Operators

Operators are symbols that perform operations on variables and values. Python has several types of operators for different tasks like arithmetic, comparison, logical operations, and more.

Types of Operators are --

1. Arithmetic operators: +, -, *, / etc.
2. Assignment operators: =, +=, -= etc.
3. Comparison operators: ==, >, >=, <, != etc.
4. Logical operators: and, or, not.
5. bitwise operater -- & | ~ >> << ^
6. identity operater -- is, is not,
7. Menbership oprater -- in , not in

```
===========================================================
```

1. Arithmetic operators

Arithmetic operators perform mathematical operations like addition, subtraction, multiplication, division, etc.

There are 7 types of arithmetic operators.

1. addition (+)
2. subtraction (-)
3. multiplication (*)
4. division (/)
5. Floor division (//)
6. modulus (%)
7. Exponentiation (**)

Eg -

a = 12

b=8

print(a+b)

Output - 20

Q. Take two inputs from the user and perform all arithmetic operators and print the outputs

In [1]:

```
num1=eval(input("Enter the first number :-"))
num2=eval(input("Enter the second number :-"))

sum = num1+num2
```

```
sub= num1-num2
mul= num1*num2
div= num1/num2
module= num1 % num2
flote= num1 // num2
expon= num1 ** num2
print("Addition is:-", sum, "\nsubtraction is:-", sub,  "\nmultiple is:-",mul,
      "\nflote is:-", flote,  "\nexpon is:-",expon)
```

```
Addition is:- 4
subtraction is:- 0
multiple is:- 4
divisionn is:- 1.0
modules is:- 0
flote is:- 1
expon is:- 4
```

```
================================================================================
```

2. Assignment operator

Assignment operators are used to assign values to variables. Python also provides compound assignment operators that perform operations like addition, subtraction, multiplication, etc.

A basic assignment operator is simple =.

The assignment operator combines arithmetic operations with assignment.

Using compound assignment operators, the reassignment works better.

1. Add and assign -- (+=)
2. Subtract and assign -- (-=)
3. Multiply and assign -- (*=)
4. Divide and assign -- (/=)
5. Floor divide and assign -- (//=)
6. modulus and assign -- (%=)
7. Exponentiation and assign -- (**=)

```
================================================================================
```

3.Comparison operator

Comparison operators, also called relational operators, are used to compare two values.

Comparison operators will always provide a Boolean result that is True and False.

--> comparison operators are as follows --

1. Equal to -- (==)

2. Not Equal to -- (!=)

3. Greater than -- (>)

4. Less than -- (<)

5. Greater than or equal to -- (>=)

6. Less than or equal to -- (<=)

Comparison operators will work with numbers, but you can use them with strings as well.

Strings will be comparing the ASCII values of the string.

In [9]:

```
### take percent from user and print the grade
percent=eval(input("Enter your percent :-"))
if percent>=85  and percent<=100:
    print("Distinction")
elif percent>=60  and percent<=84:
    print("First Class")
elif percent>=50 and percent< 60 :
    print("Second Class")
elif percent>=35 and percent<50:
    print("Third Class")
elif percent<35:
    print("Fail")
else:
    print("Invalid percent")
```

```
Invalid percent
```

```
================================================================
```

4. Logical operators

Logical operators in Python are used to combine multiple conditions and return a Boolean result (True or False).

There are 3 types of logical operators.

And

Return True if both conditions are True

Or--

Return True if at least one condition is True.

Not --

Reverse the boolean value.

```
================================================================================
```

5.Identity operator

is, is not

In [4]:

```
a=2
b=2
c=a
print(a is b); # true
print(a is c); # true
print(a is not b); # false
print(a is not c); # false
```

```
True
True
False
False
```

```
================================================================================
```

6. membership operator

IN , Not IN

In [6]:

```
a = 10;
b = 20;
c = a;
# print(a in c); # error
# print(a not in c); # error
# print(a in b); # error
# print(a not in b); # error

print("a" in "apple"); # true
print("b" in "apple"); # false
print("a" not in "apple"); # false
print("b" not in "apple"); # true
```

```
True
False
False
True
```

```
================================================================================
```

In [ ]: