

Expressions Builds

An **expression** in a programming language is a combination of one or more constants, variables, operators, and functions that the programming language interprets and is put together in a single statement. Variable and data become most useful when you use them in an expression. An expression is a literal value or variable and (or a combination of literal values, variables, operators, and other expressions that can be evaluated by the Python interpreter engine to produce a result. You use operands and operators to create expression in Python.

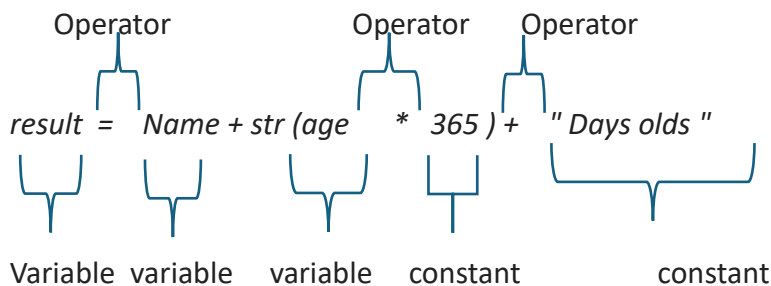
Example-1

```
Name="Asif "
```

```
age = 25;
```

```
result = Name + str (age * 365 ) + " Days olds "  # this is string expression
```

```
print (result);
```



<pre>name="Ali" age=20 result=(f"My name is {name} and I am {age*365} days old.") print (result)</pre>	
<pre>name="Ali" age=20 result2 = "my name is "+ name + " and I am " + str(age * 365) + " days old." print (result2)</pre>	

Example-2

```
Number1=20;
```

```
Number2=3;
```

```
Sum = Number1 - Nnumber2  # This is numerical expression because result will be numeric
```

Example-3

```
Number1=5
```

```
Number2=6
```

```
Result = number1 > number2  # This is Boolean expression because result will be true or false
```

Note : There are three types of expression one is string, second is numeric and third is Boolean depend on result type.

Operators

Python language supports a rich set of built-in operators. An operator is a symbol that tells the interpreter to perform certain mathematical or logical manipulations. Operators are used in program to manipulate data and variables. Python operators are binary, unary and ternary operators. A unary operator required one operand, and binary operator required two operands between the one operator. List of main types of Python operators.

Python operators can be classified into following operation types,

S.No.	Operator Type	Description
1	Arithmetic	Performs mathematical calculations.
2	Comparison/Relational	Compares operands and return a Boolean value
3	Assignment	Assign values to variables.
4	Logical	Performs Boolean operations on Boolean operands
5	Bitwise	Bitwise operators perform manipulations of data at bit level .
6	Membership	Membership operators check whether a value exists inside a sequence such as a list, tuple, string, or set..
7	Identity Operator	Identity operators check whether two variables refer to the same object .

1. Arithmetic operators: Python supports all the basic arithmetic operators. The following table shows all the basic arithmetic operators.

Symbol	Description	Example	Result
+	adds two operands	10 + 3	13
-	subtract second operands from first	10 - 3	7
*	multiply two operands	10 * 3	30
/	divide numerator by denominator	10 / 3	3.333
%	remainder of division	10 % 3	1
//	Floor division	10 // 3	3
**	Exponent/power	10 ** 3	1000

2. Relation operators: The following table shows all relation operators.

If $a = 5$, and if $b = 8$ then

Operator	Description	Example	Result
<code>==</code>	Check if two operand are equal	$a == b \rightarrow$	false
<code>!=</code>	Check if two operands are not equal.	$a != b \rightarrow$	true
<code>></code>	Check if operand on the left is greater than operand on the right	$a > b \rightarrow$	false
<code><</code>	Check operand on the left is smaller than right operand	$a < b \rightarrow$	true
<code>>=</code>	check left operand is greater than or equal to right operand	$a >= b \rightarrow$	false
<code><=</code>	Check if operand on left is smaller than or equal to right operand	$a <= b \rightarrow$	true

3. Logical operators: Python language supports following 3 logical operators. Suppose $a=1$ and $b=0$,

Operator	Description	Example
and	Int the two or more comparison expression are true then output will be true	$(5 > 3 \text{ and } 9 > 13) \rightarrow$ then false
Or	In the two or more comparison expression at least one true then output will be true	$(5 > 3 \text{ or } 9 > 3) \rightarrow$ then is true
not	If Expression is true but is Logical NOT then output will be false	$(\text{not } 5 > 3) \rightarrow$ then false

4. Assignment Operators: Assignment operators supported by Python.

Operator	Description	Example
=	assigns values from right side operands to left side operand	a=b
+=	adds right operand to the left operand and assign the result to left	a+=b is same as a=a+b
-=	subtracts right operand from the left operand and assign the result to left operand	a-=b is same as a= a- b
=	multiply left operand with the right operand and assign the result to left operand	a=b is same as a=a * b
/=	divides left operand with the right operand and assign the result to left operand	a/=b is same as a=a / b
%=	calculate modulus using two operands and assign the result to left operand	a%=b is same as a=a % b
//=	Floor division from of left operand with the right operand and assign the result to left operand	a //= b is same as a = a // b
**=	Exponent from of left operand with the right operand and assign result to left operand	a ** b is same as a = a ** b

5. Bitwise operators: Bitwise operators perform manipulations of data at **bit level**. These operators also perform **shifting of bits** from right to left. Bitwise operators are not applied to **float** or **double**, only apply Integer value.

Operator	Description
&	Bitwise AND
	Bitwise OR
^	Bitwise exclusive OR(XOR)
<<	left shift
>>	right shift
~	Bitwise NOT

Truth table for bitwise **&**, **|** and **^**

a	B	a & b	a b	a ^ b
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

The bitwise shift operators shift the bit value. The left operand specifies the value to be shifted and the right operand specifies the number of positions that the bits in the value are to be shifted. Both operands have the same precedence.

Preceding of operator

Operator precedence determines the order in which operators are evaluated. Operators with higher precedence are evaluated first. Operator precedence refers to the order in which operators execute within an expression.

The precedence of an operator specifies how "tightly" it binds two expressions together. For example, in the expression $1 + 5 * 3$, the answer is 16 and not 18 because the multiplication (" $*$ ") operator has a higher precedence than the addition (" $+$ ") operator. Parentheses may be used to force precedence, if necessary. For instance: $(1 + 5) * 3$ evaluates to 18.

Use of parentheses, even when not strictly necessary, can often increase readability of the code by making grouping explicit rather than relying on the implicit operator precedence and associativity. The following table lists the Python operators, ordered from highest to lowest precedence.

Operator (symbols)	Type Operator	Associativity
++ --	Increment and decrement	Right to left
!	Logical Not	Right to Left
/ * %	Multiply , division and modulus	Left to right
+ - .	Addition , subtraction and concatenation string	Left to Right
== <= >= > < != <>	Relational or comparisons	None
&& 	Logical And , Or	Left to Right
= += -= *= /= %=	Assignment operator	Right to left

Exercise

Theory Questions

1. What is Operator and describe of relational operators with symbols.
2. What difference between the binary and unary operators?
3. What do you mean by preceding Operators?
4. What do you mean by BODMAS operation?
5. What difference between the (= and ==) operators.

Practical Questions

1. Write a simple program to calculate total numbers of days you lived up to until now
Hint: Input your age (in years) and then calculate number of days
`number_of_days= age(in years) * 365`
2. Write a code to calculate and print percentage of a student where,
Total_Marks=400, Obtained_Marks=Sum of obtained marks of all 4 subjects
Percentage= (Obtained_Marks*100)/Total_Marks.
(Hint: input marks of four subjects out of 100).
3. Write a code to input Temperature in Centigrade and convert to Fahrenheit.
4. If A= 34 and B=55 then solve the following Exercise; Also Justify your answer
 - a) A= = 34 && B = = 55
 - b) A>=30 || B<=50
 - c) B= = 55 || A= =35
 - d) A !=34
 - e) A>=30 && A<35
 - f) B >50 || B<56
5. Mention the output of the following code at each step

Output

a = 33

b = 55

a += b;

c = a;

c -= b;

c *= a;

a + = 1

print (f "a={a} ,b={b} , c={c}")

6. What is the result after the execution of the following program segment?

a = 8;

b = a + 5;

```
c = b + 10;
print (f"a= {a} , b= {b} , c = {c} ")
```

7. Write down the output of following code and also justify your answer
 - a) `print( 4 + 2 - 12 * 3)` //what will be output
 - b) `print( 4 + ( 2 - 12) * 3)` //what will be output
 - c) `print ( (4 + 2) - 12 ) * 3` //what will be output
8. What value is assigned to `$ReturnValue` for each of the following expression.
 - a) `_ReturnValue = 2 == 3`
 - b) `_returnValue = "2" + "3";`
 - c) `_ReturnValue = 2 > 3`
 - d) `_ReturnValue = 2 < 3;`
 - e) `_ReturnValue = ( 2 > 3 ) and ( 2 < 3 )`
 - f) `_ReturnValue = ( 2 > 3 ) or ( 2 < 3 )`

Objective MCQ's

1. What is the value of the expression `4 * 2 + 3`?
 - a) 11
 - b) -11
 - c) 20
 - d) 24
2. The logical **and** operator return TRUE if _____
 - a) The left operand return a value is true
 - b) The right operand return a value is true
 - c) The left and right operand both return a value true then will be true
 - d) The left and right operand both return a value false then will be false
3. Which arithmetic operator can be used as both prefix and postfix operators?
 - a) ++
 - b) -
 - c) +
 - d) *
4. Which of the following values can be assigned to a Boolean variable?
 - a) True or False
 - b) False
 - c) 1
 - d) Yes

5. What value is assigned to the `_ReturnValue` variable in the statement `ReturnValue = 100 != 200 ;`
 - a) True
 - b) False
 - c) 100
 - d) 200

6. Which of the following is an example of initializing a variable.
 - a) `num= 2;`
 - b) `num >= 2;`
 - c) `num <= 2;`
 - d) `num == 2;`

7. A relational operator is used to
 - a) Combine the values
 - b) Distinguish different types of variables.
 - c) Change variables to logical values
 - d) Compare the values

8. Operators are used to perform some operation on the _____.
 - a) Operands
 - b) Function
 - c) Variables
 - d) Constant