String Functions

String Functions

Methods	Description
upper()	converts the string to uppercase
lower()	converts the string to lowercase
partition()	returns a tuple
replace()	replaces substring inside
find()	returns the index of first occurrence of substring
rstrip()	removes trailing characters
split()	splits string from left
startswith()	checks if string starts with the specified string
isnumeric()	checks numeric characters
index()	returns index of substring

Definition and Usage

The `lower()` method returns a string where all characters are lower case.

Symbols and Numbers are ignored.

Syntax

```
string.lower()
```

Parameter Values

No parameters

Lower case the string:

```
txt = "Hello my FRIENDS"

x = txt.lower()

print(x)
```

Definition and Usage

The `upper()` method returns a string where all characters are in upper case.

Symbols and Numbers are ignored.

Syntax

```
string.upper()
```

Parameter Values

No parameters

Example

Upper case the string:

```
txt = "Hello my friends"

x = txt.upper()

print(x)
```

Definition and Usage

The `title()` method returns a string where the first character in every word is upper case. Like a header, or a title.

If the word contains a number or a symbol, the first letter after that will be converted to upper case.

Syntax

```
string.title()
```

Parameter Values

No parameters.

Example

Make the first letter in each word upper case:

```
txt = "Welcome to my 2nd world"

x = txt.title()

print(x)
```

Definition and Usage

The `replace()` method replaces a specified phrase with another specified phrase.

Note: *All* occurrences of the specified phrase will be replaced, if nothing else is specified.

Syntax

```
string.replace(oldvalue, newvalue, count)
```

Parameter Values

Parameter	Description
oldvalue	Required. The string to search for
newvalue	Required. The string to replace the old value with
count	Optional. A number specifying how many occurrences of the old value you want to replace. Default is all

Example

Replace all occurrence of the word "one":

```
txt = "one one was a race horse, two two was one too."

x = txt.replace("one", "three")

print(x)
```

Definition and Usage

The `split()` method splits a string into a list.

You can specify the separator, default separator is any whitespace.

Note: When maxsplit is specified, the list will contain the specified number of elements *plus one*.

Syntax

```
string.split(separator, maxsplit)
```

Parameter Values

Parameter	Description
separator	Optional. Specifies the separator to use when splitting the string. By default any whitespace is a separator
maxsplit	Optional. Specifies how many splits to do. Default value is -1, which is "all occurrences"

Example

Use a hash character as a separator:

```
txt = "apple#banana#cherry#orange"

x = txt.split("#")

print(x)
```

Example

Split the string into a list with max 2 items:

```
txt = "apple#banana#cherry#orange"

# setting the maxsplit parameter to 1, will return a list with 2 elements!
x = txt.split("#", 1)

print(x)
```

The `capitalize()` method returns a string where the first character is upper case, and the rest is lower case.

Example

Upper case the first letter in this sentence:

```
txt = "hello, and welcome to my world."

x = txt.capitalize()

print (x)
```

The `center()` method will center align the string, using a specified character (space is default) as the fill character.

Syntax

```
string.center(length, character)
```

Parameter Values

Parameter	Description
length	Required. The length of the returned string
character	Optional. The character to fill the missing space on each side. Default is " " (space)

Example

Using the letter "O" as the padding character:

```
txt = "banana"

x = txt.center(20, "O")

print(x)
```

The `count()` method returns the number of times a specified value appears in the string.

Syntax

```
string.count(value, start, end)
```

Parameter Values

Parameter	Description
value	Required. A String. The string to value to search for
start	Optional. An Integer. The position to start the search. Default is 0
end	Optional. An Integer. The position to end the search. Default is the end of the string

Example

Search from position 10 to 24:

```
txt = "I love apples, apple are my favorite fruit"

x = txt.count("apple", 10, 24)

print(x)
```

The `endswith()` method returns True if the string ends with the specified value, otherwise False.

Syntax

```
string.endswith(value, start, end)
```

Parameter Values

Parameter	Description
value	Required. The value to check if the string ends with
start	Optional. An Integer specifying at which position to start the search
end	Optional. An Integer specifying at which position to end the search

Example

Check if the string ends with the phrase "my world.":

```
txt = "Hello, welcome to my world."

x = txt.endswith("my world.")

print(x)
```

The `find()` method finds the first occurrence of the specified value.

The `find()` method returns -1 if the value is not found.

The `find()` method is almost the same as the `index()` method, the only difference is that the `index()` method raises an exception if the value is not found. (See example below)

Syntax

```
string.find(value, start, end)
```

Parameter Values

Parameter	Description
value	Required. The value to search for
start	Optional. Where to start the search. Default is 0
end	Optional. Where to end the search. Default is to the end of the string

Example

Where in the text is the first occurrence of the letter "e"?:

```
txt = "Hello, welcome to my world."

x = txt.find("e")

print(x)
```

Example

If the value is not found, the find() method returns -1, but the index() method will raise an exception:

```
txt = "Hello, welcome to my world."

print(txt.find("q"))
print(txt.index("q"))
```

isdecimal()	isdigit()	isnumeric()
Example of string with decimal characters: **"12345"** **"12"** **"98201"**	Example of string with digits: **"12345"** **"123³"** **"³"**	Example of string with numerics: **"12345"** **"½¼"** **"½"** **"12345½"**
Returns 'true' if all characters of the string are decimal.	Returns 'true' if all characters of the string are digits.	Returns 'true if all characters of the string are numeric.

The `join()` method takes all items in an iterable and joins them into one string.
A string must be specified as the separator.

Syntax

```
string.join(iterable)
```

Parameter Values

Parameter	Description
iterable	Required. Any iterable object where all the returned values are strings

Example: join()

```
sep = ','
names = ['Steve', 'Bill', 'Ravi', 'Jonathan'] # list
print(sep.join(names))

mystr = 'Hello' # string
print(sep.join(mystr))

nums = ('1', '2', '3', '4') # tuple
print(sep.join(nums))

langs = {'Python', 'C#', 'Java', 'C++'} # set
print(sep.join(langs))
```

Output

```
'Steve,Bill,Ravi,Jonathan'
'H,e,l,l,o'
'1,2,3,4'
'Python,C#,Java,C++'
```

The seperator string can be of any length or char, as shown below.

```
sep = '-->'
mystr = 'Hello' # string
print(sep.join(mystr))

sep = '****'
nums = ('1', '2', '3', '4') # tuple
print(sep.join(nums))
```

Output

```
'SteveõBillõRaviõJonathan'
'H-->e-->l-->l-->o'
'1****2****3****4'
```

The elements of an iterable must be string elements, otherwise it will raise a `TypeError`

Example: join() with Numbers

Copy

```
nums = (1, 2, 3, 4, 5)
print(','.join())
```

Output

```
Traceback (most recent call last):
    ','.join((1,2,3,4,5))
TypeError: sequence item 0: expected str instance, int found
```

Definition and Usage

The `strip()` method removes any leading (spaces at the beginning) and trailing (spaces at the end) characters (space is the default leading character to remove)

Syntax

```
string.strip(characters)
```

Parameter Values

Parameter	Description
characters	Optional. A set of characters to remove as leading/trailing characters

Example

Remove spaces at the beginning and at the end of the string:

```
txt = "     banana     "

x = txt.strip()

print("of all fruits", x, "is my favorite")
```

Example

Remove the leading and trailing characters:

```
txt = ",,,,,rrttgg.....banana....rrr"

x = txt.strip(",.grt")

print(x)
```

Output: First it will remove “,” next “r, t, g” and “.” these are leading [beginning] characters
Trailing characters [end characters] are “.” and “r”

```
banana
```

Python String rstrip()

In this tutorial, we will learn about the Python String rstrip() method with the help of examples.

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

Example

```
title = 'Python Programming    '

# remove trailing whitespace from title
result = title.rstrip()
print(result)

# Output: Python Programming
```

Python String Formatting (f-Strings)

Python **f-Strings** make it really easy to print values and variables. For example,

```
name = 'Cathy'
country = 'UK'

print(f'{name} is from {country}')
```

Run Code »

Output

```
Cathy is from UK
```

Here, `f'{name} is from {country}'` is an **f-string**.

This new formatting syntax is powerful and easy to use. From now on, we will use f-Strings to print strings and variables.