==========

Operators and Expressions in Python

==========

=>An Operator is a symbol which is used to perform certation on data / objects.

=>If two or more Variables / Objects connected with Operators then it is called Expression.

=>In Python Programming, we have 7 operators. They are

1) Arithmetic Operators
2) Assignment Operator
3) Relational Operators
4) Logical Operators
5) Bitwise Operators (Most Imp)
6) Membership Operators
   a) in
   b) not in
7) Identity Operators
   a) is
   b) is not

Note: Python does not support Unary Operator ++ and --
Python does not support Ternary Operator (? :)

Note: Python Programming its own ternary Operator (if else operator)

==========

Arithmetic Operators

==========

=>These Operators are used for performing Arithmetic Operations such as addition, subtraction, multiplication etc.

=>If any Arithmetic Operators connected with two or more Variables then it is called Arithmetic Expression.

=>The following Tables gives list of Arithmetic Operators

==========

Assignment Operator

==========

=>The Symbol of Assignment Operator is =

=>The purpose of Assignment Operator is that that To transfer Right Hand Side Value / Expression to Left Hand Side variable. **=>**We can use Assignment Operator in two ways. They are

a) Single Line Assignment Operator
b) Multi Line Assignment Operator

---

a) Single Line Assignment Operator:

=>Syntax:- VarName=Value

(OR)

VarName1=VarName2

(OR)

Varname=Expression

=>This Single Line Assignment Operator, we can transfer Single Right Hand Side Value / Expression to Left Hand Side variable.

Examples:

```
>>> a=10
>>> b=20
>>> c=a+b
>>> print(a,b,c)---------10 20 30
```

```
-----------------------------------------
>>> sno=100
>>> sname="Rossum"
>>> print(sno,sname)--------100   Rossum
-------------------------------------------------------------------------
```

b) Multi Line Assignment Operator:

```
-------------------------------------------------------------------------
```

=>With Multi Line Assignment Operator, we can transfer Multiple Right Hand Side Values / Expressions to Left Hand Side variables.

=>Syntax:- Var1,Var2...Var-n=Val1,Val2....Val-n

(OR)

Var1,Var2...Var-n=Expr-1,Expr-2...Expr-n

Examples:

```
------------------
>>> a,b=10,20
>>> add,sub,mul=a+b,a-b,a*b
>>> print(a,b)---------10 20
>>> print(add,sub,mul)----------30 -10 200
-----------------------------------------
>>> sno,sname,marks=10,"Rossum",44.44
>>> print(sno,sname,marks)-------------------------10 Rossum 44.44
Note:-
>>> a=10,20,30,40,50,60
>>> print(a,type(a))---------------(10, 20, 30, 40, 50, 60) <class 'tuple'>
```

Relational Operators

=>The porpose of Relational Operators is that " To Compare two Values "

=>The two or more Variables connected with Relational Operators is called Relational Expression.

=>The Result of Relational Expression always gives either True or False (bool value)

=>The following table gives list of Relational Operators along with meaning and examples.

Note: We can't apply Relational Operators(>,<,>= , <=) between complex values

Examples:

```
--------------------
>>>  print((2+3j)>=(2.0+3.0j))-----TypeError:  '>='  not  supported  between instances of 'complex' and 'complex'

-------------------------------------------------------------------------
>>> a=2+3j
>>> b=2+3j
>>> print(a==b)---------------True
>>> print(a>b)----TypeError: '>' not supported between instances of 'complex' and 'complex'
>>> print(a!=b)-----False
>>>  print(a>=b)-----TypeError:  '>='  not  supported  between  instances  of 'complex' and 'complex'
```

==========

Logical Operators

==========

=>The purpose of Logical Operators is that b" To combile two or more Relational Expressions and compare two or more values:"
=>If two or more Relational Expressions are connected with Logical Operators then it is called Logical Expression or Logical Condition (Compund Condition).
=>The result of Logical Expression is either to be True or False.
=>In Python Programming, we have 3 types of Logical Operatoors and they are given in the following table.

==========

Bitwise Operators (Most Imp)

==========

=>Bitwise operators are applicable on Integer Data Only but not on floating point values bcoz floating values does not have certainity its value. **=>**The internal flow of Bitwise Operators is that " First They converts Integer Data into Binary Format and Process the binary data on the basis of BIT by BIT and finally gives the result in the form of Decimal Number System."
=>In Python Programming, we have the following Bitwise Operators.

1. Bitwise Left Shift Operators (<<)
2. Biwise Right Shift Operator (>>)
3. Bitwise AND Operator (&)
4. Bitwise OR Operator (|)
5. Bitwise Complement Operator (~)
6. Bitwise XOR Operator (^)

==========

6) Membership Operators

==========

=>The purpose of Membership Operators in python is that "To check the existence of Value in Iterable Object".
=>An Iterable Object is one which contains multiple elements(Sequence, List, set and dict).
=>In Python Programming, we have 2 types of Membership Operators. They are

1. in
2. not in

---

1) in:

Syntax:- value in Iterable_object
=>The 'in' operator returns True provides "value" present Iterable_object
=>The 'in' operator returns False provides "value" not present Iterable_object

---

2) not in:

Syntax:- value not in Iterable_object
=>The 'not in' operator returns True provides "value" not present Iterable_object
=>The 'not in' operator returns False provides "value" present Iterable_object

==========

7) Identity Operators (Python Command Prompt)

==========

=>The Purpose of Identity Operators is that "To compare the memory address of two objects ".

=>In Python Programming, we have type of Identity Operators. They are

1. is
2. is not
3.

---

1) is :

Syntax:- object1 is object2

=>The 'is' operator returns True provided both object1 and object2 contains same address.

=>The is' operator returns False provided both object1 and object2 contains different address.

---

2) is not :

Syntax:- object1 is not object2

=>The 'is not ' operator returns True provided both object1 and object2 contains Different address.

=>The is not ' operator returns False provided both object1 and object2 contains Same address.