Python Arithmetic Operators

Let a =10 and b=20

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| + Addition | Adds values on either side of the operator. | a + b = 30 |
| - Subtraction | Subtracts right hand operand from left hand operand. | a – b = -10 |
| \* Multiplication | Multiplies values on either side of the operator | a \* b = 200 |
| / Division | Divides left hand operand by right hand operand | b / a = 2 |
| % Modulus | Divides left hand operand by right hand operand and returns remainder | b % a = 0 |
| \*\* Exponent | Performs exponential (power) calculation on operators | a\*\*b =10 to the power 20 |
| // | Floor Division - The division of operands where the result is the quotient in which the digits after the decimal point are removed. But if one of the operands is negative, the result is floored, i.e., rounded away from zero (towards negative infinity) − | 9//2 = 4 and 9.0//2.0 = 4.0, -11//3 = -4, -11.0//3 = -4.0 |

Python Comparison Operators

These operators compare the values on either sides of them and decide the relation among them. They are also called Relational operators.

Let

A=10 and

B =20

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| == | If the values of two operands are equal, then the condition becomes true. | (a == b) is not true. |
| != | If values of two operands are not equal, then condition becomes true. | (a != b) is true. |
| <> | If values of two operands are not equal, then condition becomes true. | (a <> b) is true. This is similar to != operator. |
| > | If the value of left operand is greater than the value of right operand, then condition becomes true. | (a > b) is not true. |
| < | If the value of left operand is less than the value of right operand, then condition becomes true. | (a < b) is true. |
| >= | If the value of left operand is greater than or equal to the value of right operand, then condition becomes true. | (a >= b) is not true. |
| <= | If the value of left operand is less than or equal to the value of right operand, then condition becomes true. | (a <= b) is true. |

Python Assignment Operators

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| = | Assigns values from right side operands to left side operand | c = a + b assigns value of a + b into c |
| += Add AND | It adds right operand to the left operand and assign the result to left operand | c += a is equivalent to c = c + a |
| -= Subtract AND | It subtracts right operand from the left operand and assign the result to left operand | c -= a is equivalent to c = c - a |
| \*= Multiply AND | It multiplies right operand with the left operand and assign the result to left operand | c \*= a is equivalent to c = c \* a |
| /= Divide AND | It divides left operand with the right operand and assign the result to left operand | c /= a is equivalent to c = c / a |
| %= Modulus AND | It takes modulus using two operands and assign the result to left operand | c %= a is equivalent to c = c % a |
| \*\*= Exponent AND | Performs exponential (power) calculation on operators and assign value to the left operand | c \*\*= a is equivalent to c = c \*\* a |
| //= Floor Division | It performs floor division on operators and assign value to the left operand | c //= a is equivalent to c = c // a |

Python Bitwise Operators

Bitwise operator works on bits and performs bit by bit operation. Assume if a = 60; and b = 13; Now in the binary format their values will be 0011 1100 and 0000 1101 respectively. Following table lists out the bitwise operators supported by Python language with an example each in those, we use the above two variables (a and b) as operands −

a = 0011 1100

b = 0000 1101

-----------------

a&b = 0000 1100

a|b = 0011 1101

a^b = 0011 0001

~a  = 1100 0011

There are following Bitwise operators supported by Python language

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| & Binary AND | Operator copies a bit to the result if it exists in both operands | (a & b) (means 0000 1100) |
| | Binary OR | It copies a bit if it exists in either operand. | (a | b) = 61 (means 0011 1101) |
| ^ Binary XOR | It copies the bit if it is set in one operand but not both. | (a ^ b) = 49 (means 0011 0001) |
| ~ Binary Ones Complement | It is unary and has the effect of 'flipping' bits. | (~a ) = -61 (means 1100 0011 in 2's complement form due to a signed binary number. |
| << Binary Left Shift | The left operands value is moved left by the number of bits specified by the right operand. | a << 2 = 240 (means 1111 0000) |
| >> Binary Right Shift | The left operands value is moved right by the number of bits specified by the right operand. | a >> 2 = 15 (means 0000 1111) |

Python Logical Operators

There are following logical operators supported by Python language. Assume variable a holds 10 and variable b holds 20 then

[ [Show Example](https://www.tutorialspoint.com/python/logical_operators_example.htm) ]

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| and Logical AND | If both the operands are true then condition becomes true. | (a and b) is true. |
| or Logical OR | If any of the two operands are non-zero then condition becomes true. | (a or b) is true. |
| not Logical NOT | Used to reverse the logical state of its operand. | Not(a and b) is false. |

Python Membership Operators

Python’s membership operators test for membership in a sequence, such as strings, lists, or tuples. There are two membership operators as explained below −

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| in | Evaluates to true if it finds a variable in the specified sequence and false otherwise. | x in y, here in results in a 1 if x is a member of sequence y. |
| not in | Evaluates to true if it does not finds a variable in the specified sequence and false otherwise. | x not in y, here not in results in a 1 if x is not a member of sequence y. |

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| is | Evaluates to true if the variables on either side of the operator point to the same object and false otherwise. | x is y, here **is** results in 1 if id(x) equals id(y). |
| is not | Evaluates to false if the variables on either side of the operator point to the same object and true otherwise. | x is not y, here **is not** results in 1 if id(x) is not equal to id(y). |

Python Identity Operators

Identity operators compare the memory locations of two objects. There are two Identity operators explained below

Python Operators Precedence

The following table lists all operators from highest precedence to lowest.

[ [Show Example](https://www.tutorialspoint.com/python/operators_precedence_example.htm) ]

|  |  |
| --- | --- |
| **Sr.No.** | **Operator & Description** |
| 1 | **\*\***  Exponentiation (raise to the power) |
| 2 | **~ + -**  Complement, unary plus and minus (method names for the last two are +@ and -@) |
| 3 | **\* / % //**  Multiply, divide, modulo and floor division |
| 4 | **+ -**  Addition and subtraction |
| 5 | **>> <<**  Right and left bitwise shift |
| 6 | **&**  Bitwise 'AND' |
| 7 | **^ |**  Bitwise exclusive `OR' and regular `OR' |
| 8 | **<= < > >=**  Comparison operators |
| 9 | **<> == !=**  Equality operators |
| 10 | **= %= /= //= -= += \*= \*\*=**  Assignment operators |
| 11 | **is is not**  Identity operators |
| 12 | **in not in**  Membership operators |
| 13 | **not or and**  Logical operators |

NUMBERS

Number data types store numeric values. They are immutable data types, means that changing the value of a number data type results in a newly allocated object.

Python supports four different numerical types −

* **int (signed integers)** − They are often called just integers or ints, are positive or negative whole numbers with no decimal point.
* **long (long integers )** − Also called longs, they are integers of unlimited size, written like integers and followed by an uppercase or lowercase L.
* **float (floating point real values)** − Also called floats, they represent real numbers and are written with a decimal point dividing the integer and fractional parts. Floats may also be in scientific notation, with E or e indicating the power of 10 (2.5e2 = 2.5 x 102 = 250).
* **complex (complex numbers)** − are of the form a + bJ, where a and b are floats and J (or j) represents the square root of -1 (which is an imaginary number). The real part of the number is a, and the imaginary part is b. Complex numbers are not used much in Python programming.
* Here are some examples of numbers

|  |  |  |  |
| --- | --- | --- | --- |
| **int** | **long** | **float** | **complex** |
| 10 | 51924361L | 0.0 | 3.14j |
| 100 | -0x19323L | 15.20 | 45.j |
| -786 | 0122L | -21.9 | 9.322e-36j |
| 080 | 0xDEFABCECBDAECBFBAEL | 32.3+e18 | .876j |

Number Type Conversion

Python converts numbers internally in an expression containing mixed types to a common type for evaluation. But sometimes, you need to coerce a number explicitly from one type to another to satisfy the requirements of an operator or function parameter.

* Type **int(x)** to convert x to a plain integer.
* Type **long(x)** to convert x to a long integer.
* Type **float(x)** to convert x to a floating-point number.
* Type **complex(x)** to convert x to a complex number with real part x and imaginary part zero.
* Type **complex(x, y)** to convert x and y to a complex number with real part x and imaginary part y. x and y are numeric expressions

Mathematical Functions

Python includes following functions that perform mathematical calculations.

|  |  |
| --- | --- |
| **Sr.No.** | **Function & Returns ( description )** |
| 1 | [abs(x)](https://www.tutorialspoint.com/python/number_abs.htm)  The absolute value of x: the (positive) distance between x and zero. |
| 2 | [ceil(x)](https://www.tutorialspoint.com/python/number_ceil.htm)  The ceiling of x: the smallest integer not less than x |
| 3 | [cmp(x, y)](https://www.tutorialspoint.com/python/number_cmp.htm)  -1 if x < y, 0 if x == y, or 1 if x > y |
| 4 | [exp(x)](https://www.tutorialspoint.com/python/number_exp.htm)  The exponential of x: ex |
| 5 | [fabs(x)](https://www.tutorialspoint.com/python/number_fabs.htm)  The absolute value of x. |
| 6 | [floor(x)](https://www.tutorialspoint.com/python/number_floor.htm)  The floor of x: the largest integer not greater than x |
| 7 | [log(x)](https://www.tutorialspoint.com/python/number_log.htm)  The natural logarithm of x, for x> 0 |
| 8 | [log10(x)](https://www.tutorialspoint.com/python/number_log10.htm)  The base-10 logarithm of x for x> 0. |
| 9 | [max(x1, x2,...)](https://www.tutorialspoint.com/python/number_max.htm)  The largest of its arguments: the value closest to positive infinity |
| 10 | [min(x1, x2,...)](https://www.tutorialspoint.com/python/number_min.htm)  The smallest of its arguments: the value closest to negative infinity |
| 11 | [modf(x)](https://www.tutorialspoint.com/python/number_modf.htm)  The fractional and integer parts of x in a two-item tuple. Both parts have the same sign as x. The integer part is returned as a float. |
| 12 | [pow(x, y)](https://www.tutorialspoint.com/python/number_pow.htm)  The value of x\*\*y. |
| 13 | [round(x [,n])](https://www.tutorialspoint.com/python/number_round.htm)  **x** rounded to n digits from the decimal point. Python rounds away from zero as a tie-breaker: round(0.5) is 1.0 and round(-0.5) is -1.0. |
| 14 | [sqrt(x)](https://www.tutorialspoint.com/python/number_sqrt.htm)  The square root of x for x > 0 |

Random Number Functions

Random numbers are used for games, simulations, testing, security, and privacy applications. Python includes following functions that are commonly used.

|  |  |
| --- | --- |
| **Sr.No.** | **Function & Description** |
| 1 | [choice(seq)](https://www.tutorialspoint.com/python/number_choice.htm)  A random item from a list, tuple, or string. |
| 2 | [randrange ([start,] stop [,step])](https://www.tutorialspoint.com/python/number_randrange.htm)  A randomly selected element from range(start, stop, step) |
| 3 | [random()](https://www.tutorialspoint.com/python/number_random.htm)  A random float r, such that 0 is less than or equal to r and r is less than 1 |
| 4 | [seed([x])](https://www.tutorialspoint.com/python/number_seed.htm)  Sets the integer starting value used in generating random numbers. Call this function before calling any other random module function. Returns None. |
| 5 | [shuffle(lst)](https://www.tutorialspoint.com/python/number_shuffle.htm)  Randomizes the items of a list in place. Returns None. |
| 6 | [uniform(x, y)](https://www.tutorialspoint.com/python/number_uniform.htm)  A random float r, such that x is less than or equal to r and r is less than y |

Trigonometric Functions

Python includes following functions that perform trigonometric calculations.

|  |  |
| --- | --- |
| **Sr.No.** | **Function & Description** |
| 1 | [acos(x)](https://www.tutorialspoint.com/python/number_acos.htm)  Return the arc cosine of x, in radians. |
| 2 | [asin(x)](https://www.tutorialspoint.com/python/number_asin.htm)  Return the arc sine of x, in radians. |
| 3 | [atan(x)](https://www.tutorialspoint.com/python/number_atan.htm)  Return the arc tangent of x, in radians. |
| 4 | [atan2(y, x)](https://www.tutorialspoint.com/python/number_atan2.htm)  Return atan(y / x), in radians. |
| 5 | [cos(x)](https://www.tutorialspoint.com/python/number_cos.htm)  Return the cosine of x radians. |
| 6 | [hypot(x, y)](https://www.tutorialspoint.com/python/number_hypot.htm)  Return the Euclidean norm, sqrt(x\*x + y\*y). |
| 7 | [sin(x)](https://www.tutorialspoint.com/python/number_sin.htm)  Return the sine of x radians. |
| 8 | [tan(x)](https://www.tutorialspoint.com/python/number_tan.htm)  Return the tangent of x radians. |
| 9 | [degrees(x)](https://www.tutorialspoint.com/python/number_degrees.htm)  Converts angle x from radians to degrees. |
| 10 | [radians(x)](https://www.tutorialspoint.com/python/number_radians.htm)  Converts angle x from degrees to radians. |

Mathematical Constants

The module also defines two mathematical constants −

|  |  |
| --- | --- |
| **Sr.No.** | **Constants & Description** |
| 1 | **pi**  The mathematical constant pi. |
| 2 | **e**  The mathematical constant e. |

## 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 |

## String Formatting Operator

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

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

|  |  |
| --- | --- |
| **Format Symbol** | **Conversion** |
| %c | character |
| %s | string conversion via str() prior to formatting |
| %i | signed decimal integer |
| %d | signed decimal integer |
| %u | unsigned decimal integer |
| %o | octal integer |
| %x | hexadecimal integer (lowercase letters) |
| %X | hexadecimal integer (UPPERcase letters) |
| %e | exponential notation (with lowercase 'e') |
| %E | exponential notation (with UPPERcase 'E') |
| %f | floating point real number |
| %g | the shorter of %f and %e |
| %G | the shorter of %f and %E |

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

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

## 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 | [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. |
| 3 | [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. |
| 4 | [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. |
| 5 | [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'. |
| 6 | [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. |
| 7 | [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. |
| 8 | [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. |
| 9 | [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. |
| 10 | [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. |
| 11 | [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. |
| 12 | [isdigit()](https://www.tutorialspoint.com/python/string_isdigit.htm)  Returns true if string contains only digits and false otherwise. |
| 13 | [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. |
| 14 | [isnumeric()](https://www.tutorialspoint.com/python/string_isnumeric.htm)  Returns true if a unicode string contains only numeric characters and false otherwise. |
| 15 | [isspace()](https://www.tutorialspoint.com/python/string_isspace.htm)  Returns true if string contains only whitespace characters and false otherwise. |
| 16 | [istitle()](https://www.tutorialspoint.com/python/string_istitle.htm)  Returns true if string is properly "titlecased" and false otherwise. |
| 17 | [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. |
| 18 | [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. |
| 19 | [len(string)](https://www.tutorialspoint.com/python/string_len.htm)  Returns the length of the string |
| 20 | [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. |
| 21 | [lower()](https://www.tutorialspoint.com/python/string_lower.htm)  Converts all uppercase letters in string to lowercase. |
| 22 | [lstrip()](https://www.tutorialspoint.com/python/string_lstrip.htm)  Removes all leading whitespace in string. |
| 23 | [maketrans()](https://www.tutorialspoint.com/python/string_maketrans.htm)  Returns a translation table to be used in translate function. |
| 24 | [max(str)](https://www.tutorialspoint.com/python/string_max.htm)  Returns the max alphabetical character from the string str. |
| 25 | [min(str)](https://www.tutorialspoint.com/python/string_min.htm)  Returns the min alphabetical character from the string str. |
| 26 | [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. |
| 27 | [rfind(str, beg=0,end=len(string))](https://www.tutorialspoint.com/python/string_rfind.htm)  Same as find(), but search backwards in string. |
| 28 | [rindex( str, beg=0, end=len(string))](https://www.tutorialspoint.com/python/string_rindex.htm)  Same as index(), but search backwards in string. |
| 29 | [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. |
| 30 | [rstrip()](https://www.tutorialspoint.com/python/string_rstrip.htm)  Removes all trailing whitespace of string. |
| 31 | [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. |
| 32 | [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. |
| 33 | [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. |
| 34 | [strip([chars])](https://www.tutorialspoint.com/python/string_strip.htm)  Performs both lstrip() and rstrip() on string. |
| 35 | [swapcase()](https://www.tutorialspoint.com/python/string_swapcase.htm)  Inverts case for all letters in string. |
| 36 | [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. |
| 37 | [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. |
| 38 | [upper()](https://www.tutorialspoint.com/python/string_upper.htm)  Converts lowercase letters in string to uppercase. |
| 39 | [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). |
| 40 | [isdecimal()](https://www.tutorialspoint.com/python/string_isdecimal.htm)  Returns true if a unicode string contains only decimal characters and false otherwise. |

List

Python includes following list methods

|  |  |
| --- | --- |
| **Sr.No.** | **Methods with Description** |
| 1 | [list.append(obj)](https://www.tutorialspoint.com/python/list_append.htm)  Appends object obj to list |
| 2 | [list.count(obj)](https://www.tutorialspoint.com/python/list_count.htm)  Returns count of how many times obj occurs in list |
| 3 | [list.extend(seq)](https://www.tutorialspoint.com/python/list_extend.htm)  Appends the contents of seq to list |
| 4 | [list.index(obj)](https://www.tutorialspoint.com/python/list_index.htm)  Returns the lowest index in list that obj appears |
| 5 | [list.insert(index, obj)](https://www.tutorialspoint.com/python/list_insert.htm)  Inserts object obj into list at offset index |
| 6 | [list.pop(obj=list[-1])](https://www.tutorialspoint.com/python/list_pop.htm)  Removes and returns last object or obj from list |
| 7 | [list.remove(obj)](https://www.tutorialspoint.com/python/list_remove.htm)  Removes object obj from list |
| 8 | [list.reverse()](https://www.tutorialspoint.com/python/list_reverse.htm)  Reverses objects of list in place |
| 9 | [list.sort([func])](https://www.tutorialspoint.com/python/list_sort.htm)  Sorts objects of list, use compare func if given |

Tuple

## Built-in Tuple Functions

Python includes the following tuple functions −

|  |  |
| --- | --- |
| **Sr.No.** | **Function with Description** |
| 1 | [cmp(tuple1, tuple2)](https://www.tutorialspoint.com/python/tuple_cmp.htm)  Compares elements of both tuples. |
| 2 | [len(tuple)](https://www.tutorialspoint.com/python/tuple_len.htm)  Gives the total length of the tuple. |
| 3 | [max(tuple)](https://www.tutorialspoint.com/python/tuple_max.htm)  Returns item from the tuple with max value. |
| 4 | [min(tuple)](https://www.tutorialspoint.com/python/tuple_min.htm)  Returns item from the tuple with min value. |
| 5 | [tuple(seq)](https://www.tutorialspoint.com/python/tuple_tuple.htm)  Converts a list into tuple. |

**Dictionary**

Python includes following dictionary methods −

|  |  |
| --- | --- |
| **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 |

## Set Methods

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

|  |  |
| --- | --- |
| **Method** | **Description** |
| [add()](https://www.w3schools.com/python/ref_set_add.asp) | Adds an element to the set |
| [clear()](https://www.w3schools.com/python/ref_set_clear.asp) | Removes all the elements from the set |
| [copy()](https://www.w3schools.com/python/ref_set_copy.asp) | Returns a copy of the set |
| [difference()](https://www.w3schools.com/python/ref_set_difference.asp) | Returns a set containing the difference between two or more sets |
| [difference\_update()](https://www.w3schools.com/python/ref_set_difference_update.asp) | Removes the items in this set that are also included in another, specified set |
| [discard()](https://www.w3schools.com/python/ref_set_discard.asp) | Remove the specified item |
| [intersection()](https://www.w3schools.com/python/ref_set_intersection.asp) | Returns a set, that is the intersection of two other sets |
| [intersection\_update()](https://www.w3schools.com/python/ref_set_intersection_update.asp) | Removes the items in this set that are not present in other, specified set(s) |
| [isdisjoint()](https://www.w3schools.com/python/ref_set_isdisjoint.asp) | Returns whether two sets have a intersection or not |
| [issubset()](https://www.w3schools.com/python/ref_set_issubset.asp) | Returns whether another set contains this set or not |
| [issuperset()](https://www.w3schools.com/python/ref_set_issuperset.asp) | Returns whether this set contains another set or not |
| [pop()](https://www.w3schools.com/python/ref_set_pop.asp) | Removes an element from the set |
| [remove()](https://www.w3schools.com/python/ref_set_remove.asp) | Removes the specified element |
| [symmetric\_difference()](https://www.w3schools.com/python/ref_set_symmetric_difference.asp) | Returns a set with the symmetric differences of two sets |
| [symmetric\_difference\_update()](https://www.w3schools.com/python/ref_set_symmetric_difference_update.asp) | inserts the symmetric differences from this set and another |
| [union()](https://www.w3schools.com/python/ref_set_union.asp) | Return a set containing the union of sets |
| [update()](https://www.w3schools.com/python/ref_set_update.asp) | Update the set with the union of this set and others |
|  |  |
| Conditional  Statements: |  |
| Sr.No. | Statement & Description |
| 1 | [if statements](https://www.tutorialspoint.com/python/python_if_statement.htm)  An if statement consists of a boolean expression followed by one or more statements. |
| 2 | [if...else statements](https://www.tutorialspoint.com/python/python_if_else.htm)  An if statement can be followed by an optional else statement, which executes when the boolean expression is FALSE. |
| 3 | [nested if statements](https://www.tutorialspoint.com/python/nested_if_statements_in_python.htm)  You can use one if or else if statement inside another if or else if statement(s). |

## Overview of OOP Terminology

* **Class** − A user-defined prototype for an object that defines a set of attributes that characterize any object of the class. The attributes are data members (class variables and instance variables) and methods, accessed via dot notation.
* **Class variable** − A variable that is shared by all instances of a class. Class variables are defined within a class but outside any of the class's methods. Class variables are not used as frequently as instance variables are.
* **Data member** − A class variable or instance variable that holds data associated with a class and its objects.
* **Function overloading** − The assignment of more than one behavior to a particular function. The operation performed varies by the types of objects or arguments involved.
* **Instance variable** − A variable that is defined inside a method and belongs only to the current instance of a class.
* **Inheritance** − The transfer of the characteristics of a class to other classes that are derived from it.
* **Instance** − An individual object of a certain class. An object obj that belongs to a class Circle, for example, is an instance of the class Circle.
* **Instantiation** − The creation of an instance of a class.
* **Method** − A special kind of function that is defined in a class definition.
* **Object** − A unique instance of a data structure that's defined by its class. An object comprises both data members (class variables and instance variables) and methods.
* **Operator overloading** − The assignment of more than one function to a particular operator.

**File handling:**

|  |  |
| --- | --- |
| **Sr.No.** | **Modes & Description** |
| 1 | **r**  Opens a file for reading only. The file pointer is placed at the beginning of the file. This is the default mode. |
| 2 | **rb**  Opens a file for reading only in binary format. The file pointer is placed at the beginning of the file. This is the default mode. |
| 3 | **r+**  Opens a file for both reading and writing. The file pointer placed at the beginning of the file. |
| 4 | **rb+**  Opens a file for both reading and writing in binary format. The file pointer placed at the beginning of the file. |
| 5 | **w**  Opens a file for writing only. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing. |
| 6 | **wb**  Opens a file for writing only in binary format. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing. |
| 7 | **w+**  Opens a file for both writing and reading. Overwrites the existing file if the file exists. If the file does not exist, creates a new file for reading and writing. |
| 8 | **wb+**  Opens a file for both writing and reading in binary format. Overwrites the existing file if the file exists. If the file does not exist, creates a new file for reading and writing. |
| 9 | **a**  Opens a file for appending. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing. |
| 10 | **ab**  Opens a file for appending in binary format. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing. |
| 11 | **a+**  Opens a file for both appending and reading. The file pointer is at the end of the file if the file exists. The file opens in the append mode. If the file does not exist, it creates a new file for reading and writing. |
| 12 | **ab+**  Opens a file for both appending and reading in binary format. The file pointer is at the end of the file if the file exists. The file opens in the append mode. If the file does not exist, it creates a new file for reading and writing. |

## The *file* Object Attributes

Once a file is opened and you have one *file* object, you can get various information related to that file.

Here is a list of all attributes related to file object −

|  |  |
| --- | --- |
| **Sr.No.** | **Attribute & Description** |
| 1 | **file.closed**  Returns true if file is closed, false otherwise. |
| 2 | **file.mode**  Returns access mode with which file was opened. |
| 3 | **file.name**  Returns name of the file. |
| 4 | **file.softspace**  Returns false if space explicitly required with print, true otherwise. |