String Operations- Concatenation

To concatenate, or combine, two strings you can use the + operator.

Example:

```
a = "Hello"
b = "World"
c = a + b
print(c)
```

Output:

```
HelloWorld
```

String Operations- Repetition

The repetition operator is denoted by a '*' symbol and is useful for repeating strings to a certain length.

Example:

```
str = 'Python program'
print(str*3)
```

The above lines of code will display the following outputs:

```
Python programPython programPython program
```

String Operations- Membership Operators(in and not in)

An “in” operator keyword returns True. If the given element / entire substring is present in the given string, otherwise it returns False.

A “not in” operator keyword returns True. If the given element / entire substring is not present in the given string, otherwise it returns False.

Syntax:

```
<string_element> in <string_variable_name>
```

OR

```
<string_element> not in <string_variable_name>
```

Example: in

```
str1="I live in delhi"
str2="live"
print("str2 is a substring of str1:",str2 in str1)
```

Output:

```
str2 is a substring of str1: True
```

Example: not in

```
str1="I live in delhi"
str2="hello"
print("str2 is a substring of str1:",str2 not in str1)
```

Output:

```
str2 is a substring of str1: True
```

String Operations- Slicing

You can return a range of characters by using the slice syntax.

Specify the start index and the end index, separated by a colon, to return a part of the string.

Example

Get the characters from position 2 to position 5 (not included):

```
b = "Hello, World!"
print(b[2:5])
```

Contd..

Slice From the Start

By leaving out the start index, the range will start at the first character:

Example

Get the characters from the start to position 5 (not included):

```
b = "Hello, World!"
print(b[:5])
```

Slice To the End

By leaving out the *end* index, the range will go to the end:

Example

Get the characters from position 2, and all the way to the end:

```
b = "Hello, World!"
print(b[2:])
```

Contd..

Negative Indexing

Use negative indexes to start the slice from the end of the string:

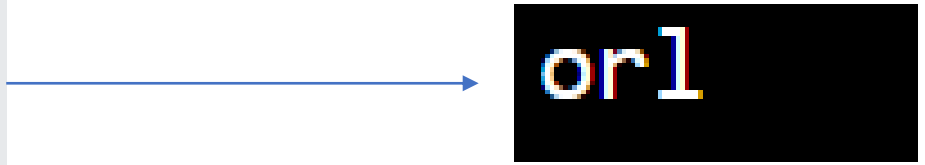
Example

Get the characters:

From: "o" in "World!" (position -5)

To, but not included: "d" in "World!" (position -2):

```
b = "Hello, World!"
print(b[-5:-2])
```

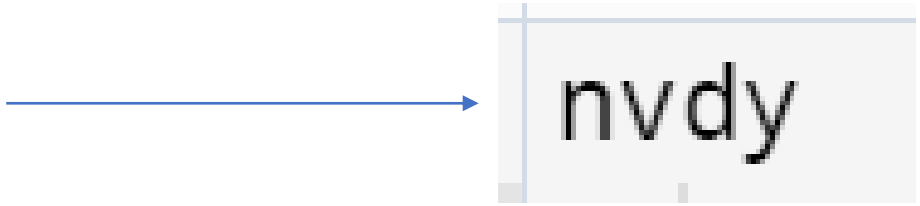


Contd..

Slicing By Skipping Values:

main.py

```
1  s='navodaya'
2  print(s[0:9:2])
```



Traversing the String

Traversing means to visit each index of the string.

Traversing Using while loop:

```
s='India'
i=0
while i<5:
    print(s[i])
    i=i+1
```

```
PS C
I
n
d
i
a
```

Traversing Using for loop:

```
s='India'
for i in s:
    print(i)
```

```
I
n
d
i
a
```

```
s='India'
for i in range(0,5):
    print(s[i])
```

```
I
n
d
i
a
```

Built in function: String

1. len() function

Syntax

```
len(object)
```

Object: must be sequence

the `len()` function returns the number of characters in the string.

Example

Return the number of characters in a string:

```
mylist = "Hello"
x = len(mylist)
```

Output: 5

2. capitalize()

The `capitalize()` method returns a string where the first character is upper case, and the rest is lower case.

Syntax

```
string.capitalize()
```

Example

The first character is converted to upper case, and the rest are converted to lower case:

```
txt = "python is FUN!"

x = txt.capitalize()

print (x)
```

```
Python is fun!
```

3. title()

The `title()` method returns a string where the first character in every word is upper case. Like a header, or a title.

If the word contains a number or a symbol, the first letter after that will be converted to upper case.

Syntax

```
string.title()
```

Example

Make the first letter in each word upper case:

```
txt = "Welcome to my 2nd world"

x = txt.title()

print(x)
```

```
Welcome To My 2Nd World
```

4. lower()

The `lower()` method returns a string where all characters are lower case.

Symbols and Numbers are ignored.

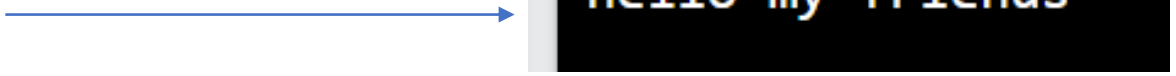
Syntax

```
string.lower()
```

Example

Lower case the string:

```
txt = "Hello my FRIENDS"

x = txt.lower()

print(x)
```



```
hello my friends
```

5. upper()

The `upper()` method returns a string where all characters are in upper case.

Symbols and Numbers are ignored.

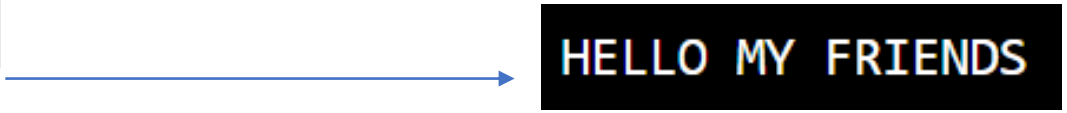
Syntax

```
string.upper()
```

Example

Upper case the string:

```
txt = "Hello my friends"

x = txt.upper()

print(x)
```



```
HELLO MY FRIENDS
```

6. count()

The `count()` method returns the number of times a specified value appears in the string.

Syntax

```
string.count(value, start, end)
```

Parameter Values

Parameter	Description
value	Required. A String. The string to value to search for
start	Optional. An Integer. The position to start the search. Default is 0
end	Optional. An Integer. The position to end the search. Default is the end of the string

Contd..

Example

Search from position 10 to 24:

```
txt = "I love apples, apple are my favorite fruit"

x = txt.count("apple", 10, 24)

print(x)
```

Output: 1

7. find()

The `find()` method finds the first occurrence of the specified value.

The `find()` method returns -1 if the value is not found.

The `find()` method is almost the same as the `index()` method, the only difference is that the `index()` method raises an exception if the value is not found. (See example below)

Syntax

```
string.find(value, start, end)
```

Parameter Values

Parameter	Description
value	Required. The value to search for
start	Optional. Where to start the search. Default is 0
end	Optional. Where to end the search. Default is to the end of the string

Contd..

Example

Where in the text is the first occurrence of the letter "e"?:

```
txt = "Hello, welcome to my world."

x = txt.find("e")

print(x)
```

Output: 1

8. index()

The `index()` method finds the first occurrence of the specified value.

Syntax

```
string.index(value, start, end)
```

Parameter Values

Parameter	Description
value	Required. The value to search for
start	Optional. Where to start the search. Default is 0
end	Optional. Where to end the search. Default is to the end of the string

Contd..

Example

Where in the text is the first occurrence of the letter "e"?:

```
txt = "Hello, welcome to my world."

x = txt.index("e")

print(x)
```

Output: 1

9. endswith()

The `endswith()` method returns True if the string ends with the specified value, otherwise False.

Syntax

```
string.endswith(value, start, end)
```

Parameter Values

Parameter	Description
value	Required. The value to check if the string ends with
start	Optional. An Integer specifying at which position to start the search
end	Optional. An Integer specifying at which position to end the search

Contd..

Example

Check if the string ends with a punctuation sign (.):

```
txt = "Hello, welcome to my world."

x = txt.endswith(".")

print(x)
```


Output: True

10. startswith()

The `startswith()` method returns True if the string starts with the specified value, otherwise False.

Syntax

```
string.startswith(value, start, end)
```

Parameter Values

Parameter	Description
value	Required. The value to check if the string starts with
start	Optional. An Integer specifying at which position to start the search
end	Optional. An Integer specifying at which position to end the search

Contd..

Example

Check if position 7 to 20 starts with the characters "wel":

```
txt = "Hello, welcome to my world."

x = txt.startswith("wel", 7, 20)

print(x)
```

Output: True

11. isalnum()

The `isalnum()` method returns True if all the characters are alphanumeric, meaning alphabet letter (a-z) and numbers (0-9).

Example of characters that are not alphanumeric: (space)!#%&? etc.

Syntax

```
string.isalnum()
```

Parameter Values

No parameters.

Contd..

Example

Check if all the characters in the text is alphanumeric:

```
txt = "Company 12"

x = txt.isalnum()

print(x)
```


Output: False

12. isalpha()

The `isalpha()` method returns True if all the characters are alphabet letters (a-z).

Example of characters that are not alphabet letters: (space)!#%&? etc.

Syntax

```
string.isalpha()
```

Parameter Values

No parameters.

Contd..

Example

Check if all the characters in the text is alphabetic:

```
txt = "Company10"

x = txt.isalpha()

print(x)
```

Output: False

13. isdigit()

The `isdigit()` method returns True if all the characters are digits, otherwise False.

Example:

```
a = "131313"

print(a.isdigit())
```

Output: True

14. islower()

The `islower()` method returns True if all the characters are in lower case, otherwise False.

Numbers, symbols and spaces are not checked, only alphabet characters.

Syntax

```
string.islower()
```

Example

Check if all the characters in the texts are in lower case:

```
a = "Hello world!"
b = "hello 123"
c = "mynameisPeter"

print(a.islower())
print(b.islower())
print(c.islower())
```

15. isupper()

The `isupper()` method returns True if all the characters are in upper case, otherwise False.

Numbers, symbols and spaces are not checked, only alphabet characters.

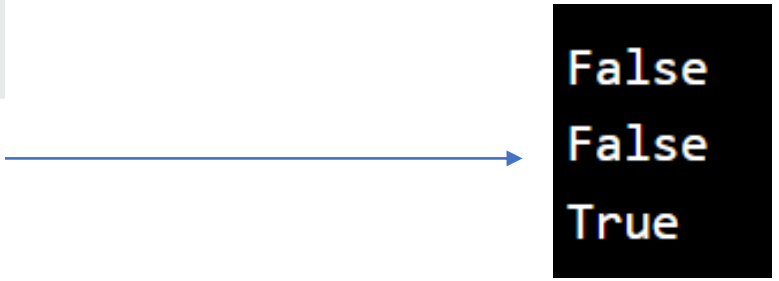
Syntax

```
string.isupper()
```

Example

Check if all the characters in the texts are in upper case:

```
a = "Hello World!"
b = "hello 123"
c = "MY NAME IS PETER"

print(a.isupper())
print(b.isupper())
print(c.isupper())
```



16. isspace()

The `isspace()` method returns True if all the characters in a string are whitespaces, otherwise False.

Syntax

```
string.isspace()
```

Example

Check if all the characters in the text are whitespaces:

```
txt = "    "

x = txt.isspace()

print(x)
```

17. lstrip()

The `lstrip()` method removes any leading characters (space is the default leading character to remove)

Syntax

```
string.lstrip(characters)
```

Parameter Values

Parameter	Description
characters	Optional. A set of characters to remove as leading characters

Example

Remove spaces to the left of the string:

```
txt = "     banana     "

x = txt.lstrip()

print("of all fruits", x, "is my favorite")
```



```
of all fruits banana     is my favorite
```

18. rstrip()

The `rstrip()` method removes any trailing characters (characters at the end a string), space is the default trailing character to remove.

Syntax

```
string.rstrip(characters)
```

Parameter Values

Parameter	Description
characters	Optional. A set of characters to remove as trailing characters

Contd..

Example

Remove any white spaces at the end of the string:

```
txt = "     banana     "

x = txt.rstrip()

print("of all fruits", x, "is my favorite")
```

```
of all fruits     banana is my favorite
```

Example

Remove the trailing characters if they are commas, periods, s, q, or w:

```
txt = "banana,,,,,ssqqqww....."

x = txt.rstrip(",.qsw")

print(x)
```

```
banana
```

19. strip()

The `strip()` method removes any leading, and trailing whitespaces.

Leading means at the beginning of the string, trailing means at the end.

You can specify which character(s) to remove, if not, any whitespaces will be removed.

Syntax

```
string.strip(characters)
```

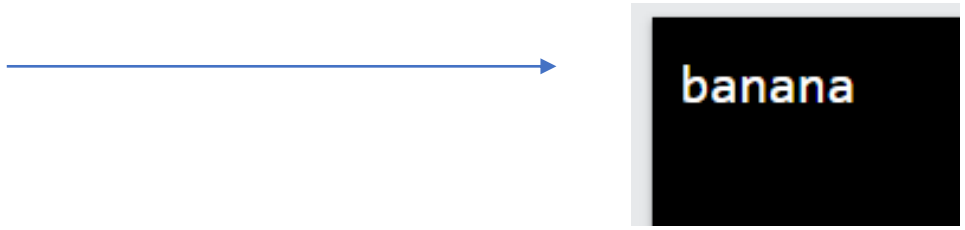
Parameter Values

Parameter	Description
characters	Optional. A set of characters to remove as leading/trailing characters

Contd..

```
txt = "     banana     "

x = txt.strip()

print("of all fruits", x, "is my favorite")
```


```
txt = ",,,,,rrttgg.....banana....rrr"

x = txt.strip(",.grt")

print(x)
```



20. replace()

The `replace()` method replaces a specified phrase with another specified phrase.

Syntax

```
string.replace(oldvalue, newvalue, count)
```

Parameter Values

Parameter	Description
oldvalue	Required. The string to search for
newvalue	Required. The string to replace the old value with
count	Optional. A number specifying how many occurrences of the old value you want to replace. Default is all occurrences

Contd..

```
txt = "I like bananas"

x = txt.replace("bananas", "apples")

print(x)
```

```
I like apples
```

Example

Replace all occurrence of the word "one":

```
txt = "one one was a race horse, two two was one too."

x = txt.replace("one", "three")

print(x)
```

```
three three was a race horse, two two was three too."
```

21. join()

The `join()` method takes all items in an iterable and joins them into one string.

A string must be specified as the separator.

Syntax

```
string.join(iterable)
```

Example:

```
s='hello'
str="*".join(s)
print(str)
```



```
h*e*l*l*o
```

22. partition()

The `partition()` method searches for a specified string, and splits the string into a tuple containing three elements.

The first element contains the part before the specified string.

The second element contains the specified string.

The third element contains the part after the string.

Note: This method searches for the *first* occurrence of the specified string.

Syntax

```
string.partition(value)
```

Parameter	Description
value	Required. The string to search for

Example:

```
s="I love video games "
str=s.partition('video')
print(str)
print(type(str))
```



```
('I love ', 'video', ' games ')
<class 'tuple'>
```

23. split()

The `split()` method splits a string into a list.

You can specify the separator, default separator is any whitespace.

Note: When maxsplit is specified, the list will contain the specified number of elements *plus one*.

Syntax

```
string.split(separator, maxsplit)
```

Parameter Values

Parameter	Description
separator	Optional. Specifies the separator to use when splitting the string. By default any whitespace is a separator
maxsplit	Optional. Specifies how many splits to do. Default value is -1, which is "all occurrences"

Example 1:

```
txt = "hello, my name is Ram, I am 17 years old"

x = txt.split(", ")

print(x)
```

```
['hello', 'my name is Ram', 'I am 17 years old']
```

Example 2:

```
s="I,love,video,games "
str=s.split(",",2)
print(str)
```

```
['I', 'love', 'video,games ']
```

Note:

```
# setting the maxsplit parameter to 2, will return a list with 3 elements!
```

Some Practice Questions

Q1. Write a program to count the frequency of a character in a string.

```
str=input("Enter any String")
ch=input("Enter the character")
print(str.count(ch))
```

Q2. Write a program to accept a string and return a string having first letter of each word in capital.

```
str=input("Enter any String")
print(str.title())
```

Q4. Write a program to reverse a string.

```
str=input("Enter any String")
print(str[::-1])
```

Contd..

Q3. Write a program to accept a string and display the following:

1. **Number of uppercase characters**
2. **Numbers of lowercase characters**
3. **Total number of alphabets**
4. **Number of digits**

```
str=input("Enter any String")
u=0
L=0
d=0
l = len(str)
for i in range(l):
    if str[i].isupper():
        u = u+1
    if str[i].islower():
        L = L + 1
    if str[i].isdigit():
        d = d+1
print("Total Upper Case Characters are: ", u)
print("Total Lower Case Characters are: ", L)
print("Total Characters are: ", L + u)
print("Total digits are: ", d)
```