Introduction to Programming in Pyret

Programming languages involve different *datatypes*, such as Numbers, Strings, and Booleans.

- Numbers are values like `1`, `0.4`, `1/3`, and `-8261.003`.
  - Numbers are *usually* used for quantitative data and other values are *usually* used as categorical data.
  - In Pyret, any decimal *must* start with a 0. `0.22` is valid, but `.22` is not.
- Strings are values like `"Emma"`, `"Rosanna"`, `"Jen and Ed"`, or even `"08/28/1980"`.
  - In Pyret, all strings *must* be surrounded in quotation marks.
- Booleans are either `true` or `false`.

Operators (like `+`, `-`, `*`, `<`, etc.) work the same way in Pyret that they do in math.

- Operators are written between values, for example: `4 + 2`.
- In Pyret, operators must always have a space around them. `4 + 2` is valid, but `4+2` is not.
- If an expression has different operators, parentheses must be used to show order of operations. `4 + 2 + 6` and `4 + (2 * 6)` are valid, but `4 + 2 * 6` is not.

Applying Functions also works the way it does in math. The function name is first, followed by a list of **arguments** in parentheses.

- In math this could look like `f(5)` or `f(g(10, 4))`.
- In Pyret this could look like `star(50, "solid", "red")`.
- There are many other Pyret functions, for example `num-sqr`, `num-sqrt` `triangle`, `star`, `string-repeat`, etc.

Functions have **contracts**, which help explain how a function should be used. Every contract has three parts:

- The *Name* of the function - literally, what it's called.
- The *Domain* of the function - what *types of values* the function consumes, and in what order.
- The *Range* of the function - what *type of value* the function produces.

Value Definitions (like `x = 4`, or `y = 9 + 6`) also work the way they do in math. Every value definition starts with a *name*, followed by an equals sign, and then an expression. Once a value is defined, it can be refered to by name.