Data Science and Generative AI

by SATISH @

https://sathyatech.com

Python Programming

Python Functions

A function is a block of code which only runs when it is called.
You can pass data, known as parameters, into a function.
A function can return data as a result.

Creating a Function

In Python a function is defined using the def keyword:

Example

```
def my_function():
  print("Hello from a function")
```

Calling a Function

To call a function, use the function name followed by parenthesis:

Example

```
def my_function():
  print("Hello from a function")

my_function()
```

Arguments

Information can be passed into functions as arguments.
Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.
The following example has a function with one argument (fname). When the function is called, we pass along a first name, which is used inside the function to print the full name:

Example

```
def my_function(fname):
  print(fname + " Refsnes")

my_function("Emil")
my_function("Tobias")
my_function("Linus")
```

Arguments are often shortened to *args* in Python documentations.

Parameters or Arguments?

The terms *parameter* and *argument* can be used for the same thing: information that are passed into a function.
From a function's perspective:
A parameter is the variable listed inside the parentheses in the function definition.
An argument is the value that is sent to the function when it is called.

Number of Arguments

By default, a function must be called with the correct number of arguments. Meaning that if your function expects 2 arguments, you have to call the function with 2 arguments, not more, and not less.

Example

This function expects 2 arguments, and gets 2 arguments:

```
def my_function(fname, lname):
  print(fname + " " + lname)

my_function("Emil", "Refsnes")
```

If you try to call the function with 1 or 3 arguments, you will get an error:

Example

This function expects 2 arguments, but gets only 1:

```
def my_function(fname, lname):
  print(fname + " " + lname)

my_function("Emil")
```

Arbitrary Arguments, *args

If you do not know how many arguments that will be passed into your function, add a * before the parameter name in the function definition.

This way the function will receive a *tuple* of arguments, and can access the items accordingly:

Example

If the number of arguments is unknown, add a * before the parameter name:

```
def my_function(*kids):
  print("The youngest child is " + kids[2])

my_function("Emil", "Tobias", "Linus")
```

Arbitrary Arguments are often shortened to **args* in Python documentations.

Keyword Arguments

You can also send arguments with the *key* = *value* syntax.

This way the order of the arguments does not matter.

Example

```
def my_function(child3, child2, child1):
  print("The youngest child is " + child3)

my_function(child1 = "Emil", child2 = "Tobias", child3 = "Linus")
```

The phrase *Keyword Arguments* are often shortened to *kwargs* in Python documentations.

Arbitrary Keyword Arguments, **kwargs

If you do not know how many keyword arguments that will be passed into your function, add two asterisk: ** before the parameter name in the function definition.

This way the function will receive a *dictionary* of arguments, and can access the items accordingly:

Example

If the number of keyword arguments is unknown, add a double ** before the parameter name:

```
def my_function(**kid):
  print("His last name is " + kid["lname"])

my_function(fname = "Tobias", lname = "Refsnes")
```

Arbitrary Kword Arguments are often shortened to ***kwargs* in Python documentations.

Default Parameter Value

The following example shows how to use a default parameter value.

If we call the function without argument, it uses the default value:

Example

```
def my_function(country = "Norway"):
  print("I am from " + country)

my_function("Sweden")
my_function("India")
my_function()
my_function("Brazil")
```

Passing a List as an Argument

You can send any data types of argument to a function (string, number, list, dictionary etc.), and it will be treated as the same data type inside the function.

E.g. if you send a List as an argument, it will still be a List when it reaches the function:

Example

```
def my_function(food):
  for x in food:
    print(x)

fruits = ["apple", "banana", "cherry"]

my_function(fruits)
```

Return Values

To let a function return a value, use the return statement:

Example

```
def my_function(x):
  return 5 * x
```

```
print(my_function(3))
print(my_function(5))
print(my_function(9))
```

The pass Statement

function definitions cannot be empty, but if you for some reason have a function definition with no content, put in the pass statement to avoid getting an error.

Example

```
def myfunction():
  pass
```

Positional-Only Arguments

You can specify that a function can have ONLY positional arguments, or ONLY keyword arguments.

To specify that a function can have only positional arguments, add , / after the arguments:

Example

```
def my_function(x, /):
  print(x)

my_function(3)
```

Without the , / you are actually allowed to use keyword arguments even if the function expects positional arguments:

Example

```
def my_function(x):
  print(x)

my_function(x = 3)
```

But when adding the , / you will get an error if you try to send a keyword argument:

Example

```
def my_function(x, /):
  print(x)

my_function(x = 3)
```

Keyword-Only Arguments

To specify that a function can have only keyword arguments, add *, *before* the arguments:

Example

```
def my_function(*, x):
  print(x)

my_function(x = 3)
```

Without the *, you are allowed to use positionale arguments even if the function expects keyword arguments:

Example

```
def my_function(x):
  print(x)

my_function(3)
```

But when adding the *, / you will get an error if you try to send a positional argument:

Example

```
def my_function(*, x):
  print(x)

my_function(3)
```

Combine Positional-Only and Keyword-Only

You can combine the two argument types in the same function.

Any argument *before* the / , are positional-only, and any argument *after* the *, are keyword-only.

Example

```
def my_function(a, b, /, *, c, d):
  print(a + b + c + d)

my_function(5, 6, c = 7, d = 8)
```

Recursion

Python also accepts function recursion, which means a defined function can call itself.

Recursion is a common mathematical and programming concept. It means that a function calls itself. This has the benefit of meaning that you can loop through data to reach a result.

The developer should be very careful with recursion as it can be quite easy to slip into writing a function which never terminates, or one that uses excess amounts of memory or processor power. However, when written correctly recursion can be a very efficient and mathematically-elegant approach to programming.

In this example, tri_recursion() is a function that we have defined to call itself ("recurse"). We use the k variable as the data, which decrements (-1) every time we recurse. The recursion ends when the condition is not greater than 0 (i.e. when it is 0).

To a new developer it can take some time to work out how exactly this works, best way to find out is by testing and modifying it.

Example

Recursion Example

```
def tri_recursion(k):
  if(k > 0):
    result = k + tri_recursion(k - 1)
    print(result)
```

```
  else:
    result = 0
  return result

print("\n\nRecursion Example Results")
tri_recursion(6)
```

Exercise:
Create a function named my_function.
:
print("Hello from a function")

Python String Methods

Python has a set of built-in methods that you can use on strings.

Note: All string methods returns new values. They do not change the original string.

Method	Description
capitalize()	Converts the first character to upper case
casefold()	Converts string into lower case
center()	Returns a centered string
count()	Returns the number of times a specified value occurs in a string
endswith()	Returns true if the string ends with the specified value
expandtabs()	Sets the tab size of the string
find()	Searches the string for a specified value and returns the position of where it was found
format()	Formats specified values in a string
format_map()	Formats specified values in a string

index()	Searches the string for a specified value and returns the position of where it was found
isalnum()	Returns True if all characters in the string are alphanumeric
isalpha()	Returns True if all characters in the string are in the alphabet
isascii()	Returns True if all characters in the string are ascii characters
isdecimal()	Returns True if all characters in the string are decimals
isdigit()	Returns True if all characters in the string are digits
isidentifier()	Returns True if the string is an identifier
islower()	Returns True if all characters in the string are lower case
isnumeric()	Returns True if all characters in the string are numeric
isprintable()	Returns True if all characters in the string are printable
isspace()	Returns True if all characters in the string are whitespaces
istitle()	Returns True if the string follows the rules of a title
isupper()	Returns True if all characters in the string are upper case
join()	Converts the elements of an iterable into a string
ljust()	Returns a left justified version of the string
lower()	Converts a string into lower case
lstrip()	Returns a left trim version of the string
maketrans()	Returns a translation table to be used in translations
partition()	Returns a tuple where the string is parted into three parts
replace()	Returns a string where a specified value is replaced with a specified value
rfind()	Searches the string for a specified value and returns the last position of where it was found

rindex()	Searches the string for a specified value and returns the last position of where it was found
rjust()	Returns a right justified version of the string
rpartition()	Returns a tuple where the string is parted into three parts
rsplit()	Splits the string at the specified separator, and returns a list
rstrip()	Returns a right trim version of the string
split()	Splits the string at the specified separator, and returns a list
splitlines()	Splits the string at line breaks and returns a list
startswith()	Returns true if the string starts with the specified value
strip()	Returns a trimmed version of the string
swapcase()	Swaps cases, lower case becomes upper case and vice versa
title()	Converts the first character of each word to upper case
translate()	Returns a translated string
upper()	Converts a string into upper case
zfill()	Fills the string with a specified number of 0 values at the beginning

Python String capitalize() Method

Upper case the first letter in this sentence:

```
txt = "hello, and welcome to my world."
x = txt.capitalize()
print (x)
```

Definition and Usage

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

Syntax

string.capitalize()

Parameter Values

No parameters

More Examples

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 String casefold() Method

Example

Make the string lower case:

```
txt = "Hello, And Welcome To My World!"
x = txt.casefold()
print(x)
```

Definition and Usage

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

This method is similar to the lower() 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()

Parameter Values

No parameters

Python String center() Method

Example

Print the word "banana", taking up the space of 20 characters, with "banana" in the middle:

```
txt = "banana"
x = txt.center(20)
print(x)
```

Definition and Usage

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

Syntax

string.center(*length, character*)

Parameter Values

Parameter	Description
length	Required. The length of the returned string
character	Optional. The character to fill the missing space on each side. Default is " " (space)

More Examples

Example

Using the letter "O" as the padding character:

```
txt = "banana"
x = txt.center(20, "O")
print(x)
```

Python String count() Method

Example

Return the number of times the value "apple" appears in the string:

```
txt = "I love apples, apple are my favorite fruit"
x = txt.count("apple")
print(x)
```

Definition and Usage

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

---

More Examples

Example

Search from position 10 to 24:

```
txt = "I love apples, apple are my favorite fruit"
x = txt.count("apple", 10, 24)

print(x)
```

Ex-2:

```
email=str(input("Enter an email : ") )

x = email.count('@')

if x == 1:

  print("Valid format")

else:

  print("Invalid format!")
```

Python String endswith() Method

Example

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

```
txt = "Hello, welcome to my world."
x = txt.endswith(".")
print(x)
```

Definition and Usage

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

---

More Examples

Example

Check if the string ends with the phrase "my world.":

```
txt = "Hello, welcome to my world."
x = txt.endswith("my world.")
print(x)
```

Example

Check if position 5 to 11 ends with the phrase "my world.":

```
txt = "Hello, welcome to my world."
x = txt.endswith("my world.", 5, 11)
print(x)
```

Python String expandtabs() Method

Example

Set the tab size to 2 whitespaces:

```
txt = "H\te\tl\tl\to"
x =  txt.expandtabs(2)
print(x)
```

Definition and Usage

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

More Examples

Example

See the result using different tab sizes:

```
txt = "H\te\tl\tl\to"

print(txt)
print(txt.expandtabs())
print(txt.expandtabs(2))
print(txt.expandtabs(4))
print(txt.expandtabs(10))
```

Python String find() Method

Example

Where in the text is the word "welcome"?:

```
txt = "Hello, welcome to my world."
x = txt.find("welcome")
print(x)
```

Definition and Usage

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

---

More Examples

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)
```

Example

Where in the text is the first occurrence of the letter "e" when you only search between position 5 and 10?:

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

x = txt.find("e", 5, 10)

print(x)
```

Example

If the value is not found, the find() method returns -1, but the index() method will raise an exception:

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

print(txt.find("q"))
print(txt.index("q"))
```

Python String format() Method

Example

Insert the price inside the placeholder, the price should be in fixed point, two-decimal format:

```
txt = "For only {price:.2f} dollars!"
print(txt.format(price = 49))
```

Definition and Usage

The format() method formats the specified value(s) and insert them inside the string's placeholder.

The placeholder is defined using curly brackets: {}. Read more about the placeholders in the Placeholder section below.

The format() method returns the formatted string.

Syntax

string.format(*value1, value2...*)

Parameter Values

Parameter	Description
value1, value2...	Required. One or more values that should be formatted and inserted in the string.

The values are either a list of values separated by commas, a key=value list, or a combination of both.

The values can be of any data type.

The Placeholders

The placeholders can be identified using named indexes {price}, numbered indexes {0}, or even empty placeholders {}.

Example

Using different placeholder values:

```
txt1 = "My name is {fname}, I'm {age}".format(fname = "John", age = 36)
txt2 = "My name is {0}, I'm {1}".format("John",36)
txt3 = "My name is {}, I'm {}".format("John",36)
```

Python String index() Method

Example

Where in the text is the word "welcome"?:

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

x = txt.index("welcome")
print(x)
```

Definition and Usage

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

The index() method raises an exception if the value is not found.

The index() method is almost the same as the find() method, the only difference is that the find() method returns -1 if the value is not found. (See example below)

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

---

More Examples

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)
```

Example

Where in the text is the first occurrence of the letter "e" when you only search between position 5 and 10?:

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

Example

If the value is not found, the find() method returns -1, but the index() method will raise an exception:

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

print(txt.find("q"))
print(txt.index("q"))
```

Python String isalnum() Method

Check if all the characters in the text are alphanumeric:

```
txt = "Company12"
x = txt.isalnum()
print(x)
```

Definition and Usage

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.

More Examples

Example

Check if all the characters in the text is alphanumeric:

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

Python String isalpha() Method

Check if all the characters in the text are letters:

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

Definition and Usage

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.

More Examples

Example

Check if all the characters in the text is alphabetic:

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

Python String isascii() Method

Example

Check if all the characters in the text are ascii characters:

```
txt = "Company123"
x = txt.isascii()
print(x)
```

Definition and Usage

The isascii() method returns True if all the characters are ascii characters (a-z).

Syntax

string.isascii()

Parameter Values

No parameters.

Python String isdecimal() Method

Example

Check if all the characters in a string are decimals (0-9):

```
txt = "1234"
x = txt.isdecimal()

print(x)
```

Definition and Usage

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

This method can also be used on unicode objects. See example below.

Syntax

string.isdecimal()

Parameter Values

No parameters.

More Examples

Example

Check if all the characters in the unicode are decimals:

```
a = "\u0030" #unicode for 0
b = "\u0047" #unicode for G

print(a.isdecimal())
print(b.isdecimal())
```

Python String isidentifier() Method

Example

Check if the string is a valid identifier:

```
txt = "Demo"
x = txt.isidentifier()
print(x)
```

Definition and Usage

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()

Parameter Values

No parameters.

More Examples

Example

Check if the strings are valid identifiers:

```
a = "MyFolder"
b = "Demo002"
c = "2bring"
d = "my demo"

print(a.isidentifier())
print(b.isidentifier())
print(c.isidentifier())
print(d.isidentifier())
```

Python String islower() Method

Example

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

```
txt = "hello world!"
x = txt.islower()
print(x)
```

Definition and Usage

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()

Parameter Values

No parameters.

More Examples

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())
```

Python String isnumeric() Method

Example

Check if all the characters in the text are numeric:

```
txt = "565543"

x = txt.isnumeric()
print(x)
```

Definition and Usage

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()

Parameter Values

No parameters.

More Examples

Example

Check if the characters are numeric:

```
a = "\u0030" #unicode for 0
b = "\u00B2" #unicode for &sup2;
c = "10km2"
d = "-1"
e = "1.5"

print(a.isnumeric())
print(b.isnumeric())
```

```
print(c.isnumeric())
print(d.isnumeric())
print(e.isnumeric())
```

Python String isprintable() Method

Example

Check if all the characters in the text are printable:

```
txt = "Hello! Are you #1?"
x = txt.isprintable()
print(x)
```

Definition and Usage

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()

Parameter Values

No parameters.

Python String isspace() Method

Example

Check if all the characters in the text are whitespaces:

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

Definition and Usage

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

Syntax

string.isspace()

Parameter Values

No parameters.

More Examples

Example

Check if all the characters in the text are whitespaces:

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

Python String istitle() Method

Example

Check if each word start with an upper case letter:

```
txt = "Hello, And Welcome To My World!"
x = txt.istitle()
print(x)
```

Definition and Usage

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()

Parameter Values

No parameters.

More Examples

Example

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.istitle())
print(b.istitle())
print(c.istitle())
print(d.istitle())
```

Python String isupper() Method

Example

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

```
txt = "THIS IS NOW!"
x = txt.isupper()
print(x)
```

Definition and Usage

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()

Parameter Values

No parameters.

More Examples

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())
```

Python String isdigit() Method

Example

Check if all the characters in the text are digits:

```
txt = "50800"
x = txt.isdigit()
print(x)
```

Definition and Usage

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()

Parameter Values

No parameters.

More Examples

Example

Check if all the characters in the text are digits:

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

Python String join() Method

Example

Join all items in a tuple into a string, using a hash character as separator:

```
myTuple = ("John", "Peter", "Vicky")
x = "#".join(myTuple)
print(x)
```

Definition and Usage

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

---

More Examples

Example

Join all items in a dictionary into a string, using the word "TEST" as separator:

```
myDict = {"name": "John", "country": "Norway"}
mySeparator = "TEST"
x = mySeparator.join(myDict)
print(x)
```

Python String lower() Method

Example

Lower case the string:

```
txt = "Hello my FRIENDS"
x = txt.lower()
print(x)
```

Definition and Usage

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

Symbols and Numbers are ignored.

Syntax

string.lower()

Parameter Values

No parameters

Python String maketrans() Method

Example

Create a mapping table, and use it in the translate() method to replace any "S" characters with a "P" character:

```
txt = "Hello Sam!"
mytable = str.maketrans("S", "P")
print(txt.translate(mytable))
```

Definition and Usage

The maketrans() method returns a mapping table that can be used with the translate() method to replace specified characters.

Syntax

str.maketrans(*x, y, z*)

Parameter Values

Parameter	Description
x	Required. If only one parameter is specified, this has to be a dictionary describing how to perform the replace. If two or more parameters are specified, this parameter has to be a string specifying the characters you want to replace.
y	Optional. A string with the same length as parameter x. Each character in the first parameter will be replaced with the corresponding character in this string.
z	Optional. A string describing which characters to remove from the original string.

More Examples

Example

Use a mapping table to replace many characters:

```
txt = "Hi Sam!"
x = "mSa"
y = "eJo"
mytable = str.maketrans(x, y)
print(txt.translate(mytable))
```

Example

The third parameter in the mapping table describes characters that you want to remove from the string:

```
txt = "Good night Sam!"
x = "mSa"
y = "eJo"
z = "odnght"
mytable = str.maketrans(x, y, z)
print(txt.translate(mytable))
```

Example

The maketrans() method itself returns a dictionary describing each replacement, in unicode:

```
txt = "Good night Sam!"
x = "mSa"
y = "eJo"
z = "odnght"
print(str.maketrans(x, y, z))
```

Python String partition() Method

Example

Search for the word "bananas", and return a tuple with three elements:

1 - everything before the "match"
2 - the "match"
3 - everything after the "match"

```
txt = "I could eat bananas all day"
x = txt.partition("bananas")
print(x)
```

Definition and Usage

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 Values

Parameter Description

value Required. The string to search for

More Examples

Example

If the specified value is not found, the partition() method returns a tuple containing: 1 - the whole string, 2 - an empty string, 3 - an empty string:

```
txt = "I could eat bananas all day"
x = txt.partition("apples")
print(x)
```

Python String replace() Method

Example

Replace the word "bananas":

```
txt = "I like bananas"
x = txt.replace("bananas", "apples")
print(x)
```

Definition and Usage

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

Note:*All* occurrences of the specified phrase will be replaced, if nothing else is specified.

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

---

More Examples

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)
```

Example

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)
```

Python String rfind() Method

Example

Where in the text is the *last* occurrence of the string "casa"?:

```
txt = "Mi casa, su casa."
x = txt.rfind("casa")
print(x)
```

Definition and Usage

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() 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

---

More Examples

Example

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

```
txt = "Hello, welcome to my world."
x = txt.rfind("e")
print(x)
```

Example

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)
```

Example

If the value is not found, the rfind() method returns -1, but the rindex() method will raise an exception:

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

print(txt.rfind("q"))
print(txt.rindex("q"))
```

Python String rindex() Method

Example

Where in the text is the *last* occurrence of the string "casa"?:

```
txt = "Mi casa, su casa."
x = txt.rindex("casa")
print(x)
```

Definition and Usage

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

The rindex() method raises an exception if the value is not found.

The rindex() method is almost the same as the rfind() method. See example below.

Syntax

string.rindex(*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

---

More Examples

Example

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

```
txt = "Hello, welcome to my world."
x = txt.rindex("e")
print(x)
```

Example

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.rindex("e", 5, 10)
print(x)
```

Example

If the value is not found, the rfind() method returns -1, but the rindex() method will raise an exception:

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

print(txt.rfind("q"))
print(txt.rindex("q"))
```

Python String ljust() Method

Example

Return a 20 characters long, left justified version of the word "banana":

```
txt = "banana"
x = txt.ljust(20)

print(x, "is my favorite fruit.")
```

Note: In the result, there are actually 14 whitespaces to the right of the word banana.

Definition and Usage

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).

More Examples

Example

Using the letter "O" as the padding character:

```
txt = "banana"
x = txt.ljust(20, "O")
print(x)
```

Python String rjust() Method

Example

Return a 20 characters long, right justified version of the word "banana":

```
txt = "banana"
x = txt.rjust(20)
print(x, "is my favorite fruit.")
```

Note: In the result, there are actually 14 whitespaces to the left of the word banana.

Definition and Usage

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).

More Examples

Example

Using the letter "O" as the padding character:

```
txt = "banana"
x = txt.rjust(20, "O")
print(x)
```

Python String rsplit() Method

Example

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

```
txt = "apple, banana, cherry"

x = txt.rsplit(", ")

print(x)
```

Definition and Usage

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() method.

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

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"

More Examples

Example

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)
```

Python String rstrip() Method

Example

Remove any white spaces at the end of the string:

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

Definition and Usage

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

---

More Examples

Example

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

```
txt = "banana,,,,,ssqqqww....."
x = txt.rstrip(",.qsw")
print(x)
```

Python String split() Method

Split a string into a list where each word is a list item:

```
txt = "welcome to the jungle"
x = txt.split()
print(x)
```

Definition and Usage

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"

More Examples

Example

Split the string, using comma, followed by a space, as a separator:

```
txt = "hello, my name is Peter, I am 26 years old"
x = txt.split(", ")
print(x)
```

Example

Use a hash character as a separator:

```
txt = "apple#banana#cherry#orange"
x = txt.split("#")

print(x)
```

Example

Split the string into a list with max 2 items:

```
txt = "apple#banana#cherry#orange"

# setting the maxsplit parameter to 1, will return a list with 2 elements!
x = txt.split("#", 1)

print(x)
```

Python String split() Method

Example

Split a string into a list where each word is a list item:

```
txt = "welcome to the jungle"
x = txt.split()
print(x)
```

Definition and Usage

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"

More Examples

Example

Split the string, using comma, followed by a space, as a separator:

```
txt = "hello, my name is Peter, I am 26 years old"
x = txt.split(", ")
print(x)
```

Example

Use a hash character as a separator:

```
txt = "apple#banana#cherry#orange"
x = txt.split("#")
print(x)
```

Python String splitlines() Method

Example

Split a string into a list where each line is a list item:

```
txt = "Thank you for the music\nWelcome to the jungle"

x = txt.splitlines()
print(x)
```

Definition and Usage

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

Syntax

string.splitlines(*keeplinebreaks*)

Parameter Values

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

---

Python String startswith() Method

Example

Check if the string starts with "Hello":

```
txt = "Hello, welcome to my world."
x = txt.startswith("Hello")
print(x)
```

Definition and Usage

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

---

More Examples

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)
```

Python String strip() Method

Example

Remove spaces at the beginning and at the end of the string:

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

Definition and Usage

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

More Examples

Example

Remove the leading and trailing characters:

```
txt = ",,,,,rrttgg.....banana....rrr"
x = txt.strip(",.grt")
print(x)
```

Python String swapcase() Method

Example

Make the lower case letters upper case and the upper case letters lower case:

```
txt = "Hello My Name Is PETER"
x = txt.swapcase()
print(x)
```

Definition and Usage

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

Syntax

string.swapcase()

Parameter Values

No parameters.

Python String title() Method

Example

Make the first letter in each word upper case:

```
txt = "Welcome to my world"
x = txt.title()
print(x)
```

Definition and Usage

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()

Parameter Values

No parameters.

More Examples

Example

Make the first letter in each word upper case:

```
txt = "Welcome to my 2nd world"
x = txt.title()
print(x)
```

Example

Note that the first letter after a non-alphabet letter is converted into a upper case letter:

```
txt = "hello b2b2b2 and 3g3g3g"

x = txt.title()
print(x)
```

Python String translate() Method

Example

Replace any "S" characters with a "P" character:

```
#use a dictionary with ascii codes to replace 83 (S) with 80 (P):
mydict = {83:  80}
txt = "Hello Sam!"
print(txt.translate(mydict))
```

Definition and Usage

The translate() method returns a string where some specified characters are replaced with the character described in a dictionary, or in a mapping table.

Use the maketrans() method to create a mapping table.

If a character is not specified in the dictionary/table, the character will not be replaced.

If you use a dictionary, you must use ascii codes instead of characters.

Syntax

string.translate(*table*)

Parameter Values

Parameter	Description
table	Required. Either a dictionary, or a mapping table describing how to perform the replace

---

More Examples

Example

Use a mapping table to replace "S" with "P":

```
txt = "Hello Sam!"
mytable = str.maketrans("S", "P")
print(txt.translate(mytable))
```

Example

Use a mapping table to replace many characters:

```
txt = "Hi Sam!"
x = "mSa"
y = "eJo"
mytable = str.maketrans(x, y)
print(txt.translate(mytable))
```

Example

The third parameter in the mapping table describes characters that you want to remove from the string:

```
txt = "Good night Sam!"
x = "mSa"
y = "eJo"
z = "odnght"
mytable = str.maketrans(x, y, z)
print(txt.translate(mytable))
```

Example

The same example as above, but using a dictionary instead of a mapping table:

```
txt = "Good night Sam!"
mydict = {109: 101, 83: 74, 97: 111, 111: None, 100: None, 110: None, 103: None, 104: None, 116: None}
print(txt.translate(mydict))
```

Python String upper() Method

Example

Upper case the string:

```
txt = "Hello my friends"
x = txt.upper()
print(x)
```

Definition and Usage

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

Symbols and Numbers are ignored.

Syntax

string.upper()

Parameter Values

No parameters

Python String zfill() Method

Example

Fill the string with zeros until it is 10 characters long:

```
txt = "50"
x = txt.zfill(10)
print(x)
```

Definition and Usage

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 desired length of the string

---

More Examples

Example

Fill the strings with zeros until they are 10 characters long:

```
a = "hello"
b = "welcome to the jungle"
c = "10.000"
print(a.zfill(10))
print(b.zfill(10))
print(c.zfill(10))
```