Mc Graw Hill

Because learning changes everything.®

TITLE: Python Programming: Problem Solving, Packages and Libraries

Lecture PPT: Chapter 3 (Operators In Python)

Learning Objectives

- Understand the difference between use of "=" in Python and in mathematics.
- Realize that "=" is used for "assignment" and not for "equality testing".
- Identify the various types of operators used in Python, namely assignment; arithmetic; comparison; Logical; bit-wise and special operators.
- Comprehend the "order of precedence" of operators.
- Use the six "comparison" operators (<, ≤, >, ≥, ==, !=), the two identity operators (is, is not) and the two membership operators (in, not in).
- List and use the three logical operators (and, or, not).
- List and use the six bitwise operators (&, |, ~, ^, ≫, ≪)

Assignment

- **In Python, the '=' sign or token is used to "assign" values to a variable, and hence, it is called the "assignment" operator.**
- **For instance, x = 5 is a simple assignment operator that assigns the value 5 on the right to the variable x on the left.**
- **The assignment operator "=" can be thought of as an arrow ←.**
- **Thus a = 5 actually means a←5.**
- **So you can think of the variable "a" as a "box" which "contains" the value 5.**

Reassignment in Python

- There are many compound operators (or reassignment operators) in Python, such as a += 5 (Which means a←a+5, that is, assign to “a” value 5 more than “a”) that adds to the variable and later assigns the same. It is equivalent to a = a + 5.
- So when you do a = a + 5, you are actually doing a “reassignment” or an “update” of the value of the variable “a”.
- One can think of the “re-assignment” in a = a + 5 as a two-step process:-
  - Evaluate the expression on the right-hand side (RHS).
  - Let the variable name on the left-hand side (LHS) refer to this new resulting object.
  - The fact that the variable (“a” here) appears on both sides does not matter.
- In a reassignment expression, such as a = a + 5, the variable ‘a’ must pre-exist.
- So, if you try to use a reassignment expression, such as a = a + 5, before “initializing” the variable “a”, it will lead to an error.

Various Assignment Operators in Python (Along with examples)

Operator	Example	Equivalent to
=	a = 10	a=10 (a← 10 (Assign a value of 10 to variable a))
+=	a += 15	a=a+15 (a←a+15 (Increase the value of a by 15))
-=	a -= 5	a = a – 5 (a←a-5 (Decrease the value of a by 5)
*=	a *= 15	a = a * 15 (a← a*15 (Multiply a by 15 and assign it to a)
/=	a /= 15	a = a / 15 (a←a/15 (Divide a by1 5 and assign the result to variable a)
%=	a %= 5	a = a % 5 (a←(a mod5) ie assign to variable a the remainder on division by 5)
//=	a //= 5	a = a // 5 (a← Quotient of a divided by 5, this is also called floor function)
**=	a **=5	a = a ** 5 (a← a^5 assign to a the value of $(a)^5$)

Common Mathemetical Operators (With Examples)

Sr	Operator	Meaning	Example
1	+	When acting on a single operand, acts as a “unary” plus	+20
		When acting on two operands, it adds them	x + y
2	-	When acting on a single operand, it acts as a “unary” minus	x – y
		When acting on two operands, it subtracts the “right” operand from the “left”.	-20
3	*	Multiplies the two operands	x * y
4	/	Divide the left operand by the operand on the right (Result is always a “float”)	x / y (Note 6/2 gives 3.0 not 3)
5	//	Floor division (Gives the integer to the left on the number line)	x // y(Example (7//3 → 2 and 7//3→-3 not -2)
6	%	Modulus operator. Gives remainder of division of left operand by right.	x % y (Example 13%8→5)
7	**	Exponent- The operand on the left is raised to the power of the operand on the right	x **y

Order of precedence of operators

The rules for "precedence of operators" in Python are the same as in mathematics as follows:

- **Parentheses have the highest precedence. So an expression in "parentheses" is always evaluated first.**
- **After "parentheses", "exponents" have the highest precedence.**
- **Next come multiplication and division operators, which have the same precedence.**
- **Finally, there are addition and subtraction operators, which also have the same precedence.**
- **Operators having the same precedence are evaluated from left-to-right. This evaluation of operators of same precedence from left to right is called "left- associative". For instance, in an expression such as "x – y + z", "x- y" gets precedence over "-y+z" since both "-" and "+" have the same precedence but the operand "-" is to the left of the operand "+"**

Types of operands and result of operation

When an operator acts on two operands, there are the following possibilities:

- **The two operands are of the same "type" and the result of operation is also of the same "type". A good example is adding two integers.**
- **The two operands are of the same "type" but the result of operation is a different type. An example of this is division of an integer by another integer. The result is always a "float".**
- **The two operands are of different types, so the result is of one of the types of the operands. An example of this is when you multiply a float with an integer. The result is a float.**

Whenever an operation is done on operands, the programmer should be aware that

- **different operations may give different types of result.**
- **This topic relates to "casting" which is dealt with later.**

Comparison Operators

SNo	Type	Operator	Meaning
1	Comparison operators	<	True if left operand is strictly less than the right operand
2		<=	True if left operand is either less than or equal to the right operand
3		>	True if the left operand is strictly greater than the right operand.
4		>=	True if the left operand is either equal to or greater than the right operand.
5		==	True if left operand equals right operand
6		!=	True if left operand is not equal to the right operand.

Identity Operators

SNo	Type	Operator	Meaning
7	Identity operators	is	Object identity. Evaluates to True if the variables on either side of the “is” operator refer to the same object and False otherwise.
8		is not	Negated object identity. Evaluates to False if the variables on either side of the operator refer to the same object and True otherwise.

Membership Operators

SNo	Type	Operator	Meaning
9	Membership operators	in	Evaluates to True if it finds a variable (on the LHS) in the specified sequence (on the RHS) and False otherwise.
10		not in	Evaluates to True if it does not find a variable (on the LHS) in the specified sequence (on the RHS) and False otherwise.

Logical Operators

Operator	Meaning	Example
and	True if both the operands are True	"x and y" will be True if both x and y are True
or	True if either 1 or both the operands are True	x or y will be True if either or both x and y are True
not	Operates on a single operand and complements it.(Unary operator)	"not x" is True if x is False. "not x" is False if x is True.

Bit-wise operators

S.No	Operator	Type	Meaning
1	>>	Unary	Bitwise right shift
2	<<	Unary	Bitwise Left shift
3	&	Binary	Bitwise AND
4	|	Binary	Bitwise OR
5	~	Binary	Bitwise NOT
6	^	Binary	Bitwise XOR

The following are important concepts regarding bit-wise operators:

- **They may be unary or binary.**
- **They treat the operand(s) as a "string" of "binary digits".**
- **They are called bitwise operators since they act "bit by bit".**

Y

X

Because learning changes everything.®

Thank You!

For any queries or feedback contact us at:

support.india@mheducation.com

1800-103-5875

www.mheducation.co.in