We can create strings in python 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 −

```
str1 =  ‘Hello Python!’
str2 = ″I love coding″
```

You can also write multi line strings by using triple quotes as follows.

```
str2 = ″″″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 it is displayed.
″″″
```

Accessing Values in Strings

Unlike C or C++, python does not support a character type these are treated as strings of length one.

To access substring of a string, we use the square brackets for slicing along with the index or indices to obtain your substring. For example −

```
str1 =  ‘Hello Python!’
str2 = ″I love coding″

print(″str1[0]: ″, str1[0]) # str1[0]: H
print(″str2[1:5]: ″, str2[1:5]) # str2[1:5]:   lov
```

Updating Strings

You can "update" an existing string by (re)assigning a variable to another string as follows.

```
str1 =  ‘Hello Python!’
str1 = ″Hello coding!″
```

String Formatting Operator

One of Python's coolest features is the string format operator %. This operator is unique to strings and makes up for the pack of having functions from C's printf() family. Following is a simple example −

```
print ″My name is %s and I am %d years old!″ % (’Ellie’, 25)
```

When the above code is executed, it produces the following result −

```
My name is Ellie and I am 25 years old!
```

Here is the list of complete set of symbols which can be used along with % −

Format Symbol	Conversion
%c	character
%s	string conversion via str() prior to formatting
%i	signed decimal integer
%d	signed decimal 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
%u	unsigned decimal integer
%o	octal integer

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

Symbol	Functionality
*	argument specifies width or precision
-	left justification
+	display the sign
#	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 '%'
m.n.	m is the minimum total width and n is the number of digits to display after the decimal point (if appl.)
<sp>	leave a blank space before a positive number
(var)	mapping variable (dictionary arguments)