
lstrip()

rstrip()

Syntax

str.rstrip([chars])

The method **lstrip()** returns a copy of the string in which specified char(s) have been stripped from right side of string. If char is not specified then space is taken as default.



```
>>>
>>> str = "$$$Time is money$$$$$"
>>> str.rstrip("$")
'$$$Time is money'
>>>
```

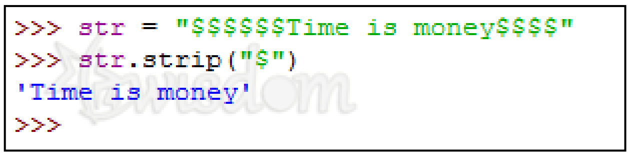
rstrip

strip()

Syntax

str.strip([chars])

The method **lstrip()** returns a copy of the string in which specified char(s) have been stripped from both side of string. If char is not specified then space is taken as default.



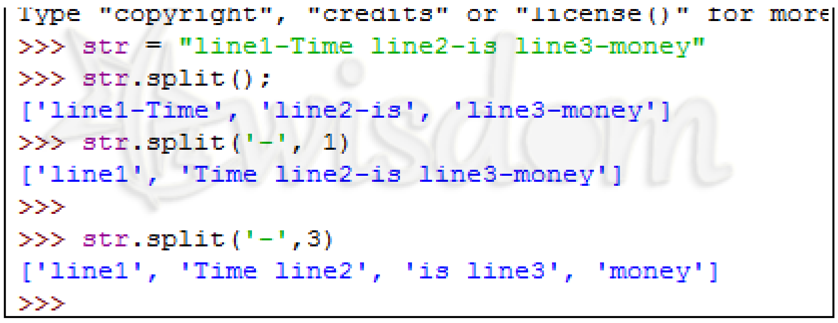
```
>>> str = "$$$$$$Time is money$$$$"
>>> str.strip("$")
'Time is money'
>>>
```

strip

split()

The method **splits** returns a list of all words in the string, delimiter separates the words. If delimiter is not specified then whitespace is taken as delimiter, paramter *num* **Syntax**

str.split("delimiter", num)



```
Type "copyright", "credits" or "license()" for more
>>> str = "line1-Time line2-is line3-money"
>>> str.split();
['line1-Time', 'line2-is', 'line3-money']
>>> str.split('-', 1)
['line1', 'Time line2-is line3-money']
>>>
>>> str.split('-',3)
['line1', 'Time line2', 'is line3', 'money']
>>>
```

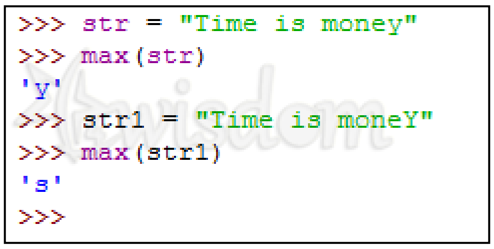
split()

max()

Syntax

max(str)

The method **max()** returns the max character from string *str* according to ASCII value. in first print statement y is max character, because ASCII code of "y" is 121. In second print statement "s" is max character, ASCII code of "s" is 115.



```
>>> str = "Time is money"
>>> max(str)
'y'
>>> str1 = "Time is moneY"
>>> max(str1)
's'
>>>
```

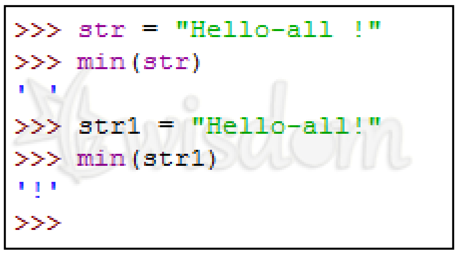
max()

min()

Syntax

min(str)

The method **min()** returns the min character from string *str* according to ASCII value.



```
>>> str = "Hello-all !"
>>> min(str)
' '
>>> str1 = "Hello-all!"
>>> min(str1)
'!'
>>>
```

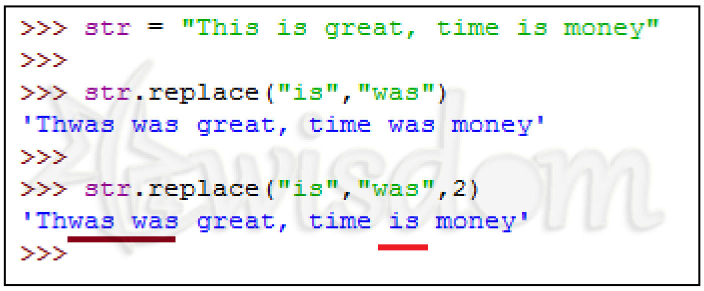
min()

replace()

Syntax

str.replace(old, new[, max])

The method **replace()** returns the string in which occurrences of string specified by parameter *old* have been replaced with string specified by parameter *new* . The parameter max defined how



```
>>> str = "This is great, time is money"
>>>
>>> str.replace("is","was")
'Thwas was great, time was money'
>>>
>>> str.replace("is","was",2)
'Thwas was great, time is money'
>>>
```

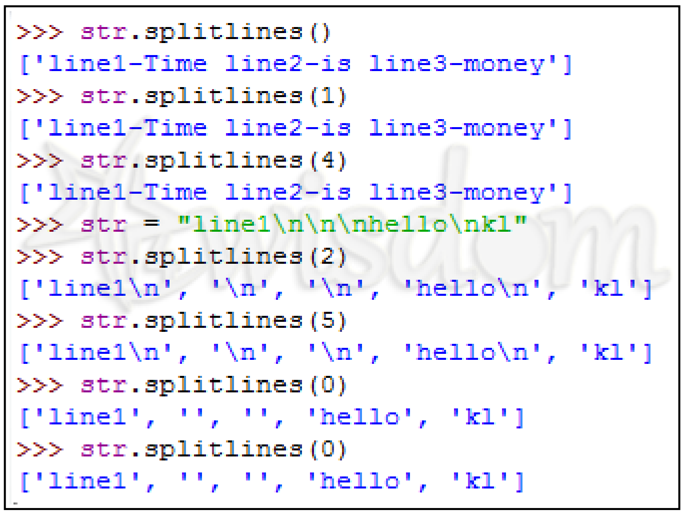
replace()

splitlines()

Syntax

str.splitlines(num) The method **splitlines()** returns a list with all the lines in string, In *num* is specified then it would include line break.

num -- This is any number, if present then it would be assumed that line breaks need to be included in the lines.



```
>>> str.splitlines()
['line1-Time line2-is line3-money']
>>> str.splitlines(1)
['line1-Time line2-is line3-money']
>>> str.splitlines(4)
['line1-Time line2-is line3-money']
>>> str = "line1\n\n\nhello\nkl"
>>> str.splitlines(2)
['line1\n', '\n', '\n', 'hello\n', 'kl']
>>> str.splitlines(5)
['line1\n', '\n', '\n', 'hello\n', 'kl']
>>> str.splitlines(0)
['line1', '', '', 'hello', 'kl']
>>> str.splitlines(0)
['line1', '', '', 'hello', 'kl']
```

splitlines()

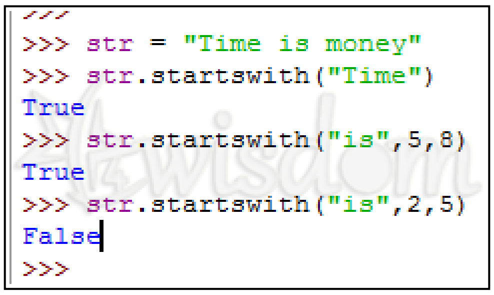
When you supplied *num* greater than 1 it include line break.

startswith()

Syntax

str.startswith(str1, beg=0,end=len(string));

The method **startswith()** return true if a string *str* starts with the string specified by parameter str1. parameter *beg* and *end* are used to slice the string *str* .



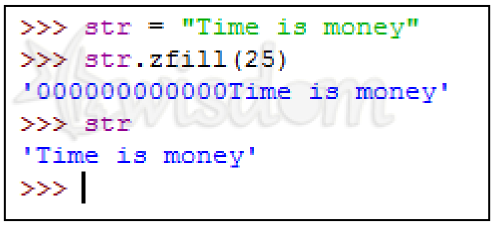
```
>>> str = "Time is money"
>>> str.startswith("Time")
True
>>> str.startswith("is",5,8)
True
>>> str.startswith("is",2,5)
False
>>>
```

startswith()

Syntax

str.zfill(width)

This method pads the string on the left with zeros to fill width.



```
>>> str = "Time is money"
>>> str.zfill(25)
'000000000000Time is money'
>>> str
'Time is money'
>>> |
```

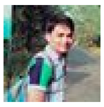
zfill()

You have leaned about strings operation in Python, I hope you enjoyed the chapter.

Like Share 2 people like this. Be the first of your friends.

« Previous Next »

0 Comments Sort by **Oldest**



Add a comment...

Facebook Comments Plugin

« Previous Next »

Home page About Us Contact Top