A program is a sequence of expressions and statements.

- **Expression Forms**

  - **Literal Value**
    
    function-name : function by name from a definition or from our language
    ➔ true! false! boolean
    ➔ (list literal-value etc) ; list
    ➔ inserted/pasted image
    ➔ number as decimal or fraction
    ➔ "characters" ; test
  
  - **Variable Reference**
    ➔ variable-name ; from a definition or from our language
  
  - **Function Call**
    ➔ (function-name argument-expression etc)
  
  - **Statement Forms**

- **Definition of a Variable or Function**

  (define variable-name value-expression)
  ➔ (define (function-name parameter-name etc) "header" body-expression)

- **Assertion / Test**

  (same! expression expression) ➔ (true! condition-expression)
  ➔ (false! condition-expression)

- **Reveal Algebraic Evaluation**

  ... (function-name literal-value etc) ... ; except literal lists

  - For a function from a definition: copy its body and substitute the arguments in place of the parameter names wherever the names occur in the body.
  - If number of arguments doesn't match number of parameters: an error

  - For combine or map: match its rule's first pattern to determine f a b c ...

  - Then substitute those into its rule's second pattern (the "template")
  - If the expression doesn't match its pattern: an error

  (map f (list a b c etc)) ➔ (list (f a) (f b) (f c) etc)
  (combine f (list a b c etc)) ➔ (f a b c etc)

  - For any other function from our language: substitute directly computed value.
  - If wrong number or type of arguments for that function: an error

  ... variable-name ...
  ➔ literal-value
  ➔ Substitute the value that was computed when the variable was defined.

- **Function Design**

- **Goal Example**

  (same! (function-name example-argument etc) literal-result)

- **Full Design**

  (same! (function-name example-argument etc) fully-generalizable-expression)

  The generalizable expression contains only uses the argument(s) as-is.

- **Partial Design**

  (same! (function-name example-argument etc) partially-general-expression)

  The partially general expression is not fully generalizable, but not just literal.

- **Equality Predicate**

  (true! (same? (+ 1 1) 2)) ➔ (false! (same? (+ 1 1) 3))

- **Type Predicates**

  (true! (function? flip)) ➔ (true! (list? (list "z" #t)))
  (true! (boolean? #false)) ➔ (true! (number? -12))
  (true! (text? "Hi")) ➔ (true! (image? #f))

- **Function Predicates**

  (true! (unary? flip)) ➔ (false! (binary? flip))

- **List Functions**

  (same! (list (filled-triangle 9) (zero? 0) (+ 2 3) "hi")
  ➔ (list #true 5 "hi")
  (same! (map - (list 3 1 7) (list -3 -1 -7))
  ➔ (list 3 1 7))
  (true! (empty? (list)))
  ➔ (same! (length (list #true 5 "hi") 4)
  (same! (first (list #true 5 "hi")))
  ➔ (same! (rest (list #true 5 "hi")))
  (same! (reverse (list #true 5 "hi")) (list "hi") #true #f))

- **Image Functions**

  (same! (mirror (hi)) ➔ (same! (scale-width (hi) 1.5))
  (same! (flip (hi)) ➔ (same! (wider (hi)))
  (same! (rotate 30) (hi)) ➔ (same! (thinner (hi)))
  (same! (clockwise (hi)) ➔ (same! (scale-height (hi) 1.5))
  (same! (anti-clockwise (hi)) ➔ (same! (taller (hi)))
  (same! (scale 1 1.5) (hi)) ➔ (same! (shorter (hi)))
  (same! (enlarge (hi)) ➔ (same! (flip (hi)))
  (same! (filled-triangle 9) ➔ (same! (triangle 9)
  (same! (filled-circle 9) ➔ (same! (circle 9))
  (same! (filled-square 9) ➔ (same! (square 9))
  (same! (filled-oval 9 15)) ➔ (same! (oval 9 15))
  (same! (filled-rectangle 9 15)) ➔ (same! (rectangle 9 15))
  (same! (width (oval 9 15)) 9)) ➔ (same! (height (oval 15) 15))

  (same! (above ● △) ➔ (same! (beside ● △)
  (same! (align-lefts ● △) ➔ (same! (align-tops ● △)
  (same! (align-rights ● △) ➔ (same! (align-bottoms ● △)

- **Numeric Functions**

  (same! (+ 2 10 3) 15) ➔ (same! (- 12) -12)
  (same! (* 2 10 3) 60) ➔ (same! (- 12 3) 9)
  (same! (/ 12 3) 4)

  (true! (zero? 0)) ➔ (false! (positive? -3))
  (true! (positive? 12)) ➔ (false! (positive? 0))

- **Text Functions**

  (same! (text-length "one") 3) ➔ (same! (text-join "Hi" = human!) "Hi human!"
  (same! (text->image "Hi")) ➔ (same! (text-list "Hi"))
  (list "Hi" "Hi" "Hi")