Python (Strings)

Python Strings is all about the working with strings and manipulation of Strings, using String Operators and string methods and Functions. First, let us understand that how do we declare the strings in python programming language. We can declar and print the strings by placing them in single Quotes ('..'), Double Quotes (".."), and using the print function too. Python Strings are Immutable (An Object with a fixed value).

Using the Strings in single quotes ('...')

```
>>> 'hello world'

'hello world'
```

Below command will give an error.

```
>>> 'let's start'   # this will give us an error

 File ",stdin>, line 1

  'let's do it'

        ^

SyntaxError: invalid syntax
```

**To overcome that we must use escape character **

```
>>> 'let\'start'

"let's start"
```

Using the Strings in Double quotes ("...")

```
>>> "let's start"      # using double quotes to avoid escape character

"let's start"
```

Using the Strings in Print() function

```
>>>print("let's start")    # we hav enclose the strings in double quotation inside print
funtion

let's start
```

Using the 3 double quotes start and end of the string allows us to print the data including spaces and newlines.

```
>>>print("""let's
...start
...Now""")
let's
start
now
```

String Concatenation:

Multiple Strings can be concatenated using (+) symbol. Let us see the example of concatenating the strings.

Example:

```
>>> x="hello"
>>> y="world"
>>>x+y
'helloworld'
```

String Repetition:

String repetition can be performed by using the (*) symbol. Let us see the example of repetition of strings.

Example:

```
>>> 3*"hello"
'hellohellohello'
```

Strings are indexed with each character in a memory location when assigned to a variable. The indexed number starts from zero '0' from first character till he end of the string. Whereas, reverse indexing starts with '-1' from right to left until the starting character. Let us try few examples of retrieving the characters from a word PYTHON in either ways.

Example:

```
>>> x="P Y T H O N"    # Word python is written without spaces
     0 1 2 3 4 5
    -6-5-4-3-2-1
>>>x[3]
'H'
>>>x[-5]
'Y'
>>>x[:4]               # Starting from first character, 4th position excluded
'PYTH'
>>>x[:-4]              # Starting from fourth character from right, 4th position excluded
'PY'
>>>X[0:]               # Starting from first character till end
'PYTHON'
>>>x[-6:]              # Starting from -6th position until start ie., -1 postion
'PYTHON'
```

String Methods in Python:

Python String Methods	Description
capitalize()	Returns the String with first Character as Capital Letter
casefold()	Returns a casefolded copy
center(width[, fillchar])	This will pads the string with a character specified
count(sub[, start[, end]])	Returns the number of occurances of substring in string
encode(encoding="utf-8", errors="strict")	returns an encoded string
endswith(suffix[, start[, end]])	Check the string if it ends with the sp cified

expandtabs(tabsize=8)	Replace the tab with space
find(sub[, start[, end]])	returns the highest index
format(*args, **kwargs)	formats the string
format_map(mapping)	formats the string except the mapping is directly used
index(sub[, start[, end]])	returns the index of substring
isalnum()	checks for alphanumeric Char
isalpha()	Checks if all characters are Alphabets
isdecimal()	Checks for decimal characters
isdigit()	Checks for digit char
isidentifier()	checks for valid Identifier
islower()	checks for lowercase of all alphabets in string
isnumeric()	Checks for Numeric Char
isprintable()	Checks for Printable Char
isspace()	Checks for Whitespace Characters
istitle()	Returns true if the string is titlecased
isupper()	Checks if all characters are Uppercase
join(iterable)	returns concatenated string
ljust(width[, fillchar])	returns left-justified string
lower()	returns lowercased string
lstrip([chars])	Removes Leading Characters
partition(sep)	returns a tuple
replace(old, new[, count])	replaces the substring
rfind(sub[, start[, end]])	Returns the Highest Index
rindex(sub[, start[, end]])	Returns Highest Index but raises when substring is not found
rjust(width[, fillchar])	Returns the string right justified
rpartition(sep)	Returns a tuple

rsplit(sep=None, maxsplit=-1)	Splits String From Right
rstrip([chars])	Removes Trailing Characters
split(sep=None, maxsplit=-1)	Splits String from Left
splitlines([keepends])	Splits String at Lines
startswith(prefix[, start[, end]])	Checks if String Starts with the Specified String
strip([chars])	Removes Both Leading and Trailing Characters
swapcase()	swap uppercase characters to lowe case and vice versa
title()	Returns a Title Cased String
translate(table)	returns mapped charactered string
upper()	returns uppercased string
zfill(width)	Returns a Copy of The String Padded With Zeros