String Data Type

The most commonly used object in any project and in any programming language is String only. Hence we should aware complete information about String data type.

What is String?

Any sequence of characters within either single quotes or double quotes is considered as a String.

How to access characters of a String:

We can access characters of a string by using the following ways.

1. By using index
2. By using slice operator

Behaviour of slice operator:

s[bEgin:end:step]

step value can be either +ve or –ve

if +ve then it should be forward direction(left to right) and we have to consider bEgin to end-1

if -ve then it should be backward direction(right to left) and we have to consider bEgin to end+1

***Note:**
In the backward direction if end value is -1 then result is always empty.
In the forward direction if end value is 0 then result is always empty.

In forward direction:

default value for bEgin: 0
default value for end: length of string
default value for step: +1

In backward direction:

default value for bEgin: -1
default value for end: -(length of string+1)

Note: Either forward or backward direction, we can take both +ve and -ve values for bEgin and end index.

Mathematical Operators for String:

We can apply the following mathematical operators for Strings.

1. + operator for concatenation
2. * operator for repetition

len() in-built function:

We can use len() function to find the number of characters present in the string.

Removing spaces from the string:

We can use the following 3 methods

1. rstrip()===>To remove spaces at right hand side
2. lstrip()===>To remove spaces at left hand side
3. strip() ==>To remove spaces both sides

Finding Substrings:

We can use the following 4 methods

For forward direction:

find()
index()

For backward direction:

rfind()
rindex()

1. find():

s.find(substring)

Returns index of first occurrence of the given substring. If it is not available then we will get -1

index() method:

index() method is exactly same as find() method except that if the specified substring is not available then we will get ValueError.

Counting substring in the given String:

We can find the number of occurrences of substring present in the given string by using count() method.

1. s.count(substring) ==> It will search through out the string
2. s.count(substring, bEgin, end) ===> It will search from bEgin index to end-1 index

Replacing a string with another string:

s.replace(oldstring,newstring)

inside s, every occurrence of oldstring will be replaced with newstring.

Splitting of Strings:

We can split the given string according to specified seperator by using split() method.

l=s.split(seperator)

The default seperator is space. The return type of split() method is List

Joining of Strings:

We can join a group of strings(list or tuple) wrt the given seperator.

s=seperator.join(group of strings)

Changing case of a String:

We can change case of a string by using the following 4 methods.

1. upper()===>To convert all characters to upper case
2. lower() ===>To convert all characters to lower case
3. swapcase()===>converts all lower case characters to upper case and all upper case characters to lower case
4. title() ===>To convert all character to title case. i.e first character in every word should be upper case and all remaining characters should be in lower case.
5. capitalize() ==>Only first character will be converted to upper case and all remaining characters can be converted to lower case

Checking starting and ending part of the string:

Python contains the following methods for this purpose

1. s.startswith(substring)
2. s.endswith(substring)

To check type of characters present in a string:

Python contains the following methods for this purpose.

1) isalnum(): Returns True if all characters are alphanumeric(a to z , A to Z ,0 to9)
2) isalpha(): Returns True if all characters are only alphabet symbols(a to z,A to Z)
3) isdigit(): Returns True if all characters are digits only(0 to 9)
4) islower(): Returns True if all characters are lower case alphabet symbols
5) isupper(): Returns True if all characters are upper case aplhabet symbols
6) istitle(): Returns True if string is in title case
7) isspace(): Returns True if string contains only spaces

Formatting the Strings:

We can format the strings with variable values by using replacement operator {} and format() method.

The main objective of format() method to format string into meaningful output form.

Case- 1: Basic Formatting for default, positional and keyword arguments

Case-2: Formatting Numbers

d--->Decimal IntEger
f----->Fixed point number(float).The default precision is 6
b-->Binary format
o--->Octal Format
x-->Hexa Decimal Format(Lower case)
X-->Hexa Decimal Format(Upper case)

Note:
{:08.3f}

Total positions should be minimum 8.
After decimal point exactly 3 digits are allowed.If it is less then 0s will be placed in the last positions
If total number is < 8 positions then 0 will be placed in MSBs
If total number is >8 positions then all intEgral digits will be considered.
The extra digits we can take only 0

Note: For numbers default alignment is Right Alignment(>)

Note: We can represent only int values in binary, octal and hexadecimal and it is not possible for float values.

Note:
{:5d} It takes an intEger argument and assigns a minimum width of 5.
{:8.3f} It takes a float argument and assigns a minimum width of 8 including "." and after decimal point excatly 3 digits are allowed with round operation if required
{:05d} The blank places can be filled with 0. In this place only 0 allowed.

Case-3: Number formatting for signed numbers

While displaying positive numbers,if we want to include + then we have to write

{:+d} and {:+f}

Using plus for -ve numbers there is no use and for -ve numbers - sign will come automatically.

Case-4: Number formatting with alignment

<,>,^ and = are used for alignment
<==>Left Alignment to the remaining space
^===>Center alignment to the remaining space
>===> Right alignment to the remaining space
= ===>Forces the signed(+) (-) to the left most position

Note: Default Alignment for numbers is Right Alignment.

Case-5: String formatting with format()

Similar to numbers, we can format String values also with format() method.

s.format(string)

Note: For numbers default alignment is right where as for strings default alignment is left

Case-6: Truncating Strings with format() method

Case-7: Formatting dictionary members using format()