



# Python Strings

[< Previous](#)[Next >](#)

## String Literals

String literals in python are surrounded by either single quotation marks, or double quotation marks.

'hello' is the same as "hello" .

You can display a string literal with the `print()` function:

### Example

```
print("Hello")  
print('Hello')
```

[Try it Yourself »](#)

## 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)
```

Try it Yourself »

---

## Multiline Strings

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

### 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)
```

Try it Yourself »

Or three single quotes:

### Example

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

Try it Yourself »

**Note:** in the result, the line breaks are inserted at the same position as in the code.

ADVERTISEMENT

## Strings are Arrays

Like many other popular programming languages, strings in Python are arrays of bytes representing unicode characters.

However, Python does not have a character data type, a single character is simply a string with a length of 1.

Square brackets can be used to access elements of the string.

### Example

Get the character at position 1 (remember that the first character has the position 0):

```
a = "Hello, World!"  
print(a[1])
```

Try it Yourself »

## Slicing

You can return a range of characters by using the slice syntax.

Specify the start index and the end index, separated by a colon, to return a part of the string.

### Example

Get the characters from position 2 to position 5 (not included):

```
b = "Hello, World!"  
print(b[2:5])
```

Try it Yourself »

## Negative Indexing

Use negative indexes to start the slice from the end of the string:

### Example

Get the characters:

From: "o" in "World!" (position -5)

To, but not included: "d" in "World!" (position -2):

```
b = "Hello, World!"  
print(b[-5:-2])
```

Try it Yourself »

## String Length

To get the length of a string, use the `len()` function.

### Example

The `len()` function returns the length of a string:

```
a = "Hello, World!"  
print(len(a))
```

Try it Yourself »

# String Methods

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

## Example

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

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

Try it Yourself »

## Example

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

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

Try it Yourself »

## Example

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

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

Try it Yourself »

## Example

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

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

Try it Yourself »

## 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!']
```

Try it Yourself »

Learn more about String Methods with our [String Methods Reference](#)

---

## Check String

To check if a certain phrase or character is present in a string, we can use the keywords `in` or `not in`.

## Example

Check if the phrase "ain" is present in the following text:

```
txt = "The rain in Spain stays mainly in the plain"  
x = "ain" in txt  
print(x)
```

Try it Yourself »

## Example

Check if the phrase "ain" is NOT present in the following text:

```
txt = "The rain in Spain stays mainly in the plain"
x = "ain" not in txt
print(x)
```

Try it Yourself »

---

## 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)
```

Try it Yourself »

### Example

To add a space between them, add a " " :

```
a = "Hello"
b = "World"
c = a + " " + b
print(c)
```

Try it Yourself »

---

## 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)
```

Try it Yourself »

But we can combine strings and numbers by using the `format()` method!

The `format()` method takes the passed arguments, formats them, and places them in the string where the placeholders `{}` are:

## Example

Use the `format()` method to insert numbers into strings:

```
age = 36
txt = "My name is John, and I am {}"
print(txt.format(age))
```

Try it Yourself »

The `format()` method takes unlimited number of arguments, and are placed into the respective placeholders:

## Example

```
quantity = 3
itemno = 567
price = 49.95
myorder = "I want {} pieces of item {} for {} dollars."
print(myorder.format(quantity, itemno, price))
```



Try it Yourself »

You can use index numbers `{0}` to be sure the arguments are placed in the correct placeholders:

## Example

```
quantity = 3
itemno = 567
price = 49.95
myorder = "I want to pay {2} dollars for {0} pieces of item {1}."
print(myorder.format(quantity, itemno, price))
```

Try it Yourself »

---

## Escape Character

To insert characters that are illegal in a string, use an escape character.

An escape character is a backslash `\` followed by the character you want to insert.

An example of an illegal character is a double quote inside a string that is surrounded by double quotes:

## Example

You will get an error if you use double quotes inside a string that is surrounded by double quotes:

```
txt = "We are the so-called "Vikings" from the north."
```

Try it Yourself »

To fix this problem, use the escape character `\"` :

## Example

The escape character allows you to use double quotes when you normally would not be allowed:

```
txt = "We are the so-called \"Vikings\" from the north."
```

[Try it Yourself »](#)

Other escape characters used in Python:

Code	Result	Try it
\'	Single Quote	<a href="#">Try it »</a>
\\	Backslash	<a href="#">Try it »</a>
\n	New Line	<a href="#">Try it »</a>
\r	Carriage Return	<a href="#">Try it »</a>
\t	Tab	<a href="#">Try it »</a>
\b	Backspace	<a href="#">Try it »</a>
\f	Form Feed	
\ooo	Octal value	<a href="#">Try it »</a>
\xhh	Hex value	<a href="#">Try it »</a>

## String Methods

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

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

Method	Description
<a href="#"><u>capitalize()</u></a>	Converts the first character to upper case

<u>casefold()</u>	Converts string into lower case
<u>center()</u>	Returns a centered string
<u>count()</u>	Returns the number of times a specified value occurs in a string
<u>encode()</u>	Returns an encoded version of the string
<u>endswith()</u>	Returns true if the string ends with the specified value
<u>expandtabs()</u>	Sets the tab size of the string
<u>find()</u>	Searches the string for a specified value and returns the position of where it was found
<u>format()</u>	Formats specified values in a string
<u>format_map()</u>	Formats specified values in a string
<u>index()</u>	Searches the string for a specified value and returns the position of where it was found
<u>isalnum()</u>	Returns True if all characters in the string are alphanumeric
<u>isalpha()</u>	Returns True if all characters in the string are in the alphabet
<u>isdecimal()</u>	Returns True if all characters in the string are decimals
<u>isdigit()</u>	Returns True if all characters in the string are digits
<u>isidentifier()</u>	Returns True if the string is an identifier
<u>islower()</u>	Returns True if all characters in the string are lower case
<u>isnumeric()</u>	Returns True if all characters in the string are numeric
<u>isprintable()</u>	Returns True if all characters in the string are printable
<u>isspace()</u>	Returns True if all characters in the string are whitespaces
<u>istitle()</u>	Returns True if the string follows the rules of a title
<u>isupper()</u>	Returns True if all characters in the string are upper case
<u>join()</u>	Joins the elements of an iterable to the end of the string
<u>ljust()</u>	Returns a left justified version of the string
<u>lower()</u>	Converts a string into lower case
<u>lstrip()</u>	Returns a left trim version of the string
<u>maketrans()</u>	Returns a translation table to be used in translations

<u>partition()</u>	Returns a tuple where the string is parted into three parts
<u>replace()</u>	Returns a string where a specified value is replaced with a specified value
<u>rfind()</u>	Searches the string for a specified value and returns the last position of where it was found
<u>rindex()</u>	Searches the string for a specified value and returns the last position of where it was found
<u>rjust()</u>	Returns a right justified version of the string
<u>rpartition()</u>	Returns a tuple where the string is parted into three parts
<u>rsplit()</u>	Splits the string at the specified separator, and returns a list
<u>rstrip()</u>	Returns a right trim version of the string
<u>split()</u>	Splits the string at the specified separator, and returns a list
<u>splitlines()</u>	Splits the string at line breaks and returns a list
<u>startswith()</u>	Returns true if the string starts with the specified value
<u>strip()</u>	Returns a trimmed version of the string
<u>swapcase()</u>	Swaps cases, lower case becomes upper case and vice versa
<u>title()</u>	Converts the first character of each word to upper case
<u>translate()</u>	Returns a translated string
<u>upper()</u>	Converts a string into upper case
<u>zfill()</u>	Fills the string with a specified number of 0 values at the beginning

## Test Yourself With Exercises

### Exercise:

Use the `len` method to print the length of the string.