**Using "join" function for the string**

The join function is a more flexible way for concatenating string. With join function, you can add any character into the string.

For example, if you want to add a colon (:) after every character in the string "Python" you can use the following code.

print(":".join("Python"))

Output

P:y:t:h:o:n

**Reversing String**

By using the reverse function, you can reverse the string. For example, if we have string "12345" and then if you apply the code for the reverse function as shown below.

string="12345"

print(''.join(reversed(string)))

Output

54321

**Split Strings**

Split strings is another function that can be applied in Python let see for string "guru99 career guru99". First here we will split the string by using the command word.split and get the result.

word="guru99 career guru99"

print(word.split(' '))

Output

['guru99', 'career', 'guru99']

To understand this better we will see one more example of split, instead of space (' ') we will replace it with ('r') and it will split the string wherever 'r' is mentioned in the string

word="guru99 career guru99"

print(word.split('r'))

Output

['gu', 'u99 ca', 'ee', ' gu', 'u99']

**Important Note:**

**In Python, Strings are immutable.**

**Consider the following code**

x = "Guru99"

x.replace("Guru99","Python")

print(x)

Output

Guru99

will still return Guru99. This is because x.replace("Guru99","Python") returns **a copy of X with replacements made**

**You will need to use the following code to observe changes**

x = "Guru99"

x = x.replace("Guru99","Python")

print(x)

Output

Python

Above codes are Python 3 examples, If you want to run in Python 2 please consider following code.

**Python 2 Example**

#Accessing Values in Strings

var1 = "Guru99!"

var2 = "Software Testing"

print "var1[0]:",var1[0]

print "var2[1:5]:",var2[1:5]

#Some more examples

x = "Hello World!"

print x[:6]

print x[0:6] + "Guru99"

#Python String replace() Method

oldstring = 'I like Guru99'

newstring = oldstring.replace('like', 'love')

print newstring

#Changing upper and lower case strings

string="python at guru99"

print string.upper()

string="python at guru99"

print string.capitalize()

string="PYTHON AT GURU99"

print string.lower()

#Using "join" function for the string

print":".join("Python")

#Reversing String

string="12345"

print''.join(reversed(string))

#Split Strings

word="guru99 career guru99"

print word.split(' ')

word="guru99 career guru99"

print word.split('r')

x = "Guru99"

x.replace("Guru99","Python")

print x

x = "Guru99"

x = x.replace("Guru99","Python")

print x

Output

var1[0]: G

var2[1:5]: oftw

Hello

Hello Guru99

I love Guru99

PYTHON AT GURU99

Python at guru99

python at guru99

P:y:t:h:o:n

54321

['guru99', 'career', 'guru99']

['gu', 'u99 ca', 'ee', ' gu', 'u99']

Guru99

Python

Python has introduced a .format function which does way with using the cumbersome %d and so on for string formatting.

## Summary:

Since Python is an object-oriented programming language, many functions can be applied to Python objects. A notable feature of Python is its indenting source statements to make the code easier to read.

* Accessing values through slicing - square brackets are used for slicing along with the index or indices to obtain a substring.
  + In slicing, if range is declared [1:5], it can actually fetch the value from range [1:4]
* You can update Python String by re-assigning a variable to another string
* Method replace() returns a copy of the string in which the occurrence of old is replaced with new.
  + Syntax for method replace: oldstring.replace("value to change","value to be replaced")
* String operators like [], [ : ], in, Not in, etc. can be applied to concatenate the string, fetching or inserting specific characters into the string, or to check whether certain character exist in the string
* Other string operations include
  + Changing upper and lower case
  + Join function to glue any character into the string
  + Reversing string
  + Split string