Rohini Mittal Unit-03

Strings

- String is a sequence which is made up of one or more characters.
- Characters may be letter, digit, whitespace or any other symbol.
- String can be created by enclosing one or more characters in single, double or triple quote.

Example -

```
str1 = "Hello World!"
str2 = """Hello World!"""
str3 = 'Hello World!'
str4 = """""" Hello World!""""""
```

- Values stored in str2 and str4 can be extended to multiple lines. i.e..

```
str2 = """Hello World!
        Welcome to the world of Python"""

str4 = """""" Hello World!
        Welcome to the world of Python""""""
```

Accessing characters in a String

In Python, individual characters of a string can be accessed by using indexing.

Indexing of characters in string 'Hello World!!'

Positive Indices	0	1	2	3	4	5	6	7	8	9	10	11	12
String	H	e	l	l	o		W	o	r	l	d	!	!
Negative Indices	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

- characters are accessed either by +ve index or by -ve index.
- +ve index means string is traversed from left to right.
- -ve index means string is traversed from right to left.

Ex:
```
str1 = 'Hello World!!'
n = len(str1)
print(n)
print(str1[0])
print(str1[-n])
print(str1[n-1])
```

O/P
```
13
'H'
'H'
'!'
```

Note:
- while accessing an index out of the range will cause IndexError

Example
```
str1 = "Hello"
print(str1[8])
```
O/P
IndexError: string index out of range

- The index can also be an expression including variables & operators but expression must evaluate to an integer.

Example
```
str1 = "Hello"
print(str1[1.5])
```
TypeError: string indices must be integer

String is Immutable

A string is an immutable data type. It means that the content of the string can not be changed after it has been created.

Example

```
str1 = "Hello World!"
str1[1] = 'a'
print(str1)
```

O/P TypeError: 'str' object does not support item assignment

String Operations

Python allows certain operations on string data type which are as following-

(1) concatenation
(2) Repetition
(3) Membership
(4) Slicing

(1) concatenation

- To concatenate means to join.
- Python allows us to join two strings using concatenation operator plus which is denoted by +.

Example :

```
str1 = "Hello"
str2 = "World!"
str3 = str1 + str2
print(str3)
```

O/P HelloWorld!

2. **Repetition**

Python allows us to repeat the given string using repetition operator which is denoted by *.

Example-

```
str1 = "Hello"
str2 = str1 * 2
print(str2)
```

o/p "HelloHello"

3. **Membership**

Python has two membership operators - in
- not in

- in operator takes two string & returns True if first string appears as the substring in the second string otherwise returns False.

Example

```
str1 = "Hello"
if ('H' in str1):
    print("true")
else:
    print("false")
```

o/p true

- not in operator also takes two strings and returns True if the first string does not appear as a substring in the second string otherwise returns False.

Example

```
str1 = "Hello"
if ("Hello" not in str1):
    print("true block")
else:
    print("false block")
```

o/p false block

4. Slicing

To access range of characters in the string, the method of slicing is used. Slicing in a string is done by a slicing operator (:)

Syntax <string-name> [start : stop : step]

Example

```
str1 = "Hello World!"
print( str1[1:5])          o/p 'ello'
print ( str1[7:9])         o/p 'or'
print ( str1[3:20])        o/p 'lo World!'
print ( str1[7:2])         ← it will print empty string
```

(i) When start is not given

```
print( str1[ :5])          o/p 'Hello'
```

(It will be taken as 0)

(ii) When stop is not given

```
print( str1[6:])           o/p 'World!'
```

(It will goes till the length of string)

(iii) When step is given

```
print( str1[0:10:2])       o/p 'HlOWs'
```

(iv) When negative index is used

```
print( str1[-6:-1])        o/p 'World'
```

Note:
If we skip both start & stop. But give step as -1 then it will print the reverse order of string

```
print( str1[ : :-1])       o/p '!dlroW olleH'
```

Traversing a String

(1) Using for loop

```
str1 = "Hello World!"
for ch in str1 :
    print( ch , end = " ")
```

O/P Hello World!

(2) Using while loop

```
str1 = "Hello World!"
index = 0
while index < len(str1):
    print( str1[index] , end = " ")
    index += 1
```

O/P Hello World!

String Methods and Built-in Functions

Python has several built-in functions that allow us to work with strings. Some of the commonly used built-in functions for string manipulation are as follows :

Built-in functions for string manipulations

Method	Description	Example

len()	Returns the length of the given string	`>>> str1 = 'Hello World!'`
		`>>> len(str1)`
		`12`

title()	Returns the string with first letter of every word in the string in uppercase and rest in lowercase	`>>> str1 = 'hello WORLD!'`
		`>>> str1.title()`
		`'Hello World!'`

lower()	Returns the string with all uppercase letters converted to lowercase	`>>> str1 = 'hello WORLD!'`
		`>>> str1.lower()`
		`'hello world!'`

upper()	Returns the string with all lowercase letters converted to uppercase	`>>> str1 = 'hello WORLD!'`
		`>>> str1.upper()`
		`'HELLO WORLD!'`

count(str, start, end)	Returns number of times substring `str` occurs in the given string. If we do not give start index and end index then searching starts from index 0 and ends at length of the string	`>>> str1 = 'Hello World!`
		`Hello Hello'`
		`>>>`
		`str1.count('Hello',12,25) 2`
		`>>> str1.count('Hello')`
		`3`

find(str,start, end)	Returns the first occurrence of index of substring `str` occurring in the given string. If we do not give start and end then searching starts from index 0 and ends at length of the string. If the substring is not present in the given string, then the function returns -1	`>>> str1 = 'Hello World! Hello`
		`Hello' >>>`
		`str1.find('Hello',10,20)`
		`13`
		`>>> str1.find('Hello',15,25)`
		`19`
		`>>> str1.find('Hello')`
		`0`
		`>>> str1.find('Hee')`
		`-1`

index(str, start, end)	Same as find() but raises an exception if the substring is not present in the given string	`>>> str1 = 'Hello World!`
		`Hello  Hello'`
		`>>> str1.index('Hello')`
		`0`
		`>>> str1.index('Hee')`
		`ValueError: substring not found`

endswith()	Returns True if the given string ends with the supplied substring otherwise returns False	`>>> str1 = 'Hello World!'`
		`>>> str1.endswith('World!')`
		`True`
		`>>> str1.endswith('!')`
		`True`
		`>>> str1.endswith('lde')`
		`False`

startswith()	Returns True if the given string starts with the supplied substring otherwise returns False	`>>> str1 = 'Hello World!'`
		`>>> str1.startswith('He')`
		`True`
		`>>> str1.startswith('Hee')`
		`False`

isalnum()	Returns True if characters of the given string are either alphabets or numeric. If whitespace or special symbols are part of the given string or the string is empty it returns False	`>>> str1 = 'HelloWorld'`
		`>>> str1.isalnum()`
		`True`
		`>>> str1 = 'HelloWorld2'`
		`>>> str1.isalnum()`
		`True`
		`>>> str1 = 'HelloWorld!!'`
		`>>> str1.isalnum()`
		`False`

islower()	Returns True if the string is non-empty and has all lowercase alphabets, or has at least one character as lowercase alphabet and rest are non-alphabet characters	`>>> str1 = 'hello world!'`
		`>>> str1.islower()`
		`True`
		`>>> str1 = 'hello 1234'`
		`>>> str1.islower()`
		`True`
		`>>> str1 = 'hello ??'`
		`>>> str1.islower()`
		`True`
		`>>> str1 = '1234'`
		`>>> str1.islower()`
		`False`
		`>>> str1 = 'Hello World!'`
		`>>> str1.islower()`
		`False`

isupper()	Returns True if the string is non-empty and has all uppercase alphabets, or has at least one character as uppercase character and rest are non-alphabet characters	**>>> str1 = 'HELLO WORLD!'** **>>> str1.isupper()** **True** **>>> str1 = 'HELLO 1234'** **>>> str1.isupper()** **True** **>>> str1 = 'HELLO ??'** **>>> str1.isupper()** **True** **>>> str1 = '1234'** **>>> str1.isupper()** **False** **>>> str1 = 'Hello World!'** **>>> str1.isupper()** **False**

isspace()	Returns True if the string is non-empty and all characters are white spaces (blank, tab, newline, carriage return)	**>>> str1 = ' \n \t \r'** **>>> str1.isspace()** **True** **>>> str1 = 'Hello \n'** **>>> str1.isspace()** **False**
istitle()	Returns True if the string is non-empty and title case, i.e., the first letter of every word in the string in uppercase and rest in lowercase	**>>> str1 = 'Hello World!'** **>>> str1.istitle()** **True** **>>> str1 = 'hello World!'** **>>> str1.istitle()** **False**
lstrip()	Returns the string after removing the spaces only on the left of the string	**>>> str1 = ' Hello World! '** **>>> str1.lstrip()** **'Hello World! '**
rstrip()	Returns the string after removing the spaces only on the right of the string	**>>> str1 = ' Hello World!' >>> str1.rstrip()** **' Hello World!'**
strip()	Returns the string after removing the spaces both on the left and the right of the string	**>>> str1 = ' Hello World!' >>> str1.strip()** **'Hello World!'**

replace(old str, newstr)	Replaces all occurrences of old string with the new string	`>>> str1 = 'Hello World!'`
		`>>> str1.replace('o','*')`
		`'Hell* W*rld!'`
		`>>> str1 = 'Hello World!'`
		`>>>`
		`str1.replace('World','Country')`
		`'Hello Country!'`
		`>>> str1 = 'Hello World! Hello' >>>`
		`str1.replace('Hello','Bye')`
		`'Bye World! Bye'`

join()	Returns a string in which the characters in the string have been joined by a separator	`>>> str1 = ('HelloWorld!')`
		`>>> str2 = '-' #separator >>>`
		`str2.join(str1)`
		`'H-e-l-l-o-W-o-r-l-d-!'`

partition()	Partitions the given string at the first occurrence of the substring (separator) and returns the string partitioned into three parts.
	1. Substring before the separator
	2. Separator
	3. Substring after the separator If the separator is not found in the string, it returns the whole string itself and two empty strings
		`>>> str1 = 'India is a Great Country' >>> str1.partition('is')`
		`('India ', 'is', ' a Great Country')`
		`>>> str1.partition('are')`
		`('India is a Great Country',' ',' ')`

split()	Returns a list of words delimited by the specified substring. If no delimiter is given then words are separated by space.	`>>> str1 = 'India is a Great Country' >>> str1.split()`
		`['India','is','a','Great', 'Country']`
		`>>> str1 = 'India is a Great Country'`
		`>>> str1.split('a')`
		`['Indi', ' is ', ' Gre', 't Country']`