String

String: String is a sequence of UNICODE characters. A string can be created by enclosing one or more characters in single, double or triple quotes (' ' or " " or ''' ''').

Example

```
>>> str1 = 'Python'
>>> str2 = "Python"
>>> str3 = '''Multi Line
String'''
>>> str4 = """Multi Line
String"""
```

Terminology

Whitespace Characters: Those characters in a string that represent horizontal or vertical space. Example: space (' '), tab ('\t'), and newline ('\n')

Indexing in Strings

- Indexing is used to access individual characters in a string using a numeric value.
- The index of the first character (from left) in the string is 0 and the last character is n-1
- The index can also be an expression including variables and operators but the expression must evaluate to an integer
- Forward indexing, also known as positive indexing, starts from left to right, with the first character having index 0, second character having index 1, and so on.
- Backward indexing, also known as negative indexing, starts from right to left, with the last character having index -1, second-last character having index -2, and so on

POSITIVE INDEX	0	1	2	3	4	5
STRING	P	Y	T	H	O	N
NEGATIVE INDEX	-6	-5	-4	-3	-2	-1

.

Example

```
>>> str1 = 'Python'
>>> str1[0]
'P'
```

```
>>> str1 = 'Python'
>>> str1[2+3]
'n'
```

Slicing

Slicing is the process of extracting a substring from a given string. It is done using the operator ':'.
Syntax : string[start:end:step].
start : 'start' is the starting index of the substring (inclusive).
end : 'end' is the ending index of the substring (exclusive)
step : 'step' is the difference between the index of two consecutive elements. Default value is 1

Case-1

```
print(str1[:3])
```

Output

```
PYT
```

Case-2

```
print(str1[1:4])
```

Output

```
YTH
```

Case-3

```
print(str1[0:5:2])
```

Output

```
PTO
```

Case-4

```
print(str1[-5:-1])
```

Output

```
YTHO
```

Case-5

```
print(str1[-1:-4])
```

Output

```
''
```

Case-6

```
print(str1[:-5:-1])
```

Output

```
NOHT
```

Case-7

```
print(str1[:-5])
```

Output

```
P
```

Mutable Object

If the value of an object can be changed after it is created, It is called mutable object

Example

Lists, Set and dictionaries.

Immutable Object

If the value of an object cannot be changed after it is created, it is called immutable object

Example

Numeric, String and Tuple

String is Immutable

A string is an immutable data type. It means that the value (content) of a string object cannot be changed after it has been created. An attempt to do this would lead to an error.

Example

```
>>> str1 = 'Python'
>>> str1[1]='Y'
```

Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment

Traversal of a String

Accessing the individual characters of string i.e. from first

1. Using only for Loop

```
str1 = 'Computer'
for ch in str1:
          print(ch, end=' ')
```

2. Using For Loop and range function

```
str1 = 'Computer'
for ch in range(0,len(str1)):
          print(str1[ch],end=' ')
```

3. Using while Loop

```
str1 = 'Computer'
i = 0
while i < len(str1):
          print(str1[i],end = ' ')
          i+= 1
```

String Operations

We can perform various operations on string such as concatenation, repetition, membership and slicing.

Concatenation

Operator : +

```
>>> str1 = "Python"
>>> str2 = "Programming"
>>> str1 + str2
'PythonProgramming'
```

Repetition

Use : To repeat the given string multiple times.

Repetition operator : *

```
>>> str1 = "Hello "
>>> str1*5
'Hello Hello Hello Hello Hello'
```

Membership

Membership is the process of checking whether a particular character or substring is present in a sequence or not. It is done using the 'in' and 'not in' operators.

Example

>>> str1 = "Programming in Python"

`>>> "Prog" in str1` `True`	`>>> "ming in" in str1` `True`	`>>> "Pyth " in str1` `False`	`>>> "Pyth " not in str1` `True`

String Methods/Functions

Python has several built-in functions that allow us to work with strings

len()

Returns the length of the given string

```
>>> str1 = 'Hello World!'
>>> len(str1)
12
```

capitalize()

converts the first character of a string to capital (uppercase) letter and rest in lowercase.

```
str1 = "python Programming for 11th"
str1.capitalize()
'Python programming for 11th'
```

title()

converts the first letter of every word in the string in uppercase and rest in lowercase

```
>>> str1 = "python ProGramming for 11th"
>>> str1.title()
'Python Programming For 11Th'
```

lower()

Returns the string with all uppercase letters converted to lowercase

```
>>> str1 = "PYTHON PROGRAMMING for 11th"
>>> str1.lower()
'python programming for 11th'
```

upper()

Returns the string with all lowercase letters converted to uppercase

```
>>> str1 = "python programming for 11th"
>>> str1.upper()
'PYTHON PROGRAMMING FOR 11TH'
```

count()

Returns number of times a substring occurs in the given string

```
>>> str1 = "python programming for 11th"
>>> str1.count("p")
2
>>> str1.count("pyth")
1
```

find()

Returns the index of the first occurrence of substring in the given string. If the substring is not found, it returns -1

```
>>> str1 = "python programming for 11th"
>>> str1.find('r')
8
>>> str1.find('u')
-1
```

index()

Same as find() but raises an exception if the substring is not present in the given string

```
>>> str1 = "python programming for 11th"
>>> str1.index('r')
8
>>> str1.index('u')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: substring not found
```

isalnum()

The isalnum() method returns True if all characters in the string are alphanumeric (either alphabets or numbers). If not, it returns False.

```
>>> str1 = 'HelloWorld'
>>> str1.isalnum()
True
>>> str1 = 'HelloWorld2'
>>> str1.isalnum()
True
```

isalpha()

Returns True if all characters in the string are alphabets, Otherwise, It returns False

```
>>> 'Python'.isalpha()
True
>>> 'Python 123'.isalpha()
False
```

isdigit()

returns True if all characters in the string are digits, Otherwise, It returns False

```
>>> '1234'.isdigit()
True
>>> '123 567'.isdigit()
False
```

isspace()

Returns True if has characters and all of them are white spaces (blank, tab, newline)

```
>>> str1 = ' \n \t'
>>> str1.isspace()
True
>>> str1 = 'Hello \n'
>>> str1.isspace()
False
```

islower()

returns True if the string has letters all of them are in lower case and otherwise False.

```
>>> str1 = 'hello world!'
>>> str1.islower()
True
>>> str1 = 'hello 1234'
>>> str1.islower()
True
>>> str1 = 'hello ??'
 >>> str1.islower()
True
```

isupper()

returns True if the string has letters all of them are in upper case and otherwise False.

```
>>> str1 = 'HELLO WORLD!'
>>> str1.isupper()
True
>>> str1 = 'HELLO 1234'
>>> str1.isupper()
True
>>> str1 = 'HELLO ??'
 >>> str1.isupper()
True
```

strip() Returns the string after removing the whitespaces both on the left and the right of the string **>>> str1 = ' Hello World! '** **>>> str1.strip()** **'Hello World!'**	**lstrip()** Returns the string after removing the whitespaces only on the left of the string **>>> str1 = ' Hello World! '** **>>> str1.lstrip()** **'Hello World! '**
rstrip() Returns the string after removing the whitespaces only on the right of the string **>>> str1 = ' Hello World! '** **>>> str1.rstrip()** **' Hello World!'**	**replace(oldstr,newstr)** used to replace a particular substring in a string with another substring **>>> str1 = 'Hello World!'** **>>> str1.replace('o','*')** **'Hell* W*rld!'**
partition() The partition() function is used to split a string into three parts based on a specified separator. **>>> 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']**

startswith()

startswith() function is used to check whether a given string starts with a particular substring.

endswith()

endswith() function is used to check whether a given string ends with a particular substring.

```
str = "Python Programming Language"
print(str.startswith("Pyt"))
print(str.startswith("Pyth"))
print(str.endswith("age"))
print(str.endswith("uage"))
print(str.startswith("Pyts"))
```

```
Output
True
True
True
True
False
```

join()

str.join(sequence)

returns a string in which the string elements of sequence have been joined by str separator. if few of the members of the sequence are not string, error is generated.

```
>>> '*-*'.join('Python')
'P*-*y*-*t*-*h*-*o*-*n'
>>> '*-*'.join(['Ajay','Abhay','Alok'])
'Ajay*-*Abhay*-*Alok'
>>> '*-*'.join(['Ajay','Abhay',123])
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: sequence item 2: expected str instance, int found
```