**Strings**

However, Python does not have a character data type, a single character is simply a string with a length of 1.

Square brackets can be used to access elements of the string.

a = "Hello" # single line print  
print(a)  
  
  
b = """Lorem ipsum dolor sit amet,   
consectetur adipiscing elit,  
sed do eiusmod tempor incididunt  
ut labore et dolore magna aliqua.""" #multi line print of string  
print(b)  
  
c = '''Lorem ipsum dolor sit amet,  
consectetur adipiscing elit,  
sed do eiusmod tempor incididunt  
ut labore et dolore magna aliqua.''' # multi line string  
print(c)

Ex2:-

for x in "banana":  
 print(x) #Loopint through string  
  
**Len():**

The len() function returns the length of a string:

b = "Hello, World!"  
 print("\n")  
 print("string lenght:" ,len(b)) #string length

**In:-**

To check if a certain phrase or character is present in a string, we can use the keyword in

s = "The best things in life is work hard"  
if "hard" in s:  
 print("Yes, 'hard', is present.") # to check the hard string is present are not.(in function)

**NOT IN :**

To check if a certain phrase or character is NOT present in a string, we can use the keyword not in.

s = "The best things in life is work hard"  
 if "expensive" not in s:  
 print("Yes, 'expensive', is not present.") #not in function of string.  
  
**Slicing:**

S[beging Index:endIndex:step]

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.

b = "Hello, World!"  
print("b:",b[2:5])  
# slice from begin  
c = "Hello, World!"  
print("c:" ,c[:5])  
# slice from end  
d = "Hello, World!"  
print("d:",d[2:])

**Negative index:**

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

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

**Upper():**

upper case are not.  
a = "Hello, python"  
print("a:",a.upper())

**Lower():-**

lower case are mnot  
a = "Hello, World!"  
print("a:",a.lower())

**Strip():-**

Whitespace is the space before and/or after the actual text, and very often you want to remove this space.

c=" hello sekhar "  
print("c:",c.strip()) #The strip() method removes any whitespace from the beginning or the end:

**Split():-**

The split() method returns a list where the text between the specified separator becomes the list items.

a="python,world"  
b=a.split(",")# to split the sting into sub string  
print(b)

**concatenate string:-**

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

a="amy"  
b="jackson "  
c="beautiful"  
d=a+b+c  
print(d) # concatenate strings using +

**format():**

The format() method takes the passed arguments, formats them, and places them in the string where the placeholders {} are. Use the format() method to insert numbers into strings

Ex1:

age = 30  
txt = "My name is sekhar, and I am {}" # {} fill the number by using format operator  
print(txt.format(age))

Ex2:

The format() method takes unlimited number of arguments, and are placed into the respective placeholders:

quantity = 4  
itemno = 500  
price = 450  
myorder = "I want {} pieces of item {} for {} dollars."  
print(myorder.format(quantity, itemno, price))  
  
ex3:-  
quantity = 4  
itemno = 500  
price = 450  
myorder = "I want to pay {2} dollars for {0} pieces of item {1}."  
print(myorder.format(quantity, itemno, price))

print("the intger is:{}".format(123))  
print("the integer is :{:08d}".format(123)) # zeros precid with zero  
print("the float is {:19.3f}".format(123.456789)) # 19 space prefix and 3 spaces postfix  
print("binary:{0:b}".format(10)) #zero and binary  
print("octal:{0:o}".format(10)) #zero and octal  
print("hexdecimal:{0:x}".format(10)) # zero and hex  
print("{:7d}".format(123))#7 means spaces  
print("{:<10d}".format(190)) #left alignment  
print("{:>10d}".format(190)) #Right alignment  
print("{:=^10d}".format(190)) #center alignment using ^ symbol

**Str methods():**

1. **capitalize():**

The capitalize() method returns a string where the first character is upper case.

*syntax:-*

*string*.capitalize()

s = "hello, and welcome to my world."  
x = s.capitalize() # capitalize the information of First letter.str.capitalize(),no parameters  
print (x)  
  
s = " 30 age of python."  
x = s.capitalize() # First letter number  
print (x)  
  
s = "ashok not prepare food"  
x = s.capitalize()  
print (x)

**2.casefold():**

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

This method is similar to the [lower()](https://www.w3schools.com/python/ref_string_lower.asp) method, but the casefold() method is stronger, more aggressive, meaning that it will convert more characters into lower case, and will find more matches when comparing two strings and both are converted using the casefold() method.

Syntax

string.casefold()

s = "HARSHA IS FOODY"  
x = s.casefold() # convert in to lower case .str.casefold() , no para meters.  
print (x)

**3.center():-**

The center() method will center align the string, using a specified character (space is default) as the fill character.

Syntax

string.center(length, character)

|  |  |
| --- | --- |
| length | Required. The length of the returned string |
| character | Optional. The character to fill the missing space on each side. Default is “”  (space) |

s = "sekhar"  
x = s.center(10,"\*") # it print data in 10 characters of with string in the center and symbol  
print (x)  
  
s = "sekhar"  
x = s.center(10)  
print (x)

**4.Endswith():**

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

Syntax

string.endswith(value, start, end)

|  |  |
| --- | --- |
| 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 |

txt = "Hello, welcome to my world."  
x = txt.endswith(".") # if it ends with . it returns True of result.  
print("x:",x)  
  
txt = "Hello, welcome to my world."  
x = txt.endswith("my world.") # my world string is endswith word it returns results True.  
print("x1:",x)  
  
txt = "Hello, welcome to my world."  
x = txt.endswith("my world.", 5, 11) # the string in between 5 to 11 characters result is True other wise False  
print("x2",x)

5.**Expandtabs():**

The expandtabs() method sets the tab size to the specified number of whitespaces.

Syntax

string.expandtabs(tabsize)

Parameter Values

|  |  |
| --- | --- |
| Parameter | Description |
| tabsize | Optional. A number specifying the tabsize. Default tabsize is 8 |

txt = "H\te\tl\tl\to"  
  
print(txt)  
print(txt.expandtabs()) # string.expandtabs(tabsize) defult tab size is 8.  
print(txt.expandtabs(2))  
print(txt.expandtabs(4))  
print(txt.expandtabs(10))

6.**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()](https://www.w3schools.com/python/ref_string_index.asp) 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 |

txt = "Hello, welcome to my world."  
x = txt.find("welcome")  
print(x)  
  
txt = "Hello, welcome to my world."  
x = txt.find("e")  
print("x:",x)  
  
txt = "Hello, welcome to my world."  
x = txt.find("e", 5, 10)  
print("x:",x)  
  
txt = "Hello, welcome to my world."  
print(txt.find("q"))  
print(txt.index("q"))

**7.Isalnum():**

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

Syntax:

String.isalnum ()

txt = "Company12"  
x = txt.isalnum()  
print(x)  
  
txt = "Company 12"  
x = txt.isalnum()  
print(x)

8.**Isalpha():**

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

Syntax:

String.isalpha()

txt = "CompanyX"  
x = txt.isalpha()  
print(x)  
  
txt = "Company10"  
x = txt.isalpha()  
print(x)

9.**Isdecimal():**

The isdecimal() method returns True if all the characters are decimals (0-9).

This method is used on unicode objects.

Syntax:

*string*.isdecimal()

txt = "\u0033" #unicode for 3  
x = txt.isdecimal()  
print(x)  
  
a = "\u0030" #unicode for 0  
b = "\u0047" #unicode for G  
print(a.isdecimal())  
print(b.isdecimal())

10.**Isdigit():**

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

Exponents, like ², are also considered to be a digit.

Syntax:

*string*.isdigit()

txt = "50800"  
x = txt.isdigit()  
print(x)  
  
a = "\u0030" #unicode for 0  
b = "\u00B2" #unicode for ²  
print(a.isdigit())  
print(b.isdigit())

11**.Isidentifer():**

The isidentifier() method returns True if the string is a valid identifier, otherwise False.

A string is considered a valid identifier if it only contains alphanumeric letters (a-z) and (0-9), or underscores (\_). A valid identifier cannot start with a number, or contain any spaces.

Syntax:

*string*.isidentifier()

txt = "Demo"  
x = txt.isidentifier()  
print(x)  
  
a = "MyFolder"  
b = "Demo002"  
c = "2bring"  
d = "my demo"  
print(a.isidentifier())  
print(b.isidentifier())  
print(c.isidentifier())  
print(d.isidentifier())

12.**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()

txt = "hello world!"  
x = txt.islower()  
print(x)  
  
a = "Hello world!"  
b = "hello 123"  
c = "mynameisPeter"  
print(a.islower())  
print(b.islower())  
print(c.islower())

13.**Isnumeric():**

The isnumeric() method returns True if all the characters are numeric (0-9), otherwise False.

Exponents, like ² and ¾ are also considered to be numeric values.

"-1" and "1.5" are NOT considered numeric values, because *all* the characters in the string must be numeric, and the - and the . are not.

Syntax

string.isnumeric()

txt = "565543"  
z="s123"  
  
x = txt.isnumeric()  
y=z.isnumeric()  
print(x)  
print(y)

14.**Isprintable():**

The isprintable() method returns True if all the characters are printable, otherwise False.

Example of none printable character can be carriage return and line feed.

Syntax

string.isprintable()

txt = "Hello sekhar"  
z="\n 12sekhar"  
x = txt.isprintable()  
y=z.isprintable()  
print("x:",x)  
print("y:",y)

15**.Isspace():**

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

Syntax

string.isspace()

txt = " "  
x = txt.isspace()  
print(x)  
  
txt = " s "  
x = txt.isspace()  
print(x)

16.**Istitle():**

The istitle() method returns True if all words in a text start with a upper case letter, AND the rest of the word are lower case letters, otherwise False.

Symbols and numbers are ignored

Syntax

string.istitle()

#Check if each word start with an upper case letter:  
  
a = "hELLO AND WELCOME TO MY WORLD"  
b = "Hello"  
c = "22 Names"  
d = "This Is %'!?"  
  
print("a:",a.istitle())  
print("b:",b.istitle())  
print("c:",c.istitle())  
print("d:",d.istitle()

17.**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()

#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:",a.isupper())  
print("b:",b.isupper())  
print("c:",c.isupper())

18.**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)

Parameter Values

|  |  |
| --- | --- |
| Parameter | Description |
| iterable | Required. Any iterable object where all the returned values are strings |

myTuple = ("John", "Peter", "Vicky")  
  
x = "@".join(myTuple) # using @ to join the string  
print(x)

19.**ljust():**

The ljust() method will left align the string, using a specified character (space is default) as the fill character.

Syntax

string.ljust(length, character)

Parameter Values

|  |  |
| --- | --- |
| Parameter | Description |
| length | Required. The length of the returned string |
| character | Optional. A character to fill the missing space (to the right of the string). Default is " " (space). |

#Return a 20 characters long, left justified version of the word "banana":  
  
txt = "banana"  
x = txt.ljust(20)  
print(x, "is my favorite fruit.")

20.**Lower():**

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

 Symbols and Numbers are ignored.

Syntax

string.lower()

#Lower case the string:  
  
txt = "Hello my FRIENDS"  
  
x = txt.lower()  
  
print(x)

21.**lstrip():**

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

string.lstrip(characters)

Parameter Values

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

#Remove spaces to the left of the string:  
  
txt = " banana "  
  
x = txt.lstrip()  
  
print("of all fruits", x, "is my favorite")

22.**Upper()**:

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

 Symbols and Numbers are ignored.

Syntax

string.upper()

#Upper case the string:  
  
txt = "Hello my friends"  
  
x = txt.upper()  
  
print(x)

23.**Swapcase():**

The swapcase() method returns a string where all the upper case letters are lower case and vice versa.

Syntax

string.swapcase()

# the lower case letters upper case and the upper case letters lower case:  
  
txt = "Hello My Name Is sekhar"  
  
x = txt.swapcase()  
print(x)

24.**Zfill():**

The zfill() method adds zeros (0) at the beginning of the string, until it reaches the specified length.

If the value of the len parameter is less than the length of the string, no filling is done.

Syntax

string.zfill(len)

Parameter Values

|  |  |
| --- | --- |
| Parameter | Description |
| len | Required. A number specifying the position of the element you want to remove |

a = "hello"  
b = "welcome to the jungle"  
c = "10.000"  
  
print(a.zfill(10)) #Fill the string with zeros until it is 10 characters long:  
print(b.zfill(10))  
print(c.zfill(10))

25.**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()

#Make the first letter in each word upper case:  
  
txt = "Welcome to my world"  
x = txt.title()  
print(x)  
  
txt = "Welcome to my 2nd python"  
y= txt.title()  
print(y)  
  
#Note that the first letter after a non-alphabet letter is converted into a upper case letter:  
  
txt = "hello b2b2b2 and 3g3g3g"  
  
z = txt.title()  
  
print(z)

26.**Splitlines():**

The splitlines() method splits a string into a list. The splitting is done at line breaks.

Syntax

string.splitlines(keeplinebreaks)

|  |  |
| --- | --- |
| Parameter | Description |
| keeplinebreaks | Optional. Specifies if the line breaks should be included (True), or not (False). Default value is False |

txt = "Thank you for the music\nWelcome to the jungle"  
  
x = txt.splitlines()  
  
print(x)

27.**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 charac |

Ex1:

txt = "     banana     "  
  
x = txt.rstrip()  
  
print("of all fruits", x, "is my favorite")

Ex2:

txt = "banana,,,,,ssqqqww....."  
  
x = txt.rstrip(",.qsw")  
  
print(x) #Remove the trailing characters if they are commas, s, q, or w:

27.**Rsplit():**

The rsplit() method splits a string into a list, starting from the right.

If no "max" is specified, this method will return the same as the [split()](https://www.w3schools.com/python/ref_string_split.asp) method.

Syntax

string.rsplit(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" |

Ex1: split a string into a list, using comma, followed by a space (, ) as the separator:

txt = "apple, banana, cherry"  
  
x = txt.rsplit(", ")  
  
print(x)

Ex2:

Split the string into a list with maximum 2 items:

txt = "apple, banana, cherry"  
  
# setting the maxsplit parameter to 1, will return a list with 2 elements!  
x = txt.rsplit(", ", 1)  
  
print(x)

28**.Rfind():**

The rfind() method finds the last occurrence of the specified value.

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

The rfind() method is almost the same as the [rindex()](https://www.w3schools.com/python/ref_string_rindex.asp) method. See example below.

Syntax

string.rfind(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 |

txt = "Mi casa, su casa."  
x = txt.rfind("casa")  
print("x:",x)  
  
txt = "Hello, welcome to my world."  
x = txt.rfind("e")  
print("x:",x)  
#Where in the text is the last occurrence of the letter "e" when you only search between position 5 and 10?:  
txt = "Hello, welcome to my world."  
x = txt.rfind("e", 5, 10)  
print(x)  
  
txt = "Hello, welcome to my world." #f the value is not found, the rfind() method returns -1, but the rindex() method will raise an exception:  
print(txt.rfind("q"))  
print(txt.rindex("q"))

29.**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 |

txt = "I like bananas" #Replace the word "bananas":  
x = txt.replace("bananas", "apples")  
print(x)  
#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)

#Replace the two first occurrence of the word "one"  
txt = "one one was a race horse, two two was one too."  
x = txt.replace("one", "three", 2)  
print(x)

30**.Rjust():**

The rjust() method will right align the string, using a specified character (space is default) as the fill character.

Syntax:

string.rjust(length, character)

Parameter Values

|  |  |
| --- | --- |
| Parameter | Description |
| length | Required. The length of the returned string |
| character | Optional. A character to fill the missing space (to the left of the string). Default is " " (space). |

#Return a 20 characters long, right justified version of the word "banana":  
  
txt = "banana"  
x = txt.rjust(20)  
print(x, "is my favorite fruit.")  
  
#Using the letter "O" as the padding character:  
  
txt = "banana"  
x = txt.rjust(20, "%")  
print(x)

31.**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 |

#Check if the string starts with "Hello":  
  
txt = "Hello, welcome to my world."  
x = txt.startswith("Hello")  
print(x)  
  
#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)

32.**Rpartition():**

The rpartition() method searches for the last occurrence of 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.

Syntax

string.rpartition(value)

Parameter Values

|  |  |
| --- | --- |
| Parameter | Description |
| value | Required. The string to search for |

txt = "I could eat bananas all day, bananas are my favorite fruit"  
x = txt.rpartition("bananas")  
print(x)  
  
#If the specified value is not found, the rpartition() method returns a tuple containing: 1 - an empty string, 2 - an empty string, 3 - the whole string:  
  
txt = "I could eat bananas all day, bananas are my favorite fruit"  
  
x = txt.rpartition("apples")  
  
print(x)