String

07 July 2024 12:24 PM

By Alok Ranjan
Linkedin- www.linkedin.com/in/alok-ranjandigu-coder
GitHub- https://github.com/alokyadav2020

STRINGS

String is a sequence which is made up of one or more UNICODE characters. Here the character can be a letter, digit, whitespace or any other symbol. A string can be created by enclosing one or more characters in single, double or triple quote.

```
print("It's alright")
print("He is called 'Johnny'")
print('He is called "Johnny"')
```

$\boxed{\text{Print( )}}$ for output

Assign String to a Variable

Assigning a string to a variable is done with the variable name followed by an equal sign and the string:

Example

```
a = "Hello"
print(a)
```

a = 'Hello'

Multiline Strings

You can assign a multiline string to a variable by using three quotes:

Full_Name = 'Alok Ranja'

Example

You can use three double quotes:

```
a = """Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua."""
print(a)
```

Accessing Characters in a String

Each individual character in a string can be accessed using a technique called indexing. The index specifies the character to be accessed in the string and is written in square brackets ([ ]). The index of the first character (from left) in the string is 0 and the last character is n-1 where n is the length of the string. If we give index value out of this range then we get an IndexError. The index must be an integer (positive, zero or negative)

```
#initializes a string str1
>>> str1 = 'Hello World!'
#gives the first character of str1
 >>> str1[0]
'H'
 #gives seventh character of str1
>>> str1[6]
'W'
#gives last character of str1
 >>> str1[11]
'!'
#gives error as index is out of range
>>> str1[15]
IndexError: string index out of range
```

Str1 = 'Hello World!'

Str1[1:10:2]

[6]
w

[6:8]

start

Str1[3:5]

(10) Str1[start : end : 1] (2) jump

end

```
>>> str1[15]
IndexError: string index out of range
```

start [6:8] wo end 12 end jump

Positive Indices	0	1	2	3	4	5	6	7	8	9	10	11
String	H	e	l	l	o		W	o	r	l	d	!
Negative Indices	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

o el ol

[-:-:3] str1[' ']

!

Upper Case

Example

The `upper()` method returns the string in upper case:

```
a = "Hello, World!"
print(a.upper())
```

Lower Case

Example

The `lower()` method returns the string in lower case:

```
a = "Hello, World!"
print(a.lower())
```

Remove Whitespace

Whitespace is the space before and/or after the actual text, and very often you want to remove this space.

Example

The `strip()` method removes any whitespace from the beginning or the end:

```
a = " Hello, World! "
print(a.strip()) # returns "Hello, World!"
```

Replace String

Example

The `replace()` method replaces a string with another string:

```
a = "Hello, World!"
print(a.replace("H", "J"))
```

Split String

The `split()` method returns a list where the text between the specified separator becomes the list items.

Example

The `split()` method splits the string into substrings if it finds instances of the separator:

```
a = "Hello, World!"
print(a.split(",")) # returns ['Hello', ' World!']
```

String Concatenation

To concatenate, or combine, two strings you can use the + operator.

Example

Merge variable `a` with variable `b` into variable `c`:

```
a = "Hello"

b = "World"

c = a + b

print(c)
```

String Format

As we learned in the Python Variables chapter, we cannot combine strings and numbers like this:

Example

```
age = 36

txt = "My name is John, I am " + age

print(txt)
```

F-Strings

F-String was introduced in Python 3.6, and is now the preferred way of formatting strings.
To specify a string as an f-string, simply put an `f` in front of the string literal, and add curly brackets `{}` as placeholders for variables and other operations.

Example

Create an f-string:

```
age = 36

txt = f"My name is John, I am {age}"

print(txt)
```

Placeholders and Modifiers

A placeholder can contain variables, operations, functions, and modifiers to format the value.

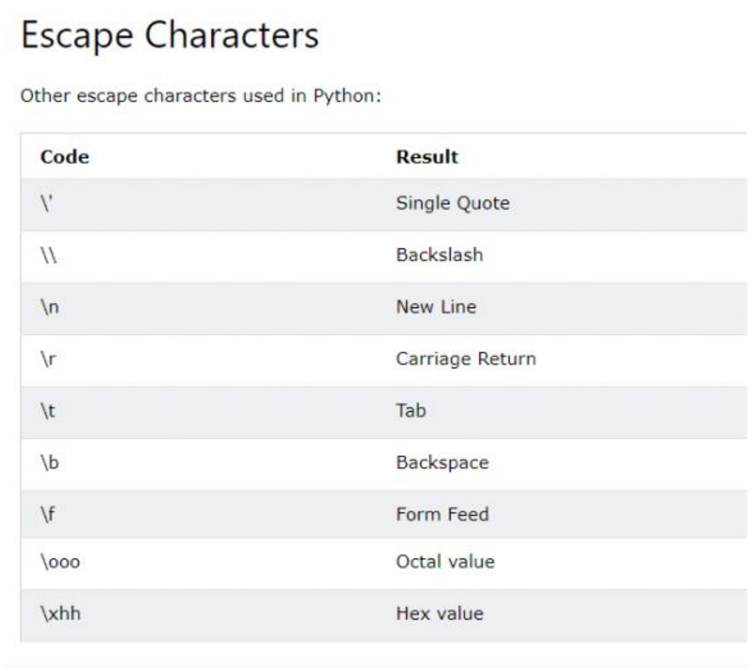
Example

Add a placeholder for the `price` variable:

```
price = 59

txt = f"The price is {price} dollars"

print(txt)
```



Escape Characters

Other escape characters used in Python:

Code	Result
\'	Single Quote
\\	Backslash
\n	New Line
\r	Carriage Return
\t	Tab
\b	Backspace
\f	Form Feed
\ooo	Octal value
\xhh	Hex value

STRING METHODS AND BUILT-IN FUNCTIONS

len() Returns the length of the given string

```
>>> str1 = 'Hello World!'
>>> len(str1)
12
```

title() Returns the string with first letter of every word in the string in uppercase and rest in lowercase

lower() Returns the string with all uppercase letters converted to lowercase

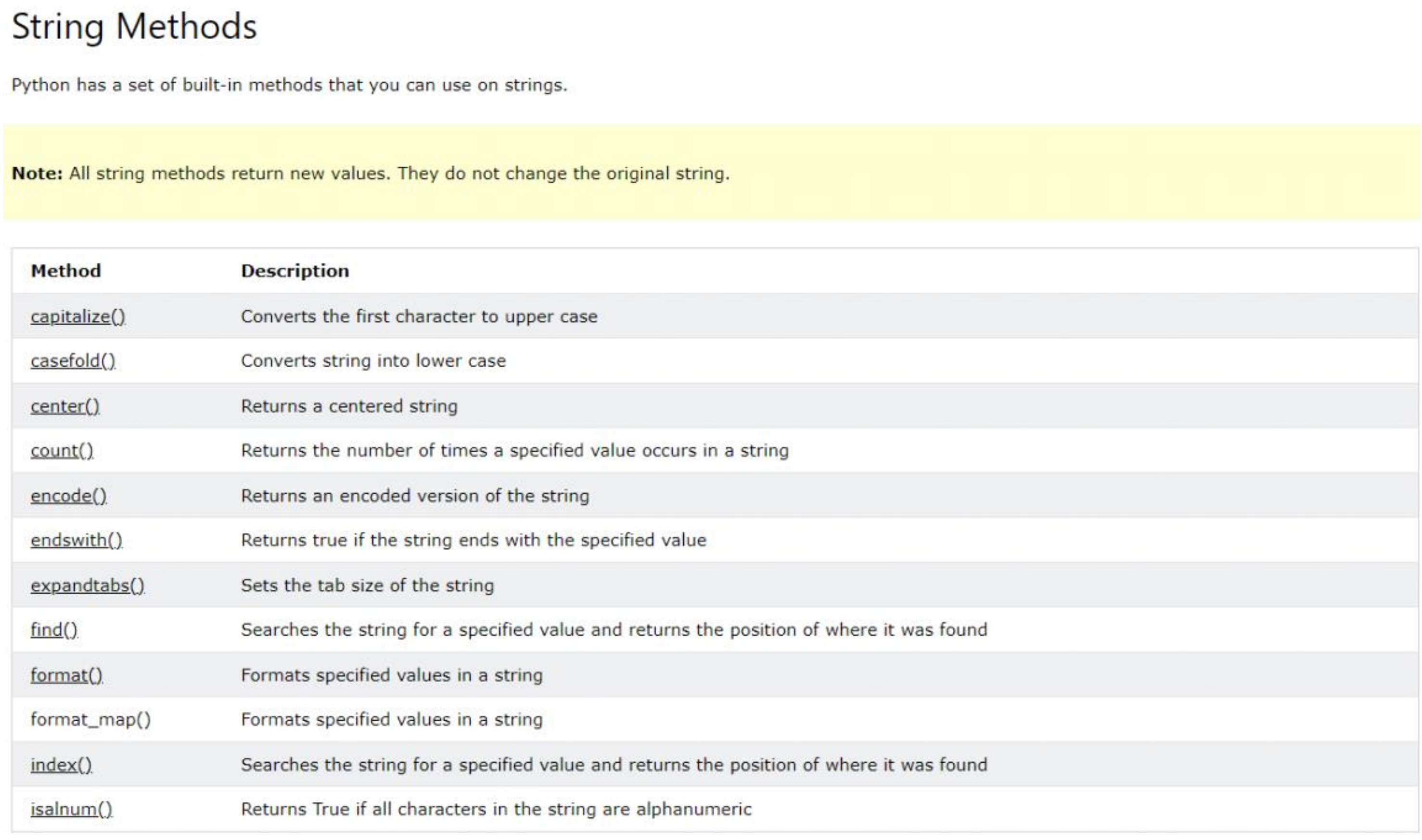
count(str, start, end) Returns number of times substring str occurs in the given string. If we do not give start index and end index then searching starts from index 0 and ends at length of the string

find(str,start, end) Returns the first occurrence of index of substring str occurring in the given string. If we do not give start and end then searching starts from index 0 and ends at length of the string. If the substring is not present in the given string, then the function returns -1.

index(str, start, end) Same as find() but raises an exception if the substring is not present in the given string.

endswith() Returns True if the given string ends with the supplied substring otherwise returns False.

startswith() Returns True if the given string starts with the supplied substring otherwise returns False



String Methods

Python has a set of built-in methods that you can use on strings.

Note: All string methods return new values. They do not change the original string.

Method	Description
capitalize()	Converts the first character to upper case
casefold()	Converts string into lower case
center()	Returns a centered string
count()	Returns the number of times a specified value occurs in a string
encode()	Returns an encoded version of the string
endswith()	Returns true if the string ends with the specified value
expandtabs()	Sets the tab size of the string
find()	Searches the string for a specified value and returns the position of where it was found
format()	Formats specified values in a string
format_map()	Formats specified values in a string
index()	Searches the string for a specified value and returns the position of where it was found
isalnum()	Returns True if all characters in the string are alphanumeric

isalpha()	Returns True if all characters in the string are in the alphabet
isascii()	Returns True if all characters in the string are ascii characters
isdecimal()	Returns True if all characters in the string are decimals
isdigit()	Returns True if all characters in the string are digits
isidentifier()	Returns True if the string is an identifier
islower()	Returns True if all characters in the string are lower case
isnumeric()	Returns True if all characters in the string are numeric
isprintable()	Returns True if all characters in the string are printable
isspace()	Returns True if all characters in the string are whitespaces
istitle()	Returns True if the string follows the rules of a title
isupper()	Returns True if all characters in the string are upper case
join()	Joins the elements of an iterable to the end of the string
ljust()	Returns a left justified version of the string
lower()	Converts a string into lower case
lstrip()	Returns a left trim version of the string
maketrans()	Returns a translation table to be used in translations
partition()	Returns a tuple where the string is parted into three parts
replace()	Returns a string where a specified value is replaced with a specified value
rfind()	Searches the string for a specified value and returns the last position of where it was found
rindex()	Searches the string for a specified value and returns the last position of where it was found
rjust()	Returns a right justified version of the string
rpartition()	Returns a tuple where the string is parted into three parts
rsplit()	Splits the string at the specified separator, and returns a list
rstrip()	Returns a right trim version of the string
split()	Splits the string at the specified separator, and returns a list
splitlines()	Splits the string at line breaks and returns a list
startswith()	Returns true if the string starts with the specified value
strip()	Returns a trimmed version of the string
swapcase()	Swaps cases, lower case becomes upper case and vice versa
title()	Converts the first character of each word to upper case
translate()	Returns a translated string
upper()	Converts a string into upper case
zfill()	Fills the string with a specified number of 0 values at the beginning

$\underline{a} = 10 \rightarrow$

$\boxed{x = 10 \quad y = 20}$

$d = \underline{30.5} \rightarrow \underline{float}$

$a = 10$
$b = 20$ $\boxed{int}$
$c = 30$

b = 20
c = 30

String

' '

" "

''' '''

""" """ → Doc String

a = 10

Name = 'alok'
Name = "alok"

a = '4' → str

a = 4 → int

b = int(a)

b = 4 → int