Lecture 18 - Operators and other stuff

Operators Part 1 - https://youtu.be/rQGaP2XJL_8
Operators Part 2 - https://youtu.be/p8sPI8_bVC8

From Amazon S3 - for download (same as youtube videos)

Operators Part 1
Operators Part 2

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
x.attribute	Attribute reference
**	Exponentiation
~*x*	Bitwise not
+*x*, -*x*	Positive, negative
*, /,%,//	Multiplication, division, remainder, integer devide
+,-	Addition, subtraction
<<, >>	Bitwise shifts
&	Bitwise AND
^	Bitwise XOR
|	Bitwise OR
in, not in, is, is not, <, <=, >, >=, <>, !=, ==	Comparisons, membership, identity
not *x*	Boolean NOT
and	Boolean AND
or	Boolean OR
lambda	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 Paraentisis 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}" )
```