Python Reference

This section contains a Python reference documentation

Built-in Functions

String Methods

List Methods

Dictionary Methods

Tuple Methods

Set Methods

File Methods

Keywords

Exceptions

Glossary

Python Built in Functions

Python has a set of built-in functions.

Function	Description
abs()	Returns the absolute value of a number
all()	Returns True if all items in an iterable object are true
any()	Returns True if any item in an iterable object is true
ascii()	Returns a readable version of an object. Replaces none-ascii characters with escape character
bin()	Returns the binary version of a number
bool()	Returns the boolean value of the specified object
bytearray()	Returns an array of bytes
bytes()	Returns a bytes object
callable()	Returns True if the specified object is callable, otherwise False
chr()	Returns a character from the specified Unicode code.
classmethod()	Converts a method into a class method
compile()	Returns the specified source as an object, ready to be executed
complex()	Returns a complex number
delattr()	Deletes the specified attribute (property or method) from the specified object
dict()	Returns a dictionary (Array)
dir()	Returns a list of the specified object's properties and methods
divmod()	Returns the quotient and the remainder when argument1 is divided by argument2
enumerate()	Takes a collection (e.g. a tuple) and returns it as an enumerate object
eval()	Evaluates and executes an expression
exec()	Executes the specified code (or object)
filter()	Use a filter function to exclude items in an iterable object
float()	Returns a floating point number
format()	Formats a specified value

frozenset()	Returns a frozenset object
getattr()	Returns the value of the specified attribute (property or method)
globals()	Returns the current global symbol table as a dictionary
hasattr()	Returns True if the specified object has the specified attribute (property/method)
hash()	Returns the hash value of a specified object
help()	Executes the built-in help system
hex()	Converts a number into a hexadecimal value
id()	Returns the id of an object
input()	Allowing user input
int()	Returns an integer number
isinstance()	Returns True if a specified object is an instance of a specified object
issubclass()	Returns True if a specified class is a subclass of a specified object
iter()	Returns an iterator object
len()	Returns the length of an object
list()	Returns a list
locals()	Returns an updated dictionary of the current local symbol table
map()	Returns the specified iterator with the specified function applied to each item
max()	Returns the largest item in an iterable
memoryview()	Returns a memory view object
min()	Returns the smallest item in an iterable
next()	Returns the next item in an iterable
object()	Returns a new object
oct()	Converts a number into an octal
open()	Opens a file and returns a file object
ord()	Convert an integer representing the Unicode of the specified character
pow()	Returns the value of x to the power of y
print()	Prints to the standard output device

property()	Gets, sets, deletes a property
range()	Returns a sequence of numbers, starting from 0 and increments by 1 (by default)
repr()	Returns a readable version of an object
reversed()	Returns a reversed iterator
round()	Rounds a numbers
set()	Returns a new set object
setattr()	Sets an attribute (property/method) of an object
slice()	Returns a slice object
sorted()	Returns a sorted list
staticmethod()	Converts a method into a static method
str()	Returns a string object
sum()	Sums the items of an iterator
super()	Returns an object that represents the parent class
tuple()	Returns a tuple
type()	Returns the type of an object
vars()	Returns the __dict__ property of an object
zip()	Returns an iterator, from two or more iterators

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
encode()	Returns an encoded version of the 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

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

Python List/Array Methods

Python has a set of built-in methods that you can use on lists/arrays.

Method	Description
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the first item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

Note: Python does not have built-in support for Arrays, but Python Lists can be used instead.

Python Dictionary Methods

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

Method	Description
clear()	Removes all the elements from the dictionary
copy()	Returns a copy of the dictionary
fromkeys()	Returns a dictionary with the specified keys and value
get()	Returns the value of the specified key
items()	Returns a list containing a tuple for each key value pair
keys()	Returns a list containing the dictionary's keys
pop()	Removes the element with the specified key
popitem()	Removes the last inserted key-value pair
setdefault()	Returns the value of the specified key. If the key does not exist: insert the key, with the specified value
update()	Updates the dictionary with the specified key-value pairs
values()	Returns a list of all the values in the dictionary

Python Tuple Methods

Python has two built-in methods that you can use on tuples.

Method	Description
count()	Returns the number of times a specified value occurs in a tuple
index()	Searches the tuple for a specified value and returns the position of where it was found

---

Python Set Methods

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

Method	Description
add()	Adds an element to the set
clear()	Removes all the elements from the set
copy()	Returns a copy of the set
difference()	Returns a set containing the difference between two or more sets
difference_update()	Removes the items in this set that are also included in another, specified set
discard()	Remove the specified item
intersection()	Returns a set, that is the intersection of two or more sets
intersection_update()	Removes the items in this set that are not present in other, specified set(s)
isdisjoint()	Returns whether two sets have a intersection or not
issubset()	Returns whether another set contains this set or not
issuperset()	Returns whether this set contains another set or not
pop()	Removes an element from the set
remove()	Removes the specified element
symmetric_difference()	Returns a set with the symmetric differences of two sets
symmetric_difference_update()	inserts the symmetric differences from this set and another
union()	Return a set containing the union of sets
update()	Update the set with another set, or any other iterable

Python File Methods

Python has a set of methods available for the file object.

Method	Description
close()	Closes the file
detach()	Returns the separated raw stream from the buffer
fileno()	Returns a number that represents the stream, from the operating system's perspective
flush()	Flushes the internal buffer
isatty()	Returns whether the file stream is interactive or not
read()	Returns the file content
readable()	Returns whether the file stream can be read or not
readline()	Returns one line from the file
readlines()	Returns a list of lines from the file
seek()	Change the file position
seekable()	Returns whether the file allows us to change the file position
tell()	Returns the current file position
truncate()	Resizes the file to a specified size
writable()	Returns whether the file can be written to or not
write()	Writes the specified string to the file
writelines()	Writes a list of strings to the file

Python Keywords

Python has a set of keywords that are reserved words that cannot be used as variable names, function names, or any other identifiers:

Keyword	Description
and	A logical operator
as	To create an alias
assert	For debugging
break	To break out of a loop
class	To define a class
continue	To continue to the next iteration of a loop
def	To define a function
del	To delete an object
elif	Used in conditional statements, same as else if
else	Used in conditional statements
except	Used with exceptions, what to do when an exception occurs
False	Boolean value, result of comparison operations
finally	Used with exceptions, a block of code that will be executed no matter if there is an exception or not
for	To create a for loop
from	To import specific parts of a module
global	To declare a global variable
if	To make a conditional statement
import	To import a module
in	To check if a value is present in a list, tuple, etc.

is	To test if two variables are equal
lambda	To create an anonymous function
None	Represents a null value
nonlocal	To declare a non-local variable
not	A logical operator
or	A logical operator
pass	A null statement, a statement that will do nothing
raise	To raise an exception
return	To exit a function and return a value
True	Boolean value, result of comparison operations
try	To make a try...except statement
while	To create a while loop
with	Used to simplify exception handling
yield	To end a function, returns a generator

Python Built-in Exceptions

The table below shows built-in exceptions that are usually raised in Python:

Exception	Description
ArithmeticError	Raised when an error occurs in numeric calculations
AssertionError	Raised when an assert statement fails
AttributeError	Raised when attribute reference or assignment fails
Exception	Base class for all exceptions
EOFError	Raised when the input() method hits an "end of file" condition (EOF)
FloatingPointError	Raised when a floating point calculation fails
GeneratorExit	Raised when a generator is closed (with the close() method)
ImportError	Raised when an imported module does not exist
IndentationError	Raised when indentation is not correct
IndexError	Raised when an index of a sequence does not exist
KeyError	Raised when a key does not exist in a dictionary
KeyboardInterrupt	Raised when the user presses Ctrl+c, Ctrl+z or Delete
LookupError	Raised when errors raised cant be found
MemoryError	Raised when a program runs out of memory
NameError	Raised when a variable does not exist
NotImplementedError	Raised when an abstract method requires an inherited class to override the method
OSError	Raised when a system related operation causes an error
OverflowError	Raised when the result of a numeric calculation is too large
ReferenceError	Raised when a weak reference object does not exist
RuntimeError	Raised when an error occurs that do not belong to any specific exceptions
StopIteration	Raised when the next() method of an iterator has no further values
SyntaxError	Raised when a syntax error occurs

TabError	Raised when indentation consists of tabs or spaces
SystemError	Raised when a system error occurs
SystemExit	Raised when the sys.exit() function is called
TypeError	Raised when two different types are combined
UnboundLocalError	Raised when a local variable is referenced before assignment
UnicodeError	Raised when a unicode problem occurs
UnicodeEncodeError	Raised when a unicode encoding problem occurs
UnicodeDecodeError	Raised when a unicode decoding problem occurs
UnicodeTranslateError	Raised when a unicode translation problem occurs
ValueError	Raised when there is a wrong value in a specified data type
ZeroDivisionError	Raised when the second operator in a division is zero

Python Glossary

This is a list of all the features explained in the Python Tutorial.

Feature	Description
Indentation	Indentation refers to the spaces at the beginning of a code line
Comments	Comments are code lines that will not be executed
Multiline Comments	How to insert comments on multiple lines
Creating Variables	Variables are containers for storing data values
Variable Names	How to name your variables
Assign Values to Multiple Variables	How to assign values to multiple variables
Output Variables	Use the print statement to output variables
String Concatenation	How to combine strings
Global Variables	Global variables are variables that belongs to the global scope
Built-In Data Types	Python has a set of built-in data types
Getting Data Type	How to get the data type of an object
Setting Data Type	How to set the data type of an object
Numbers	There are three numeric types in Python
Int	The integer number type
Float	The floating number type
Complex	The complex number type
Type Conversion	How to convert from one number type to another
Random Number	How to create a random number
Specify a Variable Type	How to specify a certain data type for a variable
String Literals	How to create string literals
Assigning a String to a Variable	How to assign a string value to a variable

Multiline Strings	How to create a multiline string
Strings are Arrays	Strings in Python are arrays of bytes representing Unicode characters
Slicing a String	How to slice a string
Negative Indexing on a String	How to use negative indexing when accessing a string
String Length	How to get the length of a string
Check In String	How to check if a string contains a specified phrase
Format String	How to combine two strings
Escape Characters	How to use escape characters
Boolean Values	True or False
Evaluate Booleans	Evaluate a value or statement and return either True or False
Return Boolean Value	Functions that return a Boolean value
Operators	Use operator to perform operations in Python
Arithmetic Operators	Arithmetic operator are used to perform common mathematical operations
Assignment Operators	Assignment operators are use to assign values to variables
Comparison Operators	Comparison operators are used to compare two values
Logical Operators	Logical operators are used to combine conditional statements
Identity Operators	Identity operators are used to see if two objects are in fact the same object
Membership Operators	Membership operators are used to test is a sequence is present in an object
Bitwise Operators	Bitwise operators are used to compare (binary) numbers
Lists	A list is an ordered, and changeable, collection
Access List Items	How to access items in a list
Change List Item	How to change the value of a list item
Loop Through List Items	How to loop through the items in a list
List Comprehension	How use a list comprehensive

Check if List Item Exists	How to check if a specified item is present in a list
List Length	How to determine the length of a list
Add List Items	How to add items to a list
Remove List Items	How to remove list items
Copy a List	How to copy a list
Join Two Lists	How to join two lists
Tuple	A tuple is an ordered, and unchangeable, collection
Access Tuple Items	How to access items in a tuple
Change Tuple Item	How to change the value of a tuple item
Loop List Items	How to loop through the items in a tuple
Check if Tuple Item Exists	How to check if a specified item is present in a tuple
Tuple Length	How to determine the length of a tuple
Tuple With One Item	How to create a tuple with only one item
Remove Tuple Items	How to remove tuple items
Join Two Tuples	How to join two tuples
Set	A set is an unordered, and unchangeable, collection
Access Set Items	How to access items in a set
Add Set Items	How to add items to a set
Loop Set Items	How to loop through the items in a set
Check if Set Item Exists	How to check if a item exists
Set Length	How to determine the length of a set
Remove Set Items	How to remove set items
Join Two Sets	How to join two sets
Dictionary	A dictionary is an unordered, and changeable, collection

Access Dictionary Items	How to access items in a dictionary
Change Dictionary Item	How to change the value of a dictionary item
Loop Dictionary Items	How to loop through the items in a tuple
Check if Dictionary Item Exists	How to check if a specified item is present in a dictionary
Dictionary Length	How to determine the length of a dictionary
Add Dictionary Item	How to add an item to a dictionary
Remove Dictionary Items	How to remove dictionary items
Copy Dictionary	How to copy a dictionary
Nested Dictionaries	A dictionary within a dictionary
If Statement	How to write an if statement
If Indentation	If statemnts in Python relies on indentation (whitespace at the beginning of a line)
Elif	elif is the same as "else if" in other programming languages
Else	How to write an if...else statement
Shorthand If	How to write an if statement in one line
Shorthand If Else	How to write an if...else statement in one line
If AND	Use the and keyword to combine if statements
If OR	Use the or keyword to combine if statements
If NOT	Use the not keyword to reverse the condition
Nested If	How to write an if statement inside an if statement
The pass Keyword in If	Use the pass keyword inside empty if statements
While	How to write a while loop
While Break	How to break a while loop
While Continue	How to stop the current iteration and continue wit the next
While Else	How to use an else statement in a while loop

For	How to write a for loop
Loop Through a String	How to loop through a string
For Break	How to break a for loop
For Continue	How to stop the current iteration and continue wit the next
Looping Through a rangee	How to loop through a range of values
For Else	How to use an else statement in a for loop
Nested Loops	How to write a loop inside a loop
For pass	Use the pass keyword inside empty for loops
Function	How to create a function in Python
Call a Function	How to call a function in Python
Function Arguments	How to use arguments in a function
*args	To deal with an unknown number of arguments in a function, use the * symbol before the parameter name
Keyword Arguments	How to use keyword arguments in a function
**kwargs	To deal with an unknown number of keyword arguments in a function, use the * symbol before the parameter name
Default Parameter Value	How to use a default parameter value
Passing a List as an Argument	How to pass a list as an argument
Function Return Value	How to return a value from a function
The pass Statement i Functions	Use the pass statement in empty functions
Function Recursion	Functions that can call itself is called recursive functions
Lambda Function	How to create anonymous functions in Python
Why Use Lambda Functions	Learn when to use a lambda function or not
Array	Lists can be used as Arrays
What is an Array	Arrays are variables that can hold more than one value

Access Arrays	How to access array items
Array Length	How to get the length of an array
Looping Array Elements	How to loop through array elements
Add Array Element	How to add elements from an array
Remove Array Element	How to remove elements from an array
Array Methods	Python has a set of Array/Lists methods
Class	A class is like an object constructor
Create Class	How to create a class
The Class __init__() Function	The __init__() function is executed when the class is initiated
Object Methods	Methods in objects are functions that belongs to the object
self	The self parameter refers to the current instance of the class
Modify Object Properties	How to modify properties of an object
Delete Object Properties	How to modify properties of an object
Delete Object	How to delete an object
Class pass Statement	Use the pass statement in empty classes
Create Parent Class	How to create a parent class
Create Child Class	How to create a child class
Create the __init__() Function	How to create the __init__() function
super Function	The super() function make the child class inherit the parent class
Add Class Properties	How to add a property to a class
Add Class Methods	How to add a method to a class
Iterators	An iterator is an object that contains a countable number of values
Iterator vs Iterable	What is the difference between an iterator and an iterable

Loop Through an Iterator	How to loop through the elements of an iterator
Create an Iterator	How to create an iterator
StopIteration	How to stop an iterator
Global Scope	When does a variable belong to the global scope?
Global Keyword	The global keyword makes the variable global
Create a Module	How to create a module
Variables in Modules	How to use variables in a module
Renaming a Module	How to rename a module
Built-in Modules	How to import built-in modules
Using the dir() Function	List all variable names and function names in a module
Import From Module	How to import only parts from a module
Datetime Module	How to work with dates in Python
Date Output	How to output a date
Create a Date Object	How to create a date object
The strftime Method	How to format a date object into a readable string
Date Format Codes	The datetime module has a set of legal format codes
JSON	How to work with JSON in Python
Parse JSON	How to parse JSON code in Python
Convert into JSON	How to convert a Python object in to JSON
Format JSON	How to format JSON output with indentations and line breaks
Sort JSON	How to sort JSON
RegEx Module	How to import the regex module
RegEx Functions	The re module has a set of functions
Metacharacters in RegEx	Metacharacters are characters with a special meaning

RegEx Special Sequences	A backslash followed by a a character has a special meaning
RegEx Sets	A set is a set of characters inside a pair of square brackets with a special meaning
RegEx Match Object	The Match Object is an object containing information about the search and the result
Install PIP	How to install PIP
PIP Packages	How to download and install a package with PIP
PIP Remove Package	How to remove a package with PIP
Error Handling	How to handle errors in Python
Handle Many Exceptions	How to handle more than one exception
Try Else	How to use the else keyword in a try statement
Try Finally	How to use the finally keyword in a try statement
raise	How to raise an exception in Python