Basic Operators

Bok, Jong Soon
javaexpert@nate.com
https://github.com/swacademy/Python

Operators

- Performs a function on one, two, or three operands and returns a result.
- An operator that requires one operand is called a ***unary operator***.
- An operator that requires two operands is a ***binary operator***.
- A ***ternary operator*** is one that requires three operands.
- Contains precedence order(top to bottom) along with their ***associativity*** (left to right or right to left).

Types of Operator

- Arithmetic Operators
- Comparison (Relational) Operators
- Assignment Operators
- Logical Operators
- Bitwise Operators
- Membership Operators
- Identity Operators

Operators in Order of Precedence

No.	Operator	Description
1	`**`	Exponentiation (raise to the power)
2	`~`, `+`, `-`	Complement, unary plus and minus (method names for the last two are +@ and -@)
3	`*`, `/`, `%`, `//`	Multiply, divide, modulo and floor division
4	`+`, `-`	Addition and subtraction
5	`<<`, `>>`	Right and left bitwise shift
6	`&`	Bitwise `'AND'`
7	`^`, `|`	Bitwise exclusive `` `OR' `` and regular `` `OR' ``
8	`<=`, `<`, `>`, `>=`	Comparison operators
9	`<>`, `==`, `!=`	Equality operators
10	`=`, `%=`, `/=`, `//=`, `-=`, `+=`, `*=`, `**=`	Assignment operators
11	`is`, `is not`	Identity operators
12	`in`, `not in`	Membership operators
13	`not`, `or`, `and`	Logical operators

Python Arithmetic Operators

- Assume variable **a** holds the value 10 and variable **b** holds the value 21.

Operator	Description	Example
+ Addition	Adds values on either side of the operator.	a + b = 31
– Subtraction	Subtracts right hand operand from left hand operand.	a – b = -11
* Multiplication	Multiplies values on either side of the operator	a * b = 210
/ Division	Divides left hand operand by right hand operand	b / a = 2.1
% Modulus	Divides left hand operand by right hand operand and returns remainder	b % a = 1

Python Arithmetic Operators (Cont.)

- Assume variable **a** holds the value 10 and variable **b** holds the value 20.

Operator	Description	Example
** Exponent	Performs exponential (power) calculation on operators.	a ** b =10^{20}
//	Floor Division - The division of operands where the result is the quotient in which the digits after the decimal point are removed. But if one of the operands is negative, the result is floored, i.e., rounded away from zero (towards negative infinity):	9 // 2 = 4 9.0 // 2.0 = 4.0 -11 // 3 = -4 -11.0 // 3 = -4.0

Python Arithmetic Operators (Cont.)

```
>>> a = 21
>>> b = 10
>>> c = 0
>>>
>>> c = a + b
>>> print ("Line 1 - Value of c is ", c)
Line 1 - Value of c is  31
>>> c = a - b
>>> print ("Line 2 - Value of c is ", c)
Line 2 - Value of c is  11
>>> c = a * b
>>> print ("Line 3 - Value of c is ", c)
Line 3 - Value of c is  210
>>> c = a / b
>>> print ("Line 4 - Value of c is ", c)
Line 4 - Value of c is  2.1
>>> c = a % b
>>> print ("Line 5 - Value of c is ", c)
Line 5 - Value of c is  1
>>>
```

```
>>> a = 2
>>> b = 3
>>> c = a ** b
>>> print ("Line 6 - Value of c is ", c)
Line 6 - Value of c is  8
>>>
>>> a = 10
>>> b = 5
>>> c = a // b
>>> print ("Line 7 - Value of c is ", c)
Line 7 - Value of c is  2
>>>
```

Python Comparison Operators

- Assume variable **a** holds the value 10 and variable **b** holds the value 20

Operator	Description	Example
==	If the values of two operands are equal, then the condition becomes true.	(a == b) is not true.
!=	If values of two operands are not equal, then condition becomes true.	(a != b) is true.

Python Comparison Operators (Cont.)

- Assume variable **a** holds the value 10 and variable **b** holds the value 20.

Operator	Description	Example
>	If the value of left operand is greater than the value of right operand, then condition becomes true.	(a > b) is not true.
<	If the value of left operand is less than the value of right operand, then condition becomes true.	(a < b) is true.
>=	If the value of left operand is greater than or equal to the value of right operand, then condition becomes true.	(a >= b) is not true.
<=	If the value of left operand is less than or equal to the value of right operand, then condition becomes true.	(a <= b) is true.

Python Comparison Operators (Cont.)

```
>>> a = 21
>>> b = 10
>>>
>>> if ( a == b) :
        print ("Line 1 - a is equal to b")
else :
        print ("Line 1 - a is not equal to b")


Line 1 - a is not equal to b
>>>
>>> if (a != b) :
        print ("Line 2 - a is not equal to b")
else :
        print ("Line 2 - a is equal to b")


Line 2 - a is not equal to b
>>>
```

```
>>> a = 21
>>> b = 10
>>>
>>> if (a < b ) :
        print ("Line 3 - a is less than b")
else :
        print ("Line 3 - a is not less than b")


Line 3 - a is not less than b
>>>
>>> if (a > b) :
        print ("Line 4 - a is greater than b")
else :
        print ("Line 4 - a is not greater than b")


Line 4 - a is greater than b
>>>
```

Python Comparison Operators (Cont.)

```
>>> a = 21
>>> b = 10
>>> a, b = b, a   # values of a and b swapped. a becomes 10, b becomes 21.
>>>
>>> if (a <= b) :
        print ("Line 5 - a is either less than or equal to b")
else :
        print ("Line 5 - a is neither less than nor equal to b")


Line 5 - a is either less than or equal to b
>>>
>>> if (b >= a) :
        print ("Line 6 - b is either greater than or equal to b")
else :
        print ("Line 6 - b is neither greater than nor equal to b")


Line 6 - b is either greater than or equal to b
>>>
```

Python Assignment Operators

- Assume variable **a** holds the value 10 and variable **b** holds the value 20.

Operator	Description	Example
=	Assigns values from right side operands to left side operand.	c = a + b assigns value of a + b into c
+= Add AND	It adds right operand to the left operand and assign the result to left operand.	c += a is equivalent to c = c + a
-= Subtract AND	It subtracts right operand from the left operand and assign the result to left operand.	c -= a is equivalent to c = c - a
*= Multiply AND	It multiplies right operand with the left operand and assign the result to left operand.	c *= a is equivalent to c = c * a

Python Assignment Operators (Cont.)

- Assume variable a holds the value 10 and variable b holds the value 20.

Operator	Description	Example
/= Divide AND	It divides left operand with the right operand and assign the result to left operand.	c /= a is equivalent to c = c / ac /= a is equivalent to c = c / a
%= Modulus AND	It takes modulus using two operands and assign the result to left operand.	c %= a is equivalent to c = c % a
**= Exponent AND	Performs exponential (power) calculation on operators and assign value to the left operand.	c **= a is equivalent to c = c ** a
//= Floor Division	It performs floor division on operators and assign value to the left operand.	c //= a is equivalent to c = c // a

Python Assignment Operators (Cont.)

```
>>> a = 21
>>> b = 10
>>> c = 0
>>>
>>> c = a + b
>>> print ("Line 1 - Value of c is ", c)
Line 1 - Value of c is  31
>>>
>>> c += a
>>> print ("Line 2 - Value of c  is ", c)
Line 2 - Value of c  is  52
>>>
>>> c *= a
>>> print ("Line 3 - Value of c is ", c)
Line 3 - Value of c is  1092
>>>
>>> c /= a
>>> print ("Line 4 - Value of c is ", c)
Line 4 - Value of c is  52.0
>>>
```

```
>>> a = 21
>>> c = 2
>>>
>>> c %= a
>>> print ("Line 5 - Value of c is ", c)
Line 5 - Value of c is  2
>>>
>>> c **= a
>>> print ("Line 6 - Value of c is ", c)
Line 6 - Value of c is  2097152
>>>
>>> c //= a
>>> print ("Line 7 - Value of c is ", c)
Line 7 - Value of c is  99864
>>>
```

Python Bitwise Operators

- Bitwise operator works on bits and performs bit-by-bit operation. Assume if a = 60; and b = 13.

Operator	Description	Example
& Binary AND	Operator copies a bit to the result if it exists in both operands.	(a & b) (means 0000 1100)
| Binary OR	It copies a bit if it exists in either operand.	(a | b) = 61 (means 0011 1101)
^ Binary XOR	It copies the bit if it is set in one operand but not both.	(a ^ b) = 49 (means 0011 0001)
~ Binary Ones Complement	It is unary and has the effect of 'flipping' bits.	(~a) = -61 (means 1100 0011 in 2's complement form due to a signed binary number.

Python Bitwise Operators (Cont.)

- Bitwise operator works on bits and performs bit-by-bit operation. Assume if a = 60; and b = 13.

Operator	Description	Example
<< Binary Left Shift	The left operands value is moved left by the number of bits specified by the right operand.	a <<= 240 (means 1111 0000)
>> Binary Right Shift	The left operands value is moved right by the number of bits specified by the right operand.	a >>= 15 (means 0000 1111)

Python Bitwise Operators (Cont.)

```
>>> a = 60                          # 60 = 0011  1100
>>> b = 13                          # 13 = 0000  1101
>>> print ("a = ", a, " : ", bin(a), ", b = ", b, " : ", bin(b))
a =  60  :  0b111100 , b =  13  :  0b1101
>>>
>>> c = a & b                       # 12 = 0000  1100
>>> print ("Result of AND is ", c, " : ", bin(c))
Result of AND is  12  :  0b1100
>>>
>>> c = a | b                       # 61 = 0011  1101
>>> print ("Result of OR is ", c, " : ", bin(c))
Result of OR is  61  :  0b111101
>>>
>>> c = a ^ b                       # 49 = 0011  0001
>>> print ("Result of XOR is ", c, " : ", bin(c))
Result of XOR is  49  :  0b110001
>>>
```

```
>>> a = 60
>>> c = 0
>>>
>>> c = ~a                                  # -61 = 1100  0011
>>> print ("Result of COMPLEMENT is ", c, " : ", bin(c))
Result of COMPLEMENT is  -61  :  -0b111101
>>>
>>> c = a << 2                              # 240 = 1111  0000
>>> print ("Result of LEFT SHIFT is ", c, " : ", bin(c))
Result of LEFT SHIFT is  240  :  0b11110000
>>>
>>> c = a >> 2                              # 15 = 0000  1111
>>> print ("Result of RIGHT SHIFT is ", c, " : ", bin(c))
Result of RIGHT SHIFT is  15  :  0b1111
>>>
```

Python Logical Operators

- Assume variable **a** holds **True** and variable **b** holds **False**.

Operator	Description	Example
and Logical AND	If both the operands are true then condition becomes true.	(a **and** b) is **False**.
or Logical OR	If any of the two operands are non-zero then condition becomes true.	(a **or** b) is **True**.
not Logical NOT	Used to reverse the logical state of its operand.	**not(a and** b) is **True**.

Python Logical Operators (Cont.)

```
>>> a = True
>>> b = False
>>>
>>> print ("a and b is ", (a and b))
a and b is  False
>>>
>>> print ("a or b is ", (a or b))
a or b is  True
>>>
>>> print ("not(a and b) is ", not(a and b))
not(a and b) is  True
>>>
```

Python Membership Operators

- Python's membership operators test for membership in a sequence, such as strings, lists, or tuples. There are two membership operators as explained below:

Operator	Description	Example
`in`	Evaluates to true if it finds a variable in the specified sequence and false otherwise.	x `in` y, here in results in a 1 if x is a member of sequence y.
`not in`	Evaluates to true if it does not finds a variable in the specified sequence and false otherwise.	x `not in` y, here not in results in a 1 if x is not a member of sequence y.

Python Membership Operators (Cont.)

```
>>> a = 10
>>> b = 20
>>> list = [1, 2, 3, 4, 5]
>>>
>>> if ( a in list ) :
            print ("Line 1 - a is available in the given list")
else :
            print ("Line 1 - a is not available in the given list")


Line 1 - a is not available in the given list
>>> if ( b not in list ) :
            print ("Line 2 - b is not available in the given list")
else :
            print ("Line 2 - b is available in the given list")


Line 2 - b is not available in the given list
>>>
>>> c = b / a
>>> if (c in list ) :
            print ("Line 3 - a is available in the given list")
else :
            print ("Line 3 - a is not available in the given list")


Line 3 - a is available in the given list
```

Python Identity Operators

- Identity operators compare the memory locations of two objects. There are two Identity operators as explained below :

Operator	Description	Example
`is`	Evaluates to true if the variables on either side of the operator point to the same object and false otherwise.	x `is` y, here is results in 1 if id(x) equals id(y).
`is not`	Evaluates to false if the variables on either side of the operator point to the same object and true otherwise.	x `is not` y, here is not results in 1 if id(x) is not equal to id(y).

Python Identity Operators (Cont.)

```
>>> a = 20
>>> b = 20
>>>
>>> print ("Line 1 ", ", a = " , a, " : ", id(a), ", b = ", b, " : ", id(b))
Line 1  , a =  20  :  10804416 , b =  20  :  10804416
>>>
>>> if( a is b ) :
            print ("Line 2 - a and b have same identity")
else :
            print ("Line 2 - a and b do not have same identity")


Line 2 - a and b have same identity
>>>
>>> if ( id(a) == id(b)) :
            print ("Line 3 - a and b have same identity")
else :
            print ("Line 3 - a and b do not have same identity")


Line 3 - a and b have same identity
>>>
```

```
>>> a = 20
>>> b = 30
>>>
>>> print ("Line 4 ", ", a = ", a, " : ", id(a), ", b = ", b, " : ", id(b))
Line 4  , a =  20  :  10804416 , b =  30  :  10804736
>>>
>>> if ( a is not b) :
            print ("Line 5 - a and b do not have same identity")
else :
            print ("Line 5 - a and b have same identity")


Line 5 - a and b do not have same identity
>>>
```