Types of Operator

Python language supports the following types of operators

1. Arithmetic Operators
2. Comparison (Relational) Operators
3. Assignment Operators
4. Logical Operators
5. Bitwise Operators
6. Membership Operators
7. Identity Operators

1.Arithmetic Operator

Operator	Meaning	Example
+	Add two operands or unary plus	x + y+ 2
-	Subtract right operand from the left or unary minus	x - y- 2
*	Multiply two operands	x * y
/	Divide left operand by the right one (always results into float)	x / y
%	Modulus - remainder of the division of left operand by the right	x % y (remainder of x/y)
//	Floor division - division that results into whole number adjusted to the left in the number line	x // y
**	Exponent - left operand raised to the power of right	x**y (x to the power y)

Example

```
x = 15 y = 4
print('x + y =',x+y)
print('x - y =',x-y)
print('x * y =',x*y)
print('x / y =',x/y)
print('x // y =',x//y)
print('x ** y =',x**y)
```

Output

```
x + y = 19
x - y = 11
x * y = 60
 x / y = 3.75
 x // y = 3
 x ** y = 50625
```

2.Comparison operators

Comparison operators are used to compare values. It returns either True or False according to the condition.

Operator	Meaning	Example
>	Greater than - True if left operand is greater than the right	x > y
<	Less than - True if left operand is less than the right	x < y
==	Equal to - True if both operands are equal	x == y
!=	Not equal to - True if operands are not equal	x != y
>=	Greater than or equal to - True if left operand is greater than or equal to the right	x >= y
<=	Less than or equal to - True if left operand is less than or equal to the right	x <= y

Example

```
x = 10 y = 12
print('x > y is',x>y)
print('x < y is',x<y)
print('x == y is',x==y)
print('x != y is',x!=y)
print('x >= y is',x>=y)
print('x <= y is',x<=y)
```

Output

```
x > y is False
x < y is True
 x == y is False
 x != y is True
 x >= y is False
x <= y is True
```

3.Logical Operators

Logical operators are the and, or, not operators.

Operator	Meaning	Example
and	True if both the operands are true	x and y
or	True if either of the operands is true	x or y
not	True if operand is false (complements the operand)	not x

Example

```
a=5>4 and 3>2
print a
b=5>4 or 3<2
print b
c=not(5>4)
print c
```

Output

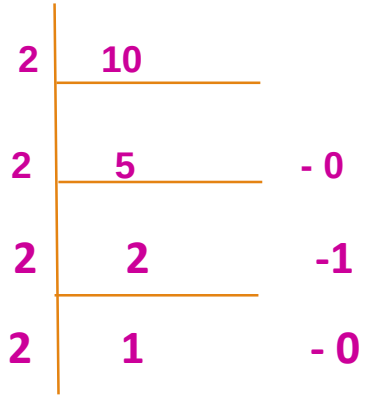
```
True
True
False
```

4.Bitwise Operators

Bitwise operator works on bits and performs bit by bit operation.

Let X= 10 (0000 1010 in binary) and Y = 4 (0000 0100 in binary)

Decimal	Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1011
10	1010



Bitwise Operators

Operator	Name	Description
&	AND	Sets each bit to 1 if both bits are 1
|	OR	Sets each bit to 1 if one of two bits is 1
^	XOR	Sets each bit to 1 if only one of two bits is 1
~	NOT	Inverts all the bits
<<	Zero fill left shift	Shift left by pushing zeros in from the right and let the leftmost bits fall off
>>	Signed right shift	Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off

Operator	Meaning	Example
&	Bitwise AND	x & y = 0 (0000 0000)
|	Bitwise OR	x | y = 14 (0000 1110)
~	Bitwise NOT	~x = -11 (1111 0101)
^	Bitwise XOR	x ^ y = 14 (0000 1110)
>>	Bitwise right shift	x >> 2 = 2 (0000 0010)
<<	Bitwise left shift	x << 2 = 40 (0010 1000)

X=10

ie X=0000 1010

Shift right

Shift 1:0000 0101 (add zero in front)

Shift 2:0000 0010

Shift Left

Shift 1:0001 0100(add zero in rear)

Shift 2:0010 1000

5.Assignment Operators

Assignment operators are used to assign values to variables:

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

6. Identity Operators

Identity operators are used to compare the objects, not if they are equal, but if they are actually the same object, with the same memory location:

Operator	Description	Example
is	Returns True if both variables are the same object	x is y
is not	Returns True if both variables are not the same object	x is not y

Example

```
x = ["apple", "banana"]

y = ["apple", "banana"]

z = x

print(x is z)          # returns True because z is the same object as x

print(x is y)          # returns False because x is not the same object as y, even if they have
the same content

print(x == y)          # to demonstrate the difference betweeen "is" and "==": this
comparison returns True because x is equal to y
```

Output

True

False

True

7.Membership Operators

Operator	Description	Example
in	Returns True if a sequence with the specified value is present in the object	x in y
not in	Returns True if a sequence with the specified value is not present in the object	x not in y

Example

```
x = ["john", "rock"]
print("rock" in x)


y = [“milk", “fruit"]
print("pineapple" not in y)
```

Output

True
True

%s - String (or any object with a string representation, like numbers)

%d – Integers

%f - Floating point numbers

%.<number of digits>f - Floating point numbers with a fixed amount of digits to the right of the dot.

String Formatting

```
>>> name="john"
>>> age=23
>>> print("%s is %d years old." % (name, age))
john is 23 years old.
>>> mylist = [1,2,3]
>>> print("A list: %s" % mylist)
A list: [1, 2, 3]
```

String Length

To get the length of a string, use the `len()` function.

Example

The `len()` function returns the length of a string:

```
a = "Hello, World!"
print(len(a))
```

Strip Method:

The `strip()` method removes any whitespace from the beginning or the end:

```
a = " Hello, World! "
print(a.strip()) # returns "Hello, World!"
```

Lower Method:

The `lower()` method returns the string in lower case:

```
a = "Hello, World!"
print(a.lower())
```

Upper method:

The upper() method returns the string in upper case:

```
a = "Hello, World!"
print(a.upper())
```

Replace method:

The replace() method replaces a string with another string:

```
a = "Hello, World!"
print(a.replace("H", "J"))
```

Split Method:

The split() method splits the string into substrings if it finds instances of the separator:

```
a = "Hello, World!"
print(a.split(",")) # returns ['Hello', ' World!']
```

String check:

```
txt = “stay home stay safe"
x = “ome" in txt
print(x)
```

txt = “stay home stay safe"
x = “ome" not in txt
print(x)

String Concatenation

To concatenate, or combine, two strings you can use the + operator.

Example

Merge variable `a` with variable `b` into variable `c`:

```
a = "Hello"
b = "World"
c = a + b
print(c)
```

To add a space between them, add a `" "`:

```
a = "Hello"
b = "World"
c = a + " " + b
print(c)
```

The format() method takes the passed arguments, formats them, and places them in the string where the placeholders {} are

Use the format() method to insert numbers into strings:

Example:1

```
age = 36
txt = "My name is John, and I am {}"
print(txt.format(age))
```

Output:

My name is John, and I am 36

Example:2

```
quantity = 3
itemno = 567
price = 49.95
myorder = "I want {} pieces of item {} for {} dollars."
print(myorder.format(quantity, itemno, price))
```

Output:

I want 3 pieces of item 567 for 49.95 dollars.

You can use index numbers {0} to be sure the arguments are placed in the correct placeholders:

Example

```
quantity = 3
itemno = 567
price = 49.95
myorder = "I want to pay {2} dollars for {0} pieces of item {1}."
print(myorder.format(quantity, itemno, price))
```

Output:

```
I want to pay 49.95 dollars for 3 pieces of item 567
```

Python has a set of built-in methods that you can use on strings.
Note: All string methods returns new values. They do not change the original string.

Method	Description
capitalize()	Converts the first character to upper case
casefold()	Converts string into lower case
center()	Returns a centered string
count()	Returns the number of times a specified value occurs in a string
encode()	Returns an encoded version of the string
endswith()	Returns true if the string ends with the specified value
expandtabs()	Sets the tab size of the string
find()	Searches the string for a specified value and returns the position of where it was found
format()	Formats specified values in a string
format_map()	Formats specified values in a string

isalnum()	Returns True if all characters in the string are alphanumeric
isalpha()	Returns True if all characters in the string are in the alphabet
isdecimal()	Returns True if all characters in the string are decimals
isdigit()	Returns True if all characters in the string are digits
isidentifier()	Returns True if the string is an identifier
islower()	Returns True if all characters in the string are lower case
isnumeric()	Returns True if all characters in the string are numeric
isprintable()	Returns True if all characters in the string are printable
isspace()	Returns True if all characters in the string are whitespaces

rsplit()	Splits the string at the specified separator, and returns a list
rstrip()	Returns a right trim version of the string
split()	Splits the string at the specified separator, and returns a list
splitlines()	Splits the string at line breaks and returns a list
startswith()	Returns true if the string starts with the specified value
strip()	Returns a trimmed version of the string
swapcase()	Swaps cases, lower case becomes upper case and vice versa
title()	Converts the first character of each word to upper case
translate()	Returns a translated string
upper()	Converts a string into upper case
zfill()	Fills the string with a specified number of 0 values at the beginning

THANK YOU