10. Python 3 – Strings

Strings are amongst the most popular types in Python. We can create them simply by enclosing characters in quotes. Python treats single quotes the same as double quotes. Creating strings is as simple as assigning a value to a variable. For example-

```
var1 = 'Hello World!'

var2 = "Python Programming"
```

Accessing Values in Strings

Python does not support a character type; these are treated as strings of length one, thus also considered a substring.

To access substrings, use the square brackets for slicing along with the index or indices to obtain your substring. For example-

```
#!/usr/bin/python3

var1 = 'Hello World!'

var2 = "Python Programming"

print ("var1[0]: ", var1[0])

print ("var2[1:5]: ", var2[1:5])
```

When the above code is executed, it produces the following result-

```
var1[0]:  H

var2[1:5]:  ytho
```

Updating Strings

You can "update" an existing string by (re)assigning a variable to another string. The new value can be related to its previous value or to a completely different string altogether. For example-

```
#!/usr/bin/python3

var1 = 'Hello World!'

print ("Updated String :- ", var1[:6] + 'Python')
```

When the above code is executed, it produces the following result-

```
Updated String :-  Hello Python
```

Escape Characters

Following table is a list of escape or non-printable characters that can be represented with backslash notation.

An escape character gets interpreted; in a single quoted as well as double quoted strings.

Backslash notation	Hexadecimal character	Description
a	0x07	Bell or alert
b	0x08	Backspace
\cx		Control-x
\C-x		Control-x
\e	0x1b	Escape
\f	0x0c	Formfeed
\M-\C-x		Meta-Control-x
\n	0x0a	Newline
\nnn		Octal notation, where n is in the range 0.7
\r	0x0d	Carriage return
\s	0x20	Space
\t	0x09	Tab

\v	0x0b	Vertical tab
\x		Character x
\xnn		Hexadecimal notation, where n is in the range 0.9, a.f, or A.F

String Special Operators

Assume string variable **a** holds 'Hello' and variable **b** holds 'Python', then-

Operator	Description	Example
+	Concatenation - Adds values on either side of the operator	a + b will give HelloPython
*	Repetition - Creates new strings, concatenating multiple copies of the same string	a*2 will give -HelloHello
[]	Slice - Gives the character from the given index	a[1] will give e
[:]	Range Slice - Gives the characters from the given range	a[1:4] will give ell
in	Membership - Returns true if a character exists in the given string	H in a will give 1
not in	Membership - Returns true if a character does not exist in the given string	M not in a will give 1
r/R	Raw String - Suppresses actual meaning of Escape characters. The syntax for raw strings is exactly the same as for normal strings with the exception of the raw string operator, the letter "r," which precedes the quotation marks. The "r" can be lowercase (r) or uppercase (R) and must be placed immediately preceding the first quote mark.	print r'\n' prints \n and print R'\n'prints \n

%	Format - Performs String formatting	See next section

String Formatting Operator

One of Python's coolest features is the string format operator %. This operator is unique to strings and makes up for the pack of having functions from C's printf() family. Following is a simple example –

```
#!/usr/bin/python3

print ("My name is %s and weight is %d kg!" % ('Zara', 21))
```

When the above code is executed, it produces the following result –

```
My name is Zara and weight is 21 kg!
```

Here is the list of complete set of symbols which can be used along with %-

Format Symbol	Conversion
%c	character
%s	string conversion via str() prior to formatting
%i	signed decimal integer
%d	signed decimal integer
%u	unsigned decimal integer
%o	octal integer
%x	hexadecimal integer (lowercase letters)
%X	hexadecimal integer (UPPERcase letters)
%e	exponential notation (with lowercase 'e')

%E	exponential notation (with UPPERcase 'E')
%f	floating point real number
%g	the shorter of %f and %e
%G	the shorter of %f and %E

Other supported symbols and functionality are listed in the following table-

Symbol	Functionality
*	argument specifies width or precision
-	left justification
+	display the sign
<sp>	leave a blank space before a positive number
#	add the octal leading zero ('0') or hexadecimal leading '0x' or '0X', depending on whether 'x' or 'X' were used.
0	pad from left with zeros (instead of spaces)
%	'%%' leaves you with a single literal '%'
(var)	mapping variable (dictionary arguments)
m.n.	m is the minimum total width and n is the number of digits to display after the decimal point (if appl.)

Triple Quotes

Python's triple quotes comes to the rescue by allowing strings to span multiple lines, including verbatim NEWLINEs, TABs, and any other special characters.

The syntax for triple quotes consists of three consecutive **single or double** quotes.

```
#!/usr/bin/python3


para_str = """this is a long string that is made up of
several lines and non-printable characters such as
TAB ( \t ) and they will show up that way when displayed.
NEWLINEs within the string, whether explicitly given like
this within the brackets [ \n ], or just a NEWLINE within
the variable assignment will also show up.
"""
print (para_str)
```

When the above code is executed, it produces the following result. Note how every single special character has been converted to its printed form, right down to the last NEWLINE at the end of the string between the "up." and closing triple quotes. Also note that NEWLINEs occur either with an explicit carriage return at the end of a line or its escape code (\n) –

```
this is a long string that is made up of
several lines and non-printable characters such as
TAB (    ) and they will show up that way when displayed.
NEWLINEs within the string, whether explicitly given like
this within the brackets [
 ], or just a NEWLINE within
the variable assignment will also show up.
```

Raw strings do not treat the backslash as a special character at all. Every character you put into a raw string stays the way you wrote it-

```
#!/usr/bin/python3
print ('C:\\nowhere')
```

When the above code is executed, it produces the following result-

```
C:\nowhere
```

Now let us make use of raw string. We would put expression in **r'expression'** as follows-

```
#!/usr/bin/python3
```

```
print (r'C:\\nowhere')
```

When the above code is executed, it produces the following result-

```
C:\\nowhere
```

Unicode String

In Python 3, all strings are represented in Unicode. In Python 2 are stored internally as 8-bit ASCII, hence it is required to attach 'u' to make it Unicode. It is no longer necessary now.

Built-in String Methods

Python includes the following built-in methods to manipulate strings-

S. No.	Methods with Description
1	**capitalize()** Capitalizes first letter of string
2	**center(width, fillchar)** Returns a string padded with *fillchar* with the original string centered to a total of *width* columns.
3	**count(str, beg= 0,end=len(string))** Counts how many times str occurs in string or in a substring of string if starting index beg and ending index end are given.
4	**decode(encoding='UTF-8',errors='strict')** Decodes the string using the codec registered for encoding. encoding defaults to the default string encoding.
5	**encode(encoding='UTF-8',errors='strict')** Returns encoded string version of string; on error, default is to raise a ValueError unless errors is given with 'ignore' or 'replace'.
6	**endswith(suffix, beg=0, end=len(string))**

	Determines if string or a substring of string (if starting index beg and ending index end are given) ends with suffix; returns true if so and false otherwise.
7	**expandtabs(tabsize=8)** Expands tabs in string to multiple spaces; defaults to 8 spaces per tab if tabsize not provided.
8	**find(str, beg=0 end=len(string))** Determine if str occurs in string or in a substring of string if starting index beg and ending index end are given returns index if found and -1 otherwise.
9	**index(str, beg=0, end=len(string))** Same as find(), but raises an exception if str not found.
10	**isalnum()** Returns true if string has at least 1 character and all characters are alphanumeric and false otherwise.
11	**isalpha()** Returns true if string has at least 1 character and all characters are alphabetic and false otherwise.
12	**isdigit()** Returns true if the string contains only digits and false otherwise.
13	**islower()** Returns true if string has at least 1 cased character and all cased characters are in lowercase and false otherwise.
14	**isnumeric()** Returns true if a unicode string contains only numeric characters and false otherwise.

15	**isspace()** Returns true if string contains only whitespace characters and false otherwise.
16	**istitle()** Returns true if string is properly "titlecased" and false otherwise.
17	**isupper()** Returns true if string has at least one cased character and all cased characters are in uppercase and false otherwise.
18	**join(seq)** Merges (concatenates) the string representations of elements in sequence seq into a string, with separator string.
19	**len(string)** Returns the length of the string
20	**ljust(width[, fillchar])** Returns a space-padded string with the original string left-justified to a total of width columns.
21	**lower()** Converts all uppercase letters in string to lowercase.
22	**lstrip()** Removes all leading whitespace in string.
23	**maketrans()** Returns a translation table to be used in translate function.

24	**max(str)** Returns the max alphabetical character from the string str.
25	**min(str)** Returns the min alphabetical character from the string str.
26	**replace(old, new [, max])** Replaces all occurrences of old in string with new or at most max occurrences if max given.
27	**rfind(str, beg=0,end=len(string))** Same as find(), but search backwards in string.
28	**rindex(str, beg=0, end=len(string))** Same as index(), but search backwards in string.
29	**rjust(width,[, fillchar])** Returns a space-padded string with the original string right-justified to a total of width columns.
30	**rstrip()** Removes all trailing whitespace of string.
31	**split(str="", num=string.count(str))** Splits string according to delimiter str (space if not provided) and returns list of substrings; split into at most num substrings if given.
32	**splitlines(num=string.count('\n'))** Splits string at all (or num) NEWLINEs and returns a list of each line with NEWLINEs removed.

33	**startswith(str, beg=0,end=len(string))** Determines if string or a substring of string (if starting index beg and ending index end are given) starts with substring str; returns true if so and false otherwise.
34	**strip([chars])** Performs both lstrip() and rstrip() on string
35	**swapcase()** Inverts case for all letters in string.
36	**title()** Returns "titlecased" version of string, that is, all words begin with uppercase and the rest are lowercase.
37	**translate(table, deletechars="")** Translates string according to translation table str(256 chars), removing those in the del string.
38	**upper()** Converts lowercase letters in string to uppercase.
39	**zfill (width)** Returns original string leftpadded with zeros to a total of width characters; intended for numbers, zfill() retains any sign given (less one zero).
40	**isdecimal()** Returns true if a unicode string contains only decimal characters and false otherwise.

String capitalize() Method

It returns a copy of the string with only its first character capitalized.

Syntax

```
str.capitalize()
```

Parameters

NA

Return Value

string

Example

```
#!/usr/bin/python3
str = "this is string example....wow!!!"
print ("str.capitalize() : ", str.capitalize())
```

Result

```
str.capitalize() :  This is string example....wow!!!
```

String center() Method

The method center() returns centered in a string of length width. Padding is done using the specified fillchar. Default filler is a space.

Syntax

```
str.center(width[, fillchar])
```

Parameters

- width - This is the total width of the string.
- fillchar - This is the filler character.

Return Value

This method returns a string that is at least width characters wide, created by padding the string with the character fillchar (default is a space).

Example

The following example shows the usage of the center() method.

```
#!/usr/bin/python3
str = "this is string example....wow!!!"
```

```
print ("str.center(40, 'a') : ", str.center(40, 'a'))
```

Result

```
str.center(40, 'a') :  aaaathis is string example....wow!!!aaaa
```

String count() Method

Description

The **count()** method returns the number of occurrences of substring sub in the range [start, end]. Optional arguments start and end are interpreted as in slice notation.

Syntax

```
str.count(sub, start= 0,end=len(string))
```

Parameters

- **sub -** This is the substring to be searched.
- **start** - Search starts from this index. First character starts from 0 index. By default search starts from 0 index.
- **end** - Search ends from this index. First character starts from 0 index. By default search ends at the last index.

Return Value

Centered in a string of length width.

Example

```
#!/usr/bin/python3
str="this is string example....wow!!!"
sub='i'
print ("str.count('i') : ", str.count(sub))
sub='exam'
print ("str.count('exam', 10, 40) : ", str.count(sub,10,40))
```

Result

```
str.count('i') :  3
str.count('exam', 4, 40) :
```

String decode() Method

Description

The **decode()** method decodes the string using the codec registered for encoding. It defaults to the default string encoding.

Syntax

```
Str.decode(encoding='UTF-8',errors='strict')
```

Parameters

- **encoding** - This is the encodings to be used. For a list of all encoding schemes please visit: Standard Encodings.

- **errors** - This may be given to set a different error handling scheme. The default for errors is 'strict', meaning that encoding errors raise a UnicodeError. Other possible values are 'ignore', 'replace', 'xmlcharrefreplace', 'backslashreplace' and any other name registered via codecs.register_error()..

Return Value

Decoded string.

Example

```
#!/usr/bin/python3

Str = "this is string example....wow!!!";

Str = Str.encode('base64','strict');

print "Encoded String: " + Str

print "Decoded String: " + Str.decode('base64','strict')
```

Result

```
Encoded String:  b'dGhpcyBpcyBzdHJpbmcgZXhhbXBsZS4uLi53b3chISE='

Decoded String:  this is string example....wow!!!
```

String encode() Method

Description

The **encode()** method returns an encoded version of the string. Default encoding is the current default string encoding. The errors may be given to set a different error handling scheme.

Syntax

```
str.encode(encoding='UTF-8',errors='strict')
```

Parameters

- **encoding -** This is the encodings to be used. For a list of all encoding schemes please visit: Standard Encodings.

- **errors -** This may be given to set a different error handling scheme. The default for errors is 'strict', meaning that encoding errors raise a UnicodeError. Other possible values are 'ignore', 'replace', 'xmlcharrefreplace', 'backslashreplace' and any other name registered via codecs.register_error().

Return Value

Decoded string.

Example

```
#!/usr/bin/python3
import base64
Str = "this is string example....wow!!!"
Str=base64.b64encode(Str.encode('utf-8',errors='strict'))
print ("Encoded String: " , Str)
```

Result

```
Encoded String:  b'dGhpcyBpcyBzdHJpbmcgZXhhbXBsZS4uLi53b3chISE='
```

String endswith() Method

Description

It returns True if the string ends with the specified suffix, otherwise return False optionally restricting the matching with the given indices start and end.

Syntax

```
str.endswith(suffix[, start[, end]])
```

Parameters

- **suffix -** This could be a string or could also be a tuple of suffixes to look for.

- **start** - The slice begins from here.

- **end** - The slice ends here.

Return Value

TRUE if the string ends with the specified suffix, otherwise FALSE.

Example

```
#!/usr/bin/python3
Str='this is string example....wow!!!'
suffix='!!'
print (Str.endswith(suffix))
print (Str.endswith(suffix,20))
suffix='exam'
print (Str.endswith(suffix))
print (Str.endswith(suffix, 0, 19))
```

Result

```
True
True
False
True
```

String expandtabs() Method

Description

The **expandtabs()** method returns a copy of the string in which the tab characters ie. '\t' are expanded using spaces, optionally using the given tabsize (default 8)..

Syntax

```
str.expandtabs(tabsize=8)
```

Parameters

tabsize - This specifies the number of characters to be replaced for a tab character '\t'.

Return Value

This method returns a copy of the string in which tab characters i.e., '\t' have been expanded using spaces.

Example

```
#!/usr/bin/python3
str = "this is\tstring example....wow!!!"
print ("Original string: " + str)
print ("Defualt exapanded tab: " +  str.expandtabs())
print ("Double exapanded tab: " +  str.expandtabs(16))
```

Result

```
Original string: this is        string example....wow!!!
Defualt exapanded tab:         this is string example....wow!!!
Double exapanded tab: this is         string example....wow!!!
```

String find() Method

Description

The find() method determines if the string str occurs in string, or in a substring of string if the starting index beg and ending index end are given.

Syntax

```
str.find(str, beg=0 end=len(string))
```

Parameters

- **str** - This specifies the string to be searched.
- **beg** - This is the starting index, by default its 0.
- **end** - This is the ending index, by default its equal to the lenght of the string.

Return Value

Index if found and -1 otherwise.

Example

```
#!/usr/bin/python3
str1 = "this is string example....wow!!!"
str2 = "exam";
print (str1.find(str2))
print (str1.find(str2, 10))
print (str1.find(str2, 40))
```

Result

```
15
15
-1
```

String index() Method

Description

The index() method determines if the string str occurs in string or in a substring of string, if the starting index beg and ending index end are given. This method is same as find(), but raises an exception if sub is not found.

Syntax

```
str.index(str, beg=0 end=len(string))
```

Parameters

- **str** - This specifies the string to be searched.
- **beg** - This is the starting index, by default its 0.
- **end** - This is the ending index, by default its equal to the length of the string.

Return Value

Index if found otherwise raises an exception if str is not found.

Example

```
#!/usr/bin/python3
str1 = "this is string example....wow!!!"
str2 = "exam";
print (str1.index(str2))
print (str1.index(str2, 10))
print (str1.index(str2, 40))
```

Result

```
15
```

```
15
Traceback (most recent call last):
  File "test.py", line 7, in
    print (str1.index(str2, 40))
ValueError: substring not found
shell returned 1
```

String isalnum() Method

Description

The **isalnum()** method checks whether the string consists of alphanumeric characters.

Syntax

Following is the syntax for isalnum() method-

```
str.isa1num()
```

Parameters

NA

Return Value

This method returns true if all the characters in the string are alphanumeric and there is at least one character, false otherwise.

Example

The following example shows the usage of isalnum() method.

```
#!/usr/bin/python3
str = "this2016"  # No space in this string
print (str.isalnum())
str = "this is string example....wow!!!"
print (str.isalnum())
```

When we run the above program, it produces the following result-

```
True
False
```

String isalpha() Method

Description

The **isalpha()** method checks whether the string consists of alphabetic characters only.

Syntax

Following is the syntax for islpha() method-

```
str.isalpha()
```

Parameters

NA

Return Value

This method returns true if all the characters in the string are alphabetic and there is at least one character, false otherwise.

Example

The following example shows the usage of isalpha() method.

```
#!/usr/bin/python3
str = "this";  # No space & digit in this string
print (str.isalpha())
str = "this is string example....wow!!!"
print (str.isalpha())
```

Result

```
True
False
```

String isdigit() Method

Description

The method isdigit() checks whether the string consists of digits only.

Syntax

Following is the syntax for isdigit() method-

```
str.isdigit()
```

Parameters

NA

Return Value

This method returns true if all characters in the string are digits and there is at least one character, false otherwise.

Example

The following example shows the usage of isdigit() method.

```
#!/usr/bin/python3
str = "123456";  # Only digit in this string
print (str.isdigit())
str = "this is string example....wow!!!"
print (str.isdigit())
```

Result

```
True
False
```

String islower() Method

Description

The **islower()** method checks whether all the case-based characters (letters) of the string are lowercase.

Syntax

Following is the syntax for islower() method-

```
str.islower()
```

Parameters

NA

Return Value

This method returns true if all cased characters in the string are lowercase and there is at least one cased character, false otherwise.

Example

The following example shows the usage of islower() method.

```
#!/usr/bin/python3
str = "THIS is string example....wow!!!"
print (str.islower())
str = "this is string example....wow!!!"
print (str.islower())
```

Result

```
False
True
```

String isnumeric() Method

Description

The **isnumeric()** method checks whether the string consists of only numeric characters. This method is present only on unicode objects.

Note: Unlike Python 2, all strings are represented in Unicode in Python 3. Given below is an example illustrating it.

Syntax

Following is the syntax for isnumeric() method-

```
str.isnumeric()
```

Parameters

NA

Return Value

This method returns true if all characters in the string are numeric, false otherwise.

Example

The following example shows the usage of isnumeric() method.

```
#!/usr/bin/python3
str = "this2016"
print (str.isnumeric())
str = "23443434"
```

```
print (str.isnumeric())
```

Result

```
False
True
```

String isspace() Method

Description

The **isspace()** method checks whether the string consists of whitespace..

Syntax

Following is the syntax for isspace() method-

```
str.isspace()
```

Parameters

NA

Return Value

This method returns true if there are only whitespace characters in the string and there is at least one character, false otherwise.

Example

The following example shows the usage of isspace() method.

```
#!/usr/bin/python3
str = "       "
print (str.isspace())
str = "This is string example....wow!!!"
print (str.isspace())
```

Result

```
True
False
```

String istitle() Method

Description

The **istitle()** method checks whether all the case-based characters in the string following non-casebased letters are uppercase and all other case-based characters are lowercase.

Syntax

Following is the syntax for istitle() method-

```
str.istitle()
```

Parameters

NA

Return Value

This method returns true if the string is a titlecased string and there is at least one character, for example uppercase characters may only follow uncased characters and lowercase characters only cased ones. It returns false otherwise.

Example

The following example shows the usage of istitle() method.

```
#!/usr/bin/python3
str = "This Is String Example...Wow!!!"
print (str.istitle())
str = "This is string example....wow!!!"
print (str.istitle())
```

Result

```
True
False
```

String isupper() Method

Description

The **isupper()** method checks whether all the case-based characters (letters) of the string are uppercase.

Syntax

Following is the syntax for isupper() method-

```
str.isupper()
```

Parameters

NA

Return Value

This method returns true if all the cased characters in the string are uppercase and there is at least one cased character, false otherwise.

Example

The following example shows the usage of isupper() method.

```
#!/usr/bin/python3
str = "THIS IS STRING EXAMPLE....WOW!!!"
print (str.isupper())
str = "THIS is string example....wow!!!"
print (str.isupper())
```

Result

```
True
False
```

String join() Method

Description

The **join()** method returns a string in which the string elements of sequence have been joined by str separator.

Syntax

Following is the syntax for join() method-

```
str.join(sequence)
```

Parameters

sequence - This is a sequence of the elements to be joined.

Return Value

This method returns a string, which is the concatenation of the strings in the sequence **seq.** The separator between elements is the string providing this method.

Example

The following example shows the usage of join() method.

```
#!/usr/bin/python3
s = "-"
seq = ("a", "b", "c") # This is sequence of strings.
print (s.join( seq ))
```

Result

```
a-b-c
```

String len() Method

Description

The **len()** method returns the length of the string.

Syntax

Following is the syntax for len() method –

```
len( str )
```

Parameters

NA

Return Value

This method returns the length of the string.

Example

The following example shows the usage of len() method.

```
#!/usr/bin/python3
str = "this is string example....wow!!!"
print ("Length of the string: ", len(str))
```

Result

```
Length of the string:  32
```

String ljust() Method

Description

The method ljust() returns the string left justified in a string of length width. Padding is done using the specified fillchar (default is a space). The original string is returned if width is less than len(s).

Syntax

Following is the syntax for ljust() method –

```
str.ljust(width[, fillchar])
```

Parameters

- **width** - This is string length in total after padding.
- **fillchar** - This is filler character, default is a space.

Return Value

This method returns the string left justified in a string of length width. Padding is done using the specified fillchar (default is a space). The original string is returned if width is less than len(s).

Example

The following example shows the usage of ljust() method.

```
#!/usr/bin/python3

str = "this is string example....wow!!!"

print str.ljust(50, '*')
```

Result

```
this is string example....wow!!!******************
```

String lower() Method

Description

The method lower() returns a copy of the string in which all case-based characters have been lowercased.

Syntax

Following is the syntax for lower() method –

```
str.lower()
```

Parameters

NA

Return Value

This method returns a copy of the string in which all case-based characters have been lowercased.

Example

The following example shows the usage of lower() method.

```
#!/usr/bin/python3
str = "THIS IS STRING EXAMPLE....WOW!!!"
print (str.lower())
```

Result

```
this is string example....wow!!!
```

String lstrip() Method

Description

The **lstrip()** method returns a copy of the string in which all chars have been stripped from the beginning of the string (default whitespace characters).

Syntax

Following is the syntax for lstrip() method-

```
str.lstrip([chars])
```

Parameters

chars - You can supply what chars have to be trimmed.

Return Value

This method returns a copy of the string in which all chars have been stripped from the beginning of the string (default whitespace characters).

Example

The following example shows the usage of lstrip() method.

```
#!/usr/bin/python3
str = "     this is string example....wow!!!"
print (str.lstrip())
str = "*****this is string example....wow!!!*****"
print (str.lstrip('*'))
```

Result

```
this is string example....wow!!!
this is string example....wow!!!*****
```

String maketrans() Method

Description

The **maketrans()** method returns a translation table that maps each character in the intabstring into the character at the same position in the outtab string. Then this table is passed to the translate() function.

Note: Both intab and outtab must have the same length.

Syntax

Following is the syntax for maketrans() method-

```
str.maketrans(intab, outtab]);
```

Parameters

- **intab** - This is the string having actual characters.
- **outtab** - This is the string having corresponding mapping character.

Return Value

This method returns a translate table to be used translate() function.

Example

The following example shows the usage of maketrans() method. Under this, every vowel in a string is replaced by its vowel position –

```
#!/usr/bin/python3
intab = "aeiou"
```

```
outtab = "12345"
trantab = str.maketrans(intab, outtab)
str = "this is string example....wow!!!"
print (str.translate(trantab))
```

Result

```
th3s 3s str3ng 2x1mpl2....w4w!!!
```

String max() Method

Description

The **max()** method returns the max alphabetical character from the string str.

Syntax

Following is the syntax for max() method-

```
max(str)
```

Parameters

str - This is the string from which max alphabetical character needs to be returned.

Return Value

This method returns the max alphabetical character from the string str.

Example

The following example shows the usage of max() method.

```
#!/usr/bin/python3
str = "this is a string example....really!!!"
print ("Max character: " + max(str))
str = "this is a string example....wow!!!"
print ("Max character: " + max(str))
```

Result

```
Max character: y
Max character: x
```

String min() Method

Description

The **min()** method returns the min alphabetical character from the string str.

Syntax

Following is the syntax for min() method-

```
min(str)
```

Parameters

str - This is the string from which min alphabetical character needs to be returned.

Return Value

This method returns the max alphabetical character from the string str.

Example

The following example shows the usage of min() method.

```
#!/usr/bin/python3
str = "www.tutorialspoint.com"
print ("Min character: " + min(str))
str = "TUTORIALSPOINT"
print ("Min character: " + min(str))
```

Result

```
Min character: .
Min character: A
```

String replace() Method

Description

The **replace()** method returns a copy of the string in which the occurrences of old have been replaced with new, optionally restricting the number of replacements to max.

Syntax

Following is the syntax for replace() method-

```
str.replace(old, new[, max])
```

Parameters

- **old** - This is old substring to be replaced.
- **new** - This is new substring, which would replace old substring.
- **max** - If this optional argument max is given, only the first count occurrences are replaced.

Return Value

This method returns a copy of the string with all occurrences of substring old replaced by new. If the optional argument max is given, only the first count occurrences are replaced.

Example

The following example shows the usage of replace() method.

```
#!/usr/bin/python3
str = "this is string example....wow!!! this is really string"
print (str.replace("is", "was"))
print (str.replace("is", "was", 3))
```

Result

```
thwas was string example....wow!!! thwas was really string
thwas was string example....wow!!! thwas is really string
```

String rfind() Method

Description

The **rfind()** method returns the last index where the substring str is found, or -1 if no such index exists, optionally restricting the search to string[beg:end].

Syntax

Following is the syntax for rfind() method-

```
str.rfind(str, beg=0 end=len(string))
```

Parameters

- **str** - This specifies the string to be searched.
- **beg** - This is the starting index, by default its 0.
- **end** - This is the ending index, by default its equal to the length of the string.

Return Value

This method returns last index if found and -1 otherwise.

Example

The following example shows the usage of rfind() method.

```
#!/usr/bin/python3
str1 = "this is really a string example....wow!!!"
str2 = "is"
print (str1.rfind(str2))
print (str1.rfind(str2, 0, 10))
print (str1.rfind(str2, 10, 0))
print (str1.find(str2))
print (str1.find(str2, 0, 10))
print (str1.find(str2, 10, 0))
```

Result

```
5
5
-1
2
2
-1
```

String rindex() Method

Description

The **rindex()** method returns the last index where the substring str is found, or raises an exception if no such index exists, optionally restricting the search to string[beg:end].

Syntax

Following is the syntax for rindex() method-

```
str.rindex(str, beg=0 end=len(string))
```

Parameters

- **str** - This specifies the string to be searched.

- **beg** - This is the starting index, by default its 0.
- **len** - This is ending index, by default its equal to the length of the string.

Return Value

This method returns last index if found otherwise raises an exception if str is not found.

Example

The following example shows the usage of rindex() method.

```
#!/usr/bin/python3
str1 = "this is really a string example....wow!!!"
str2 = "is"
print (str1.rindex(str2))
print (str1.rindex(str2,10))
```

Result

```
5
Traceback (most recent call last):
  File "test.py", line 5, in
    print (str1.rindex(str2,10))
ValueError: substring not found
```

String rjust() Method

Description

The **rjust()** method returns the string right justified in a string of length width. Padding is done using the specified fillchar (default is a space). The original string is returned if width is less than len(s).

Syntax

Following is the syntax for rjust() method-

```
str.rjust(width[, fillchar])
```

Parameters

- **width** - This is the string length in total after padding.
- **fillchar** - This is the filler character, default is a space.

Return Value

This method returns the string right justified in a string of length width. Padding is done using the specified fillchar (default is a space). The original string is returned if the width is less than len(s).

Example

The following example shows the usage of rjust() method.

```
#!/usr/bin/python3
str = "this is string example....wow!!!"
print (str.rjust(50, '*'))
```

Result

```
******************this is string example....wow!!!
```

String rstrip() Method

Description

The **rstrip()** method returns a copy of the string in which all chars have been stripped from the end of the string (default whitespace characters).

Syntax

Following is the syntax for rstrip() method-

```
str.rstrip([chars])
```

Parameters

chars - You can supply what chars have to be trimmed.

Return Value

This method returns a copy of the string in which all chars have been stripped from the end of the string (default whitespace characters).

Example

The following example shows the usage of rstrip() method.

```
#!/usr/bin/python3
str = "     this is string example....wow!!!     "
print (str.rstrip())
str = "*****this is string example....wow!!!*****"
```

```
print (str.rstrip('*'))
```

Result

```
    this is string example....wow!!!
*****this is string example....wow!!!
```

String split() Method

Description

The **split()** method returns a list of all the words in the string, using str as the separator (splits on all whitespace if left unspecified), optionally limiting the number of splits to num.

Syntax

Following is the syntax for split() method-

```
str.split(str="", num=string.count(str)).
```

Parameters

- **str** - This is any delimeter, by default it is space.
- **num** - this is number of lines to be made

Return Value

This method returns a list of lines.

Example

The following example shows the usage of split() method.

```
#!/usr/bin/python3
str = "this is string example....wow!!!"
print (str.split( ))
print (str.split('i',1))
print (str.split('w'))
```

Result

```
['this', 'is', 'string', 'example....wow!!!']
['th', 's is string example....wow!!!']
['this is string example....', 'o', '!!!']
```

String splitlines() Method

Description

The **splitlines()** method returns a list with all the lines in string, optionally including the line breaks (if num is supplied and is true).

Syntax

Following is the syntax for splitlines() method-

```
str.splitlines( num=string.count('\n'))
```

Parameters

num - This is any number, if present then it would be assumed that the line breaks need to be included in the lines.

Return Value

This method returns true if found matching with the string otherwise false.

Example

The following example shows the usage of splitlines() method.

```
#!/usr/bin/python3
str = "this is \nstring example....\nwow!!!"
print (str.splitlines( ))
```

Result

```
['this is ', 'string example....', 'wow!!!']
```

String startswith() Method

Description

The **startswith()** method checks whether the string starts with str, optionally restricting the matching with the given indices start and end.

Syntax

Following is the syntax for startswith() method-

```
str.startswith(str, beg=0,end=len(string));
```

Parameters

- **str** - This is the string to be checked.
- **beg** - This is the optional parameter to set start index of the matching boundary.
- **end** - This is the optional parameter to set start index of the matching boundary.

Return Value

This method returns true if found matching with the string otherwise false.

Example

The following example shows the usage of startswith() method.

```
#!/usr/bin/python3
str = "this is string example....wow!!!"
print (str.startswith( 'this' ))
print (str.startswith( 'string', 8 ))
print (str.startswith( 'this', 2, 4 ))
```

Result

```
True
True
False
```

String strip() Method

Description

The **strip()** method returns a copy of the string in which all chars have been stripped from the beginning and the end of the string (default whitespace characters).

Syntax

Following is the syntax for strip() method –

```
str.strip([chars]);
```

Parameters

chars - The characters to be removed from beginning or end of the string.

Return Value

This method returns a copy of the string in which all the chars have been stripped from the beginning and the end of the string.

Example

The following example shows the usage of strip() method.

```
#!/usr/bin/python3
str = "*****this is string example....wow!!!*****"
print (str.strip( '*' ))
```

Result

```
this is string example....wow!!!
```

String swapcase() Method

Description

The **swapcase()** method returns a copy of the string in which all the case-based characters have had their case swapped.

Syntax

Following is the syntax for swapcase() method-

```
str.swapcase();
```

Parameters

NA

Return Value

This method returns a copy of the string in which all the case-based characters have had their case swapped.

Example

The following example shows the usage of swapcase() method.

```
#!/usr/bin/python3
str = "this is string example....wow!!!"
print (str.swapcase())
str = "This Is String Example....WOW!!!"
print (str.swapcase())
```

Result

```
THIS IS STRING EXAMPLE....WOW!!!
tHIS iS sTRING eXAMPLE....wow!!!
```

String title() Method

Description

The **title()** method returns a copy of the string in which first characters of all the words are capitalized.

Syntax

Following is the syntax for title() method-

```
str.title();
```

Parameters

NA

Return Value

This method returns a copy of the string in which first characters of all the words are capitalized.

Example

The following example shows the usage of title() method.

```
#!/usr/bin/python3
str = "this is string example....wow!!!"
print (str.title())
```

Result

```
This Is String Example....Wow!!!
```

String translate() Method

Description

The method translate() returns a copy of the string in which all the characters have been translated using table (constructed with the maketrans() function in the string module), optionally deleting all characters found in the string deletechars.

Syntax

Following is the syntax for translate() method-

```
str.translate(table[, deletechars]);
```

Parameters

- **table** - You can use the maketrans() helper function in the string module to create a translation table.
- **deletechars** - The list of characters to be removed from the source string.

Return Value

This method returns a translated copy of the string.

Example

The following example shows the usage of translate() method. Under this, every vowel in a string is replaced by its vowel position.

```
#!/usr/bin/python3
from string import maketrans   # Required to call maketrans function.
intab = "aeiou"
outtab = "12345"
trantab = maketrans(intab, outtab)
str = "this is string example....wow!!!";
print (str.translate(trantab))
```

Result

```
th3s 3s str3ng 2x1mpl2....w4w!!!
```

Following is the example to delete 'x' and 'm' characters from the string-

```
#!/usr/bin/python3
from string import maketrans   # Required to call maketrans function.
intab = "aeiouxm"
outtab = "1234512"
trantab = maketrans(intab, outtab)
str = "this is string example....wow!!!";
print (str.translate(trantab))
```

Result

```
th3s 3s str3ng 21pl2....w4w!!!
```

String upper() Method

Description

The **upper()** method returns a copy of the string in which all case-based characters have been uppercased.

Syntax

Following is the syntax for upper() method –

```
str.upper()
```

Parameters

NA

Return Value

This method returns a copy of the string in which all case-based characters have been uppercased.

Example

The following example shows the usage of upper() method.

```
#!/usr/bin/python3
str = "this is string example....wow!!!"
print ("str.upper : ",str.upper())
```

Result

```
str.upper :  THIS IS STRING EXAMPLE....WOW!!!
```

String zfill() Method

Description

The **zfill()** method pads string on the left with zeros to fill width.

Syntax

Following is the syntax for zfill() method-

```
str.zfill(width)
```

Parameters

width - This is final width of the string. This is the width which we would get after filling zeros.

Return Value

This method returns padded string.

Example

The following example shows the usage of zfill() method.

```
#!/usr/bin/python3
str = "this is string example....wow!!!"
print ("str.zfill : ",str.zfill(40))
print ("str.zfill : ",str.zfill(50))
```

Result

```
str.zfill :  00000000this is string example....wow!!!
str.zfill :  000000000000000000this is string example....wow!!!
```

String isdecimal() Method

Description

The **isdecimal()** method checks whether the string consists of only decimal characters. This method are present only on unicode objects.

Note: Unlike in Python 2, all strings are represented as Unicode in Python 3. Given Below is an example illustrating it.

Syntax

Following is the syntax for isdecimal() method-

```
str.isdecimal()
```

Parameters

NA

Return Value

This method returns true if all the characters in the string are decimal, false otherwise.

Example

The following example shows the usage of isdecimal() method.

```
#!/usr/bin/python3

str = "this2016"

print (str.isdecimal())

str = "23443434"

print (str.isdecimal())
```

Result

```
False

True
```