Part 06 - Operators

Part 06 - Operators

Mathematical

Name	Syntax example	Comments
Multiplication	`a * b`	
Division	`a / b`	
Remainder	`a % b`	Often called mod or modulus
Addition	`a + b`	
Subtraction	`a - b`	
Exponent	`a ** b`	Do not confuse this with Bitwise Xor ^
Bitshift Right	`a >> b`	
Bitshift Left	`a << b`	
Bitwise And	`a & b`	
Bitwise Or	`a | b`	
Bitwise Xor	`a ^ b`	

The mathematical operators can also be used in the syntax `a <operator>= b`, for example, `a += b`.
This is merely a shortcut for `a = a <operator> b`, or in our example `a = a + b`.

Relational and Logical

Name	Syntax Example	Comments
Less Than	`a < b`	
Greater Than	`a > b`	
Less Than or Equal To	`a <= b`	
Greater Than or Equal To	`a >= b`	

Equality	`a == b`	
Inequality	`a != b`	
Logical And	`a and b`	Only use when a and b are boolean values
Logical Or	`a or b`	Only use when a and b are boolean values
Logical Not	`not a`	Only use when a is a boolean value

Types

Name	Syntax Example	Comments
Typecast	`a cast string`	
Typecast	`a as string`	
Type Equality/Compatibility	`a isa string`	
Type Retrieval	`typeof(string)`	
Type Retrieval	`a.GetType()`	

Primary

Name	Syntax Example	Comments
Member	`A.B`	Classes are described in Part 08 - Classes
Function Call	`f(x)`	Functions are described in Part 07 - Functions
Post Increment	`i++`	See Difference between Pre and Post Increment/Decrement
Post Decrement	`i--`	See Difference between Pre and Post Increment/Decrement
Constructor Call	`o = MyClass()`	Classes are described in Part 08 - Classes

Unary

Name	Syntax Example	Comments
Negative Value	`-5`	
Pre Increment	`++i`	See Difference between Pre and Post Increment/Decrement
Pre Decrement	`--i`	See Difference between Pre and Post Increment/Decrement
Grouping	`(a + b)`	

Difference between Pre and Post Increment/Decrement

When writing inline code, Pre Increment/Decrement (`+i`/`-i`) commit the action, then return its new value, whereas Post Increment/Decrement (`i+`/`i-`) return the current value, then commit the change.

preincrement vs. postincrement

```
num = 0
for i in range(5):
    print num++
print '---'
num = 0
for i in range(5):
    print ++num
```

Output

```
0
1
2
3
4
---
1
2
3
4
5
```

✅ **Recommendation**

To make your code more readable, avoid using the incrementors and decrementors.
Instead, use `i += 1` and `i -= 1`.

Exercises

1. Put your hands on a wall, move your left foot back about 3 feet, move the right foot back 2 feet.

Go on to **Part 07 - Functions**