Previously covered

~ << >>

Bitwise operators (cont)

& - AND

| - OR

^ - XOR, EOR exclusive-OR

AND &

```
  11001010   <- x
& 01111100   <- y
----------
  01001000
```

zeros in MASK

x & y

x & MASK

it is often helpful to think of one of the values as a "mask"

1 iff both bits are 1

Result is the first operand but with bits cleared where the mask has zeros

PORTA = 0b10101111

PORTA = PORTA & 0b11111100

↳ PORTA is now 0b10101100

OR |

```
  11001010   <- x
| 01111100   <- y
----------
  11111110
```

x | MASK

Result is the first operand but with bits set where the mask has ones

PORTA = PORTA | 0b00001000

This bit of PORTA is set

EOR ^

```
  11001010   <- x
^ 01111100   <- mask
----------
  10110110
```

Excludes case where both bits are 1

Result is the first operand but with bits toggled where the mask has ones

PORTA = PORTA ^ 0b00000010

toggles in this position

Result is 0 or 1

Non-zero is True

Logical operators

! logical "NOT"

X	!X	
1	0	← False
5	0	
-50	0	
0	1	← True

&& logical AND

X	Y	X && Y	
0	0	0	
0	5	0	← False
42	0	0	
3	7	1	← True

|| logical OR

X	Y	X || Y
0	0	0
0	17	1
-1	0	1
-47	101106	1

There is NO Logical EOR

Relational Operators ↓

Relational Operators

aka "evaluates to"

`==` "Returns" True (1) if operands are equal

`x==5` False (0) otherwise

↖ 1 if x is 5
0 otherwise

Don't confuse with `x=5` ← assignment

`!=` not equals (opposite)

`<`
`>` as you would think
`<=`
`>=`

`x < 7` True (equals 1) iff the value of x is less than 7 (else 0)

`++x` `x++` } add one to x ⟹ increment

`--x` `x--` } subtract 1 from x ⟹ decrement

x changes

`++x` and `--x` evaluate to x after change

`x++` and `x--` evaluate to x before change

Examples ↓

```
x = 5      y = ++x    ← y = 6  x = 6
x = 5      y = x++    ← y = 5  x = 6
```

The left side must be an "L-value" - something you can assign to e.g. x but not x+7

Assignment

= "equals" ← right side is assigned to left

Left side ← changes

x = y x gets assigned y's value

x += y ← x = x + y add y to x

These all evaluate to the result (new value)

-=
*=
/=
%=
<<=
>>=
&=
|=
^=

x -= y x = x - y

x *= 2 x = x * 2

x <<= 1 same as x = x * 2

x &= 0b0000111 vs x & 0b0000111

↖ x change

y = x+=5

↑ y is also the new value of x

?:

x = y<7 ? 5 : 3

test — result if true — result if false