Introduction to Python String

- A string is just a sequence of characters.
- It is among the most popular data types in Python.
- It can be created simply by enclosing characters in quotes.
- Python provides a rich set of operators, functions, and methods for working with strings which helps to access and extract portions of strings and also manipulate and modify string data.

What is a string in python?

- Technically speaking, an immutable sequence of data is known as a string in Python.
- In simple words, as discussed in the case of a crossword, a Python string is nothing but an array of characters, but a computer does not understand characters.
- It only understands the language of 0's and 1's.
- So these characters are converted to a number so that the computer can understand it clearly.
- This conversion is known as encoding. ASCII and Unicode are some of the popular encodings used.
- To put it simply, Python strings are a sequence of Unicode characters.

Creating a String in Python

•A string in Python can be easily created using single, double or even triple quotes. The characters that form the string are enclosed within any of these quotes.

•Note- Triple quotes are generally used when we are working with multiline strings in Python and docstring in Python.

Reassigning Strings in Python

•Since strings in Python are immutable, we can update the values of strings simply by reassigning these strings with new values.

The strings cannot be partially replaced, so it is always replaced completely with a new string.

•Let's see how that's done –

```
message="Welcome to PythonLife“

print(message)

message="Hello World" #reassigning a new string

print(message)
```

•The output of the above snippet will be –

```
Welcome to PythonLife

Hello World
```

From the output, it is evident that the string variable message was reassigned with a new string, which gets printed to the console in the last line.

How to access characters in a python string?

As we know, a string is made up of a sequence of characters. Thus, in Python, we have some techniques to reference or index a particular character in a string.

The two primary ways of accessing characters of a string in Python are-

1. Indexing –

The length of a string can be easily calculated using the len() function . The length of a string provides us with a range of the string. In python, we can perform indexing on any string as long as it is within the range. The index is always referred as to be an integer. Using any other data type would give rise to a TypeError.

Indexing can either be positive or negative. Positive indexing starts from 0 and starts indexing from the beginning of the string. On the other hand, negative indexing starts from -1, and indexing starts from the end of the string.

2. Slicing –

Unlike indexing, slicing returns a range of characters using the slicing syntax.

The syntax looks like –

‘start[start_index: end_index] ‘

While performing slicing operations, one thing to remember is that the start index value is always included while that of the last index is always excluded.

Slicing can also be negative or positive, just like indexing in Python!

How to delete a string?

•We have established that strings are immutable. In simple words, it means that once a string is assigned, we cannot make any changes to it. We cannot remove or delete any character of the string.

•What we can do is delete the Python string entirely. We can do so by using the del keyword.

Example Program of String Operators in Python

```
message1="Hello World!"
message2="Welcome to PythonLife"
print(message1+message2) # + operator
print(message1*3) # * operator
print(message1[6]) # [] operator
print(message2[0:7]) # [:] operator
str1="Hello"
if str1 in message1:# in operator
        print("It is present!")
if str1 not in message2:# not in operator
        print("It is not present!")
```

Output –

Hello World!Welcome to PythonLife

Hello World!Hello World!Hello World!

w

Welcome

It is present!

It is not present!

String Manipulation

To manipulate strings, we can use some of Python's built-in methods.

Creation

word = "Hello World"

>>> print word

Hello World

Accessing

Use [ ] to access characters in a string

word = "Hello World"

letter=word[0]

>>> print letter

H

Length

word = "Hello World"

>>> len(word)

11

Finding

word = "Hello World">>> print word.count('l') # count how many times l is in the string

3

>>> print word.find("H") # find the word H in the string

0

>>> print word.index("World") # find the letters World in the string

6

Count

s = "Count, the number of spaces"

>>> print s.count(' ')

8

Slicing

Use [# : #] to get set of letter

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

word = "Hello World"

print word[0] #get one char of the word

print word[0:1] #get one char of the word (same as above)

print word[0:3] #get the first three char

print word[:3] #get the first three char

print word[-3:] #get the last three char

print word[3:] #get all but the three first char

print word[:-3] #get all but the three last character

word = "Hello World"

word[start:end] # items start through end-1

word[start:] # items start through the rest of the list

word[:end] # items from the beginning through end-1

word[:] # a copy of the whole list

Split Strings

word = "Hello World"

>>> word.split(' ') # Split on whitespace

['Hello', 'World']

Startswith / Endswith

word = "hello world"

>>> word.startswith("H")

True

>>> word.endswith("d")

True

>>> word.endswith("w")

False

Repeat Strings

print "."* 10 # prints ten dots

>>> print "." * 10

....

Replacing

word = "Hello World"

>>> word.replace("Hello", "Goodbye")

'Goodbye World'

Changing Upper and Lower Case Strings

string = "Hello World"

>>> print string.upper()

HELLO WORLD

>>> print string.lower()

hello world

>>> print string.title()

Hello World

>>> print string.capitalize()

Hello world

>>> print string.swapcase()

hELLO wORLD

Reversing

string = "Hello World"

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

d l r o W o l l e H

Strip

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

word = "Hello World"
Strip off newline characters from end of the string

```
>>> print word.strip('

')
```

Hello World

strip() #removes from both ends

lstrip() #removes leading characters (Left-strip)

rstrip() #removes trailing characters (Right-strip)

```
>>> word = " xyz "

>>> print word
xyz

>>> print word.strip()
xyz

>>> print word.lstrip()
xyz

>>> print word.rstrip()
```

xyz

Concatenation

To concatenate strings in Python use the “+” operator.

"Hello " + "World" # = "Hello World"

"Hello " + "World" + "!"# = "Hello World!"

Join

>>> print ":".join(word) # #add a : between every char

H:e:l:l:o: :W:o:r:l:d

>>> print " ".join(word) # add a whitespace between every char

H e l l o W o r l d

Extra information:

lower()	Converts a string into lower case
lstrip()	Returns a left trim version of the string
replace()	Returns a string where a specified value is replaced with a specified value

rstrip() Returns a right trim version of the string

split() Splits the string at the specified separator, and returns a list

startswith() Returns true if the string starts with the specified value

strip() Returns a trimmed version of the string

title() Converts the first character of each word to upper case

upper() Converts a string into upper case

count() Returns the number of times a specified value occurs in a string

endswith() Returns true if the string ends with the specified value

find() Searches the string for a specified value and returns the position of where it was found

format() Formats specified values in a string

index() Searches the string for a specified value and returns the position of where it was found

isalnum() Returns True if all characters in the string are alphanumeric

isalpha() Returns True if all characters in the string are in the alphabet

join() Converts the elements of an iterable into a string