# Definitions

|  |  |
| --- | --- |
| Element | One element is a defined input of a function  f(x) = x+1; |
| Function | A function is a set of definitions that share the same identifier |
| Identifier | The name of the function. Used when defining and calling the function.  Identifier(x) = x+1; |

# Types

## Assignment Operators

|  |  |  |
| --- | --- | --- |
| Operator | Name | Description |
| = | Equals | Lazy assignment, calculated results are stored in memory. |
| <- | Gets | Lazy assignment, calculated results are not stored in memory. |

## Mathematical Operators

|  |  |
| --- | --- |
| Operator | Name |
| a+b | Addition |
| a-b | Subtraction |
| a\*b | Multiplication |
| a/b | Division |
| a%b | Modulus |
| \*\* | Exponentiation |

## Logical Operators

|  |  |
| --- | --- |
| Operator | Name |
| a&&b | And |
| a||b | Or |
| ! | Not |

## Relational Operators

|  |  |
| --- | --- |
| Operator | Name |
| a>b | Greater than |
| a<b | Less than |
| a>=b | Greater than or equal to |
| a<=b | Less than or equal to |
| a==b | Equal to |
| a!=b | Not equal to |

## List Operators

|  |  |
| --- | --- |
| Operator | Description |
| [a..b] | Generates a List of all integers between a and b inclusive |
| #a | Length of a list |
| a:b | Concatenation of two lists |
| ::a | Concatenation of a list |
| :+a | Sum of a finite function |
| :\*a | Product of a finite function |
| :&x | Returns true element if ALL elements are true |
| :|x | Returns true element if ANY element is true |

## Layout

|  |  |
| --- | --- |
| Operator | Description |
| /\*\* | Function documentation comment opener |
| /\* | Generic comment opener |
| \*/ | Generic comment closer |
| , | Separates arguments |
| | | Where, used to define variables in inputs |
| ; | End of function |
| ( |  |
| ) |  |

# Examples

#### Example 1: Fibonacci Sequence

The Fibonacci sequence is the sequence of numbers where each number being defined as the sum of the two previous numbers. The first 10 numbers of the sequence are: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34.

/\*\*fibonacci sequence is declared recursively\*/

/\*declaring known fib numbers\*/

fib(0) <- 0;

fib(1) <- 1;

/\*declaring unknown fib numbers recursively\*/

fib(x) = fib(x-1) + fib(x-2);

#### Example 2: FizzBuzz

FizzBuzz is a maths game where each multiple of 3 is replaced with fizz, each multiple of 5 is replaced with buzz, and multiples of both 3 and 5 are replaced with fizzbuzz. The first 20 values of the sequence are: 1, 2, fizz, 4, buzz, fizz, 7, 8, fizz, buzz, 11, fizz, 13, 14, fizzbuzz, 16, 17, fizz ,19, buzz.

/\*\*defining fizz recursively\*/

fizz(x<3) <- "";

fizz(3) <- "fizz";

fizz(x) <- fizz(x-3);

/\*\*defining buzz recursively\*/

buzz(x<5) <- "";

buzz(5) <- "buzz";

buzz(x) <- buzz(x-5);

/\*\*the output is the concatenation of both fizz and buzz\*/

stringout(x) = fizz(x)+buzz(x);

/\*\*if the cardinality of out for x is 0, return x, else return the concatenation of out\*/

singleout(x|stringout(x) == "") = x;

singleout(x) = stringout(x);

fizzbuzz(x>0) <- mapsingleout([0..x]);