Operators in Java:

Operators are special symbols in Java that carry out arithmetic or logical computation. The value that the operator operates on is called the operand.

Operators are used to perform specific operations on one more operands (variables) and provides a result

Java supports all three types:

An operator that performs an action on one operand is called a Unary Operator

(**++, - -**)

An operator that performs an action on two operands is called a Binary Operator

(+,-, /,*,and more)

An operator that performs an action on three operands is called ternary operator (**? :**)

Java supports the following operators:

1. **Arithmetic Operators**
2. **Comparison/Relational Operators**
3. **Assignment Operators**
4. **Boolean Logical Operators**
5. **Bitwise Operators**
6. **Unary Operators or increment/decrement operators**
7. **Ternary Operator(Conditional Operator)**
8. **Special Operator**

Arithmetic Operators:

Arithmetic operators are used to perform mathematical operations like addition (+), subtraction (-), multiplication (*), dividing (/), remainder (%) etc.

Java does not support operator overloading, but there are certain operators which are overloaded by java itself like '+' operator which behaves differently when applied to different operands.

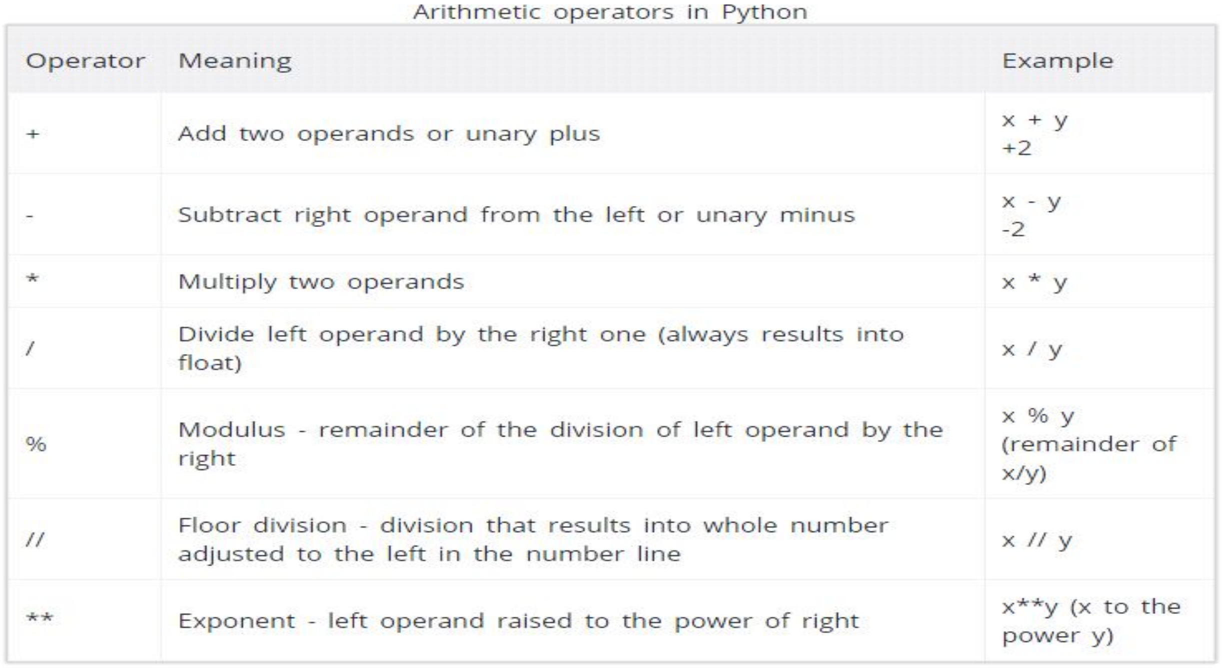
Ex: If + operator is used and one of the operand is a string, then the other operand is converted to a String automatically and concatenated.

Ex: System.out.println statement when the string in the quotes is concatenated with the values of the result or individual primitives

In addition to these operators arithmetic compound assignment operators are also provided by java : +=, -=, /=, *=, %=

a += b;	// evaluated as a=a+b;	a -=b;	//evaluated as a=a-b;
a *=b;	//evaluated as a=a*b;	a /=b;	//evaluated as a=a/b;
a %=b;	//evaluated as a=a%b;		

Arithmetic operators in Python



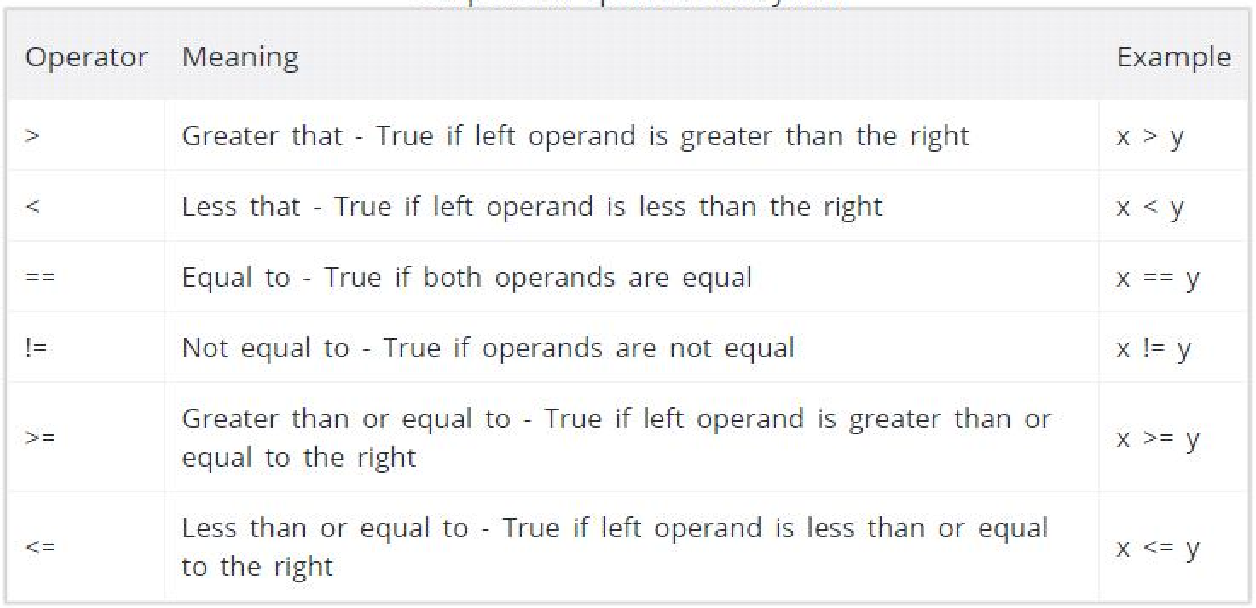
Operator	Meaning	Example
+	Add two operands or unary plus	x + y +2
-	Subtract right operand from the left or unary minus	x - y -2
*	Multiply two operands	x * y
/	Divide left operand by the right one (always results into float)	x / y
%	Modulus - remainder of the division of left operand by the right	x % 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)

Relational/Comparison Operators:

Relational operators in java return either true or false as a Boolean type.

Relational operators in C++ returns an integer where the integer value of zero may be interpreted as false and any non-zero value may be interpreted as true .

Comparision operators in Python



Operator	Meaning	Example
>	Greater that - True if left operand is greater than the right	x > y
<	Less that - 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

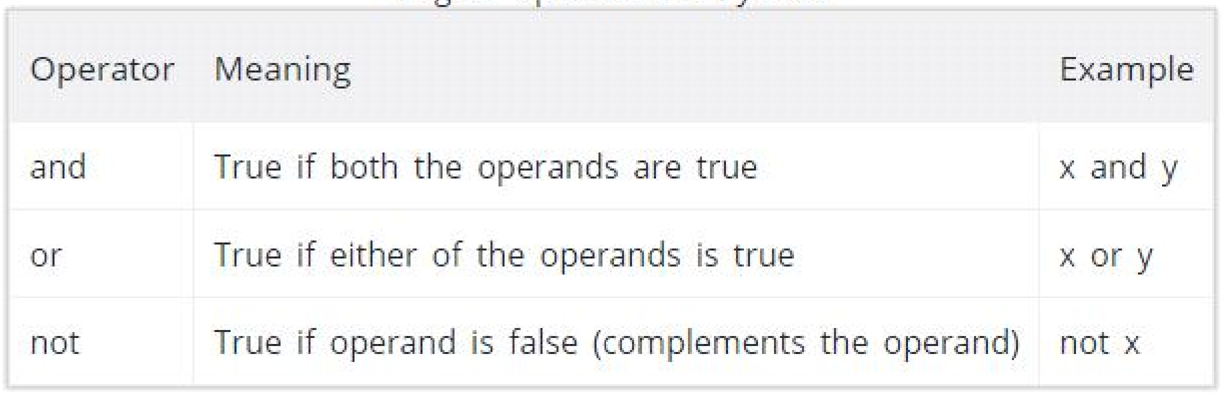
Boolean Logical Operators

- **Logical AND (&&)** operator is used to simultaneously evaluate two conditions or expressions with relational operators. If expressions on both the sides (left and right side) of the logical operator are true, then the whole expression is true. For example, If we have an expression (a>b) && (b>c), then the whole expression is true only if both expressions are true. That is, if b is greater than a and c.
- **Logical OR (||)** operator is used to simultaneously evaluate two conditions or expressions with relational operators. If one or both the expressions of the logical operator is true, then the whole expression is true. For example, If we have an expression (a>b) || (b>c), then the whole expression is true if either b is greater than a or b is greater than c.

- **Logical not (!)** operator takes a single expression and negates the value of the expression. Logical NOT produces a zero if the expression evaluates to a non-zero value and produces a 1 if the expression produces a zero. In other words, it just reverses the value of the expression. For example, a = 10, b b = !a; Now, the value of b = 0. The value of a is not zero, therefore, !a = 0. The value of !a is assigned to b, hence, the result.

Note:
Logical operators are the and, or, not operators. Based on Boolean Algebra. Returns results as either True or False

Logical operators in Python



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

Assignment operators :

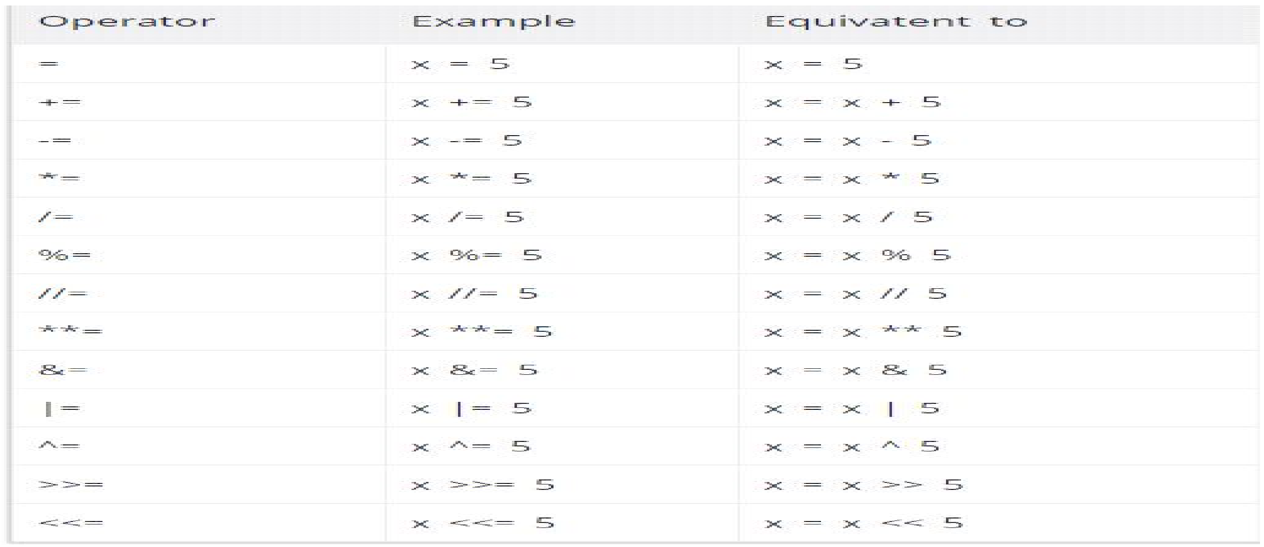
It sets the value of a variable(or expression) to some new value.

The simple '=' operator sets the left-hand operand to the value of the right-hand operand.

The assignment operator has right to left associativity.

Statement a=b=0;would assign 0 to b then b to a. Java supports the following list of shortcut or compound assignment operators: =, +=,-=, *=, /=, %=,//=, **=, &=, |=,^=, <<=, >>=

These operators allow you to combine two operators into one: one fixed as assignment plus another one



Operator	Example	Equivatent to
=	x = 5	x = 5
+=	x += 5	x = x + 5
-=	x -= 5	x = x - 5
*=	x *= 5	x = x * 5
/=	x /= 5	x = x / 5
%=	x %= 5	x = x % 5
//=	x //= 5	x = x // 5
**=	x **= 5	x = x ** 5
&=	x &= 5	x = x & 5
|=	x |= 5	x = x | 5
^=	x ^= 5	x = x ^ 5
>>=	x >>= 5	x = x >> 5
<<=	x <<= 5	x = x << 5

Bitwise Operators

Bitwise operators include and, or, xor, not, right shift ,left shift and unsigned right shift.

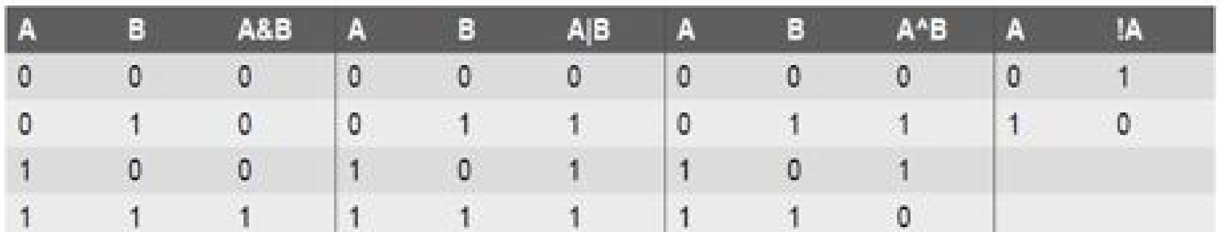
In java bitwise operators operate on int and long values.

If any of the operand is shorter than an int, it is automatically promoted to int before the operations are performed

- It performs bit by bit operation on bits

Operator	Operation Performed	Description
&	**a&b**	**1 if both bits are 1**
|	**a|b**	**1 if either of the bits is 1**
^	**a^b**	**1 if both bits are different**
~	**~a**	**Complement the bits**
<<	**a<<b**	**Shift the bits left by b positions. Zero bits are added from the LSB side. Bits are discarded from the MSB side**
>>	**a>>b**	**Shift the bits right by b positions. Sign bits are copied from the MSB side. Bits are discarded from the LSB side**
>>>	**a>>>b**	**Shift the bits right by b positions. Zero bits are added from the MSB side. Bits are discarded from the LSB side**

- The truth tables of these bitwise operators are given below.



A	B	A&B	A	B	A|B	A	B	A^B	A	!A
0	0	0	0	0	0	0	0	0	0	1
0	1	0	0	1	1	0	1	1	1	0
1	0	0	1	0	1	1	0	1		
1	1	1	1	1	1	1	1	0		

Note: Integer values are represented in Binary.

Ex: Decimal number 4 is represented as100 in binary and 5is represented as 101. Negative integers are always represented in 2's complement form.

Unary Operators or Increment/Decrement Operators:

Unary Operators as the name suggests, are applied to only one operand.

They are as follows ++ , --

Increment and Decrement Operators:

Increment and Decrement operators can be applied to all integers and floating-point types. They can be used either in prefix(++x,--x) or postfix(x++,x--) mode.

Prefix Increment/Decrement Operation

int x=2;

int y=++x;// x=3,y=3

int z=--x;// x=1,z=1

Postfix Increment/Decrement Operation

int x=2;

int y=x++;// x=3,y=2

int z=x--;// x=1,z=2

Ternary Operator

Ternary operators are applied to three operands. This conditional operator(?:) decides on the basis of the first expression, which of the two expressions to be evaluated

Operand1 ? Operand2 : Operan3

Operand 1 must be of Boolean type or an expression producing a Boolean result.

If operand1 is true, then operand2 is returned.

If operand1 is false, then operand3 is returned.

This operator is similar to an if conditioned statement

For Example:

If the value of x is less than y, ”Y is greater” string is returned and stored in the variable: greater, else “X” is greater is returned and stored in the variable: greater.

Ex: finding larger number from given two numbers(int a,b).

int max=a>b?a:b.

Special Operator:

The java **instanceof** operator is used to test whether the object is an instance of the specified type (class or subclass or interface).

The instanceof in java is also known as type comparison operator because it compares the instance with type. It returns either true or false. If we apply the instanceof operator with any variable that has null value, it returns false.

Ex: 1

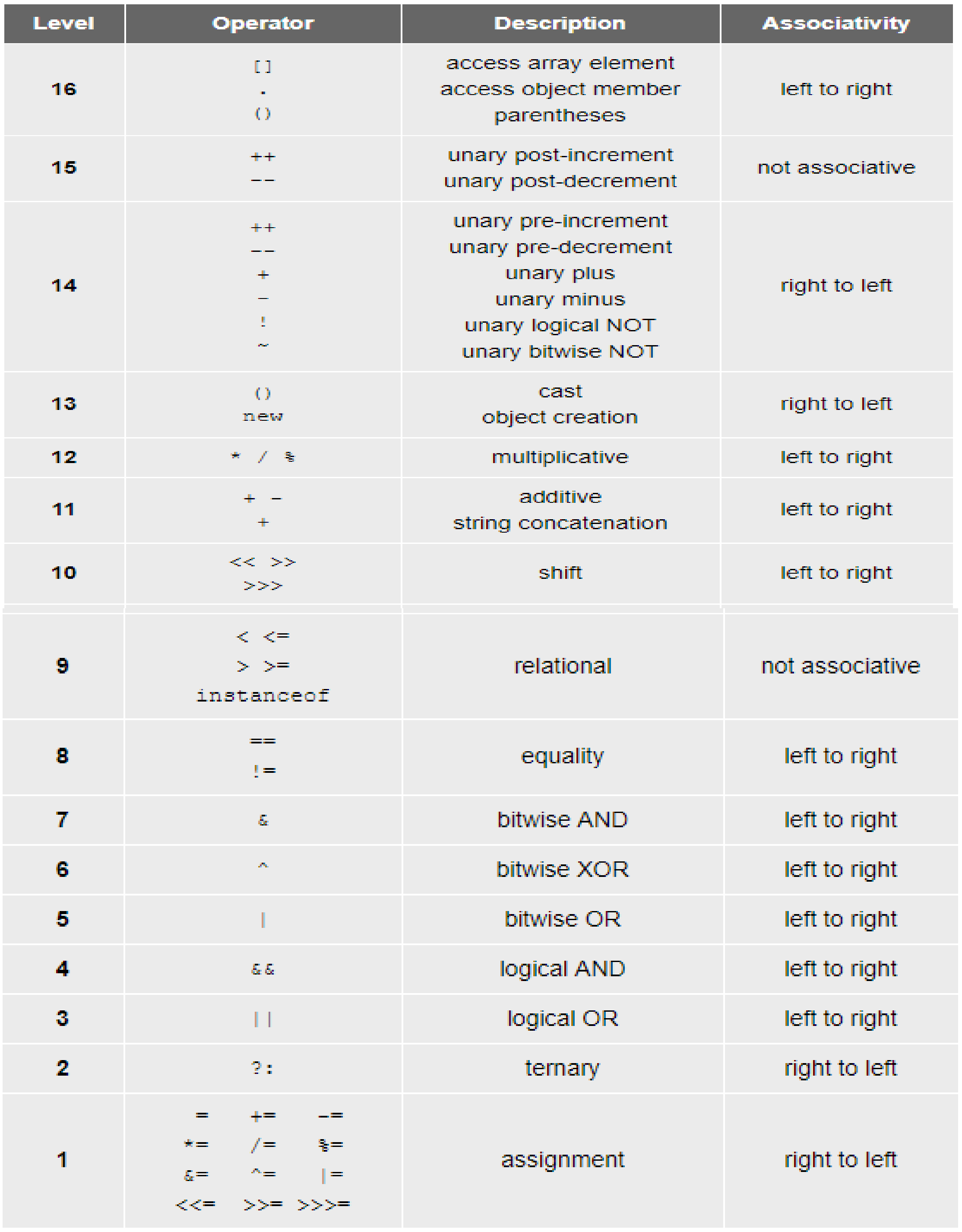
```
public class Test
{
    public static void main(String[] args)
    {
      Test t= new Test();
      System.out.println(t instanceof Test);
      }
}
```

Ex:2

```
public class Test {
   public static void main(String args[]) {
      String name = "James";
     // following will return true since name is type of String
      boolean result = name instanceof String;
      System.out.println( result );
   }
}
```

Precedence Rules and Associativity

Precedence rules are used to determine the order of evaluation priority in case there are two operators with different precedence



Level	Operator	Description	Associativity

16	`[]`
	`.`
	`()`
		access array element
		access object member
		parentheses
			left to right

15	`++` `--`	unary post-increment unary post-decrement	not associative

14	`++`
	`--`
	`+`
	`-`
	`!`
	`~`
		unary pre-increment
		unary pre-decrement
		unary plus
		unary minus
		unary logical NOT
		unary bitwise NOT
			right to left

13	`()`
	`new`
		cast
		object creation
			right to left

12	`* / %`	multiplicative	left to right

11	`+ -`
	`+`
		additive
		string concatenation
			left to right

10	`<< >>`
	`>>>`
		shift	left to right

9	`< <=`
	`> >=`
	`instanceof`
		relational	not associative

8	`==`
	`!=`
		equality	left to right

7	`&`	bitwise AND	left to right
6	`^`	bitwise XOR	left to right
5	`|`	bitwise OR	left to right
4	`&&`	logical AND	left to right
3	`||`	logical OR	left to right
2	`?:`	ternary	right to left

1	`= += -=`
	`*= /= %=`
	`&= ^= |=`
	`<<= >>= >>>=`
		assignment	right to left

Ex: When two operators share an operand the operator with the higher *precedence* goes first. For example, 1 + 2 * 3 is treated as 1 + (2 * 3), whereas 1 * 2 + 3 is treated as (1 * 2) + 3 since multiplication has a higher precedence than addition

Expressions

- An **expression** is any legal combination of symbols (like variables, constants and operators) that represents a value. In Python, an expression must have at least one operand (variable or constant) and can have one or more operators. On evaluating an expression, we get a value. *Operand* is the value on which operator is applied.
- ***Constant Expressions*:** One that involves only constants. **Example: 8 + 9 – 2**
- ***Integral Expressions:*** One that produces an integer result after evaluating the expression. **Example: a = 10**
- ***Floating Point Expressions:*** One that produces floating point results. **Example: a * b / 2**
- ***Relational Expressions:*** *One that returns either true or false value.* **Example: c = a>b**
- ***Logical Expressions:*** One that combines two or more relational expressions and returns a value as *True* or *False*. **Example: a>b && y! = 0**
- ***Bitwise Expressions*:** One that manipulates data at bit level. **Example: x = y&z**
- ***Assignment Expressions*:**One that assigns a value to a variable. **Example:c = a + b or c = 10**