Elisha Vernee Hart

Revise The Operators Covered So Far

Operators are symbols that carryout computation. These operators include arithmetic operators such as +, -, /, %, //, *, **

The operator symbol “+” will perform addition if both or one of the operand is Numeric or Boolean. For an example:
4+True
Output: 5

Due to True being 1. Therefore 4+1=5.

The operator symbol “+” will perform if both operands are sequenced or are the same type. For example, ‘hello’ + [23] will show an error.

The operator symbol “*” will perform multiplication if both or one of the operand is Numeric or Boolean. An example is 2*3 and the output shows 6.

The operator symbol “*” will perform repetition if one of the operand is sequenced and the other is “int.” For example, 2 * ‘Python’ the output will show ‘PythonPython’

The operator symbol “/” performs division. It always returns the float result. For example, 12/34 and the output will show 0.3529411764705886.

Another example is 12/3 and the output showing 4.0.

The operator symbol “//” is called floor division. If both the operands are int the ans or answer will be int. If one of the operand is a float the output will be float. For example, 34//35 the output will show 0.

The power of operator is “**” An example would be 2**3 and the output would be 8 since 2 to the 3rd power is equal to 8.

Relational Operator or Conditional Operator which is >, >=, <, <=, ==, !=. For example, 3 >= 3 and the output will show True. The output will always be a Boolean value.

Logical Operators include and, or, not. When using the word “and”, if one of the operand is False, the output will be False. For example, 1 and False will be False.

If both the operands are False values, like 0, 0.0, False, then the output will be the first operand.
For example, in the compiler this was typed:
‘ ‘ and 0.0
The output will be the first operand.
The output showed ‘ ‘

Another example is 0j and 0
The output will be 0j.

If both the operands are True which includes any int, any float, any complex or for example 0j, any string. The output will be the second operand.
For example:
6.0 and True
The output showed True

Another example:
3 and ‘ ‘
The output is ‘ ‘

The word “not” always inverts the results, it always returns Boolean values.

Bitwise operators includes &: bitwise and
|: bitwise or
~: bitwise not [complement]
^: bitwise XOR
<<: left shift
>>: right shift

& if the bits are 1 output is 1, otherwise output is 0.
For example:

```
  1001
+ 1111
______
1001
```

| if one of the bit is 1 output is 1. If both are 0 output will be 0.
An example, 4 | 10
The output is 14.

~ will invert the individual bits. For example ~12 the output is -13.

^ if one of the bit is 1, output will be 1, if both are the same output is 0. For example, 1^2 the output is 3.

<< left shift, shifts the bits towards the left side. For example, 2 << 1 the output is 4.

>> right shift, shifts the bits towards the right side. For example, 14>>2 and the output is 3.

The advanced assignment operators includes: +=, -=, /=, //=, %=, *=, **=, &=, |=, ^=, <<=, >>=...

Assignment operator is assigning the literal or expression on its right to the variable on its left. An example is: a = 34

```
                id(a)
                Output 140504033533264
```

Membership Operator includes words such as "in", "not in." It checks whether the subsequence is present in the given sequence such as list, tuple, str, bytes, bytearray.

Precedence Operator definition is, if there are multiple operators which operator will be evaluated first. Is decided by the precedence rules.

Special Operator which includes the Identity Operator and the Membership Operator.

```
a=2
b=2
print(id(a), id(b))
Output shows: 140504033532240   140504033532240
```

It can return True if both the operands are not the same object internally.