Phụ lục I. Danh sách toán tử và phụ lục

Operator	Name	Explanation	Examples
+	Plus	Adds the two objects	`3 + 5` gives `8`. `'a' + 'b'` gives `'ab'`.
-	Minus	Either gives a negative number or gives the subtraction of one number from the other	`-5.2` gives a negative number. `50 - 24` gives `26`.
*	Multiply	Gives the multiplication of the two numbers or returns the string repeated that many times.	`2 * 3` gives `6`. `'la' * 3` gives `'lalala'`.
**	Power	Returns x to the power of y	`3 ** 4` gives `81` (i.e. `3 * 3 * 3 * 3`)
/	Divide	Divide x by y	`4/3` gives `1` (division of integers gives an integer). `4.0/3` or `4/3.0` gives`1.3333333333333333`
//	Floor Division	Returns the floor of the quotient	`4 // 3.0` gives `1.0`
%	Modulo	Returns the remainder of the division	`8%3` gives `2`. `-25.5%2.25` gives `1.5` .
<<	Left Shift	Shifts the bits of the number to the left by the number of bits specified. (Each number is represented in memory by bits or binary digits i.e. 0 and 1)	`2 << 2` gives `8`. `2` is represented by `10` in bits. Left shifting by 2 bits gives `1000` which represents the decimal `8`.
>>	Right Shift	Shifts the bits of the number to the right by the number of bits specified.	`11 >> 1` gives `5`. `11` is represented in bits by `1011` which when right shifted by 1 bit gives`101` which is nothing but decimal `5`.
&	Bitwise AND	Bitwise AND of the numbers	`5 & 3` gives `1`.
|	Bit-wise OR	Bitwise OR of the numbers	`5 | 3` gives `7`
^	Bit-wise XOR	`5 ^ 3` gives `6`	Â
~	Bit-wise invert	The bit-wise inversion of x is -(x+1)	`~5` gives `-6`.

Operator	Name	Explanation	Examples
<	Less Than	Returns whether x is less than y. All comparison operators return 1 for true and 0 for false. This is equivalent to the special variables `True` and `False` respectively. Note the capitalization of these variables' names.	`5 < 3` gives `0` (i.e. `False`) and `3 < 5` gives `1` (i.e. `True`). Comparisons can be chained arbitrarily: 3 < 5 < 7 gives `True`.
>	Greater Than	Returns whether x is greater than y	`5 < 3` returns `True`. If both operands are numbers, they are first converted to a common type. Otherwise, it always returns `False`.
<=	Less Than or Equal To	Returns whether x is less than or equal to y	`x = 3; y = 6; x <= y` returns `True`.
>=	Greater Than or Equal To	Returns whether x is greater than or equal to y	x = 4; y = 3; x >= 3 returns `True`.
==	Equal To	Compares if the objects are equal	`x = 2; y = 2; x == y` returns `True`. `x = 'str'; y = 'stR'; x == y` returns`False`. `x = 'str'; y = 'str'; x == y` returns `True`.
!=	Not Equal To	Compares if the objects are not equal	`x = 2; y = 3; x != y` returns `True`.
not	Boolean NOT	If x is `True`, it returns `False`. If x is `False`, it returns `True`.	`x = True; not y` returns `False`.
and	Boolean AND	`x and y` returns `False` if x is `False`, else it returns evaluation of y	`x = False; y = True; x and y` returns `False` since x is False. In this case, Python will not evaluate y since it knows that the value of the expression will has to be false (since x is False). This is called short-circuit evaluation.
or	Boolean OR	If x is `True`, it returns True, else it returns evaluation of y	`x = True; y = False; x or y` returns `True`. Short-circuit evaluation applies here as well.