String Methods

10 June 2024 10:32 AM

Difference between function and methods

Function	Methods
Functions are independent block of code which is used for performing one task	Methods are the functions which are defined inside the class

Def len(value) :
_____
____
	Class str( ) :
	Def lower (self):

Function can be called or executed directly	In order to execute methods we need either class or object reference
Syntax: Function name(value/variable)	Syntax: Object reference.Method name( ) Or Classreference. Methode Name( )
Len, print, type, id	s.lower( )
Function can be used on all data types	Methods are only used with specific data types

Built-in Methods of string:
All the methods can Perform some operation on string but they will not modify actual value of string
Case conversion method of strings

Methods_name	**Syntax of method**	**Functionality**
lower( )	S.lower( )	It is used for converting each and every character of given string into lower case and returns converted string
upper( )	S.upper( )	It is used for converting each and every character of given string into upper case and returns converted string
title( )	S.title( )	First character of each and every word of a given string will be converted into upper case and remaining characters into lower case and returns converted string
capitalize( )	S.capitalize( )	First character of entire string will be converted into upper case and remaining characters into lower case and returns converted string
swapcase( )	S.swapecase( )	It is used for converting all lower case to upper case and all upper cases to lower case and returns the converted string

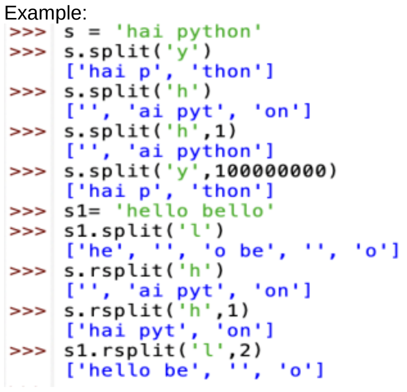
Case verification methods

islower( )	S.islower( )	Returns true if string satisfies lower method else returns false
isupper( )	S.isupper( )	Returns true if string satisfies upper method else returns false
istitle( )	S.istitle( )	Returns true if string satisfies title method else returns false
isalpha( )	S.isalpha( )	Returns true if string contains only alphabets else returns false
isdigit( )	S.isdigit( )	Returns true if string contains only digits else returns false
isalnum( )	S.isalum( )	Returns true if string contains only alphabets or digit or combination of both else returns false
isspace( )	S.isspace( )	Returns true if string contains only space else returns false

Arguments are the essential values required for the function/Methods for its execution
Mandatory argument ———> direct names
Default argument ————> [names]

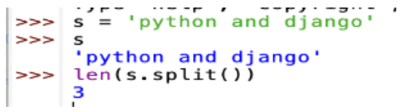
split( )	**rsplit( )**
It is used for splitting the given string based on given decimator	It is used for splitting the given string based on given decimator
Syntax: split([delimator],[count]) Default values For delimiter = space For count = how many times decimator is repeated	Syntax: rsplit([delimator],[count]) Default values For delimiter = space For count = how many times decimator is repeated
Output format of split is a list	Output format of rsplit is a list
Direction of considering the delimiter happens in left to right	Direction of considering the delimiter happens in right to left

Example:



```
>>> s = 'hai python'
>>> s.split('y')
    ['hai p', 'thon']
>>> s.split('h')
    ['', 'ai pyt', 'on']
>>> s.split('h',1)
    ['', 'ai python']
>>> s.split('y',100000000)
    ['hai p', 'thon']
>>> s1= 'hello bello'
>>> s1.split('l')
    ['he', '', 'o be', '', 'o']
>>> s.rsplit('h')
    ['', 'ai pyt', 'on']
>>> s.rsplit('h',1)
    ['hai pyt', 'on']
>>> s1.rsplit('l',2)
    ['hello be', '', 'o']
```

Write a program to print how many words are present in given string :



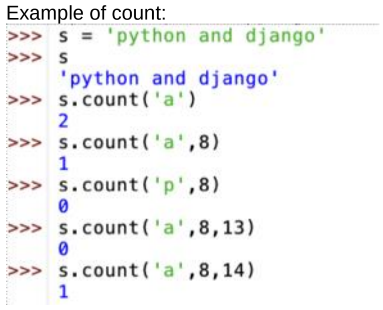
```
>>> s = 'python and django'
>>> s
'python and django'
>>> len(s.split())
3
```

Method name	Syntax of method	Functionality

count( )	count(substring,[start_index],[end_index])
	Default value:
	start_index —-> 0
	end_index ——> len(string)
		It is used for returning how many times given substring is repeated in given string

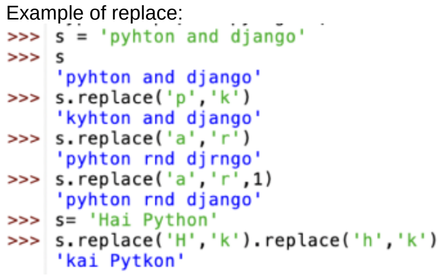
replace( )	replace(old string,newstring,[count]) Default value: count=no. of times ur old string is repeated	It is used for replacing the existing character with new character

Example of count:



```
>>> s = 'python and django'
>>> s
'python and django'
>>> s.count('a')
2
>>> s.count('a',8)
1
>>> s.count('p',8)
0
>>> s.count('a',8,13)
0
>>> s.count('a',8,14)
1
```

Example of replace:

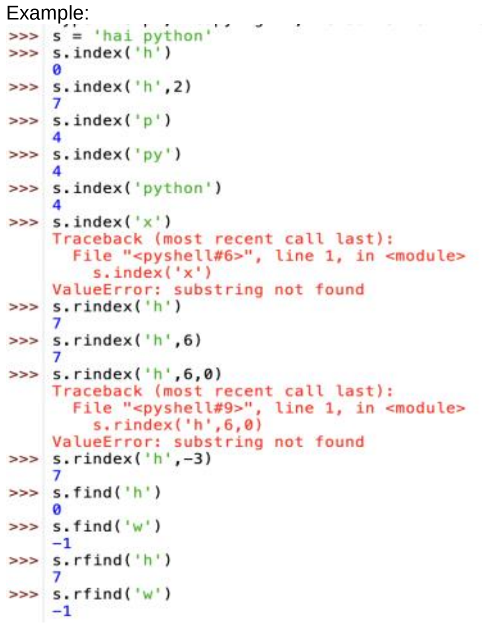


```
>>> s = 'pyhton and django'
>>> s
'pyhton and django'
>>> s.replace('p','k')
'kyhton and django'
>>> s.replace('a','r')
'pyhton rnd djrngo'
>>> s.replace('a','r',1)
'pyhton rnd django'
>>> s= 'Hai Python'
>>> s.replace('H','k').replace('h','k')
'kai Pytkon'
```

index( )	find( )
Syntax: index('value',[start_index],[end_index])	Syntax: find('value',[start_index],[end_index])
It is used for returning the index position If value is present else it returns error as output	It is used for returning the index position If value is present else it returns -1 as output
Direction Left to right	Direction Left to right

rindex( )	rfind( )
Syntax: rindex('value',[start_index],[end_index])	Syntax: rfind('value',[start_index],[end_index])
It is used for returning the index position If value is present else it returns error as output	It is used for returning the index position If value is present else it returns -1 as output
Direction Right to left	Direction Right to left

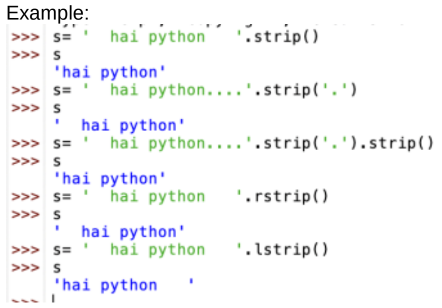
Example:



```
>>> s = 'hai python'
>>> s.index('h')
0
>>> s.index('h',2)
7
>>> s.index('p')
4
>>> s.index('py')
4
>>> s.index('python')
4
>>> s.index('x')
Traceback (most recent call last):
  File "<pyshell#6>", line 1, in <module>
    s.index('x')
ValueError: substring not found
>>> s.rindex('h')
7
>>> s.rindex('h',6)
7
>>> s.rindex('h',6,0)
Traceback (most recent call last):
  File "<pyshell#9>", line 1, in <module>
    s.rindex('h',6,0)
ValueError: substring not found
>>> s.rindex('h',-3)
7
>>> s.find('h')
0
>>> s.find('w')
-1
>>> s.rfind('h')
7
>>> s.rfind('w')
-1
```

Methods	Syntax	Functionality
s.strip( ), s.rstrip( ) ——> for right side only s.lstrip( ) ——> for left side	strip([unwanted_character]), rstrip([UWC]), lstrip([UWC])	These methods are used for removing the leading spaces from the given string and it can remove unwanted character as well

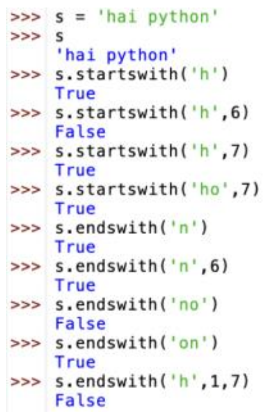
Example:



```
>>> s= '  hai python   '.strip()
>>> s
'hai python'
>>> s= '  hai python....'.strip('.')
>>> s
'  hai python'
>>> s= '  hai python....'.strip('.').strip()
>>> s
'hai python'
>>> s= '  hai python   '.rstrip()
>>> s
'  hai python'
>>> s= '  hai python   '.lstrip()
>>> s
'hai python   '
```

Startswith :
It is used for checking whether the string is starting with specified substring or not, if it is starting it will return true else it return false
Syntax:
startswith(substring,[start_index],[end_index])

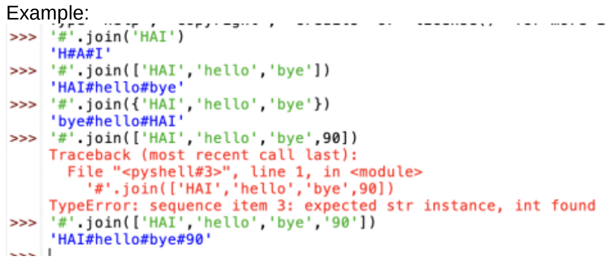
Endswith :
It is used for checking whether the string is ending with specified substring or not, if it is ending it will return true else it return false.
Syntax:
endswith(substring,[start_index],[end_index])



```
>>> s = 'hai python'
>>> s
'hai python'
>>> s.startswith('h')
True
>>> s.startswith('h',6)
False
>>> s.startswith('h',7)
True
>>> s.startswith('ho',7)
True
>>> s.endswith('n')
True
>>> s.endswith('n',6)
True
>>> s.endswith('no')
False
>>> s.endswith('on')
True
>>> s.endswith('h',1,7)
False
```

join( )	Glue character'.join(iterable/collection) Note: Collection should have only string as elements	It is used for creating new string by joining the elements of given collection with glue character

Example:

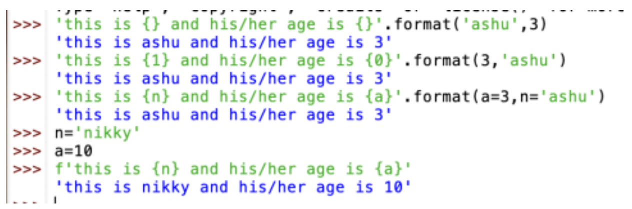


```
>>> '#'.join('HAI')
'H#A#I'
>>> '#'.join(['HAI','hello','bye'])
'HAI#hello#bye'
>>> '#'.join({'HAI','hello','bye'})
'bye#hello#HAI'
>>> '#'.join(['HAI','hello','bye',90])
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    '#'.join(['HAI','hello','bye',90])
TypeError: sequence item 3: expected str instance, int found
>>> '#'.join(['HAI','hello','bye','90'])
'HAI#hello#bye#90'
```

Format:

format( )	Content { } content{ }'.format(value1,value2) f'content(val1}content{val2}'	1. It is used for creating dynamic string 2. We need to create place holders 3. Place holders are created by using { }

Example:



```
>>> 'this is {} and his/her age is {}'.format('ashu',3)
'this is ashu and his/her age is 3'
>>> 'this is {1} and his/her age is {0}'.format(3,'ashu')
'this is ashu and his/her age is 3'
>>> 'this is {n} and his/her age is {a}'.format(a=3,n='ashu')
'this is ashu and his/her age is 3'
>>> n='nikky'
>>> a=10
>>> f'this is {n} and his/her age is {a}'
'this is nikky and his/her age is 10'
```