Python For Beginners

Strings 5

```
This is a new serie of articles here at Python for beginners, that are supposed
to be a starting point for completely beginners of Python.

See it as a cheat sheet, reference, manual or whatever you want.

The purpose is to very short write down the basics of Python.

This page will describe Strings in Python.
```

What is a string?

```
A string is a list of characters in order.

A character is anything you can type on the keyboard in one keystroke,
like a letter, a number, or a backslash.

Strings can have spaces: "hello world".

An empty string is a string that has 0 characters.

Python strings are immutable

Python recognize as strings everything that is delimited by quotation marks
(" " or ' ').
```

Accessing Strings

```
Use [ ] to access characters in a string:
word = "computer"
letter = word[0]

Use [ # :#] to get set of letters
word= word[0:3]

To pick from beginning to a set point:
word = [:4]

To pick from set point to end:
word = [3:]

To pick starting from the end:
word = [-1]
```

Quotes

```
Strings can be enclosed in single quotes
print 'Hello World in single quotes'

Strings can also be enclosed in double quotes
print "Hello World in double quotes"
```

Search

Follow us on Twitter

Follow @pythonbeginners

Categories

- Basics
- Cheatsheet
- Code snippets
- Development
- Dictionary
- Error Handling
- Lists
- Loops
- Modules
- Strings
- System & OS
- Uncategorized
- Web & Internet

Follow us on Twitter

pythonbeginners on Twitter

Follow

957 people follow pythonbeginners

gstoyano onlinegr ieski dbosulli discomea

dbta abhijeet tupperwa fvcprodu daniel_v

```
Strings can be continued on the next line
print "This string is continued on the\
next line (in the source) but a newline is not added"
```


Raw Strings

```
Raw strings exists so that you can more conveniently express strings that would
be modified by escape sequence processing.

This is most especially useful when writing out regular expressions, or other
forms of code in string literals.

print r"The newline character is represented with \n"
```

Concatenate Strings

```
In Python there are a few different ways to concatenating strings.

Concatenation combines two (or more) strings into a new string object.

You can use the + operator, like this:
print "You can concatenate two " + "strings with the '+' operator."

str1 = "Hello"
str2 = "World"
str1 + str2     # concatenation: a new string

String literals may be concatenated by a space
word = 'left' "right" 'left'

Any string expression may be concatenated by a +
word = wordA + "\n" + wordB
```


Reverse Strings

```
string = "Hello World"

print ' '.join(reversed(string))
>>Output:
d l r o W o l l e H
```

Changing Upper and Lower Case Strings

```
string = "Hello World"

print string.lower()

print string.upper()

print string.title()
```

Replace Strings

```
string = "Hello World"
```

string.replace("Hello", "Goodbye")

Repeat Strings

```
print "."* 10 # prints ten dots( print string * n ; prints the string n times)
```

Split Strings

```
The split function splits a single string into a string array using the separator
defined.

If no separator is defined, whitespace is used.
```

```
x = 'blue,red,green'
x.split(",")
['blue', 'red', 'green']

word = "This is some random text"
words2 = word.split(" ")
print words
['This', 'is', 'some', 'random', 'text']
```

Startswith / Endswith

```
Checking if a string starts or ends with a substring:
s = "hello world"

s.startswith("hello")
True

s.endswith("rld")
True
```

Strip Strings

```
Python strings have the strip(), lstrip(), rstrip() methods for removing
any character from both ends of a string.

If the characters to be removed are not specified then white-space will be removed.
```

```
string = "Hello World"

#Strip off newline characters from end of the string
print string.strip('\n')

strip()     #removes from both ends
lstrip()    #removes leading characters (Left-strip)
rstrip()    #removes trailing characters (Right-strip)

spacious = "   xyz   "
print spacious.strip()

spacious = "   xyz   "
print spacious.lstrip()

spacious =  "xyz   "
print spacious.rstrip()
```

Slicing Strings

```
Strings have indices, so we can refer to every position of a string with its
corresponding index.

Keep in mind that python, as many other languages, starts to count from 0!!
```

```
print string[1]         #get one char of the word
print string[1:2]       #get one char of the word (same as above)
print string[1:3]       #get the first three char
print string[:3]        #get the first three char
print string[-3:]       #get the last three char
print string[3:]        #get all but the three first char
print string[:-3]       #get all but the three last character

x = "my string"

x[start:end]            # items start through end-1
x[start:]               # items start through the rest of the list
x[:end]                 # items from the beginning through end-1
x[:]                    # a copy of the whole list
```

Formatting Strings

```
String formatting with %
```

```
%f      # used for floating point

x = 'apple'
y = 'lemon'
z = "The items in the basket are %s and %s" % (x,y)

Note: Make sure to use a tuple for the values.

String formatting using { }

The pairs of empty curly braces {} serve as place-holders for the variables that
we want to place inside the string.

We then pass these variables into our string as inputs of the strings format()
method in order.

With this method, we don’t have to change our integer into string types first
the format method does that for us automatically.

fname = "Joe"
lname = "Who"
age = "24"

#Example of how to use the format() method:
print "{} {} is {} years ".format(fname, lname, age)

#Another really cool thing is that we don't have to provide the inputs in the
#same order, if we number the place-holders.

print "{0} {1} is {2} years".format(fname, lname, age)
```

Join Strings

```
This method takes a list of strings and joins them together with the calling
string in between each element.
```



```

>>> ' '.join(['the', 'cat', 'sat', 'on', 'the', 'mat'])
'the cat sat on the mat'

#Let's look at one more example of using the Join method:
#creating a new list
>>> music = ["Abba","Rolling Stones","Black Sabbath","Metallica"]

#Join a list with an empty space
>>> print ' '.join(music)

#Join a list with a new line
>>> print "\n".join(music)
```

Testing Strings

```
A string in Python can be tested for truth value.

The return type will be in Boolean value (True or False)
```

```
my_string = "Hello World"

my_string.isalnum()         #check if all char are numbers
my_string.isalpha()         #check if all char in the string are alphabetic
my_string.isdigit()         #test if string contains digits
my_string.istitle()         #test if string contains title words
my_string.isupper()         #test if string contains upper case
my_string.islower()         #test if string contains lower case
my_string.isspace()         #test if string contains spaces
my_string.endswith('d')     #test if string endswith a d
my_string.startswith('H')   #test if string startswith H
```

Built-in String Methods

```
String methods are working on the string its called from! (using dot notation)
```

```
To manipulate strings, we can use some of Pythons built-in methods
```

```
string.upper()              #get all-letters in uppercase
string.lower()              #get all-letters in lowercase
string.capitalize()         #capitalize the first letter
string.title()              #capitalze the first letter of words
string.swapcase()           #converts uppercase and lowercase
string.strip()              #remove all white spaces
string.lstrip()             #removes whitespace from left
string.rstrip()             #removes whitespace from right
string.split()              #splitting words
string.split(',')           #split words by comma
string.count('l')           #count how many times l is in the string
string.find('Wo')           #find the word Wo in the string
string.index("Wo")          #find the letters Wo in the string
":".join(string)            #add a : between every char
" ".join(string)            #add a whitespace between every char
len(string)                 #find the length of the string
string.replace('World', 'Tomorrow') #replace string World with Tomorrow
```

Sources

https://github.com/adaptives/python-examples
http://en.wikibooks.org/wiki/Python_Programming