Mystery Language: Numbers

We don't intend for solving these mysteries to take too long. If you have spent five hours on any part of this assignment please stop and check in with the module instructors or the TAs.

We are developing a new language, one feature at a time. We try multiple prototypes for each feature, but keep losing the source code. Your task is to tell the different variants apart.

Documentation

This documentation includes all of the syntactic constructs we have implemented so far. The latest features to be added or modified are highlighted.

Let

Syntax	Behavior
`<name> = <expr>`	Evaluates `expr`, binds the result to `name`. Cannot be the last statement in a block.

Block

Syntax	Behavior

`block:`
`  ...`
`end`
	Evaluates each of their statements in order, and evaluates to the value of the final expression in the block.

Parentheses

Syntax	Behavior
`(<expr>)`	Parentheses are used for grouping expressions. They do not otherwise change the meaning of the program.

Number Literals

Syntax	Behavior
`9000.1` `-42`	Evaluation of a number literal yields a numeric value. Integer and decimal numbers are supported.

Binary Arithmetic Operations

Syntax	Behavior

`<expr> + <expr>`
`<expr> - <expr>`
`<expr> * <expr>`
`<expr> / <expr>`
	These arithmetic operations raise exceptions if their arguments do not evaluate to numbers. Otherwise, they perform the specified operation. (Note: operators like + must be surrounded by whitespace.)

Today's Feature: Numbers

Focus on number representation and results of numeric operations.

We had our interns cobble together some prototypes. Unfortunately they left, and we lost track of which is which.

Setup

To install the languages, follow the installation guide.

The language variants for this assignment are:
`#lang Numbers1` (Core)
`#lang Numbers2` (Core)
`#lang Numbers3` (Core)

To run them all at once, use:
`#lang Numbers`

Submission

The version of Racket and the Mystery Languages should be 7.0.

Submit your classifiers and explanation via the corresponding Moodle form.