COMP 110

CL02 - Expressions

Last Lecture

- Data Types
 - float (decimal, e.g. 2.0)
 - o int (whole number, e.g. 2)
 - str (string of characters, e.g. "Hello")
 - bool (evaluates to True or False e.g. True, 2 >= 3)
- Check type
 - o type()
- Change type
 - o str(), float(), int()

Expressions

- Something that evaluates at runtime
- Every expression evaluates to a specific typed value
- Examples

```
0 1 + 2 * 3
```

- 0
- 0 1.0 * 2.0
- o "Hello" + " World!"
- o 1 > 3

Numerical Operators

Operator Name	Symbol
Addition	+
Subtraction/Negation	-
Multiplication	*
Division	/
Exponentiation	**
Remainder "modulo"	%

Addition +

If numerical objects, add the values together

```
\circ \quad 1 + 1 \rightarrow 2
```

$$\circ$$
 1.0 + 2.0 \rightarrow 3.0

If strings, concatenate them

The result type depends on the operands

```
o float + float → float
```

- \circ int + int \rightarrow int
- o float + int → float
- o int + float → float
- \circ str + str \rightarrow str

Addition +

• If numerical objects, add the values together

```
0.01 + 1 \rightarrow 2
0.01 + 2.0 \rightarrow 3.0
```

If strings, concatenate them

```
o "Comp" + "110" → "Comp110"
```

The result type depends on the operands

```
o float + float → float
```

- \circ int + int \rightarrow int
- o float + int → float
- o int + float → float
- \circ str + str \rightarrow str

Question: What happens when you try to add incompatible types?

Subtraction/Negation -

Meant strictly for numerical types

- \circ 3 2 \rightarrow 1
- \circ 4.0 2.0 \rightarrow 2.0
- \circ 4.0 2 \rightarrow 2.0
- $\circ (1+1) \rightarrow -2$

The result type depends on the operands

- \circ int int \rightarrow int
- \circ float int \rightarrow float
- \circ int float \rightarrow float

Multiplication *

If numerical objects, multiply the values

```
\begin{array}{ccc} \circ & 1 * 1 \rightarrow 1 \\ \circ & 1.0 * 2.0 \rightarrow 2.0 \end{array}
```

If string and int, repeat the string

```
    "Hello" * 3 → "HelloHelloHello"
```

The result type depends on the operands

```
o float * float → float
```

```
\circ int * int \rightarrow int
```

- o float * int → float
- o int * float → float
- \circ str * int \rightarrow str

Division /

- Meant strictly for numerical types
 - \circ 3/2 \rightarrow 1.5
 - \circ 4.0 / 2.0 \rightarrow 2.0
 - \circ 4/2 \rightarrow 2.0
- Division results in a float
 - o float / float → float
 - \circ int / int \rightarrow float
 - o float / int → float
 - o int / float → float

Exponentiation **

- Meant strictly for numerical types

 - \circ 2.0 ** 2.0 \rightarrow 4.0
- The result type depends on the operands
 - o float ** float → float
 - \circ int ** int \rightarrow int
 - o float ** int → float
 - o int ** float → float

Remainder "modulo"

- Calculates the remainder when you divide two numbers
- Meant strictly for numerical types
 - \circ 5 % 2 \rightarrow 1
 - \circ 6 % 3 \rightarrow 0
- The result type depends on the operands
 - \circ int % int \rightarrow int
 - o float % float → float
 - o float % int → float
 - o int % float → float
- Note:
 - o If x is even, $x \% 2 \rightarrow 0$
 - \circ If x is odd, x % 2 \rightarrow 1

Order Of Operations

- P()
- F **
- MD * / %
- AS + -
- Tie? Evaluate Left to Right

Relational Operators

Operator Name	Symbol
Equal?	==
Less than?	<
Greater than?	>
Less than or equal to? (At most)	<=
Greater than or equal to? (At least)	>=
Not equal?	!=

Relational Operators

- Always result in a bool (True or False)
- Equals (==) and Not Equal (!=)
 - Can be used for all primitive types we've learned so far! (bool, int, float, str)
- Every other type
 - Just use on floats and ints
 - (Can technically use on all primitive types)

Practice! Simplify and Type

Simplify: 2 + 4 / 2 * 2

(Reminder: P E M D A S)

Simplify: 2 + 4 / 2 * 2

What type is 2 + 4 / 2 * 2?

Simplify: 220 >= int(("1" + "1" + "0") * 2)

Mods Practice! Simplify

- 7 % 2
- 8 % 4
- 7%4
- Any even number % 2
- Any odd number % 2