

Lecture 18 - Operators and other stuff

This is kind of the “stuff” that is in python - with a few exceptions.

We need to cover some aerators. We have used

Operator	Specification
+	add
-	subtract
*	multiply
/	divide resulting in a float (remainder)
//	divide integer result
%	remainder from division
==	equal
!=	not equal

Let's be much more specific in what these operators do.

```
a = 2
b = 3
c = 4

x = a + b * c

# Will Print 14 becasue 3 * 4 = 12, then add + 2
print ( f"x={x}" )
```

What we are seeing is “precedence” of operators. Most languages on a computer use a “precedence” process.

Operator	Description
()	Parentheses (grouping)
f(args...)	Function call
x[index:index]	Slicing
x[index]	Subscription

<code>x.attribute</code>	Attribute reference
<code>**</code>	Exponentiation
<code>~x</code>	Bitwise not
<code>+x, -x</code>	Positive, negative
<code>*, /, %</code>	Multiplication, division, remainder
<code>+, -</code>	Addition, subtraction
<code><<, >></code>	Bitwise shifts
<code>&</code>	Bitwise AND
<code>^</code>	Bitwise XOR
<code> </code>	Bitwise OR
<code>in, not in, is, is not, <, <=, >, >=, <>, !=, ==</code>	Comparisons, membership, identity
<code>not x</code>	Boolean NOT
<code>and</code>	Boolean AND
<code>or</code>	Boolean OR
<code>lambda</code>	Lambda expression

So we can use parens to order. You will notice in the table that they are at the top of the list.

```

a = 2
b = 3
c = 4

x = ( a + b ) * c

# Use Parenthesis to order
print ( f"x={x}" )

```

Computers representation is to use on/off signals - so a number is a set of on/off signals.

For humans this is 1's and 0's

This means that we have operators that work on binary data.

```
a = 4
```

```
b = 2
```

```
x = a << b
```

```
print ( f"x={x}" )
```

```
y = ( a * 2 ) * 2
```

```
print ( f"y={y}" )
```

```
a = 4
```

```
z = ~a
```

```
print ( f"z={z}" )
```

This leads to how computers represent negative numbers! They use a “bit” to say positive or negate.

That is why when we flipped all the bits in the value we got a negate number.

So what about division:

```
a = 32
```

```
b = 3
```

```
x = a >> b
```

```
print ( f"x={x}" )
```

And basic boolean operations (What is a boolean operation):

And:

```
a = 2
```

```
b = 3
```

```
c = a & b
```

```
print ( f"c={c}" )
```

Or:

```
a = 2
b = 3
c = a | b

print ( f"c={c}" )
```

Exclusive Or:

```
a = 2
b = 3
c = a ^ b

print ( f"c={c}" )
```