```
"""
Strings 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:
"""

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

```
Hello
Hello
```

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

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

```
e
```

```
# ToLoop through the letters in the word "banana":

for x in "banana":
  print(x)
```

```
b
a
n
a
n
a
```

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

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

```
13
```

```
#To Check if "free" is present in the following text:

txt = "The best things in life are free!"
print("free" in txt)
#-------------------------------------------------------------------------------------------
# To Print only if "free" is present:

txt = "The best things in life are free!"
if "free" in txt:
  print("Yes, 'free' is present.")
```

```
True
Yes, 'free' is present.
```

```
# To Check if "expensive" is NOT present in the following text:

txt = "The best things in life are free!"
print("expensive" not in txt)
#-------------------------------------------------------------------------------------------

#To print only if "expensive" is NOT present:

txt = "The best things in life are free!"
if "expensive" not in txt:
  print("No, 'expensive' is NOT present.")
```

```
True
No, 'expensive' is NOT present.
```

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

b = "Hello, World!"
print(b[2:5])
#-------------------------------------------------------------------------------------------
# To Get the characters from the start to position 5 (not included):

b = "Hello, World!"
print(b[:5])
#-------------------------------------------------------------------------------------------
# To Get the characters from position 2, and all the way to the end:

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

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])
```

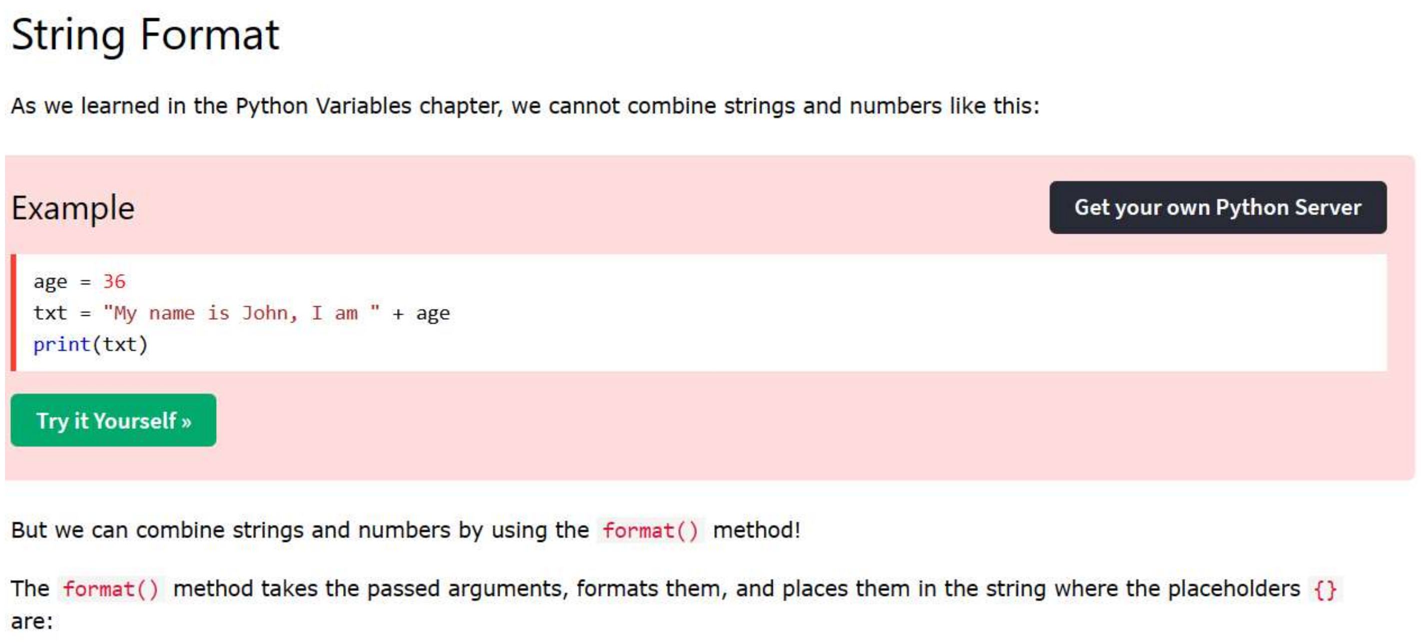
```
llo
Hello
llo, World!
orl
```

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

a = "Hello, World!"
print(a.upper())
#----------------------------------------------------------------------------------------------
#The lower() method returns the string in lower case:

a = "Hello, World!"
print(a.lower())
#----------------------------------------------------------------------------------------------
#The strip() method removes any whitespace from the beginning or the end:

a = " Hello, World! "
print(a.strip()) # returns "Hello, World!"
#----------------------------------------------------------------------------------------------
#The replace() method replaces a string with another string. The text before the comma is th
# before the comma with

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

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

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



#----------------------------------------------------------------------------------------------
```

```
HELLO, WORLD!
hello, world!
Hello, World!
Jello, World!
['Hello', ' World!']
```

```
"""
String Concatenation
To concatenate, or combine, two strings you can use the + operator.

Exampl

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

a = "Hello"
b = "World"
c = a + b
print(c)
#To add a space between them, add a " ":

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

```
     HelloWorld
     Hello World
```



String Format

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

Example

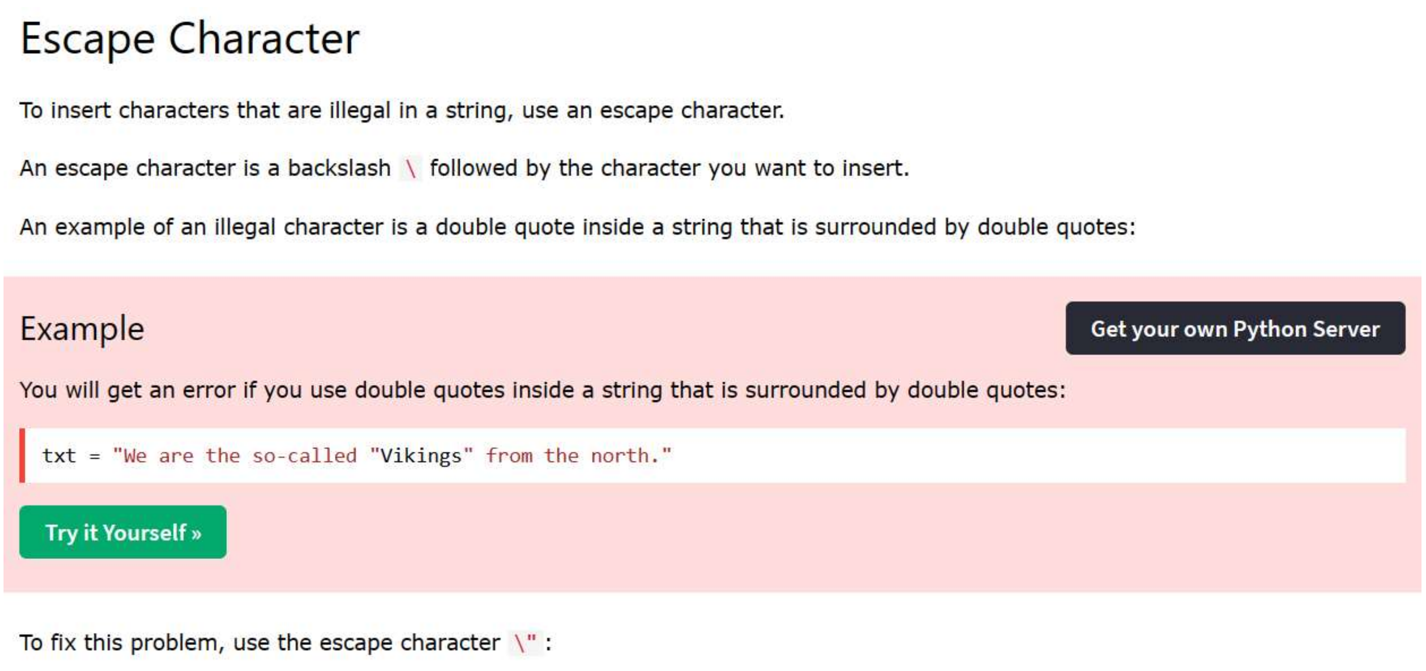
```
age = 36
txt = "My name is John, I am " + age
print(txt)
```

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))
#-------------------------------------------------------------------------------------------
#The format() method takes unlimited number of arguments, and are placed into the respective
quantity = 3
itemno = 567
price = 49.95
myorder = "I want {} pieces of item {} for {} dollars."
print(myorder.format(quantity, itemno, price))
#-------------------------------------------------------------------------------------------
#You can use index numbers {0} to be sure the arguments are placed in the correct placeholde
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))
```

```
My name is John, and I am 36
I want 3 pieces of item 567 for 49.95 dollars.
I want to pay 49.95 dollars for 3 pieces of item 567.
```



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:

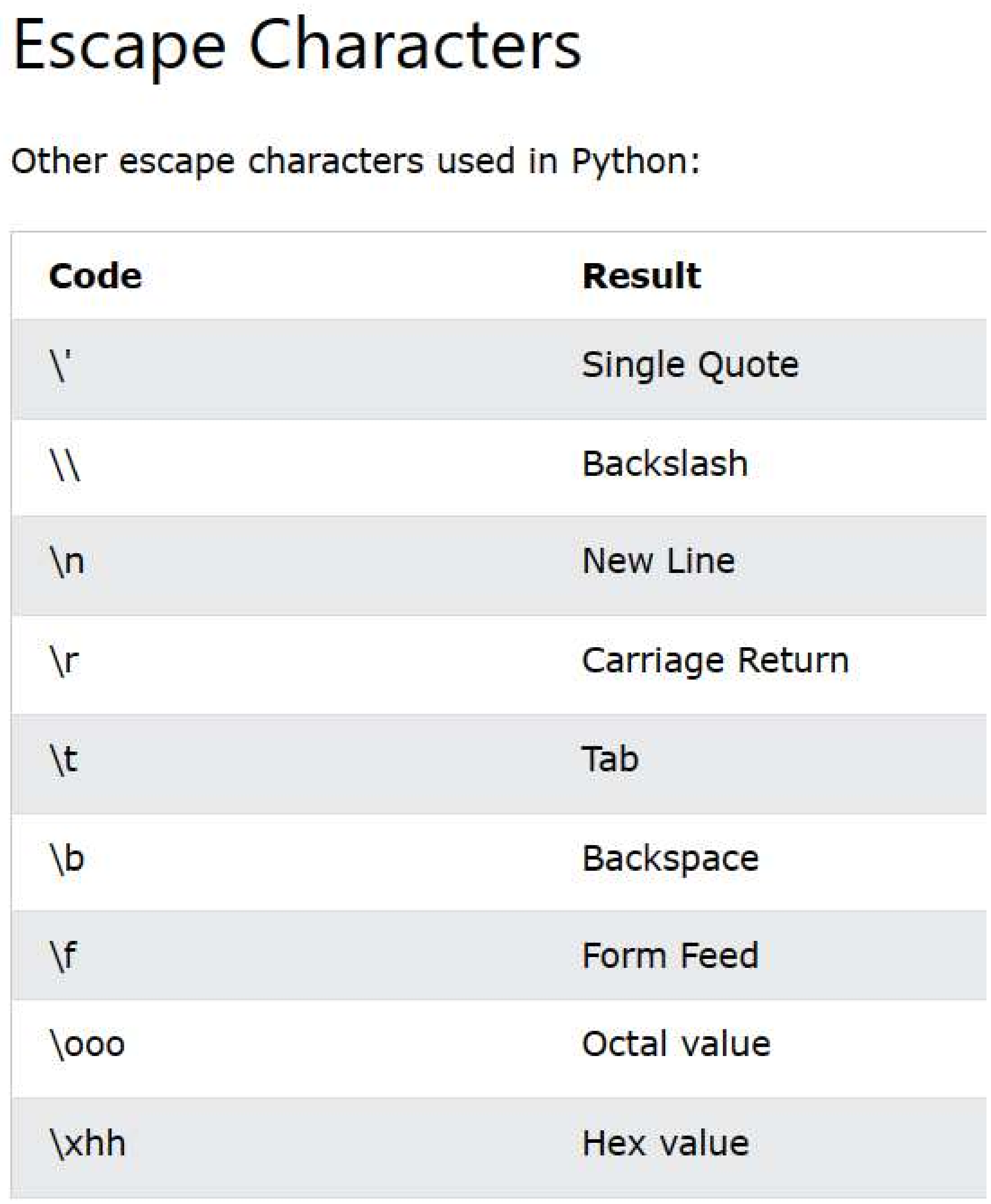
Example

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."
```

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

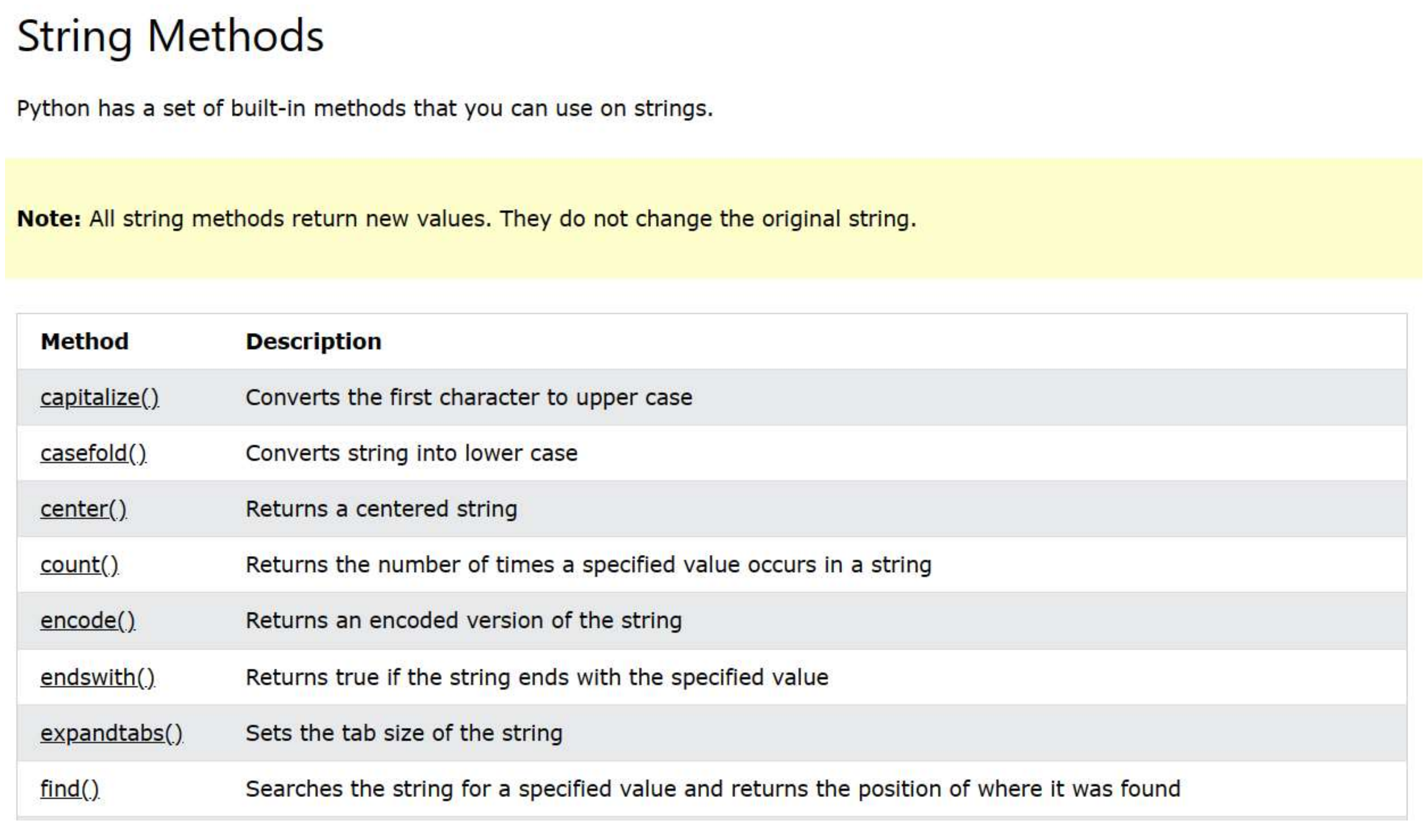
```
"""
Example
The escape character allows you to use double quotes when you normally would not be allowed:
"""

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



Escape Characters

Other escape characters used in Python:

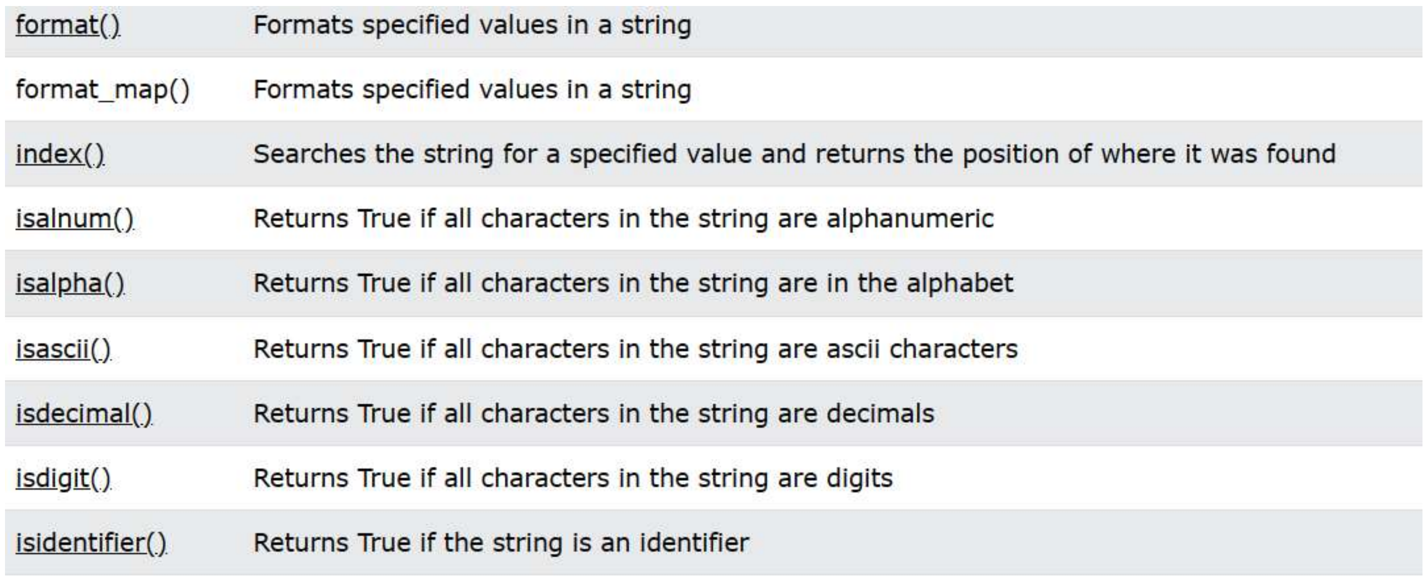
Code	Result
\\'	Single Quote
\\\\	Backslash
\\n	New Line
\\r	Carriage Return
\\t	Tab
\\b	Backspace
\\f	Form Feed
\\ooo	Octal value
\\xhh	Hex value



String Methods

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