OPERATORS IN PYTHON

Operators

Operators can be defined as symbols that are used to perform operations on operands.

OR

These are tokens that trigger some computation/action when applied to variables or other objects.

Operators

Types of Operators

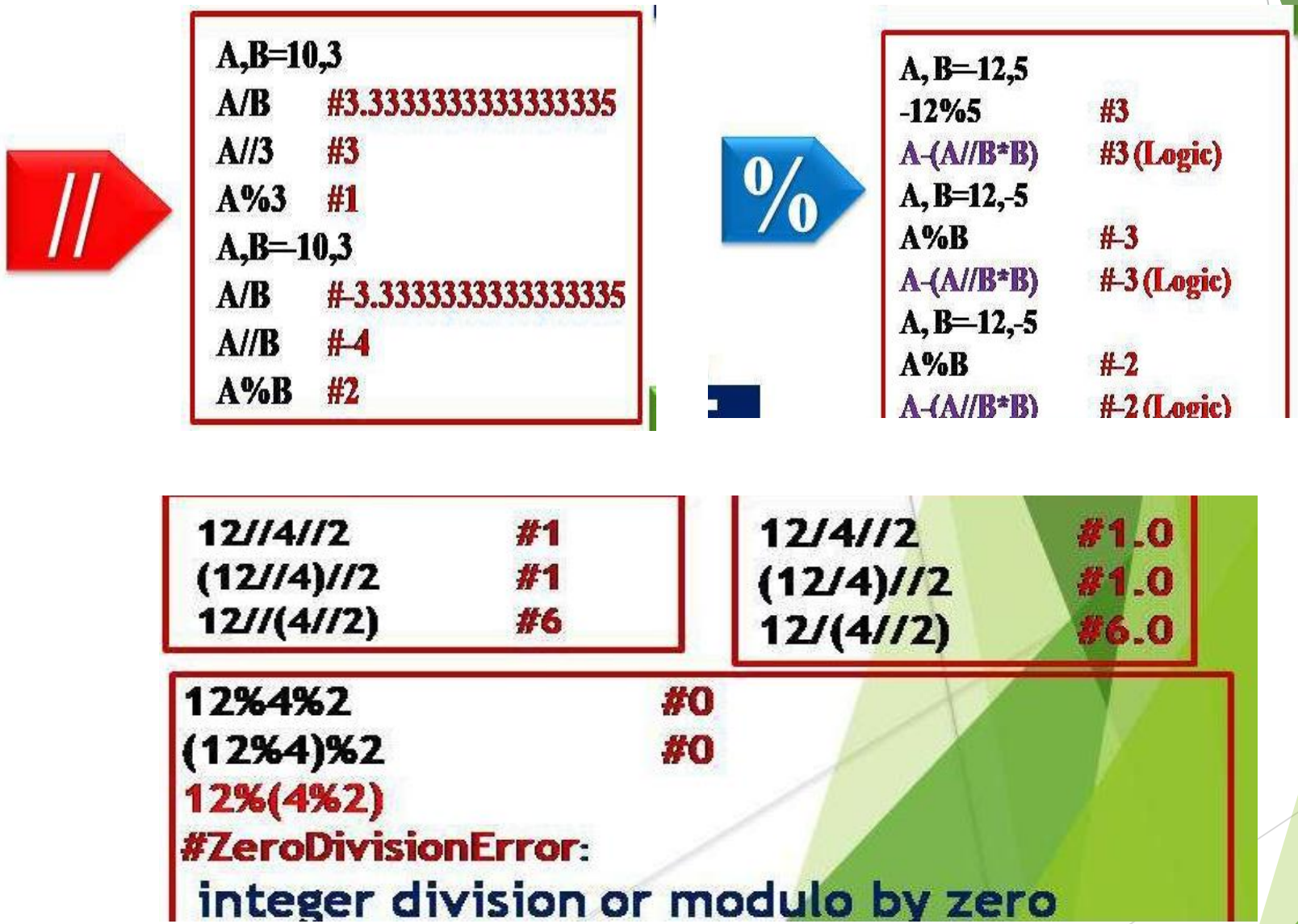
1. Arithmetic Operators. (+,-,*,/,%,**,//)
2. Relational Operators. (>,<,>=,<=,==,!=)
3. Assignment Operators. (=)
4. Logical Operators.(and,or,not)
5. Bitwise Operators (&, |,^)
6. Membership Operators. (in,not in)
7. Identity Operators. (is, is not)
8. Shift Operator. (<<,>>)
9. Arithmetic-assignment Operator. (/=,+=,-=,%=,**=,//=)

Operators continue

1. Arithmetic Operators

Arithmetic Operators are used to perform arithmetic operations like addition, multiplication, division etc.

Operators	Description	Example
+	perform addition of two number	a+b
-	perform subtraction of two number	a-b
/	perform division of two number	a/b
*****	perform multiplication of two number	a*b
%	Modulus = returns remainder	a%b
//	Floor Division = remove digits after the decimal point	a//b
******	Exponent = perform raise to power	a**b


//
A,B=10,3
A/B #3.3333333333333335
A//3 #3
A%3 #1
A,B=-10,3
A/B #-3.3333333333333335
A//B #-4
A%B #2
%
A, B=-12,5
-12%5 #3
A-(A//B*B) #3 (Logic)
A, B=12,-5
A%B #-3
A-(A//B*B) #-3 (Logic)
A, B=-12,-5
A%B #-2
A-(A//B*B) #-2 (Logic)
12//4//2 #1
(12//4)//2 #1
12//(4//2) #6
12/4//2 #1.0
(12/4)//2 #1.0
12/(4//2) #6.0
12%4%2 #0
(12%4)%2 #0
12%(4%2)
#ZeroDivisionError:
integer division or modulo by zero

Operators continue

2. Relational Operators

Relational Operators are used to compare the values.

Operators	Description	Example
==	Equal to, return true if a equals to b	a == b
!=	Not equal, return true if a is not equals to b	a != b
>	Greater than, return true if a is greater than b	a > b
>=	Greater than or equal to , return true if a is greater than b or a is equals to b	a >= b
<	Less than, return true if a is less than b	a < b
<=	Less than or equal to , return true if a is less than b or a is equals to b	a <= b

Expressions and Statement

a. Expression : - which is evaluated and produce result. e.g. (20 + 4) / 4
b. Statement :- instruction that does something.
   - e.g
   - a = 20
   - print("Calling in proper sequence")

Operators continue

3. Assignment Operators
Used to assign values to the variables.

Operators	Description	Example
=	Assigns values from right side operands to left side operand	a=b

Operators continue

4. Logical Operators

Logical Operators are used to perform logical operations on the given two variables or values.

Operators	Description	Example
and	return true if both condition are true	x and y
or	return true if either or both condition are true	x or y
not	reverse the condition	not(a>b)

```
a=30
b=20
if(a==30 and b==20):
   print('hello')
```

Output :-

hello

Operators continue

5. Bitwise Operators

Bitwise operators are used to compare (binary) numbers:

Bitwise AND Operator

A = 10 => 1010 (Binary)
B = 7 => 111 (Binary)
A & B = 1010
&
0111
= 0010
= 2 (Decimal)

Bitwise OR Operator

A = 10 => 1010 (Binary)
B = 7 => 111 (Binary)
A | B = 1010
|
0111
= 1111
= 15 (Decimal)

Bitwise XOR Operator

A = 10 => 1010 (Binary)
B = 7 => 111 (Binary)
A ^ B = 1010
^
0111
= 1101
= 13 (Decimal)

1*1=0
0*1=1
1*0=1

Bitwise Ones' Complement Operator

complement of a number 'A' is equal to -(A+1).

A = 10 => 1010 (Binary)
~A = ~1010
= -(1010+1)
= -(1011)
= -11 (Decimal)

Bitwise Left Shift Operator

A = 10 => 1010 (Binary)
A<<2 = 1010<<2
= 101000
= 40 (Decimal)

Bitwise Right Shift Opera

A = 10 => 1010 (Binary)
A>>2 = 1010>>2
= 10
= 2 (Decimal)

Operators continue

6. Membership Operators

The membership operators in Python are used to validate whether a value is found within a sequence such as such as strings, lists, or tuples.

Operators	Description	Example
in	return true if value exists in the sequence, else false.	a in list
not in	return true if value does not exists in the sequence, else false.	a not in list

E.g.

```
S = 'Python is Fun'
"p" in S          #False
"P" in S          #True
'P' in S          #True
'fun' in S        #False
'Fun' in S        #True
'fun' not in S    # True
```

Operators continue

7. Identity Operators
Identity operators in Python compare the memory locations of two objects.

Operators	Description	Example
is	returns true if two variables point the same object, else false	a is b
is not	returns true if two variables point the different object, else false	a is not b

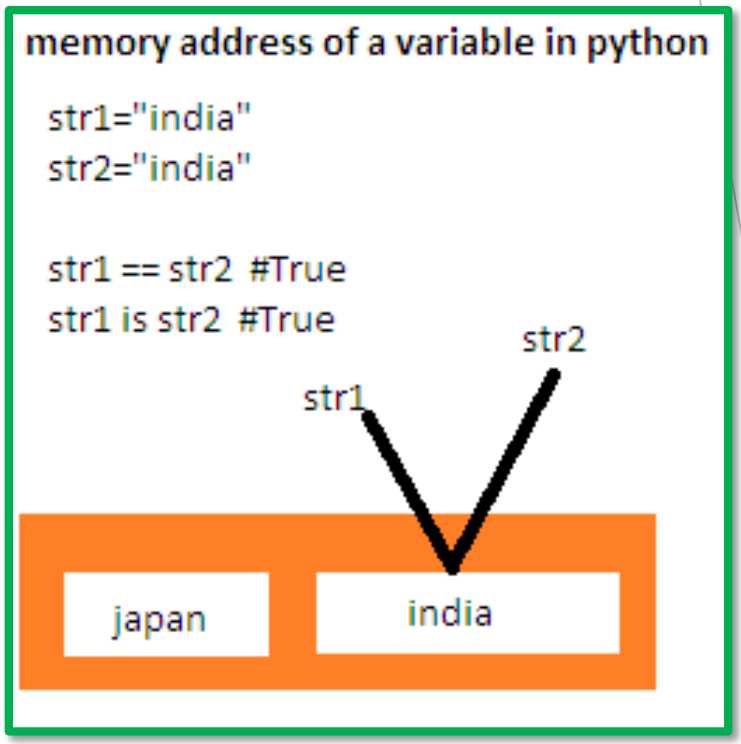
Operators continue

Examples:

```
e.g.
a = 34
b=34
if (a is b):
    print('both a and b has same identity')
else:
    print('a and b has different identity')
b=99
if (a is b):
    print('both a and b has same identity')
else:
    print('a and b has different identity')
```

Output :-

```
both a and b has same identity  a
and b has different identity
```




```
a, b=7,7
a is b   #True
id(a)    #1838247120
id(b)    #1838247120

b=8
a is b   #False
id(b)    #1838247152
```

Operators continue

8. Arithmetic -Assignment Operators
Used to assign values to the variables.

Operators	Description	Example
=	Assigns values from right side operands to left side operand	a=b
+=	Add 2 numbers and assigns the result to left operand.	a+=b
/=	Divides 2 numbers and assigns the result to left operand.	a/=b
*=	Multiply 2 numbers and assigns the result to left operand.	a*=b
-=	Subtracts 2 numbers and assigns the result to left operand.	a-=b
%=	modulus 2 numbers and assigns the result to left operand.	a%=b
//=	Perform floor division on 2 numbers and assigns the result to left operand.	a//=b
**=	calculate power on operators and assigns the result to left operand.	a**=b

PYTHON OPERATOR PRECEDENCE

PEMDAS

Parentheses | Exponentiation | Multiplication | Division | Addition | Subtraction

Operators	Meaning
()	Parentheses
**	Exponent
+x, -x,~x	Unary plus, Unary minus, Bitwise NOT
*,/, //, %	Multiplication, Division, Floor, Division, Modulus
+,-	Addition, Subtraction
<<,>>	Bitwise shift operators
&	Bitwise AND
^	Bitwise XOR
|	Bitwise OR
==,!=,>,>=,<,<=, is, is not, in, not in	Comparisons, Identity, Memebership Operators
not	Logical NOT
and	Logical AND
or	Logical OR

Barebone of a python program

a. Expression : - which is evaluated and produce result. E.g. (20 + 4) / 4
b. Statement :- instruction that does something.

```
e.g
a = 20
print("Calling in proper sequence")
```

c. Comments : which is readable for programmer but ignored by python interpreter

i. Single line comment: Which begins with # sign.
ii. Multi line comment (docstring): either write multiple line beginning with # sign or use triple quoted multiple line. E.g.

```
'''this is my
  first
 python multiline comment
'''
```

d. Function

a code that has some name and it can be reused.e.g. **keyArgFunc** in above program

e. Block & indentation : group of statements is block.indentation at same level create a block.e.g. all 3 statement of **keyArgFunc function**

Example of a python program

```
#function definition
def keyArgFunc(empname, emprole):
        print ("Emp Name: ", empname)
        print ("Emp Role: ", emprole)
        return;
A = 20
print("Calling in proper sequence")
keyArgFunc(empname = "Nick",emprole = "Manager" )
print("Calling in opposite sequence")
keyArgFunc(emprole = "Manager",empname = "Nick")
```

comment

Function

indentation

expression

statements

A python program contain the following components

a. Expression
b. Statement
c. Comments
d. Function
e. Block &n indentation