**Operators**: operators are used to performing operations on variable and values

Python divides the operators into the following groups:

* Arthematic operators
* Relational operators
* Logical operators
* Assignment operators
* Unary operators
* Bitwise operators
* Identity operators
* Membership operators

1. **Arthematic operators** : addition (+), subtraction(-), multiplication(\*), division(/), modulus(%)

Power(\*\*), floor or integer division (//)

|  |  |  |
| --- | --- | --- |
| Operators | Name | Example |
| + | Addition | a+b |
| - | Subtraction | a-b |
| \* | Multiplication | a\*b |
| / | Division | a/b |
| // | Floor or integer division | a//b |
| % | Modular division | a%b |
| \*\* | Power | a\*\*b |

1. **Relational operators:** the relational operators are less than(<), greater than(>), less than equal (<=), greater than equal(>=), equally (==) , not equal (!=)

|  |  |  |
| --- | --- | --- |
| Operator | Name | Example |
| < | Less than | a<b |
| > | Greater than | a>b |
| <= | Less than equal | a<=b |
| >= | Greater than equal | a>=b |
| == | equally | a == b |
| != | Not equal | a != b |

1. **Logical operators:** the logical operators are and , or , not

|  |  |  |
| --- | --- | --- |
| Operator | Condition | Example |
| and | Both condition is true | a<b and b>a |
| or | At least one condition true | a<b and b<a |
| not | If its true the result is false | not(a<b) |

1. **Assignment operators**: the assignment operators are +=, -=, /=, \*=, %=

|  |  |  |
| --- | --- | --- |
| Operator | Condition | Example |
| Addition | a+=b | a = a+b |
| subtraction | a-=b | a = a-b |
| multiplication | a\*=b | a = a\*b |
| division | a/=b | a=a/b |
| modulus | a%=b | a=a%b |

1. **Unary operators** : in python unary operators operate on a single operand Examples include the unary plus (+), unary minus(-), and logical negoation(not).

**Unary plus(+):** indicates the positive value of a number (typically redundant but useful for clarity)

Ex: x= -7

y= +x

**Unary minus(-):** used to negate a number

Ex: x=5

y=-x

1. **Bitwise operators**: the bitwise operators are three types and , or, not operators

**And ( &):** if both the bits at the same place in two operands are 1, then 1 is copied to the result, otherwise , o is copied.

**Or (|):** the resulting bit will be o if both the bits are zero, otherwise, the resulting bit will be1

**^(xor):** the resulting bit will be 1 if both the bits are different otherwise, the resulting bit will be 0

**Negation (~):** it calculate the negation of each bit of the operand ~ nis – (n-11)

**<< (left shift):** a<<b means b bits will shift to left from a result is a \* 2 b

**>> (right shift):** a>>b means b bits will shift to right from a result is a// 2 b

1. **Identity operators:** identity operators are sued to compare the objects, they are actually the same object with the same memory location

Ex: l=[2,3,4,8,7,9]

1. **Membership operators:**

In --- True if the values is found in the sequence

not in – True if value is not found in the sequence.

L=[2,3,4,5,6,7,8]

print(2 in l)

print(14 in l)