Chap-5: Strings

- Concept of String:
- String Operations
- String are Immutable
- String formatting Operator
- Built-in String Methods and Functions
- Slice Operation
- ord() and chr() functions
- in() and not in() operators
- Comparing Strings
- Iterating String
- The String Module
- Practice Programs.

Concept of String:

String are collection characters (numbers, alphabate, symbols).

In string multiple characters are present together without any spacing in between.*

String are usually enclosed within sigle quote or double quote ' ', " "

e.g. String_1 = 'India'
Str_2 = 'Welcome'
Str_3 = "Abc123"
Str_4 = "123#@$"
str_5 = "Xyz#@$"
Str_6 = "Abc123#@$"

String can be expressed in two ways,

1] Single line String:

It is written in single line only, it is used to represent heading or introduction.
It is written within ' ' or " "

e.g
str_1 = 'hello'
Str_2 = "World_1@"
str_3 = "123"
Str_4 = '124@abd'
str_5 = ' '
str_6 = 'a'

2] Multiline String:

It can be written in more than one line, it is written within """ """ or '''' ''''

e.g
```
str_1 = """India is my country.
         I love my country."""
str_2 = '''' Hello
         This is multi-line
         String
''''
```

1] Creating or Declaring a String:

In Python, we can create a string by following syntax;

```
String_name = "Characters"
```

where String_name - valid variable
Characters - alphabate, number, symbol.

e.g.
```
str_1 = "Happy"
str_2 = 'Happy'
```

2] Accessing elements of list:

As string is a collection of character, each character present in collection can be accessed. By using index value characters can be accessed.

String is converted in the form of tuple of Character.

a] Using Positive Index: char. present at appropriate index value.

Syntax: string_name [index]

e.g.
```
str = "Hello You"
a = str[0]
print(a)
print[str[1], str[3])
print(str[3&+1])
```

b) Using Negative Index: from the back-side of a string.

string_name[-ve Index]

e.g.
```
str = "Lucky Vishwakarma"
a = str[-1]
print(a)
print(str[-3])
print(str[-9], str[-6-1])
```

c) Using slice (:) operator:

using slice we can access mutiple characters of string (i.e from strating to ending index)

string_name [start_index, stop-index, step-size]

where step-size is fixed increment or decrement.

```
str = "Lucky Vishwakarma"
a = str[1: 9]
print(a)
print[-9: -3]
print[-1: -11: -1]
```

3] Traversing a String:

Traversing or Iterating a string is the method of printing all the element one-by-one.

a) Using for loop():

Syntax: for variable in string_name:
print(variable)

e.g.
```
string = "hello-you"
for i in string:
    print(i)
```

b) Using range function:
using range function we can iterate over string through index values.

Syntax:
```
for i in range(leng(string)):
    print(string[i])
i=0
while i < leng(string):
    print(string[i])
    i = i+1
```

e.g.
```
str = "India" ; i = 0
while i < len(str):
    print(str[i])
    i = i + 1

for i in range(len(str)):
    print(str[i])
```

c) Using Enumarate function:
using enumerate function we can print both index and value of a string.

Syntax:
```
str = "Good greeting"
for i in enumerate(str):
    print(i)
```

- As Strings are the immutable data-type we cannot change the or modify the string.
- Strings does not support deletion of elements.
i.e. pop(), del() & remove function cant be used with string.

4] String Operations:

a) Concatenation: it is a method which allows to join two or more string to form a single string.

Syntax: new-string = str_1 + str_2

e.g. S_1 = 'Hello' S_2 = 'World!'
∴ print (S_1 + S_2) --> 'Hello World'.

```
S1 = 'What's'
S2 = 'Up?'
S3 = S1+S2
print(S3)
```

b] Repeatition: it is a method of repeating a string multiple times.

Syntax:
```
str = 'Hello'
print(#Str * 2)
```

e.g.
```
str1 = 'lucky'
print(str1 * 3)
print('nikhil' * 2)
```

5] Different Built-in Methods and functions on String:

1) count(): it return the count of particular element present in string.

Syntax: string-name.count('element')

e.g.
```
string = 'Lucky'
a = string.count('L')
print(a)
print('Nikhil'.count('i'))
```

2) index(): it return the index of particular element present in string.

Syntax: str-name.index(element)

e.g.
```
str = 'India'
a = str.index('I')
print ('str'.index('t'))
```

3) len(): it return the length of string i.e. total no. of element of string.

Syntax: len (str-name)

e.g.
```
print (lent'India')) ; b = 'Ryuga'
a = len (b)
print (a)
```

4) max(): it return the maximum of element of string, it uses ASCII value for compa-rision.

Syntax: max ('string-name')

e.g.
```
a = 'India'
print (max (a))
print (max('abcdefghijklmnopqrst')
```

5) min(): it return the minimun of element of string, it uses ASCII value for comparision.

Syntax: min ('string-name')

e.g.
```
a = 'India'
print (min (a))
print (min ('Hello You'))
```

* string does not support sum(), append(), del() extend(), sort(), pop() insert(), clear() methods as they are immutable.

6) Sorted(): it return the string in sorted form it compare the element through their ASCII value for sorting.

Syntax: sorted(string_name)

e.g.
```
a='abcdefghijklmnopqrstuvwxyz'
print(sorted(a))
print(sorted('abcxyz,123'))
```

7) all(): it return true value when all element are true or when string is empty.

Syntax: all(string_name)

e.g.
```
a='Lucky'
print(all(a))
print(all('000'))
```

8) any(): it return true value if any of element is true or string is empty.

Syntax: any(string_name)

e.g.
```
a='Nikhil'
print(any(a))
print(any('Lucky'))
```

9) str(): this function is used to create or convert other data-type to string.

Syntax : str(other_data)

e.g.
```
a = 51.59
print (str(a))
print (str(57.92))
```

10) format(): formatting of string is method of displaying the string by using different values and variables.
It is done by 3 ways.

a) using % operator: %d - for integer
%s - for string
%f - for float
% c - for character

Syntax: "string %s string %d" % ('Lucky', 999121)

e.g. print ('There are %d type of datastructure' % 2)
```
a = '''Hello %s !
How are You.
Your score is %d''' % ('Lucky', 123)
```

b) using .format(): using this we can assign value to particular position of string.

Syntax: "String {0} and int {1} ...".format ('Hii', 1)

e.g. print("'{0} is my country.
All {1} are my Brother & Sisters. format
('India', 'Indians'))

a = "{2} x {0} = {1}". format (5, 10, 2)
print (a) >> 2 x 5 = 10

c) f" ": it allows displaying the string using the variables.
It is the simplest form used oftenly.

Syntax: f" String {var1} string {var2} ... "
a = India ; b = Indian
e.g. print (f"'{a} is my country.
All {b} are my Brother & Sisters.'")
a = 2 , b = 10 , c = 5
m = f" {a} x {c} = {10}"
print (m)

11) upper() - it convert all character of string to uppercase

Syntax - string-name.upper()

e.g. str = 'i love my country'
print (str.upper())
print ("lucky".upper())

1 isupper() - it return true value when all character of string are in upper case

print ('LUCKY'.isupper())
>> true.

12) lower() - it will convert all character of string into lowercase.

Syntax: string_name.lower()

e.g.
```
str = "INDIA IS MY COUNTRY"
print (str.lower())
print ("LUCKY".lower())
```

islower() - it will return true value when all element of string are in lowercase.

```
print ("i love my country".islower())
```

13) replace() - it replace a string with specified string.

Syntax - string_name.replace()

e.g.
```
str = "I Love My Country"
print (str.replace ("Country", "India")
print ('Hello World'.replace ('World', 'You'))
```

14) strip() - it will remove all the rightmost and leftmost spaces of string.

Syntax - str.strip()

e.g.
```
Str = "  ... India ---  "
print (str.strip())
print ('--- Lucky'.strip())
```

lstrip() - remove leftmost spaces.
rstrip() - remove rightmost spaces.

15) split() - this function will separate the string by specified operator/separator and return it in list form.

Syntax - str.split('separator')

e.g.
```
str = 'India is my country'
print(str.split(' '))
print('India-is-greate'.splite('-'))
```

16) title() - it will convert first letter of every word present in string to Upper case.

Syntax - string.name.title()

e.g.
```
str = 'india is my country'
print(str.title())
print('i love india'.title())
```

17) ord() - this function return the ASCII code of a character

Syntax -
```
str_chr = "A"
print(ord(str_chr))
print(ord('a'))
```

18) chr() - this function return the character of an ASCII code

Syntax - char(ASCII code)

Example -
```
str = 90
print(chr(90))
print(char(str))
```

String Module:

The String module contains multiple functions and constants to manipulate over string. For using, we need to import it.

Functions:

1) capwords():

It convert first letter of each letter of a string to capital.

Sytanx: string.capswords(str_name)

Example:
```
str_1 = 'India is great'
print(string.capswords(str_1))

print(string.capswords('lucky nikhil'))
```

2) upper() & lower():

upper function is used to convert all the characters of string to uppercase

lower function is used to convert all the characters of string to lower case.

Syntax - str.upper/lower(str_name)

Example -
```
str_1 = 'hello you!'
Str_2 = 'HELLO YOU!'
print(str.upper(str_1))
print(str.lower(str_2))
print(str.upper("lucky"))
print(str.lower("LUCKY")
```

3) maketrans() - this function is used to transform or replace one character with other character in a string.

Syntax - maketrans(from_ch, to_ch)

Example -
```
string1="benene"
str1= string1.maketrans('e','a')
print(str1)
print("Luck□".maketrans('□';'y')
```

(*) Constants :
String module contain various constants that contain some values.

To use them we need to import string module.

Constant	Values
• string.ascletters	'abcdefghijklmnopqrstuvwxyz ABCDEFGHIJKLMNOPQRSTU VWXYZ'
• String.lowercase	'abcdefghijklmnopqrstuvwxyz'
• String.uppercase	'ABCDEFGHIJKLMNOPQRSTUVW XYZ'
• String.digit	'0123456789'
• String.octal	'01234567'
• String.hexdigit	'0123456789ABCDEFabcdef'

• string.punctuations: '!"#$%&\'()*+,-./:;<=>?
@[\\]^_`{|}~'

• string.whitespace : '\t\n\x0b\x0c\r'