Functions for data type conversion

Function	Description
`int(x)`	Converts x to an integer.
`float(x)`	Converts x to a floating-point number.
`complex(x,y)`	Creates a complex number as x + yi.
`str(x)`	Converts x to a string representation.
`repr(x)`	Converts x to an expression string.
`eval(str)`	Evaluates str and returns an object.
`tuple(str)`	Converts str to a tuple.
`list(str)`	Converts str to a list.
`set(str)`	Converts str to a set.
`dict(str)`	Creates a dictionary of str.
`frozenset(str)`	Converts str to a frozen set.
`chr(x)`	Converts integer x to a character.
`unichr(x)`	Converts integer x to a Unicode character.
`ord(x)`	Converts single character x to its integer value.
`hex(x)`	Converts integer x to a hexadecimal string.
`oct(x)`	Converts integer x to an octal string.

Arithmetic operators

Operator	Description
`+ (addition)`	Adds operands on either side of the operator.
`- (subtraction)`	Subtracts the right-side operand from the left-side operand.
`* (multiplication)`	Multiplies operands on either side of the operator.
`/ (division)`	Divides left-side operand by right-side operand and returns the quotient.
`% (modulus)`	Divides left-side operand by right-side operand and returns remainder.
`** (exponent)`	Performs exponential (power) calculation on operators.
`// (floor division)`	Divides the left-side operand by right side operand and returns the quotient after removing the digits after the decimal point.

Relational (comparison) operators

Operator	Description
==	If the values of the two operands are equal, then the condition becomes true. Otherwise, it is false.
!=	If the values of the two operands are not equal, then the condition becomes true. Otherwise, it is false.
>	If the value of the left operand is greater than the value of the right operand, then the condition becomes true. Otherwise, it is false.
<	If the value of the left operand is less than the value of the right operand, then the condition becomes true. Otherwise, it is false.
>=	If the value of the left operand is greater than or equal to the value of the right operand, then the condition becomes true. Otherwise, it is false.
<=	If the value of the left operand is less than or equal to the value of the right operand, then the condition becomes true. Otherwise, it is false.

Assignment and in-place operators

Operator	Description
=	Assigns values from the right operand to the left operand.
+=	Adds the right operand to the left side operand and assigns the result to the left operand.
-=	Subtracts the right operand from the left operand and assigns the result to the left operand.
*=	Multiplies the right operand with the left operand and assigns the result to the left operand.
/=	Divides the left operand with the right operand and assigns the result to the left operand.
%=	Takes modulus using two operands and assigns the result to the left operand.
**=	Performs exponential (power) calculation on operators and assigns value to the left operand.
//=	Performs floor division on operators and assigns value to the left operand.

Logical operators

Operator	Description
and (logical AND)	If both the operands are true then the condition becomes true.
or (logical OR)	If any of the two operands are true then the condition becomes true.
not (logical not)	Reverses the logical state of its operand.

Bitwise operators

Operator	Description
`& (binary AND)`	Copies a bit to the result if it exists in both operands.
`| (binary OR)`	Copies a bit if it exists in either operand.
`^ (binary XOR)`	Copies the bit, if it is set in one operand but not both.

Shift operators

Operator	Description
`<< (binary left shift)`	The left operand's value is moved left by the number of bits specified by the right operand.
`>> (binary right shift`	The left operand's value is moved right by the number of bits specified by the right operand.

Membership operators

Operator	Description
`in`	Evaluates to true, if it finds a variable in the specified sequence and false otherwise.
`not in`	Evaluates to true, if it does not find a variable in the specified sequence and false otherwise.

Identity operators

Operator	Description
`is`	Evaluates to true if the variables on either side of the operator point to the same object and false otherwise.
`is not`	Evaluates to false if the variables on either side of the operator point to the same object and true otherwise.

Unary operators

Operator	Description
`- (unary minus)`	Negates the operand value.
`+ (unary plus)`	Restores the operand value.
`~ (ones complement)`	Flips bits of the operand.