Equality and Relational Operators

How do we *compare* **numerical** and **string** data?

*… with **relational and equality** operators!*

Test	Math	Operator
"is greater than?"	>	>
"is at least?"	≥	>=
"is less than?"	<	<
"is at most?"	≤	<=
"is equal to?"	=	==
"is not equal to?"	≠	!=

The `equal to` Operator is `==`

- Two equals symbols side-by-side can be read as *"is equal to?"*

`1 == 1` evaluates to `True`

`1 == 2` evaluates to `False`

- **Important!** Equality is *very different* from assignment!
  - = is read as "is bound to a value of"
  - == is read as "is equal to?"

- `b = x == y`

"The variable **b** is **assigned the result** of *evaluating* **'is x equal to y?'"**

The **not equal to** Operator is **!=**

- The ! symbol in many programming languages often means "NOT"

`1 != 1` evaluates to **`False`**

`1 != 2` evaluates to **`True`**

- **`b = x != y`**

"The variable **b** is **assigned a value** of *evaluating* **'is x *not* equal to y?'"**

Logical Type - `bool`

- Literal examples: `True`, `False`
- A `bool`, short for Boolean, can only be **one of two values**, *either* `True` or `False`.
- The next lesson will focus on `bool` operators:
  - `not`
  - `and`
  - `or`

Relational Expressions evaluate to **`bool`**s

- Notice the **evaluation** of each relational operator is a **`bool`** value

- But what is on either side of the relational expression is not a `bool` value!

```
10.0 >= 100.0
   False
1 == 1
   True
"a" < "b"
   True
"hello" == "HELLO"
   False
```

- For a **well typed program**, use **the same type** of objects on **both sides of a relational**!

Equality and Relational Precedence & Types

- These operators have **lower** precedence than arithmetic operators

- Thus:
  `1 + 1 == 2` is `True`

- Notice if == had higher precedence, then it would simplify to 1 + True which is invalid because, with strict type checking, adding a number to a boolean is non-sensible.