Expressions, Operators, and Precedence

Expressions

An expression is something which has a value.

Here are some examples:

```
EXPRESSION          => RESULT

2 + 3               => 5
2 + 3 x 4           => 14
12 x feet + inches  => (some number)
BoxVolume(3, 4, 8)  => 96
247                 => 247
7 / 2               => 3.5
```

Data Types

A data type is a set of values.

int

The *int* datatype is used to represent integers.

int = `{0, 1, -1, 2, -2, …}`

- In Python there is no limit to the size of ints.
- To convert something from another datatype to an int (e.g. from a string), you can use the built-in function `int()`.

float

The *float* datatype is used to represent numbers with decimal points.

float = `{3.5, 3.14159265, -27.6, 3.0, …}`

- 3 is an int but 3.0 is a float.
- To convert from another datatype (e.g. a string) into a float, you can use the built-in function `float()`.

bool

The *bool* datatype is used to represent boolean values.

bool = `{True, False}`

- True and False are reserved words.

str

The *str* datatype is used to represent series of characters.

str = `{'Ann', 'Hi there', 'True', '5471', ' ', '' (the empty string), …}`

- You can use the built-in function `str()` to get the string representation of another datatype.

NoneType

The *NoneType* datatype contains only the value `None`, which is meant to represent nothing.

NoneType = `{None}`

- The only operations you can perform with `None` are to check if something is equal to it, or if it's not equal to it:

  `x == None` => `True` if x is None.
  `x != None` => `True` if x is not None.

type()

You can use the inbuilt function `type()` to check the type of a variable.

Operators

Arithmetic Operators

+

The addition operator. Works as you'd expect with numbers.

- Adding two floats always gives a float:

  `1.5 + 1.5 => 3.0`

- Also concatenates strings:

  `'a' + 'b' => 'ab'`

-

Subtraction operator. As you'd expect with numbers.

- Subtracting two floats always gives a float.

*

The multiplication operator, as you'd expect with numbers.

- Multiplication only gives an int if both inputs are also ints. If either is a float, the result is a float.
- Can also be used with strings:

  `'a' * 5 => 'aaaaa'`

/

Division.

- Always returns a float:

  `6 / 2 => 3.0`

//

Floor division.

- Ignores remainder:

  `10 // 3 => 3`

- Returns an int if fed two ints, returns a float if either input is a float.

%

Modulo operator. Gives the remainder after division.

- Examples:

  `10 % 3 => 1` `50 % 2 => 0` `49 % 2 => 1`

- Connected with floor division by this identity:

  `((x // y) * y) + (x % y) => x`

Comparative Operaters

- `==`, `!=`, `<`, `>`, `<=`, `>=`
- Work as you'd expect, return True or False.
- (`<`, `>`, etc.) can be used on strings, to compare them in dictionary order:

  `'a' < 'b' => True` `'1' < '2' => True` `'11' < '1000' => False`

Boolean Operators

- `and`, `or`, `not` — all reserved words
- Take two bools and return a bool.

Evaluation of Operands

For `and` and `or`, the interpreter will only evaluate both operands if it has to:

- With `P or Q`, if P is `True`, then Q will not be evaluated, because the result is already known.
- With `P and Q`, if P is `False`, then Q will not be evaluated either.

Precedence

- Comparative operators have higher precedence than arithmetic operators.
- Arithmetic operators have higher precedence than boolean operators.
- For boolean operators, `not` has higher precedence than `and`, which has higher precedence than `or`

Handouts & Assignments

- Handout 3 - Expressions, Operators, & Precedence