Python - Strings

- Strings are amongst the most popular types in Python. We can create them simply by enclosing characters in quotes. Python treats single quotes the same as double quotes.
- Creating strings is as simple as assigning a value to a variable. For example:

```
var1 = 'Hello World!'
var2 = "Python Programming"
```

Accessing Values in Strings:

- Python does not support a character type; these are treated as strings of length one, thus also considered a substring.

- To access substrings, use the square brackets for slicing along with the index or indices to obtain your substring:

- **Example:**

```
var  1 = 'Hello World!'
var2 = "Python Programming"
print "var1[0]: ", var1[0]
print "var2[1:5]: ", var2[1:5]
```

This will produce following result:

```
var1[0]: H
var2[1:5]: ytho
```

Updating Strings:

- You can "update" an existing string by (re)assigning a variable to another string. The new value can be related to its previous value or to a completely different string altogether.

- **Example:**

```
var1 = 'Hello World!'
print "Updated String :- ", var1[:6] + 'Python'
```

This will produce following result:

```
Updated String :- Hello Python
```

Escape Characters:

Backslash notation	Hexadecimal character	Description
\a	0x07	Bell or alert
\b	0x08	Backspace
\cx		Control-x
\C-x		Control-x
\e	0x1b	Escape
\f	0x0c	Formfeed
\M-\C-x		Meta-Control-x
\n	0x0a	Newline
\nnn		Octal notation, where n is in the range 0.7
\r	0x0d	Carriage return
\s	0x20	Space
\t	0x09	Tab
\v	0x0b	Vertical tab
\x		Character x
\xnn		Hexadecimal notation, where n is in the range 0.9, a.f, or A.F

String Special Operators: Assume string variable a holds 'Hello' and variable b holds 'Python' then:

Operator	Description	Example
+	Concatenation - Adds values on either side of the operator	a + b will give HelloPython
*	Repetition - Creates new strings, concatenating multiple copies of the same string	a*2 will give -HelloHello
[]	Slice - Gives the character from the given index	a[1] will give **e**
[:]	Range Slice - Gives the characters from the given range	a[1:4] will give **ell**
in	Membership - Returns true if a character exists in the given string	**H in a** will give 1
not in	Membership - Returns true if a character does not exist in the given string	**M not in a** will give 1
r/R	Raw String - Suppress actual meaning of Escape characters.	**print r'\n'** prints \n and **print R'\n'** prints \n
%	Format - Performs String formatting	See at next section

String Formatting Operator:

Format Symbol	Conversion
%c	character
%s	string conversion via str() prior to formatting
%i	signed decimal integer
%d	signed decimal integer
%u	unsigned decimal integer
%o	octal integer
%x	hexadecimal integer (lowercase letters)
%X	hexadecimal integer (UPPERcase letters)
%e	exponential notation (with lowercase 'e')
%E	exponential notation (with UPPERcase 'E')
%f	floating point real number
%g	the shorter of %f and %e
%G	the shorter of %f and %E

Other supported symbols and functionality are listed in the following table:

Symbol	Functionality
*	argument specifies width or precision
-	left justification
+	display the sign
<sp>	leave a blank space before a positive number
#	add the octal leading zero ('0') or hexadecimal leading '0x' or '0X', depending on whether 'x' or 'X' were used.
0	pad from left with zeros (instead of spaces)
%	'%%' leaves you with a single literal '%'
(var)	mapping variable (dictionary arguments)
m.n.	m is the minimum total width and n is the number of digits to display after the decimal point (if appl.)

Triple Quotes:

- Python's triple quotes comes to the rescue by allowing strings to span multiple lines, including verbatim NEWLINEs, TABs, and any other special characters.

- The syntax for triple quotes consists of three consecutive **single or double** quotes.

```
para_str = """this is a long string that is made up of
several lines and non-printable characters such as TAB
( \t ) and they will show up that way when displayed.
NEWLINEs within the string, whether explicitly given
like this within the brackets [ \n ], or just a NEWLINE
within the variable assignment will also show up. """

print para_str;
```

Raw String:

- Raw strings don't treat the backslash as a special character at all. Every character you put into a raw string stays the way you wrote it:

```
print 'C:\\nowhere'
```

This would print following result:

```
C:\nowhere
```

Now let's make use of raw string. We would put expression in **r'expression'** as follows:

```
print r'C:\\nowhere'
```

This would print following result:

```
C:\\nowhere
```

Unicode String:

- Normal strings in Python are stored internally as 8-bit ASCII, while Unicode strings are stored as 16-bit Unicode. This allows for a more varied set of characters, including special characters from most languages in the world. I'll restrict my treatment of Unicode strings to the following:

```
print u'Hello, world!'
```

This would print following result:

```
Hello, world!
```

Built-in String Methods:

1	[capitalize()](#)
	Capitalizes first letter of string
2	[center(width, fillchar)](#)
	Returns a space-padded string with the original string centered to a total of width columns
3	[count(str, beg= 0,end=len(string))](#)
	Counts how many times str occurs in string, or in a substring of string if starting index beg and ending index end are given
3	[decode(encoding='UTF-8',errors='strict')](#)
	Decodes the string using the codec registered for encoding. encoding defaults to the default string encoding.
4	[encode(encoding='UTF-8',errors='strict')](#)
	Returns encoded string version of string; on error, default is to raise a ValueError unless errors is given with 'ignore' or 'replace'.
5	[endswith(suffix, beg=0, end=len(string))](#)
	Determines if string or a substring of string (if starting index beg and ending index end are given) ends with suffix; Returns true if so, and false otherwise
6	[expandtabs(tabsize=8)](#)
	Expands tabs in string to multiple spaces; defaults to 8 spaces per tab if tabsize not provided

7	find(str, beg=0 end=len(string))
	Determine if str occurs in string, or in a substring of string if starting index beg and ending index end are given; returns index if found and -1 otherwise
8	index(str, beg=0, end=len(string))
	Same as find(), but raises an exception if str not found
9	isa1num()
	Returns true if string has at least 1 character and all characters are alphanumeric and false otherwise
10	isalpha()
	Returns true if string has at least 1 character and all characters are alphabetic and false otherwise
11	isdigit()
	Returns true if string contains only digits and false otherwise
12	islower()
	Returns true if string has at least 1 cased character and all cased characters are in lowercase and false otherwise
13	isnumeric()
	Returns true if a unicode string contains only numeric characters and false otherwise
14	isspace()
	Returns true if string contains only whitespace characters and false otherwise

15	istitle()
	Returns true if string is properly "titlecased" and false otherwise
16	isupper()
	Returns true if string has at least one cased character and all cased characters are in uppercase and false otherwise
17	join(seq)
	Merges (concatenates) the string representations of elements in sequence seq into a string, with separator string
18	len(string)
	Returns the length of the string
19	ljust(width[, fillchar])
	Returns a space-padded string with the original string left-justified to a total of width columns
20	lower()
	Converts all uppercase letters in string to lowercase
21	lstrip()
	Removes all leading whitespace in string
22	maketrans()
	Returns a translation table to be used in translate function.
23	max(str)
	Returns the max alphabetical character from the string str

24	min(str)
	Returns the min alphabetical character from the string str
25	replace(old, new [, max])
	Replaces all occurrences of old in string with new, or at most max occurrences if max given
26	rfind(str, beg=0,end=len(string))
	Same as find(), but search backwards in string
27	rindex(str, beg=0, end=len(string))
	Same as index(), but search backwards in string
28	rjust(width,[, fillchar])
	Returns a space-padded string with the original string right-justified to a total of width columns.
29	rstrip()
	Removes all trailing whitespace of string
30	split(str="", num=string.count(str))
	Splits string according to delimiter str (space if not provided) and returns list of substrings; split into at most num substrings if given
31	splitlines(num=string.count('\n'))
	Splits string at all (or num) NEWLINEs and returns a list of each line with NEWLINEs removed

32	startswith(str, beg=0,end=len(string))
	Determines if string or a substring of string (if starting index beg and ending index end are given) starts with substring str; Returns true if so, and false otherwise
33	strip([chars])
	Performs both lstrip() and rstrip() on string
34	swapcase()
	Inverts case for all letters in string
35	title()
	Returns "titlecased" version of string, that is, all words begin with uppercase, and the rest are lowercase
36	translate(table, deletechars="")
	Translates string according to translation table str(256 chars), removing those in the del string
37	upper()
	Converts lowercase letters in string to uppercase
38	zfill (width)
	Returns original string leftpadded with zeros to a total of width characters; intended for numbers, zfill() retains any sign given (less one zero)
39	isdecimal()
	Returns true if a unicode string contains only decimal characters and false otherwise