Python Virtual Environment

Python virtual environments create a virtual installation of Python inside a project directory. Users can then install and manage Python packages for each project. This allows users to be able to install packages and modify their Python environment without fear of breaking packages installed in other environments.

What is Virtual Environment in Python?

A Python virtual environment is:

* Considered as disposable.
* Used to contain a specific Python interpreter and software libraries and binaries which are needed to support a project.
* Contained in a directory, conventionally either named venv or .venv in the project directory.
* Not considered as movable or copyable.

C:\pythonapp>python -m venv myvenv

C:\pythonapp>myvenv\scripts\activate

C:\pythonapp>myvenv\scripts\deactivate

## Python - Syntax

The Python syntax defines a set of rules that are used to create a Python Program.

1. Interactive Mode Programming (b) Script Mode Programming.

## Python Multi-Line Statements

total = item\_one + \

item\_two + \

item\_three contained within the [], {}, or () brackets do not need to use the line continuation character.

single ('), double (") and triple (''' or """)

# First comment

'''

This is a multiline

comment.

'''

## Multiple Statements on a Single Line

The semicolon ( ; )

Command Line Arguments in Python

Many programs can be run to provide you with some basic information about how they should be run. Python enables you to do this with -h −

$ python3 -h

usage: python3 [option] ... [-c cmd | -m mod | file | -] [arg] ...

Options and arguments (and corresponding environment variables):

-c cmd : program passed in as string (terminates option list)

-d : debug output from parser (also PYTHONDEBUG=x)

-E : ignore environment variables (such as PYTHONPATH)

-h : print this help message and exit

Deleting

del var1[,var2[,var3[....,varN]]]]

print(type(x))

print(type(y))

print(type(z))

## Casting Python Variables

x = str(10) # x will be '10'

y = int(10) # y will be 10

z = float(10)

> a=b=c=10

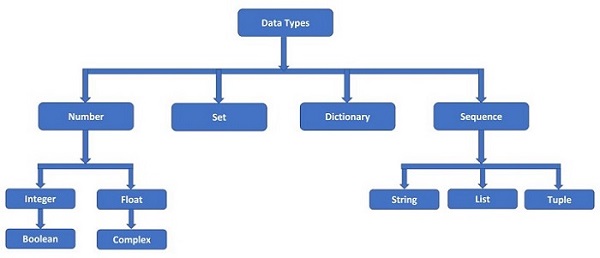
a,b,c = 1,2,"Zara Ali"

id("May")

2167264641264

## Constants in Python

treated as a constant by using all-caps names with underscores



|  |  |
| --- | --- |
| **Data Type** | **Examples** |
| Numeric | int, float, complex |
| String | str (text sequence type) |
| Sequence | list, tuple, range |
| Binary | bytes, bytearray, memoryview |
| Mapping | dict |
| Boolean | bool |
| Set | set, frozenset |
| None | NoneType |

var1 = 1 # int data type

var2 = True # bool data type

var3 = 10.023 # float data type

var4 = 10+3j # complex data type

type(123)

<class 'int'>

 Tuples can be thought of as **read-only** lists.

range(start, stop, step)

for i in range(2, 5):

print(i)

# Using bytes() function to create bytes

b1 = bytes([65, 66, 67, 68, 69])

print(b1)# b'ABCDE'

# Using prefix 'b' to create bytes

b2 = b'Hello'

print(b2)

data = bytearray(b'Hello, world!')

view = memoryview(data)

print(view)

<memory at 0x00000186FFAA3580>

### **Python Dictionary Data Type**

 curly brackets {}.

{1:'one', 2:'two', 3:'three'}

print (dict['one']) # Prints value for 'one' key

print (dict[2]) # Prints value for 2 key

print (tinydict) # Prints complete dictionary

print (tinydict.keys()) # Prints all the keys

print (tinydict.values()) # Prints all the values

### **Python Set Data Type**

{(5+6j), 3.11, 0.000123, 'Python', 2023}

|  |  |
| --- | --- |
| **Sr.No.** | **Function & Description** |
| 1 | [**Python int() function**](https://www.tutorialspoint.com/python/python-int-function.htm)  Converts x to an integer. base specifies the base if x is a string. |
| 2 | [**Python long() function**](https://www.tutorialspoint.com/python/python-long-function.htm)  Converts x to a long integer. base specifies the base if x is a string. *This function has been deprecated.* |
| 3 | [**Python float() function**](https://www.tutorialspoint.com/python/python-float-function.htm)  Converts x to a floating-point number. |
| 4 | [**Python complex() function**](https://www.tutorialspoint.com/python/python-complex-function.htm)  Creates a complex number. |
| 5 | [**Python str() function**](https://www.tutorialspoint.com/python/python-str-function.htm)  Converts object x to a string representation. |
| 6 | [**Python repr() function**](https://www.tutorialspoint.com/python/python-repr-function.htm)  Converts object x to an expression string. |
| 7 | [**Python eval() function**](https://www.tutorialspoint.com/python/python-eval-function.htm)  Evaluates a string and returns an object. |
| 8 | [**Python tuple() function**](https://www.tutorialspoint.com/python/python-tuple-function.htm)  Converts s to a tuple. |
| 9 | [**Python list() function**](https://www.tutorialspoint.com/python/python-list-function.htm)  Converts s to a list. |
| 10 | [**Python set() function**](https://www.tutorialspoint.com/python/python-set-function.htm)  Converts s to a set. |
| 11 | [**Python dict() function**](https://www.tutorialspoint.com/python/python-dict-function.htm)  Creates a dictionary. d must be a sequence of (key,value) tuples. |
| 12 | [**Python frozenset() function**](https://www.tutorialspoint.com/python/python-frozenset-function.htm)  Converts s to a frozen set. |
| 13 | [**Python chr() function**](https://www.tutorialspoint.com/python/python-chr-function.htm)  Converts an integer to a character. |
| 14 | [**Python unichr() function**](https://www.tutorialspoint.com/python/python-unichr-function.htm)  Converts an integer to a Unicode character. |
| 15 | [**Python ord() function**](https://www.tutorialspoint.com/python/python-ord-function.htm)  Converts a single character to its integer value. |
| 16 | [**Python hex() function**](https://www.tutorialspoint.com/python/python-hex-function.htm)  Converts an integer to a hexadecimal string. |
| 17 | [**Python oct() function**](https://www.tutorialspoint.com/python/python-oct-function.htm)  Converts an integer to an octal string. |

***implicit and explicit.TYPE CASTING***

a=10 # int object

<<< b=10.5 # float object

a = int(10)

Octal

<<< a = int("20", 8)

<<< a

16

### **Hexa-Decimal String to Intege**

<<< a = int("2A9", 16)

<<< a

681

str() function puts the list and tuple inside the quote symbols.

<<< obj=str(a)

<<< obj

'[1, 2, 3, 4, 5]'

list(c) tuple(c)

## Python's Unicode Support

var = "3/4"

print (var)

var = "\u00BE"

print (var)

**Literals**

x = 0O34//octal

x = 0X1C//hexa

float .

scientific

1.23E5

Complex:2+3j=

String ‘,”,”””

* Arithmetic Operators
* Comparison (Relational) Operators
* Assignment Operators
* Logical Operators
* Bitwise Operators
* Membership Operators
* Identity Operators

|  |  |  |
| --- | --- | --- |
| + | Addition | a + b = 30 |
| - | Subtraction | a – b = -10 |
| \* | Multiplication | a \* b = 200 |
| / | Division | b / a = 2 |
| % | Modulus | b % a = 0 |
| \*\* | Exponent | a\*\*b =10\*\*20 |
| // | Floor Division | 9//2 = 4 |

|  |  |  |
| --- | --- | --- |
| **Operator** | **Name** | **Example** |
| & | AND | a & b |
| | | OR | a | b |
| ^ | XOR | a ^ b |
| ~ | NOT | ~a |
| << | Zero fill left shift | a << 3 |
| >> | Signed right shift | a >> 3 |

Logical

|  |  |  |
| --- | --- | --- |
| **Operator** | **Name** | **Example** |
| and | AND | a and b |
| or | OR | a or b |
| not | NOT | not(a) |

Membership

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| in | Returns True if it finds a variable in the specified sequence, false otherwise. | a in b |
| not in | returns True if it does not finds a variable in the specified sequence and false otherwise.  **identity** | a not in b |
| Operator | Description | Example |
| is | Returns True if both variables are the same object and false otherwise. | a is b |
| is not | Returns True if both variables are not the same object and false otherwise. | a is not b |

names = ["Alice", "Bob", "Charlie"]

name\_lengths = {name: len(name) for name in names}

print(name\_lengths) # Output: {'Alice': 5, 'Bob': 3, 'Charlie': 7}

def weekday(n):

match n:

case 0: return "Monday"

case 1: return "Tuesday"

case 2: return "Wednesday"

case 3: return "Thursday"

case 4: return "Friday"

case 5: return "Saturday"

case 6: return "Sunday"

case \_: return "Invalid day number"

print (weekday(3))

print (weekday(6))

print (weekday(7))

print ("Iteration no. {}".format(count))

### Built-in Functions

Built-in functions are the functions that are always available in Python. You do not need to import any module to use them. These functions are part of the Python standard library.

#### Examples of Built-in Functions:

* **print()**: Outputs text to the console.
* **len()**: Returns the length of an object (like a string, list, or dictionary).
* **type()**: Returns the type of an object.
* **int()**: Converts a value to an integer.
* **str()**: Converts a value to a string.
* **max()**: Returns the largest item in an iterable or the largest of two or more arguments.
* **min()**: Returns the smallest item in an iterable or the smallest of two or more arguments.
* **sum()**: Sums the items of an iterable.
* **range()**: Generates a sequence of numbers.

### Functions Defined in Built-in Modules

These functions are part of Python's standard library but are organized into modules. You need to import these modules to use their functions.

#### Examples of Built-in Modules:

* **math**: Provides mathematical functions.
* **datetime**: Provides classes for manipulating dates and times.
* **random**: Implements pseudo-random number generators.
* **os**: Provides a way of using operating system-dependent functionality.
* **sys**: Provides access to some variables used or maintained by the interpreter.

Positional or Required Arguments  
def add(a, b):

return a + b

print(add(5, 3)) # Output: 8

Keyword Arguments

introduce(name="Alice", age=30)

Default Arguments

def greet(name, message="Hello"):

return f"{message}, {name}!"

print(greet("Alice")) # Output: Hello, Alice!

print(greet("Bob", "Hi")) # Output: Hi, Bob!

**Positional-only Arguments**

def add(a, b, /):

return a + b

# Correct usage

print(add(5, 3)) # Output: 8

Keyword-only Arguments

def introduce(\*, name, age):

print(f"My name is {name} and I am {age} years old.")

# Correct usage

introduce(name="Alice", age=30)

Arbitrary or Variable-length Arguments

def sum\_all(\*args):

return sum(args)

print(sum\_all(1, 2, 3, 4, 5)) # Output: 15

def print\_details(\*\*kwargs):

for key, value in kwargs.items():

print(f"{key}: {value}")

print\_details(name="Alice", age=30, city="New York")

## Nonlocal Variables

The Python variables that are not defined in either local or global scope are called nonlocal variables. They are used in nested functions.

def yourfunction():

a = 5

b = 6

# nested function

def myfunction():

# nonlocal function

nonlocal a#calling defined a

nonlocal b

a = 10

b = 20

print("variable a:", a)

print("variable b:", b)

return a+b

print (myfunction())

yourfunction()

**Global var**

z = 10

def modify\_global():

global z

z = 20

modify\_global()

print("Modified global z:", z) # Output: Modified global z: 20

**Function annotations**

def myfunction(a: int, b: int) -> int:

c = a+b

return c

print(myfunction.\_\_annotations\_\_)

|  |  |
| --- | --- |
| **Sr.No.** | **Name & Brief Description** |
| 1 | [**os**](https://www.tutorialspoint.com/python/os_file_methods.htm)  This module provides a unified interface to a number of operating system functions. |
| 2 | [**string**](https://www.tutorialspoint.com/python/python_string_methods.htm)  This module contains a number of functions for string processing |
| 3 | **re**  This module provides a set of powerful regular expression facilities. Regular expression (RegEx), allows powerful string search and matching for a pattern in a string |
| 4 | [**math**](https://www.tutorialspoint.com/python/python_maths.htm)  This module implements a number of mathematical operations for floating point numbers. These functions are generally thin wrappers around the platform C library functions. |
| 5 | **cmath**  This module contains a number of mathematical operations for complex numbers. |
| 6 | **datetime**  This module provides functions to deal with dates and the time within a day. It wraps the C runtime library. |
| 7 | **gc**  This module provides an interface to the built-in garbage collector. |
| 8 | **asyncio**  This module defines functionality required for asynchronous processing |
| 9 | **Collections**  This module provides advanced Container datatypes. |
| 10 | **Functools**  This module has Higher-order functions and operations on callable objects. Useful in functional programming |
| 11 | **operator**  Functions corresponding to the standard operators. |
| 12 | **pickle**  Convert Python objects to streams of bytes and back. |
| 13 | **socket**  Low-level networking interface. |
| 14 | **sqlite3**  A DB-API 2.0 implementation using SQLite 3.x. |
| 15 | **statistics**  Mathematical statistics functions |
| 16 | **typing**  Support for type hints |
| 17 | **venv**  Creation of virtual environments. |
| 18 | **json**  Encode and decode the JSON format. |
| 19 | **wsgiref**  WSGI Utilities and Reference Implementation. |
| 20 | **unittest**  Unit testing framework for Python. |
| 21 | **random**  Generate pseudo-random numbers |

mymodule.py

from mymodule import sum, average

from modname import \*

from modulename as alias

* \_\_file\_\_ returns the physical name of the module.
* \_\_package\_\_ returns the package to which the module belongs.
* \_\_doc\_\_ returns the docstring at the top of the module if any
* \_\_dict\_\_ returns the entire scope of the module
* \_\_name\_\_ returns the name of the module
* print ("\_\_file\_\_ attribute:", mymodule.\_\_file\_\_)
* print ("\_\_doc\_\_ attribute:", mymodule.\_\_doc\_\_)
* print ("\_\_name\_\_ attribute:", mymodule.\_\_name\_\_)

dir() built-in function returns a sorted list of strings containing the names defined by a module.

List of Python Built-in Functions

As of Python 3.12.2 version, the list of built-in functions is given below −

|  |  |
| --- | --- |
| **Sr.No.** | **Function & Description** |
| 1 | [**Python aiter() function**](https://www.tutorialspoint.com/python/python_aiter_function.htm)  Returns an asynchronous iterator for an asynchronous iterable. |
| 2 | [**Python all() function**](https://www.tutorialspoint.com/python/python_all_function.htm)  Returns true when all elements in iterable is true. |
| 3 | [**Python anext() function**](https://www.tutorialspoint.com/python/python_anext_function.htm)  Returns the next item from the given asynchronous iterator. |
| 4 | [**Python any() function**](https://www.tutorialspoint.com/python/python_any_function.htm)  Checks if any Element of an Iterable is True. |
| 5 | [**Python ascii() function**](https://www.tutorialspoint.com/python/python_ascii_function.htm)  Returns String Containing Printable Representation. |
| 6 | [**Python bin() function**](https://www.tutorialspoint.com/python/python_bin_function.htm)  Converts integer to binary string. |
| 7 | [**Python bool() function**](https://www.tutorialspoint.com/python/python_bool_function.htm)  Converts a Value to Boolean. |
| 8 | [**Python breakpoint() function**](https://www.tutorialspoint.com/python/python_breakpoint_function.htm)  This function drops you into the debugger at the call site and calls sys.breakpointhook(). |
| 9 | [**Python bytearray() function**](https://www.tutorialspoint.com/python/python_bytearray_function.htm)  Returns array of given byte size. |
| 10 | [**Python bytes() function**](https://www.tutorialspoint.com/python/python_bytes_function.htm)  Returns immutable bytes object. |
| 11 | [**Python callable() function**](https://www.tutorialspoint.com/python/python_callable_function.htm)  Checks if the Object is Callable. |
| 12 | [**Python chr() function**](https://www.tutorialspoint.com/python/python-chr-function.htm)  Returns a Character (a string) from an Integer. |
| 13 | [**Python classmethod() function**](https://www.tutorialspoint.com/python/python_classmethod_function.htm)  Returns class method for given function. |
| 14 | [**Python compile() function**](https://www.tutorialspoint.com/python/python_compile_function.htm)  Returns a code object. |
| 15 | [**Python complex() function**](https://www.tutorialspoint.com/python/python-complex-function.htm)  Creates a Complex Number. |
| 16 | [**Python delattr() function**](https://www.tutorialspoint.com/python/python_delattr_function.htm)  Deletes Attribute From the Object. |
| 17 | [**Python dict() function**](https://www.tutorialspoint.com/python/python-dict-function.htm)  Creates a Dictionary. |
| 18 | [**Python dir() function**](https://www.tutorialspoint.com/python/python_dir_function.htm)  Tries to Return Attributes of Object. |
| 19 | [**Python divmod() function**](https://www.tutorialspoint.com/python/python_divmod_function.htm)  Returns a Tuple of Quotient and Remainder. |
| 20 | [**Python enumerate() function**](https://www.tutorialspoint.com/python/python_enumerate_function.htm)  Returns an Enumerate Object. |
| 21 | [**Python eval() function**](https://www.tutorialspoint.com/python/python-eval-function.htm)  Runs Code Within Program. |
| 22 | [**Python exec() function**](https://www.tutorialspoint.com/python/python_exec_function.htm)  Executes Dynamically Created Program. |
| 23 | [**Python filter() function**](https://www.tutorialspoint.com/python/python_filter_function.htm)  Constructs iterator from elements which are true. |
| 24 | [**Python float() function**](https://www.tutorialspoint.com/python/python-float-function.htm)  Returns floating point number from number, string. |
| 25 | [**Python format() function**](https://www.tutorialspoint.com/python/python_format_function.htm)  Returns formatted representation of a value. |
| 26 | [**Python frozenset() function**](https://www.tutorialspoint.com/python/python-frozenset-function.htm)  Returns immutable frozenset object. |
| 27 | [**Python getattr() function**](https://www.tutorialspoint.com/python/python_getattr_function.htm)  Returns value of named attribute of an object. |
| 28 | [**Python globals() function**](https://www.tutorialspoint.com/python/python_globals_function.htm)  Returns dictionary of current global symbol table. |
| 29 | [**Python hasattr() function**](https://www.tutorialspoint.com/python/python_hasattr_function.htm)  Returns whether object has named attribute. |
| 30 | [**Python hash() function**](https://www.tutorialspoint.com/python/python_hash_function.htm)  Returns hash value of an object. |
| 31 | [**Python help() function**](https://www.tutorialspoint.com/python/python_help_function.htm)  Invokes the built-in Help System. |
| 32 | [**Python hex() function**](https://www.tutorialspoint.com/python/python-hex-function.htm)  Converts to Integer to Hexadecimal. |
| 33 | [**Python id() function**](https://www.tutorialspoint.com/python/python_id_function.htm)  Returns Identify of an Object. |
| 34 | [**Python input() function**](https://www.tutorialspoint.com/python/python_input_function.htm)  Reads and returns a line of string. |
| 35 | [**Python int() function**](https://www.tutorialspoint.com/python/python-int-function.htm)  Returns integer from a number or string. |
| 36 | [**Python isinstance() function**](https://www.tutorialspoint.com/python/python_isinstance_function.htm)  Checks if a Object is an Instance of Class. |
| 37 | [**Python issubclass() function**](https://www.tutorialspoint.com/python/python_issubclass_function.htm)  Checks if a Class is Subclass of another Class. |
| 38 | [**Python iter() function**](https://www.tutorialspoint.com/python/python_iter_function.htm)  Returns an iterator. |
| 39 | [**Python len() function**](https://www.tutorialspoint.com/python/python_len_function.htm)  Returns Length of an Object. |
| 40 | [**Python list() function**](https://www.tutorialspoint.com/python/python-list-function.htm)  Creates a list in Python. |
| 41 | [**Python locals() function**](https://www.tutorialspoint.com/python/python_locals_function.htm)  Returns dictionary of a current local symbol table. |
| 42 | [**Python map() function**](https://www.tutorialspoint.com/python/python_map_function.htm)  Applies Function and Returns a List. |
| 43 | [**Python memoryview() function**](https://www.tutorialspoint.com/python/python_memoryview_function.htm)  Returns memory view of an argument. |
| 44 | [**Python next() function**](https://www.tutorialspoint.com/python/python_next_function.htm)  Retrieves next item from the iterator. |
| 45 | [**Python object() function**](https://www.tutorialspoint.com/python/python_object_function.htm)  Creates a featureless object. |
| 46 | [**Python oct() function**](https://www.tutorialspoint.com/python/python-oct-function.htm)  Returns the octal representation of an integer. |
| 47 | [**Python open() function**](https://www.tutorialspoint.com/python/python_open_function.htm)  Returns a file object. |
| 48 | [**Python ord() function**](https://www.tutorialspoint.com/python/python-ord-function.htm)  Returns an integer of the Unicode character. |
| 49 | [**Python print() function**](https://www.tutorialspoint.com/python/python_print_function.htm)  Prints the Given Object. |
| 50 | [**Python property() function**](https://www.tutorialspoint.com/python/python_property_function.htm)  Returns the property attribute. |
| 51 | [**Python range() function**](https://www.tutorialspoint.com/python/python_range_function.htm)  Returns a sequence of integers. |
| 52 | [**Python repr() function**](https://www.tutorialspoint.com/python/python-repr-function.htm)  Returns a printable representation of the object. |
| 53 | [**Python reversed() function**](https://www.tutorialspoint.com/python/python_reversed_function.htm)  Returns the reversed iterator of a sequence. |
| 54 | [**Python set() function**](https://www.tutorialspoint.com/python/python-set-function.htm)  Constructs and returns a set. |
| 55 | [**Python setattr() function**](https://www.tutorialspoint.com/python/python_setattr_function.htm)  Sets the value of an attribute of an object. |
| 56 | [**Python slice() function**](https://www.tutorialspoint.com/python/python_slice_function.htm)  Returns a slice object. |
| 57 | [**Python sorted() function**](https://www.tutorialspoint.com/python/python_sorted_function.htm)  Returns a sorted list from the given iterable. |
| 58 | [**Python staticmethod() function**](https://www.tutorialspoint.com/python/python_staticmethod_function.htm)  Transforms a method into a static method. |
| 59 | [**Python str() function**](https://www.tutorialspoint.com/python/python-str-function.htm)  Returns the string version of the object. |
| 60 | [**Python super() function**](https://www.tutorialspoint.com/python/python_super_function.htm)  Returns a proxy object of the base class. |
| 61 | [**Python tuple() function**](https://www.tutorialspoint.com/python/python-tuple-function.htm)  Returns a tuple. |
| 62 | [**Python type() function**](https://www.tutorialspoint.com/python/python_type_function.htm)  Returns the type of the object. |
| 63 | [**Python vars() function**](https://www.tutorialspoint.com/python/python_vars_function.htm)  Returns the \_\_dict\_\_ attribute. |
| 64 | [**Python zip() function**](https://www.tutorialspoint.com/python/python_zip_function.htm)  Returns an iterator of tuples. |
| 65 | [**Python \_\_import\_\_() function**](https://www.tutorialspoint.com/python/python_import_function.htm)  Function called by the import statement. |
| 66 | [**Python unichr() function**](https://www.tutorialspoint.com/python/python-unichr-function.htm)  Converts a Unicode code point to its corresponding Unicode character. |
| 67 | [**Python long() function**](https://www.tutorialspoint.com/python/python-long-function.htm)  Represents integers of arbitrary size. |

Built-in Mathematical Functions

There are some additional built-in functions that are used for performing only mathematical operations in Python, they are listed below −

|  |  |
| --- | --- |
| **Sr.No.** | **Function & Description** |
| 1 | [**Python abs() function**](https://www.tutorialspoint.com/python/number_abs.htm)  The abs() function returns the absolute value of x, i.e. the positive distance between x and zero. |
| 2 | [**Python max() function**](https://www.tutorialspoint.com/python/number_max.htm)  The max() function returns the largest of its arguments or largest number from the iterable (list or tuple). |
| 3 | [**Python min() function**](https://www.tutorialspoint.com/python/number_min.htm)  The function min() returns the smallest of its arguments i.e. the value closest to negative infinity, or smallest number from the iterable (list or tuple) |
| 4 | [**Python pow() function**](https://www.tutorialspoint.com/python/pow_function.htm)  The pow() function returns x raised to y. It is equivalent to x\*\*y. The function has third optional argument mod. If given, it returns (x\*\*y) % mod value |
| 5 | [**Python round() Function**](https://www.tutorialspoint.com/python/number_round.htm)  round() is a built-in function in Python. It returns x rounded to n digits from the decimal point. |
| 6 | [**Python sum() function**](https://www.tutorialspoint.com/python/sum_function.htm)  The sum() function returns the sum of all numeric items in any iterable (list or tuple). An optional start argument is 0 by default. If given, the numbers in the list are added to start value. |

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 at next section |

Built-in String Methods

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

|  |  |
| --- | --- |
| **Sr.No.** | **Methods with Description** |
| 1 | [**capitalize()**](https://www.tutorialspoint.com/python/string_capitalize.htm)  Capitalizes first letter of string. |
| 2 | [**casefold()**](https://www.tutorialspoint.com/python/casefold_method.htm)  Converts all uppercase letters in string to lowercase. Similar to lower(), but works on UNICODE characters alos. |
| 3 | [**center(width, fillchar)**](https://www.tutorialspoint.com/python/string_center.htm)  Returns a space-padded string with the original string centered to a total of width columns. |
| 4 | [**count(str, beg= 0,end=len(string))**](https://www.tutorialspoint.com/python/string_count.htm)  Counts how many times str occurs in string or in a substring of string if starting index beg and ending index end are given. |
| 5 | [**decode(encoding='UTF-8',errors='strict')**](https://www.tutorialspoint.com/python/string_decode.htm)  Decodes the string using the codec registered for encoding. encoding defaults to the default string encoding. |
| 6 | [**encode(encoding='UTF-8',errors='strict')**](https://www.tutorialspoint.com/python/string_encode.htm)  Returns encoded string version of string; on error, default is to raise a ValueError unless errors is given with 'ignore' or 'replace'. |
| 7 | [**endswith(suffix, beg=0, end=len(string))**](https://www.tutorialspoint.com/python/string_endswith.htm)  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. |
| 8 | [**expandtabs(tabsize=8)**](https://www.tutorialspoint.com/python/string_expandtabs.htm)  Expands tabs in string to multiple spaces; defaults to 8 spaces per tab if tabsize not provided. |
| 9 | [**find(str, beg=0 end=len(string))**](https://www.tutorialspoint.com/python/string_find.htm)  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. |
| 10 | [**format(\*args, \*\*kwargs)**](https://www.tutorialspoint.com/python/python_string_format_method.htm)  This method is used to format the current string value. |
| 11 | [**format\_map(mapping)**](https://www.tutorialspoint.com/python/python_string_format_map_method.htm)  This method is also use to format the current string the only difference is it uses a mapping object. |
| 12 | [**index(str, beg=0, end=len(string))**](https://www.tutorialspoint.com/python/string_index.htm)  Same as find(), but raises an exception if str not found. |
| 13 | [**isalnum()**](https://www.tutorialspoint.com/python/string_isalnum.htm)  Returns true if string has at least 1 character and all characters are alphanumeric and false otherwise. |
| 14 | [**isalpha()**](https://www.tutorialspoint.com/python/string_isalpha.htm)  Returns true if string has at least 1 character and all characters are alphabetic and false otherwise. |
| 15 | [**isascii()**](https://www.tutorialspoint.com/python/isascii_method.htm)  Returns True is all the characters in the string are from the ASCII character set. |
| 16 | [**isdecimal()**](https://www.tutorialspoint.com/python/string_isdecimal.htm)  Returns true if a unicode string contains only decimal characters and false otherwise. |
| 17 | [**isdigit()**](https://www.tutorialspoint.com/python/string_isdigit.htm)  Returns true if string contains only digits and false otherwise. |
| 18 | [**isidentifier()**](https://www.tutorialspoint.com/python/python_string_isidentifier_method.htm)  Checks whether the string is a valid Python identifier. |
| 19 | [**islower()**](https://www.tutorialspoint.com/python/string_islower.htm)  Returns true if string has at least 1 cased character and all cased characters are in lowercase and false otherwise. |
| 20 | [**isnumeric()**](https://www.tutorialspoint.com/python/string_isnumeric.htm)  Returns true if a unicode string contains only numeric characters and false otherwise. |
| 21 | [**isprintable()**](https://www.tutorialspoint.com/python/isprintable_method.htm)  Checks whether all the characters in the string are printable. |
| 22 | [**isspace()**](https://www.tutorialspoint.com/python/string_isspace.htm)  Returns true if string contains only whitespace characters and false otherwise. |
| 23 | [**istitle()**](https://www.tutorialspoint.com/python/string_istitle.htm)  Returns true if string is properly "titlecased" and false otherwise. |
| 24 | [**isupper()**](https://www.tutorialspoint.com/python/string_isupper.htm)  Returns true if string has at least one cased character and all cased characters are in uppercase and false otherwise. |
| 25 | [**join(seq)**](https://www.tutorialspoint.com/python/string_join.htm)  Merges (concatenates) the string representations of elements in sequence seq into a string, with separator string. |
| 26 | [**ljust(width[, fillchar])**](https://www.tutorialspoint.com/python/string_ljust.htm)  Returns a space-padded string with the original string left-justified to a total of width columns. |
| 27 | [**lower()**](https://www.tutorialspoint.com/python/string_lower.htm)  Converts all uppercase letters in string to lowercase. |
| 28 | [**lstrip()**](https://www.tutorialspoint.com/python/string_lstrip.htm)  Removes all leading white space in string. |
| 29 | [**maketrans()**](https://www.tutorialspoint.com/python/string_maketrans.htm)  Returns a translation table to be used in translate function. |
| 30 | [**partition()**](https://www.tutorialspoint.com/python/partition_method.htm)  Splits the string in three string tuple at the first occurrence of separator. |
| 31 | [**removeprefix()**](https://www.tutorialspoint.com/python/removeprefix_method.htm)  Returns a string after removing the prefix string. |
| 32 | [**removesuffix()**](https://www.tutorialspoint.com/python/removesuffix_method.htm)  Returns a string after removing the suffix string. |
| 33 | [**replace(old, new [, max])**](https://www.tutorialspoint.com/python/string_replace.htm)  Replaces all occurrences of old in string with new or at most max occurrences if max given. |
| 34 | [**rfind(str, beg=0,end=len(string))**](https://www.tutorialspoint.com/python/string_rfind.htm)  Same as find(), but search backwards in string. |
| 35 | [**rindex( str, beg=0, end=len(string))**](https://www.tutorialspoint.com/python/string_rindex.htm)  Same as index(), but search backwards in string. |
| 36 | [**rjust(width,[, fillchar])**](https://www.tutorialspoint.com/python/string_rjust.htm)  Returns a space-padded string with the original string right-justified to a total of width columns. |
| 37 | [**rpartition()**](https://www.tutorialspoint.com/python/rpartition_method.htm)  Splits the string in three string tuple at the ladt occurrence of separator. |
| 38 | [**rsplit()**](https://www.tutorialspoint.com/python/rsplit_method.htm)  Splits the string from the end and returns a list of substrings. |
| 39 | [**rstrip()**](https://www.tutorialspoint.com/python/string_rstrip.htm)  Removes all trailing whitespace of string. |
| 40 | [**split(str="", num=string.count(str))**](https://www.tutorialspoint.com/python/string_split.htm)  Splits string according to delimiter str (space if not provided) and returns list of substrings; split into at most num substrings if given. |
| 41 | [**splitlines( num=string.count('\n'))**](https://www.tutorialspoint.com/python/string_splitlines.htm)  Splits string at all (or num) NEWLINEs and returns a list of each line with NEWLINEs removed. |
| 42 | [**startswith(str, beg=0,end=len(string))**](https://www.tutorialspoint.com/python/string_startswith.htm)  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. |
| 43 | [**strip([chars])**](https://www.tutorialspoint.com/python/string_strip.htm)  Performs both lstrip() and rstrip() on string. |
| 44 | [**swapcase()**](https://www.tutorialspoint.com/python/string_swapcase.htm)  Inverts case for all letters in string. |
| 45 | [**title()**](https://www.tutorialspoint.com/python/string_title.htm)  Returns "titlecased" version of string, that is, all words begin with uppercase and the rest are lowercase. |
| 46 | [**translate(table, deletechars="")**](https://www.tutorialspoint.com/python/string_translate.htm)  Translates string according to translation table str(256 chars), removing those in the del string. |
| 47 | [**upper()**](https://www.tutorialspoint.com/python/string_upper.htm)  Converts lowercase letters in string to uppercase. |
| 48 | [**zfill (width)**](https://www.tutorialspoint.com/python/string_zfill.htm)  Returns original string leftpadded with zeros to a total of width characters; intended for numbers, zfill() retains any sign given (less one zero). |
| Sr.No. | Function with Description |
| 1 | [len(list)](https://www.tutorialspoint.com/python/string_len.htm)  Returns the length of the string. |
| 2 | [max(list)](https://www.tutorialspoint.com/python/string_max.htm)  Returns the max alphabetical character from the string str. |
| 3 | [min(list)](https://www.tutorialspoint.com/python/string_min.htm)  Returns the min alphabetical character from the string str. |

**String to LIST**

s1="WORD"

print ("original string:", s1)

l1=list(s1)

ARRAY

# Creating an array of integers

my\_array = array.array('i', [1, 2, 3, 4, 5])

print(my\_array) # Output: array('i', [1, 2, 3, 4, 5])

|  |  |
| --- | --- |
| **Sr.No.** | **Method & Description** |
| 1 | [**capitalize()**](https://www.tutorialspoint.com/python/string_capitalize.htm)  Capitalizes first letter of string |
| 2 | [**casefold()**](https://www.tutorialspoint.com/python/casefold_method.htm)  Converts all uppercase letters in string to lowercase. Similar to lower(), but works on UNICODE characters alos |
| 3 | [**lower()**](https://www.tutorialspoint.com/python/string_lower.htm)  Converts all uppercase letters in string to lowercase. |
| 4 | [**swapcase()**](https://www.tutorialspoint.com/python/string_swapcase.htm)  Inverts case for all letters in string. |
| 5 | [**title()**](https://www.tutorialspoint.com/python/string_title.htm)  Returns "titlecased" version of string, that is, all words begin with uppercase and the rest are lowercase. |
| 6 | [**upper()**](https://www.tutorialspoint.com/python/string_upper.htm)  Converts lowercase letters in string to uppercase. |

## Alignment Methods

Following methods in the **str** class control the alignment of characters within the string object.

|  |  |
| --- | --- |
| **Sr.No.** | **Methods & Description** |
| 1 | [**center(width, fillchar)**](https://www.tutorialspoint.com/python/string_center.htm)  Returns a string padded with fillchar with the original string centered to a total of width columns. |
| 2 | [**ljust(width[, fillchar])**](https://www.tutorialspoint.com/python/string_ljust.htm)  Returns a space-padded string with the original string left-justified to a total of width columns. |
| 3 | [**rjust(width,[, fillchar])**](https://www.tutorialspoint.com/python/string_rjust.htm)  Returns a space-padded string with the original string right-justified to a total of width columns. |
| 4 | [**expandtabs(tabsize = 8)**](https://www.tutorialspoint.com/python/string_expandtabs.htm)  Expands tabs in string to multiple spaces; defaults to 8 spaces per tab if tabsize not provided. |
| 5 | [**zfill (width)**](https://www.tutorialspoint.com/python/string_zfill.htm)  Returns original string leftpadded with zeros to a total of width characters; intended for numbers, zfill() retains any sign given (less one zero). |

## Split and Join Methods

Python has the following methods to perform split and join operations −

|  |  |
| --- | --- |
| **Sr.No.** | **Method & Description** |
| 1 | [**lstrip()**](https://www.tutorialspoint.com/python/string_lstrip.htm)  Removes all leading whitespace in string. |
| 2 | [**rstrip()**](https://www.tutorialspoint.com/python/string_rstrip.htm)  Removes all trailing whitespace of string. |
| 3 | [**strip()**](https://www.tutorialspoint.com/python/string_strip.htm)  Performs both lstrip() and rstrip() on string |
| 4 | [**rsplit()**](https://www.tutorialspoint.com/python/rsplit_method.htm)  Splits the string from the end and returns a list of substrings |
| 5 | [**split()**](https://www.tutorialspoint.com/python/string_split.htm)  Splits string according to delimiter (space if not provided) and returns list of substrings. |
| 6 | [**splitlines()**](https://www.tutorialspoint.com/python/string_splitlines.htm)  Splits string at NEWLINEs and returns a list of each line with NEWLINEs removed. |
| 7 | [**partition()**](https://www.tutorialspoint.com/python/partition_method.htm)  Splits the string in three string tuple at the first occurrence of separator |
| 8 | [**rpartition()**](https://www.tutorialspoint.com/python/rpartition_method.htm)  Splits the string in three string tuple at the ladt occurrence of separator |
| 9 | [**join()**](https://www.tutorialspoint.com/python/string_join.htm)  Concatenates the string representations of elements in sequence into a string, with separator string. |
| 10 | [**removeprefix()**](https://www.tutorialspoint.com/python/removeprefix_method.htm)  Returns a string after removing the prefix string |
| 11 | [**removesuffix()**](https://www.tutorialspoint.com/python/removesuffix_method.htm)  Returns a string after removing the suffix string |

## Boolean String Methods

Following methods in str class return True or False.

|  |  |
| --- | --- |
| **Sr.No.** | **Methods & Description** |
| 1 | [**isalnum()**](https://www.tutorialspoint.com/python/string_isalnum.htm)  Returns true if string has at least 1 character and all characters are alphanumeric and false otherwise. |
| 2 | [**isalpha()**](https://www.tutorialspoint.com/python/string_isalpha.htm)  Returns true if string has at least 1 character and all characters are alphabetic and false otherwise. |
| 3 | [**isdigit()**](https://www.tutorialspoint.com/python/string_isdigit.htm)  Returns true if the string contains only digits and false otherwise. |
| 4 | [**islower()**](https://www.tutorialspoint.com/python/string_islower.htm)  Returns true if string has at least 1 cased character and all cased characters are in lowercase and false otherwise. |
| 5 | [**isnumeric()**](https://www.tutorialspoint.com/python/string_isnumeric.htm)  Returns true if a unicode string contains only numeric characters and false otherwise |
| 6 | [**isspace()**](https://www.tutorialspoint.com/python/string_isspace.htm)  Returns true if string contains only whitespace characters and false otherwise. |
| 7 | [**istitle()**](https://www.tutorialspoint.com/python/string_istitle.htm)  Returns true if string is properly "titlecased" and false otherwise. |
| 8 | [**isupper()**](https://www.tutorialspoint.com/python/string_isupper.htm)  Returns true if string has at least one cased character and all cased characters are in uppercase and false otherwise. |
| 9 | [**isascii()**](https://www.tutorialspoint.com/python/isascii_method.htm)  Returns True is all the characters in the string are from the ASCII character set |
| 10 | [**isdecimal()**](https://www.tutorialspoint.com/python/string_isdecimal.htm)  Checks if all the characters are decimal characters |
| 11 | [**isidentifier()**](https://www.tutorialspoint.com/python/python_string_isidentifier_method.htm)  Checks whether the string is a valid Python identifier |
| 12 | [**isprintable()**](https://www.tutorialspoint.com/python/isprintable_method.htm)  Checks whether all the characters in the string are printable |

## Find and Replace Methods

Following are the Find and Replace methods in Python −

|  |  |
| --- | --- |
| **Sr.No.** | **Method & Description** |
| 1 | [**count(sub, beg ,end)**](https://www.tutorialspoint.com/python/string_count.htm)  Counts how many times sub occurs in string or in a substring of string if starting index beg and ending index end are given. |
| 2 | [**find(sub, beg, end)**](https://www.tutorialspoint.com/python/string_find.htm)  Determine if sub 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. |
| 3 | [**index(sub, beg, end)**](https://www.tutorialspoint.com/python/string_index.htm)  Same as find(), but raises an exception if str not found. |
| 4 | [**replace(old, new [, max])**](https://www.tutorialspoint.com/python/string_replace.htm)  Replaces all occurrences of old in string with new or at most max occurrences if max given. |
| 5 | [**rfind(sub, beg, end)**](https://www.tutorialspoint.com/python/string_rfind.htm)  Same as find(), but search backwards in string. |
| 6 | [**rindex( sub, beg, end)**](https://www.tutorialspoint.com/python/string_rindex.htm)  Same as index(), but search backwards in string. |
| 7 | [**startswith(sub, beg, end)**](https://www.tutorialspoint.com/python/string_startswith.htm)  Determines if string or a substring of string (if starting index beg and ending index end are given) starts with substring sub; returns true if so and false otherwise. |
| 8 | [**endswith(suffix, beg, end)**](https://www.tutorialspoint.com/python/string_endswith.htm)  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. |

# Translation Methods

Following are the Translation methods of the string −

|  |  |
| --- | --- |
| **Sr.No.** | **Method & Description** |
| 1 | [**maketrans()**](https://www.tutorialspoint.com/python/string_maketrans.htm)  Returns a translation table to be used in translate function. |
| 2 | [**translate(table, deletechars="")**](https://www.tutorialspoint.com/python/string_translate.htm)  Translates string according to translation table str(256 chars), removing those in the del string. |

list1.append('e')

**my\_list.insert(index, new\_item)**.

list1.extend(another\_list)

**my\_list.remove(value)**, which deletes the first occurrence of value from **my\_list**.

**my\_list.pop(index)**

**my\_list.clear()**, which empties **my\_list**, leaving it with no elements.

del list2[0:2]

for item in list:

# Code block to execute

list\_name.sort(key=None, reverse=False)

words = ["pear", "apple", "banana", "kiwi", "watermelon"] words.sort(key=len)

sorted(iterable, key=None, reverse=False)

Changes to the nested objects in the copy will affect the original, and vice versa.== shallow\_copy = copy.copy(original)

deep\_copied\_list = copy.deepcopy(original\_list)

it creates a fully independent copy where changes to the copy do not affect the original, and vice versa.

original\_list = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

Methods to Add Elements to a List

The following are the methods specifically designed for adding new item/items into a list −

|  |  |
| --- | --- |
| **Sr.No.** | **Methods with Description** |
| 1 | [**list.append(obj)**](https://www.tutorialspoint.com/python/list_append.htm)  Appends object obj to list. |
| 2 | [**list.extend(seq)**](https://www.tutorialspoint.com/python/list_extend.htm)  Appends the contents of seq to list |
| 3 | [**list.insert(index, obj)**](https://www.tutorialspoint.com/python/list_insert.htm)  Inserts object obj into list at offset index |

Methods to Remove Elements from a List

The following are the methods specifically designed for removing items from a list −

|  |  |
| --- | --- |
| **Sr.No.** | **Methods with Description** |
| 1 | [**list.clear()**](https://www.tutorialspoint.com/python/python_list_clear_method.htm)  Clears all the contents of the list. |
| 2 | [**list.pop(obj=list[-1])**](https://www.tutorialspoint.com/python/list_pop.htm)  Removes and returns the last object or the object at the specified index from the list. |
| 3 | [**list.remove(obj)**](https://www.tutorialspoint.com/python/list_remove.htm)  Removes the first occurrence of object obj from the list. |

Methods to Access Elements in a List

These are the methods used for finding or counting items in a list −

|  |  |
| --- | --- |
| **Sr.No.** | **Methods with Description** |
| 1 | [**list.index(obj)**](https://www.tutorialspoint.com/python/list_index.htm)  Returns the lowest index in list that obj appears |
| 2 | [**list.count(obj)**](https://www.tutorialspoint.com/python/list_count.htm)  Returns count of how many times obj occurs in the list. |

Copying and Ordering Methods

These are the methods used for creating copies and arranging items in a list −

|  |  |
| --- | --- |
| **Sr.No.** | **Methods with Description** |
| 1 | [**list.copy()**](https://www.tutorialspoint.com/python/python_list_copy_method.htm)  Returns a copy of the list object. |
| 2 | [**list.sort([func])**](https://www.tutorialspoint.com/python/list_sort.htm)  Sorts the objects in the list in place, using a comparison function if provided. |
| 3 | [**list.reverse()**](https://www.tutorialspoint.com/python/list_reverse.htm)  Reverses the order of objects in the list in place. |

Unpack Tuple

tup1 = (10,20,30)

x, y, z = tup1

print ("x: ", x, "y: ", "z: ",z)

updated\_tuple = my\_tuple[:2] + (10,) + my\_tuple[3:]

updated\_tuple = (\*my\_tuple[:2], new\_value)

print(updated\_tuple) # Output: (1, 2, 4)

# Unpacking first two elements and the rest

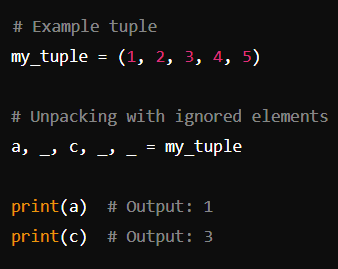
first, second, \*rest = my\_tuple

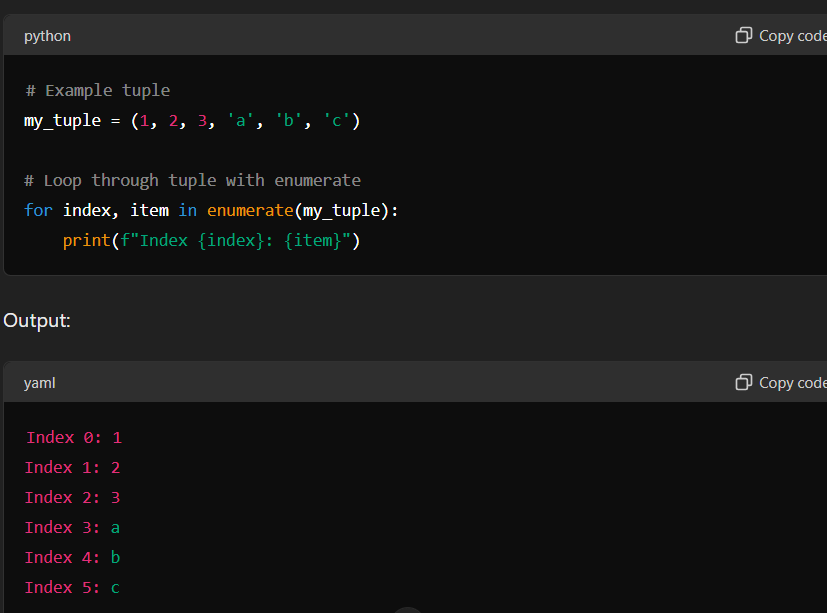
# Example nested tuple

nested\_tuple = (1, (2, 3), 4)

# Unpacking nested tuple

a, (b, c), d = nested\_tuple





|  |  |
| --- | --- |
| **Sr.No** | **Methods & Description** |
| 1 | **tuple.count(obj)**  Returns count of how many times obj occurs in tuple |
| 2 | **tuple.index(obj)**  Returns the lowest index in tuple that obj appears |

## Sets in Python

curly braces **{}** or the built-in **set()** f

my\_set.add(4)

my\_set.remove(3)

**discard()** function to remove an element from the set if it is present. Unlike remove(), discard() does not raise an error

my\_set.discard(5)

* **Union −** It combine elements from both sets using the union() function or the **|** operator.
* **Intersection −** It is used to get common elements using the intersection() function or the **&** operator.
* **Difference −** It is used to get elements that are in one set but not the other using the difference() function or the **-** operator.
* **Symmetric Difference −** It is used to get elements that are in either of the sets but not in both using the symmetric\_difference() method or the **^** operator.

# Adding an element that already exists (no change)

my\_set.add(2)

# Adding multiple elements using update()

my\_set.update([4, 5, 6])

print(my\_set) # Output: {1, 2, 3, 4, 5, 6}

# removing and returning an arbitrary element from the set

removed\_element = my\_set.pop()

# Removing all elements from the set

my\_set.clear()

In this example, we are defining two sets "s1" and "s2", and then using the difference\_update() method to remove elements from "s1" that are also in "s2" −

s1 = {1,2,3,4,5}

s2 = {4,5,6,7,8}

print ("s1 before running difference\_update: ", s1)

s1.difference\_update(s2)

print ("s1 after running difference\_update: ", s1)

After executing the above code, we get the following output −

s1 before running difference\_update: {1, 2, 3, 4, 5}

s1 after running difference\_update: {1, 2, 3}

set()

difference\_update()

symmetric\_difference() - difference operator (^) to create a new set "result\_set" containing elements that are in either "set1" or "set2" but not in both –

## Remove Uncommon Set Items

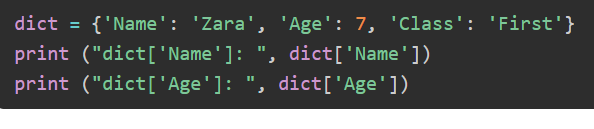
set1.intersection\_update(set2)

set.intersection(obj)

s1.symmetric\_difference\_update(s2)- it contains elements that are in either "s1" or "s2", but not in both −

**Dictionaries**

ython doesn't accept mutable objects such as list as key, and raises TypeError.

****

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}

dict['Age'] = 8; # update existing entry

dict['School'] = "DPS School"; # Add new entry

del dict['Name'];

|  |  |
| --- | --- |
| **Sr.No.** | **Methods with Description** |
| 1 | [**dict.clear()**](https://www.tutorialspoint.com/python/dictionary_clear.htm)  Removes all elements of dictionary *dict* |
| 2 | [**dict.copy()**](https://www.tutorialspoint.com/python/dictionary_copy.htm)  Returns a shallow copy of dictionary *dict* |
| 3 | [**dict.fromkeys()**](https://www.tutorialspoint.com/python/dictionary_fromkeys.htm)  Create a new dictionary with keys from seq and values *set* to *value*. |
| 4 | [**dict.get(key, default=None)**](https://www.tutorialspoint.com/python/dictionary_get.htm)  For *key* key, returns value or default if key not in dictionary |
| 5 | [**dict.has\_key(key)**](https://www.tutorialspoint.com/python/dictionary_has_key.htm)  Returns *true* if key in dictionary *dict*, *false* otherwise |
| 6 | [**dict.items()**](https://www.tutorialspoint.com/python/dictionary_items.htm)  Returns a list of *dict*'s (key, value) tuple pairs |
| 7 | [**dict.keys()**](https://www.tutorialspoint.com/python/dictionary_keys.htm)  Returns list of dictionary dict's keys |
| 8 | [**dict.setdefault(key, default=None)**](https://www.tutorialspoint.com/python/dictionary_setdefault.htm)  Similar to get(), but will set dict[key]=default if *key* is not already in dict |
| 9 | [**dict.update(dict2)**](https://www.tutorialspoint.com/python/dictionary_update.htm)  Adds dictionary *dict2*'s key-values pairs to *dict* |
| 10 | [**dict.values()**](https://www.tutorialspoint.com/python/dictionary_values.htm)  Returns list of dictionary *dict*'s values |

Built-in Functions with Dictionaries

Following are the built-in functions we can use with Dictionaries −

|  |  |
| --- | --- |
| **Sr.No.** | **Function with Description** |
| 1 | [**cmp(dict1, dict2)**](https://www.tutorialspoint.com/python/dictionary_cmp.htm)  Compares elements of both dict. |
| 2 | [**len(dict)**](https://www.tutorialspoint.com/python/dictionary_len.htm)  Gives the total length of the dictionary. This would be equal to the number of items in the dictionary. |
| 3 | [**str(dict)**](https://www.tutorialspoint.com/python/dictionary_str.htm)  Produces a printable string representation of a dictionary |
| 4 | [**type(variable)**](https://www.tutorialspoint.com/python/dictionary_type.htm)  Returns the type of the passed variable. If passed variable is dictionary, then it would return a dictionary type. |

my\_dict.update({'age': 31, 'city': 'San Francisco'})

* **keys()**: Returns a view object that displays a list of all the keys in the dictionary.

# Getting keys

keys = my\_dict.keys()

print(keys) # Output: dict\_keys(['name', 'city'])

* **values()**: Returns a view object that displays a list of all the values in the dictionary.

# Getting values

values = my\_dict.values()

print(values) # Output: dict\_values(['John', 'San Francisco'])

* **items()**: Returns a view object that displays a list of key-value tuple pairs.

# Getting key-value pairs

items = my\_dict.items()

print(items) # Output: dict\_items([('name', 'John'), ('city', 'San Francisco')])

### Summary

Val = dict.get("key")

### **Example**

d1=dict([('a', 100), ('b', 200)])

d2 = dict((('a', 'one'), ('b', 'two')))

print ('d1: ', d1)

print ('d2: ', d2)

It will produce the following **output** −

d1: {'a': 100, 'b': 200}

d2: {'a': 'one', 'b': 'two'}

d1=dict(a= 100, b=200)

d2 = dict(a='one', b='two')

print ('d1: ', d1)

print ('d2: ', d2)

It will produce the following **output** −

d1: {'a': 100, 'b': 200}

d2: {'a': 'one', 'b': 'two'}

d1.update(k1=v1, k2=v2)

### Dictionary View Objects

Python dictionaries provide three different view objects that allow you to view the keys, values, or key-value pairs of a dictionary without creating new lists or tuples. These view objects reflect changes made to the dictionary after the view object has been created.

1. **keys()**: Returns a view object that displays a list of all the keys in the dictionary.

python

Copy code

# Example dictionary

my\_dict = {'name': 'John', 'age': 30, 'city': 'New York'}

# Getting keys view object

keys\_view = my\_dict.keys()

print(keys\_view) # Output: dict\_keys(['name', 'age', 'city'])

1. **values()**: Returns a view object that displays a list of all the values in the dictionary.

python

Copy code

# Getting values view object

values\_view = my\_dict.values()

print(values\_view) # Output: dict\_values(['John', 30, 'New York'])

1. **items()**: Returns a view object that displays a list of key-value tuple pairs.

python

Copy code

# Getting items view object

items\_view = my\_dict.items()

print(items\_view) # Output: dict\_items([('name', 'John'), ('age', 30), ('city', 'New York')])

### Looping Through Dictionaries

You can iterate over dictionaries using various methods, including iterating over keys, values, or key-value pairs using the items() method.

python

Copy code

# Iterating over keys

for key in my\_dict:

print(key, my\_dict[key])

# Iterating over values

for value in my\_dict.values():

print(value)

# Iterating over key-value pairs

for key, value in my\_dict.items():

print(key, value)

### Copying Dictionaries

Python dictionaries can be copied using either the copy() method or the dictionary comprehension method (dict() constructor).

python

Copy code

# Using copy() method

copy\_dict = my\_dict.copy()

# Using dictionary comprehension

copy\_dict = dict(my\_dict)

### Nested Dictionaries

Dictionaries in Python can contain other dictionaries as values, allowing for nested data structures.

python

Copy code

# Example of nested dictionaries

employee = {

'emp\_id': 1234,

'name': 'Alice',

'contact': {

'email': 'alice@example.com',

'phone': '123-456-7890'

}

}

# Accessing nested values

print(employee['name']) # Output: Alice

print(employee['contact']['email']) # Output: alice@example.com

### Dictionary Methods

Python dictionaries come with a variety of methods for adding, modifying, accessing, and removing elements. Here are a few commonly used methods:

* **clear()**: Removes all items from the dictionary.
* **update()**: Updates the dictionary with the key-value pairs from another dictionary or iterable.
* **pop(key[, default])**: Removes and returns the value associated with the key. If the key is not found and a default value is provided, it returns the default value.
* **popitem()**: Removes and returns an arbitrary key-value pair as a tuple.

These methods, along with view objects and nesting capabilities, make dictionaries a powerful and flexible tool in Python for organizing and manipulating data.