Operators and Expressions

Operators in Python:-

Operators are special symbols in Python, that are used to perform operations on variables and values.

The value that the operator, operates on is called the operand.

Example:

```
>>> 2+3
5
```

→ Here, [+] is the operator, that performs addition.

→ [2] and [3] are the operands.

→ [5] is the output of the operation.

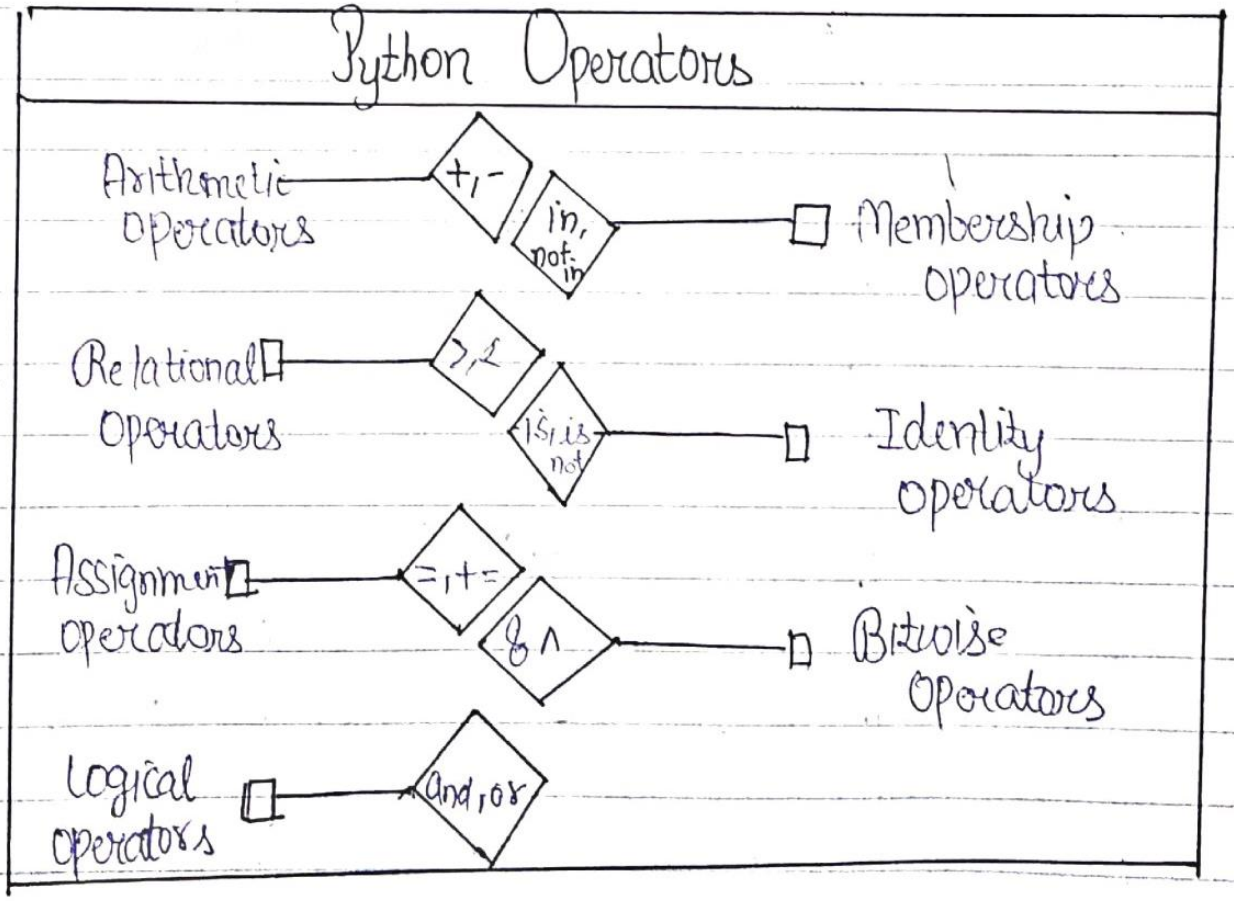
* (A sequence of operands and operators, like a+b-5, is called an expression.)

Types of operators

Python divides the operators in the following groups:

- Arithmetic operators
- Relational Operators
- Logical Operators
- Assignment Operators
- Bitwise Operators
- Membership Operators
- Identity Operators

} special operators



1. Arithmetic operators :-

Arithmetic operators are used to perform mathematical operations like addition, subtraction, multiplication and division.

Operator	Description	Syntax
+	Addition : adds two operands	a+b
-	Subtracts : subtracts two operands	a-b
*	Multiplication : multiplies two operands	a*b
/	Division (float) : divides the first operand by Second	a/b
%	Modulus : returns the remainder when first operand is divided by second	a%b
//	Floor Division : (also called integer division)	a//b
**	Power : Returns first raised to Power Second	a**b

Example of Arithmetic Operator

```
a = 15
b = 4
# Arithmetic operations
add = a+b          # Addition of numbers
sub = a-b          # Subtraction of numbers
mul = a * b        # Multiplication of number
div = a / b        # Division(float) of number
div2 = a // b      # Division (floor) of number
mod = a % b        # Modulo of both number
p = a ** b         # power

# print output
print(add)
print (sub)
print (mul)
print (div)
print (div2)
print (mod)
print (p)
```

Output
19
11
60
3.75
3
3
50625

2. Relational Operators :

Relational operators compares the value. It either returns True or False according to the condition.

Operator	Description	Syntax
>	Greater than :→ True if left operand is greater than right	a>b
<	less than :- True if left operand is less than right	a<b
==	Equal to :- True if both operands are equal.	a==b
!=	Not equal to :- True if both operands are not equal	a!=b
>=	Greater than or equal to :- True if left operand is greater than or equal to the right	a>=b
<=	less than or equal to :- True if left operand is less than or equal to the right	a<=b

Example:-
Comparison operators in Python

```
a=10
b=12

print (a>b)      # a>b is False
print (a<b)      # a<b is True
print (a==b)     # a==b is False
print (a!=b)     # a!=b is True
print (a>=b)     # a>=b is False
print (a<=b)     # a<=b is True
```

Output:-

```
False
True
False
True
False
True
```

(3) Logical Operators :-

Logical operators perform Logical AND, Logical OR and Logical NOT operations.

Operator	Description	Syntax
and	Logical AND : True if both operands are True.	a and b
or	Logical OR :- True if either of the operands is True.	a or b
not	Logical NOT :- True if operand is false.	not a

Examples of Logical Operators.

```
a = True
b = False

print(a and b)
print(a or b)
print(not a)
```

Output

```
False
True
False
```

4 Assignment Operators.

Assignment Operators are used in Python to assign values to variables. (a = 5)

Operator	Description	Syntax
=	Assign value of right side of expression to left side of operand.	c = a+b
+=	Add AND: Add right side operand with left side operand and then assign to left operand.	a += b a = a+b
-=	Subtract AND: subtract right operand from left operand then assign to left operand	a -= b a = a-b
*=	Multiply AND: multiply right operand with left operand and then assign to left operand	a *= b a = a*b
/=	Divide AND: Divide left operand with right operand and then assign to left operand	a /= b a = a/b

Operator	Description	Syntax
%=	Modulus AND: Takes modulus using left and right operands and assign result to left operand.	a%=b a = a%b
//=	Divide(floor) AND: Divide left operand with right operand and then assign the value (floor) to left operand	a//=b a = a//b
**=	exponential (Power): calculation on operators and assign value to left operand	a**=b
&=	Bitwise AND: Performe Bitwise AND on operands and assign value to left operand	a&=b a = a&b
|=	Performs Bitwise OR on operands and assign value to left operand.	a|=b a = a|b
^=	XOR: on operands and assign value to left operand	a^=b a = a^b

>>= Bitwise right Shift a = a >> b

<< = Bitwise Left Shift a = a << b

Some Examples:(i)
```
a = 10
b = 20
print(a + b)
```

(ii)
```
m = 15
m+ = 3
print(m)
```

(iii)
```
n = 4
n- = 3
print(n)
```

(IV)
```
z = 4
z* = 1
print(z)
```

Output:- 30
18
1
4

5 Bitwise Operators :-

Bitwise operators acts on bits and performs bit by bit operation.

It is used to compare (binary) number.

Operator	Description	Syntax
&	Bitwise AND	a & b
|	Bitwise OR	a | b
~	Bitwise NOT	~a
^	Bitwise XOR	a ^ b
>>	Bitwise right shift	a >>
<<	Bitwise left shift	a <<

#Examples of Bitwise operators

```
a = 10
b = 4
#bitwise AND    print(a & b)
#bitwise OR     print(a | b)
#bitwise NOT    print(~a)
#bitwise XOR    print(a ^ b)
#bitwise right  print(a >> 2)
#bitwise left   print(a << 2)
  shift
```

Output

```
0
14
-11
14
2
40
```

6 Special Operators

Python language offers some special types of operators like the identity operator or membership operator

- Identity Operators
- Membership Operators

Identity Operators :-

→ is and is not are the identity operators in Python

→ They are used to check if two values are located on the same part of the memory.

→ Two variables (values) that are equal does not imply that they are identical.

Operator	Description	Example
is	True if the operands are identical	a is True
is not	True if the operands are not identical.	a is not True

Example

```
a1 = 5
b1 = 5
a2 = 'jpwebdevelopers'
b2 = 'jpwebdevelopers'
a3 = [1,2,3]
b3 = [1,2,3]
print (a1 is b1)
print (a2 is not b2)
print (a3 is b3)
```

Output

```
True
false
false
```

Membership Operators

→ in and not in are the membership operators in Python.

→ They are used to test whether a value or variable is found in a sequence (string, list, tuple, set and dictionary)

Operator	Meaning (Description)	Example
in	True if value is found in the sequence.	5 in a
not in	True if value is not found in the sequence.	5 not in a

Examples of Membership operator

```
a = "jpwebdevelopers"
print('p' in a)
```

Output True

```
print('w' in not a)
```

Output False