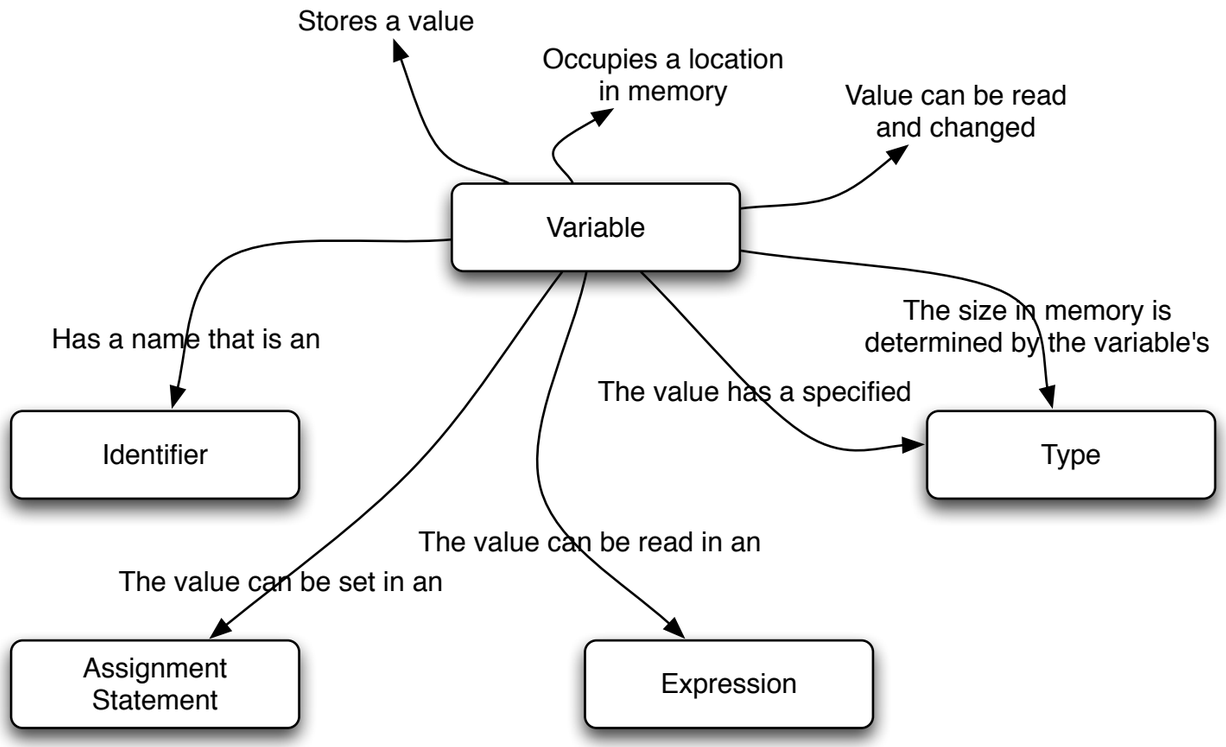
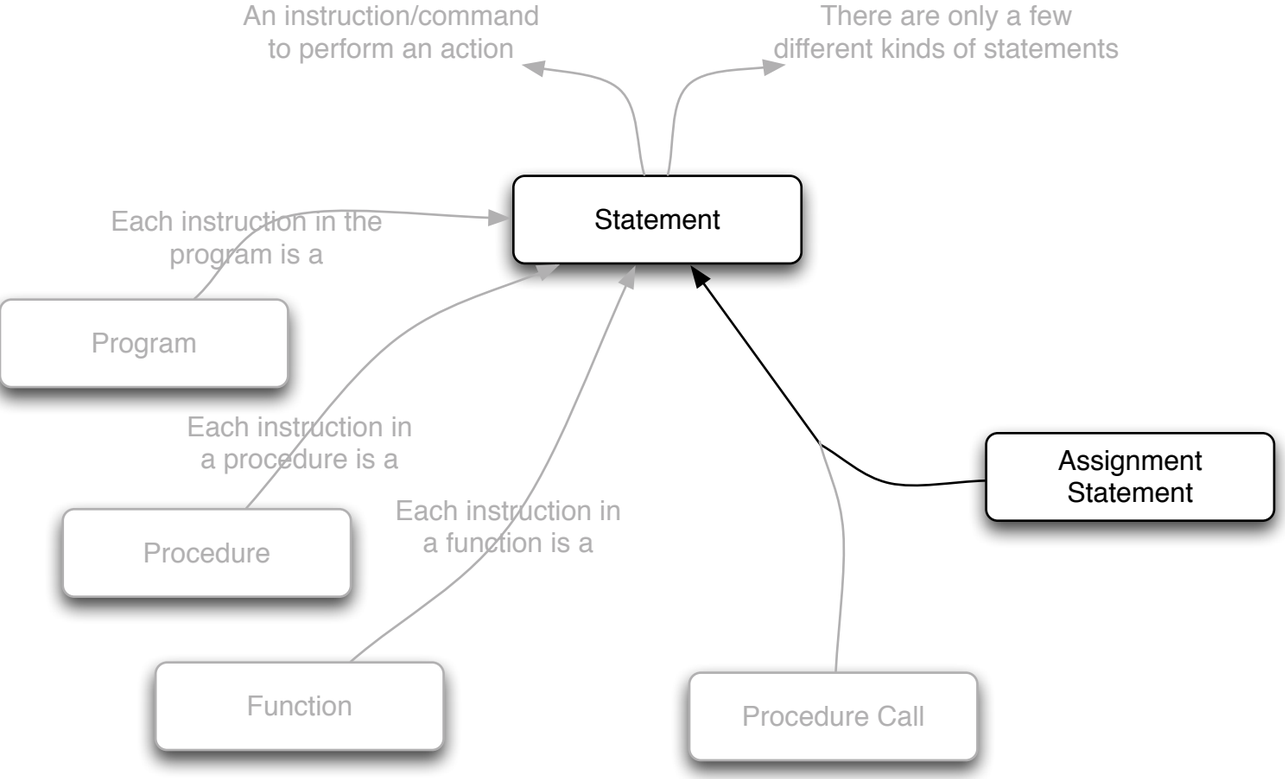
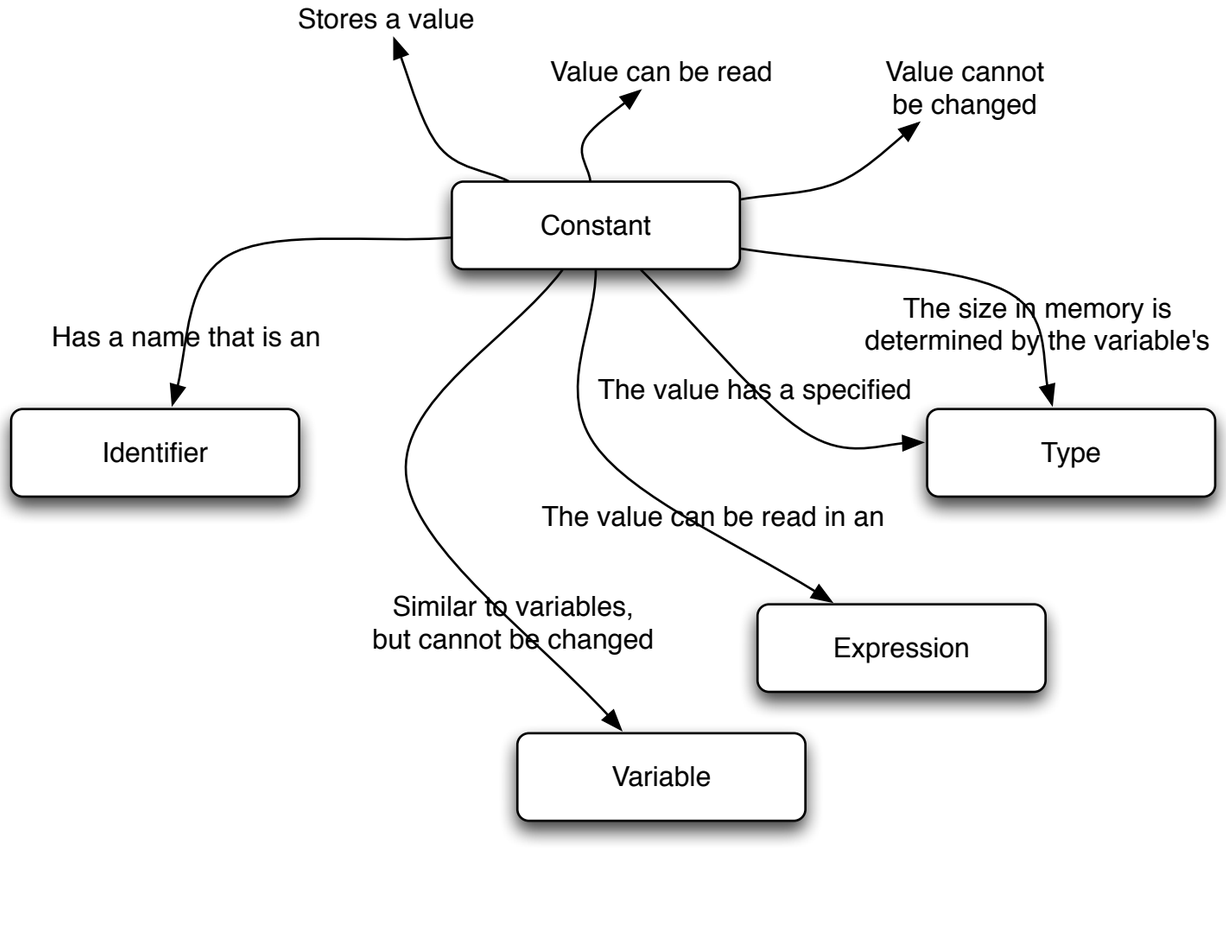
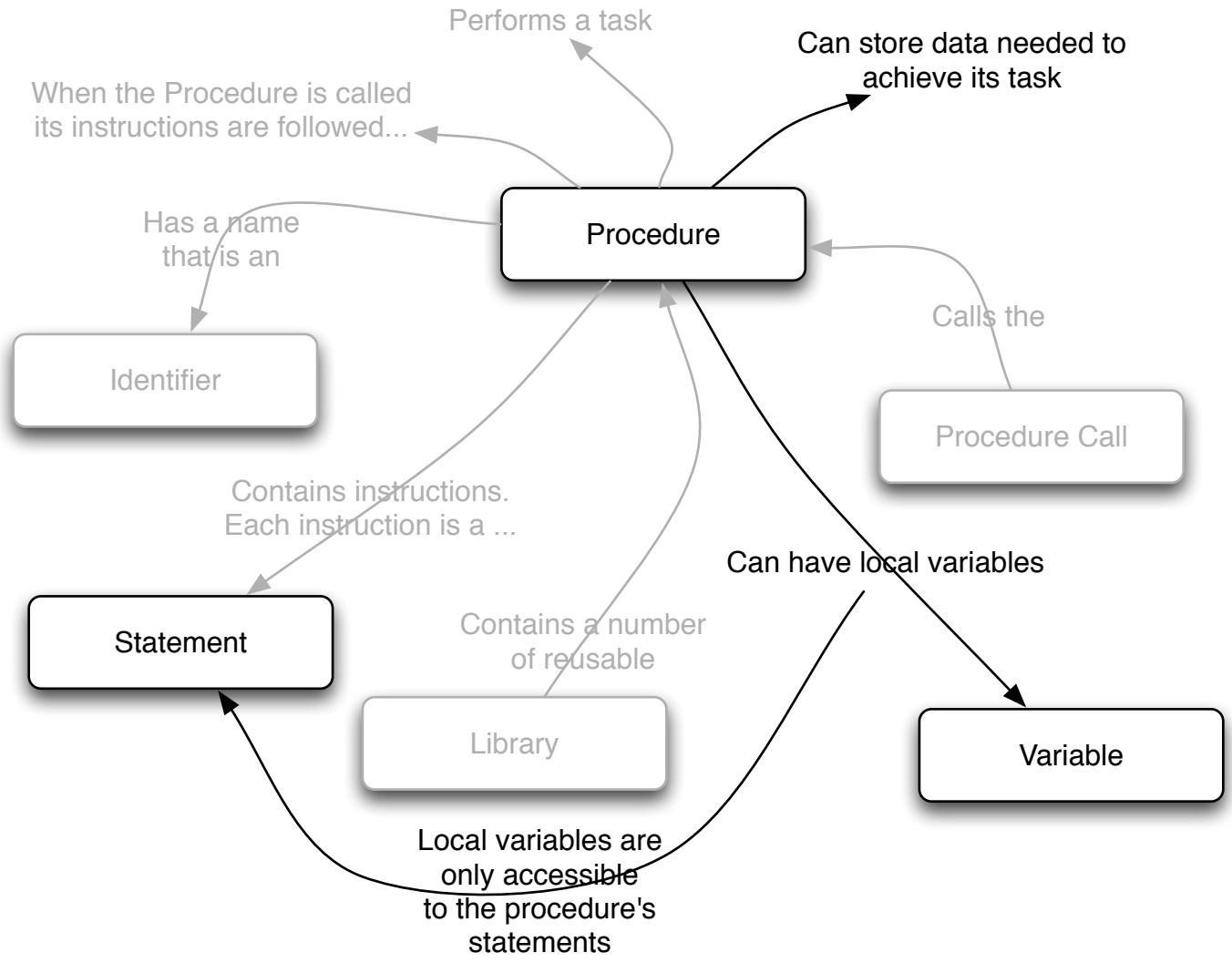
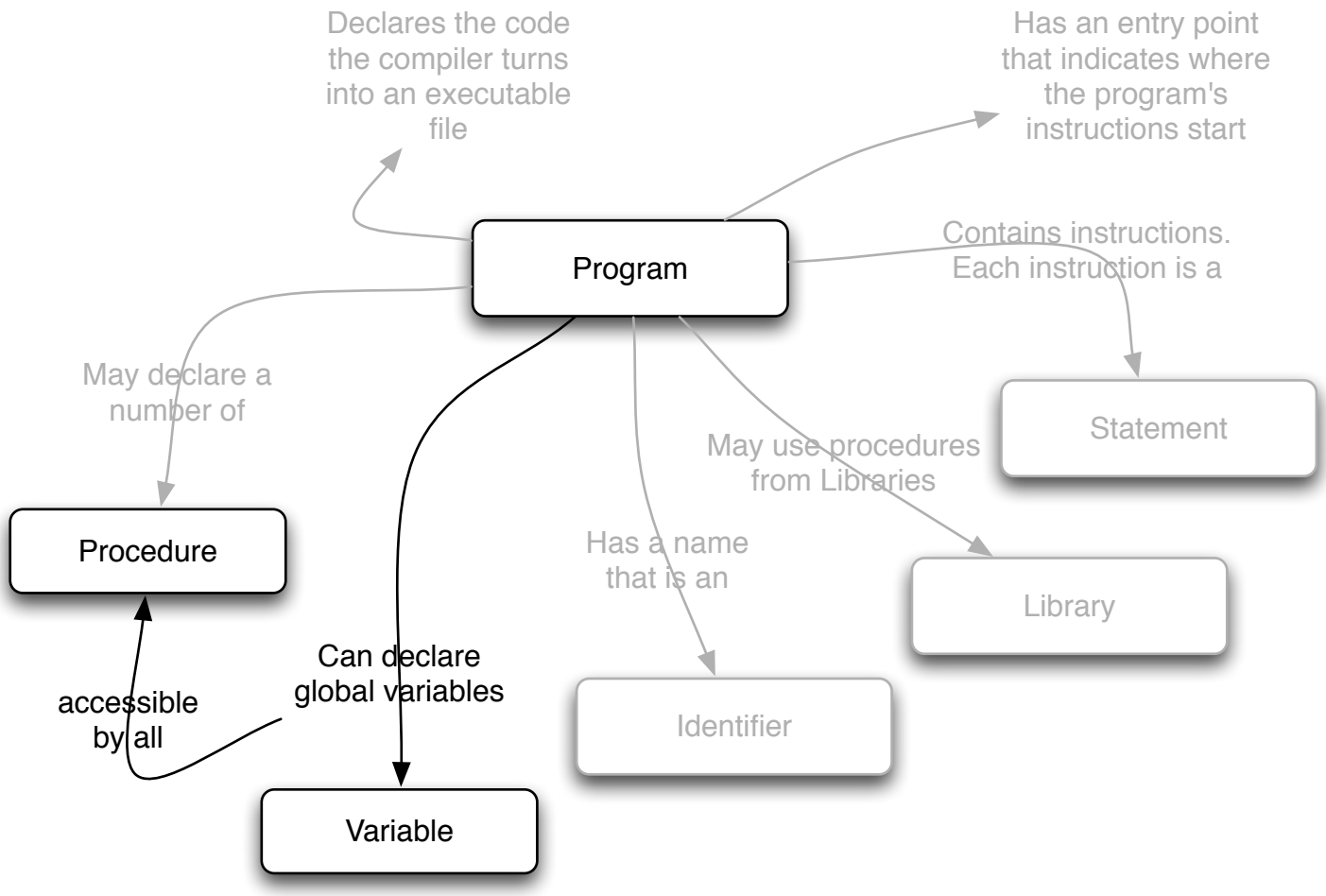
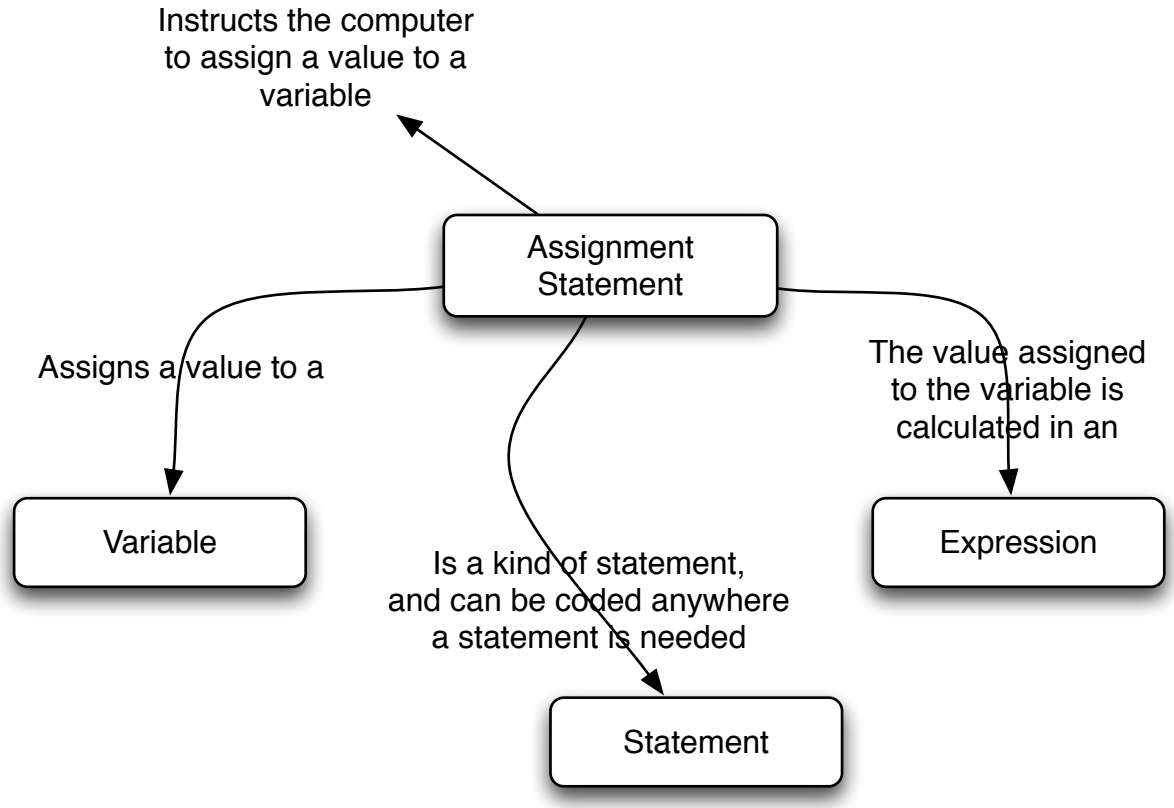

Stores a value
Occupies a location in memory
Value can be read and changed
Variable
Has a name that is an
The size in memory is determined by the variable's
The value has a specified
Identifier
Type
The value can be read in an
The value can be set in an
Assignment Statement
Expression

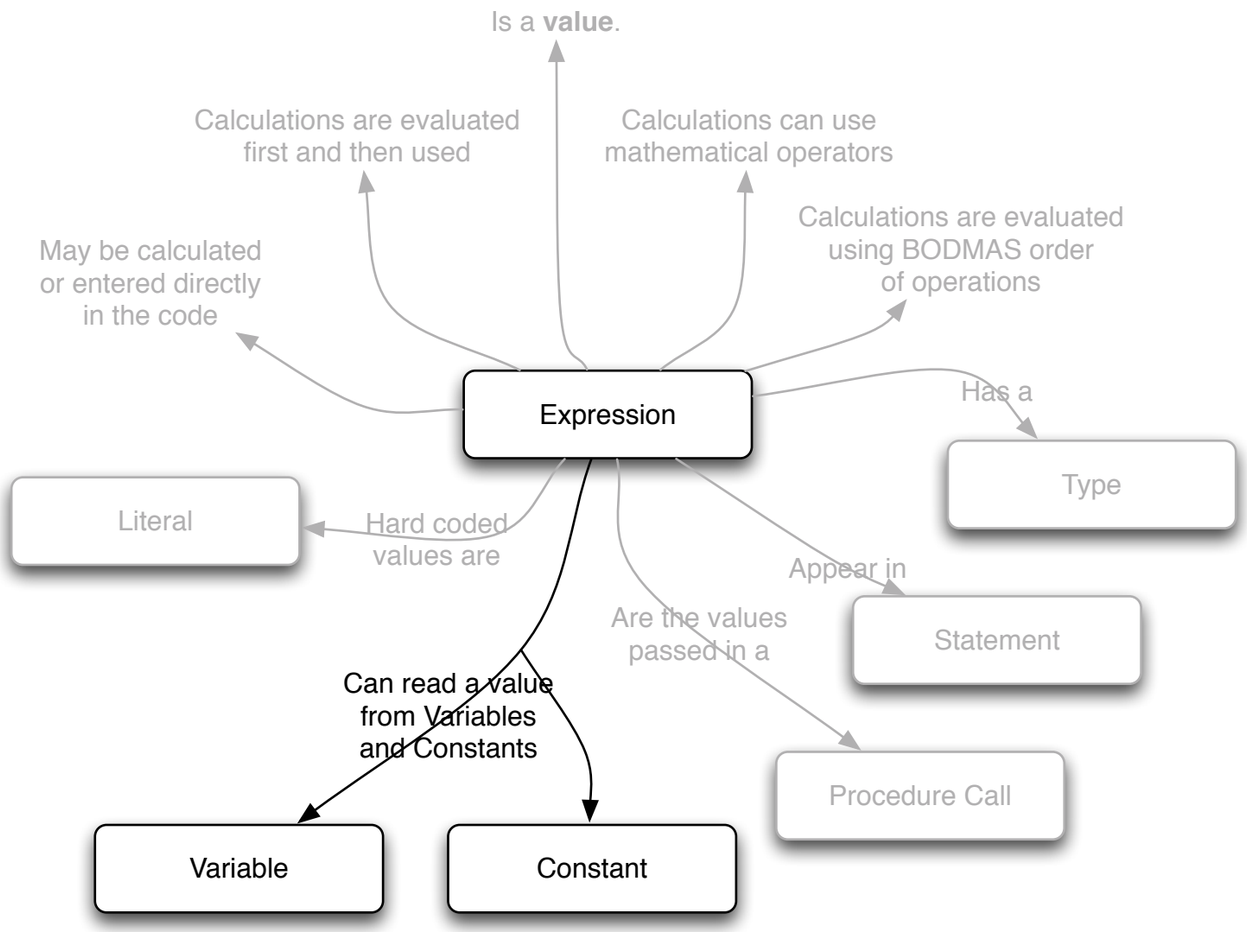

An instruction/command
to perform an action
There are only a few
different kinds of statements
Statement
Each instruction in the
program is a
Program
Each instruction in
a procedure is a
Procedure
Each instruction in
a function is a
Function
Assignment
Statement
Procedure Call

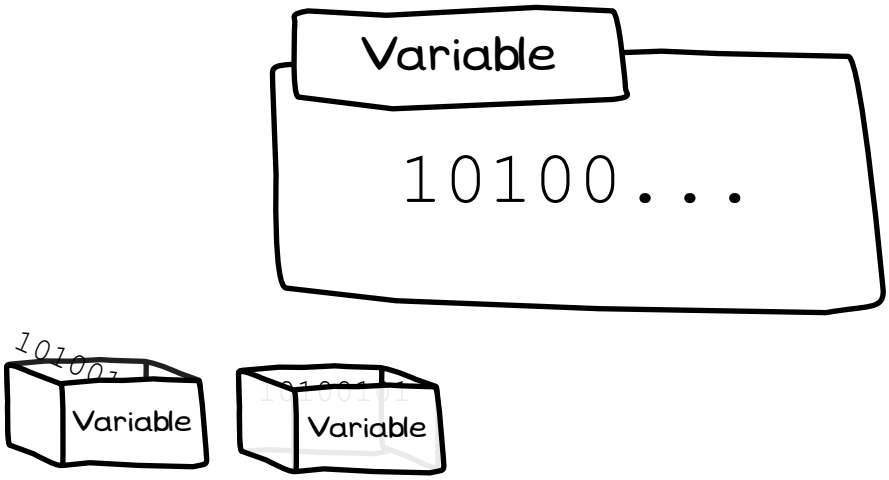

Stores a value
Value can be read
Value cannot be changed
Constant
Has a name that is an
Identifier
The size in memory is determined by the variable's
Type
The value has a specified
The value can be read in an
Expression
Similar to variables, but cannot be changed
Variable

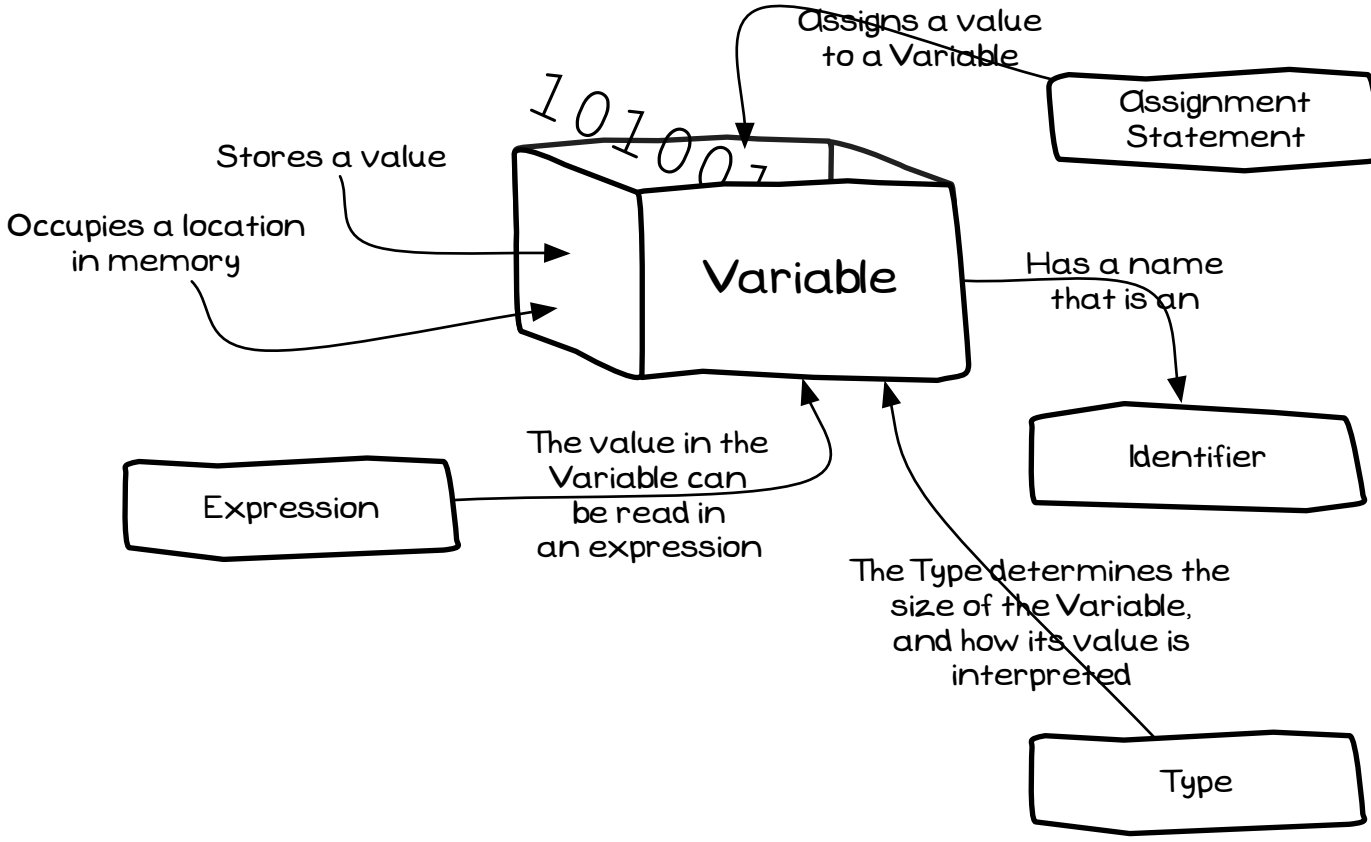

Performs a task
Can store data needed to achieve its task
When the Procedure is called its instructions are followed...
Has a name that is an
Procedure
Identifier
Calls the
Procedure Call
Contains instructions. Each instruction is a ...
Can have local variables
Statement
Contains a number of reusable
Library
Variable
Local variables are only accessible to the procedure's statements

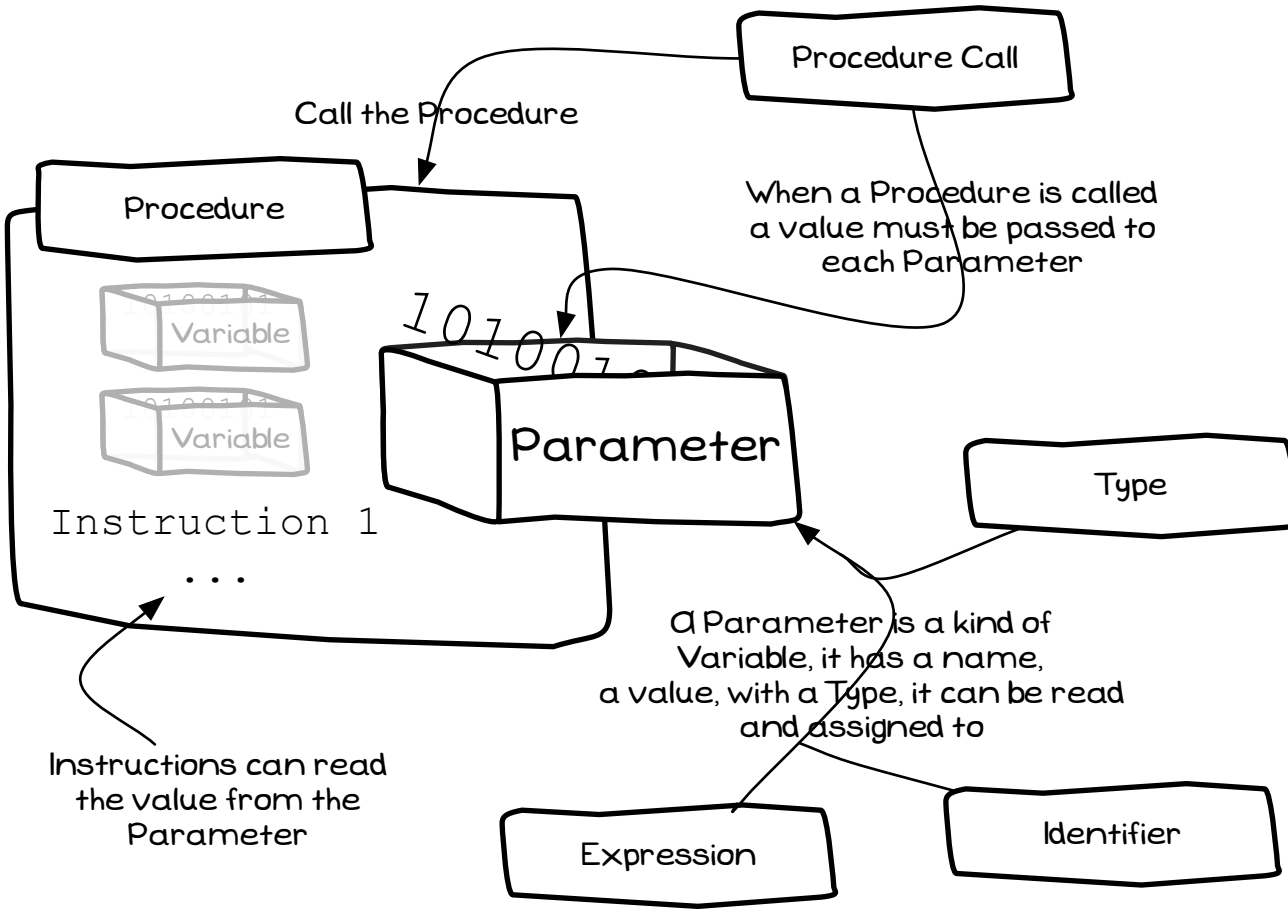

Declares the code the compiler turns into an executable file
Has an entry point that indicates where the program's instructions start
Program
Contains instructions. Each instruction is a
Statement
May declare a number of
Procedure
May use procedures from Libraries
Library
Has a name that is an
Identifier
Can declare global variables
accessible by all
Variable

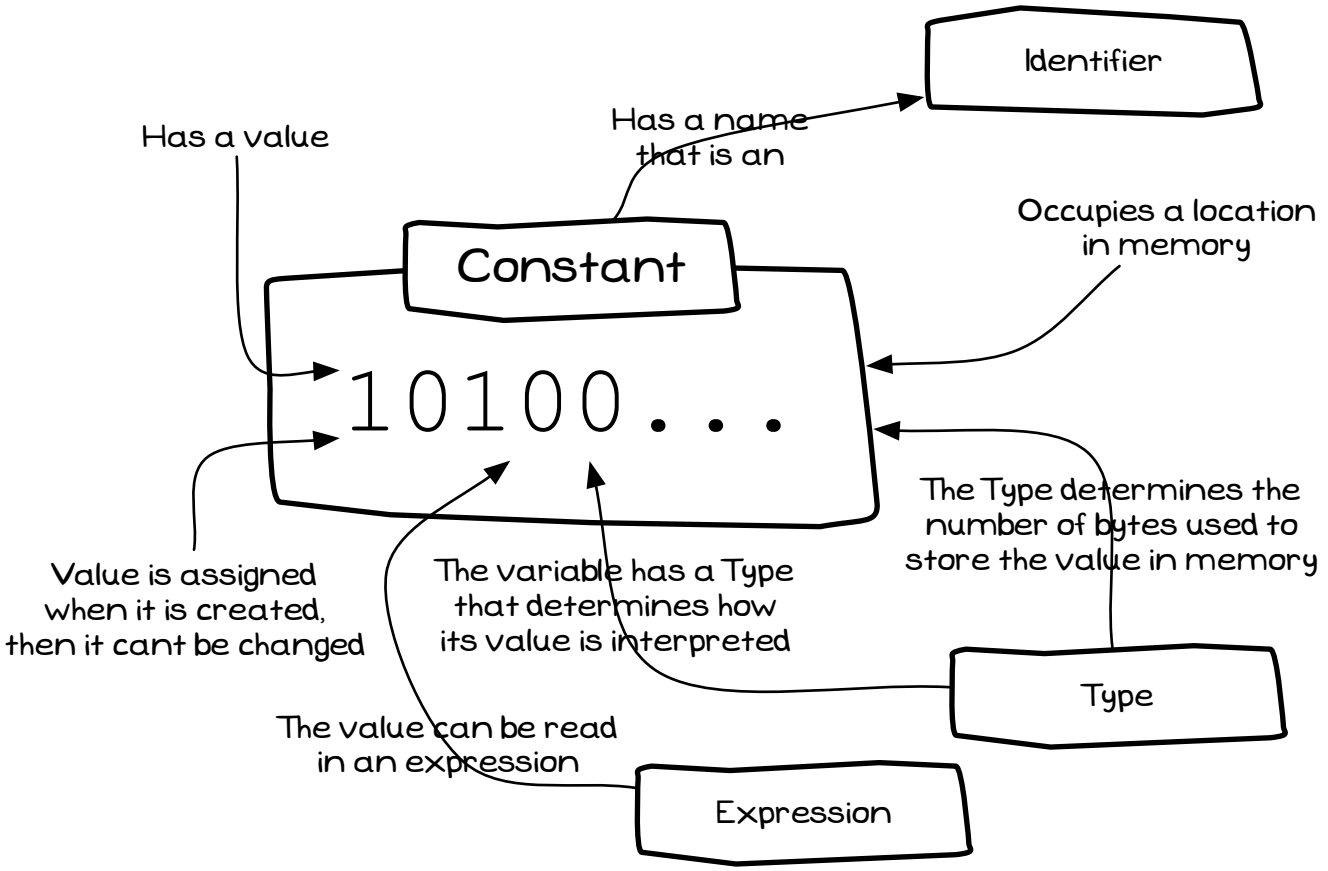

Instructs the computer to assign a value to a variable
Assignment Statement
Assigns a value to a
The value assigned to the variable is calculated in an
Variable
Is a kind of statement, and can be coded anywhere a statement is needed
Expression
Statement


Is a **value**.
Calculations are evaluated first and then used
Calculations can use mathematical operators
Calculations are evaluated using BODMAS order of operations
May be calculated or entered directly in the code
Expression
Has a
Type
Literal
Hard coded values are
Appear in
Statement
Are the values passed in a
Procedure Call
Can read a value from Variables and Constants
Variable
Constant

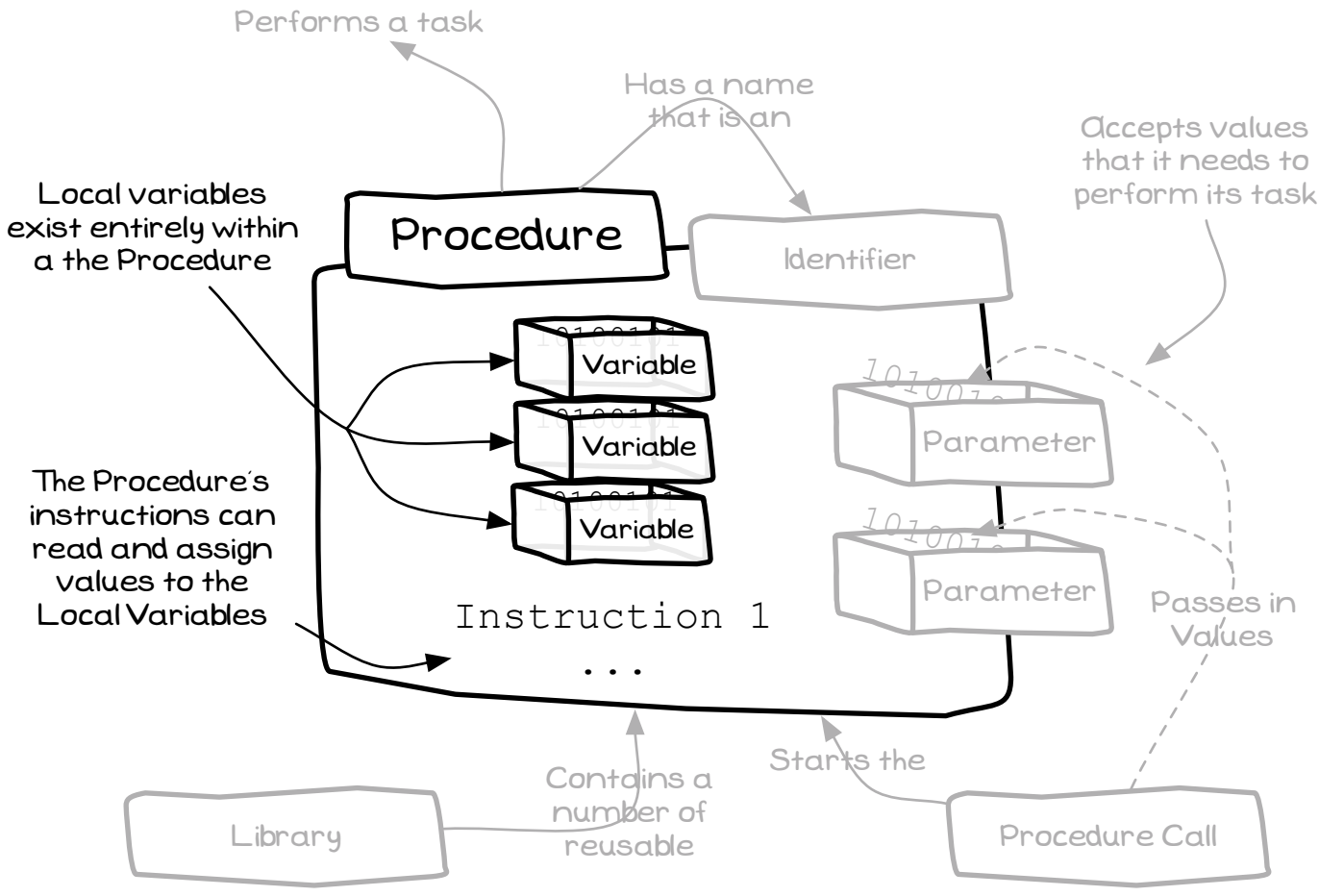

Variable
10100...
Variable
Variable

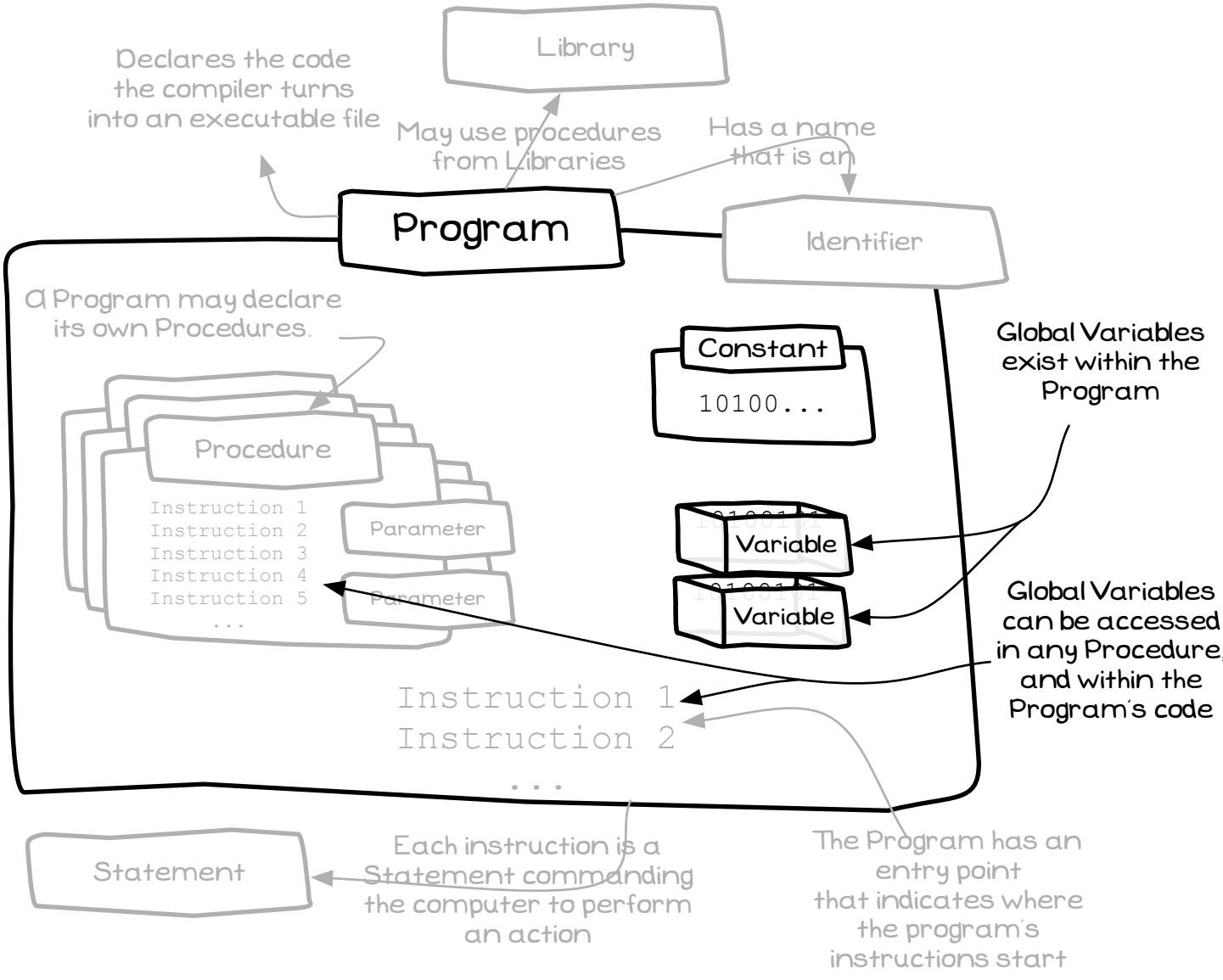

Assigns a value to a Variable
Assignment Statement
Stores a value
Occupies a location in memory
Variable
Has a name that is an
Identifier
Expression
The value in the Variable can be read in an expression
The Type determines the size of the Variable, and how its value is interpreted
Type

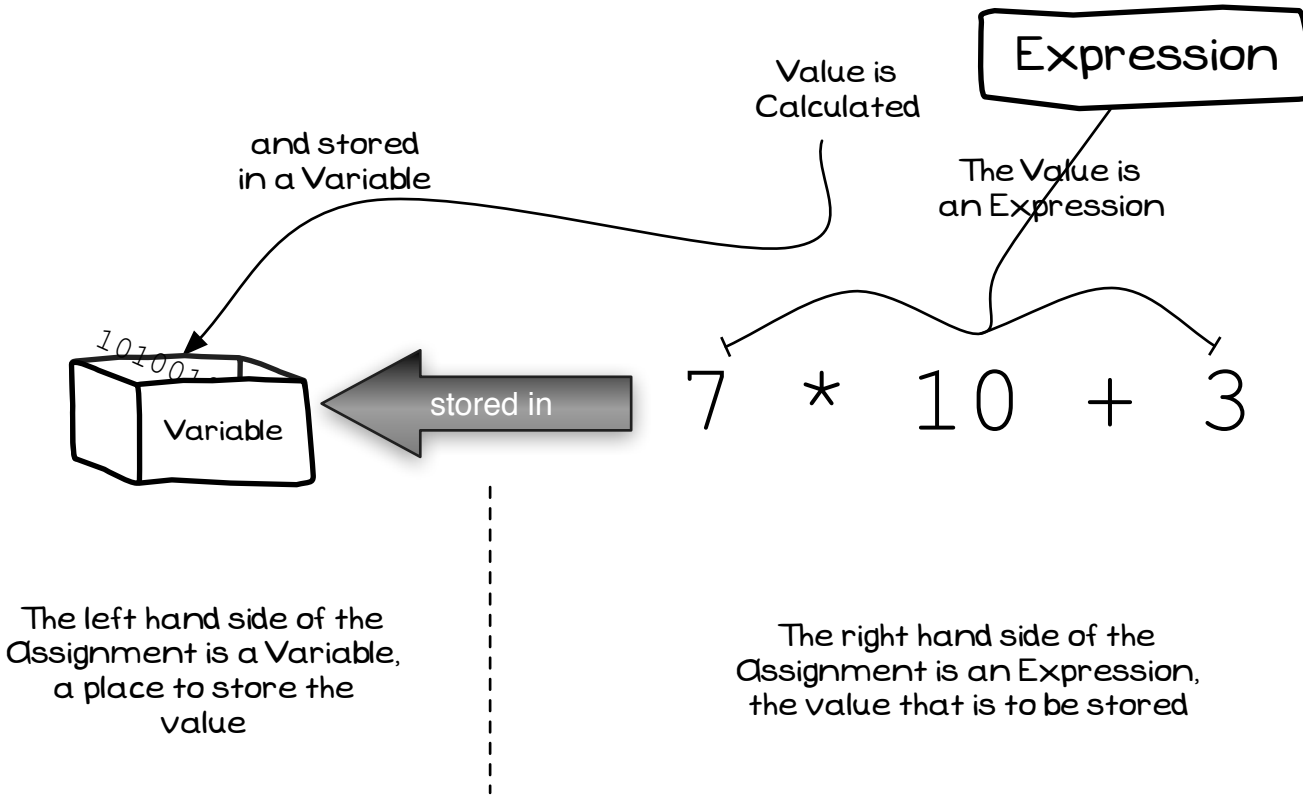

Procedure Call
Call the Procedure
Procedure
When a Procedure is called
a value must be passed to
each Parameter
Variable
Variable
Parameter
Type
Instruction 1
...
A Parameter is a kind of
Variable, it has a name,
a value, with a Type, it can be read
and assigned to
Instructions can read
the value from the
Parameter
Expression
Identifier

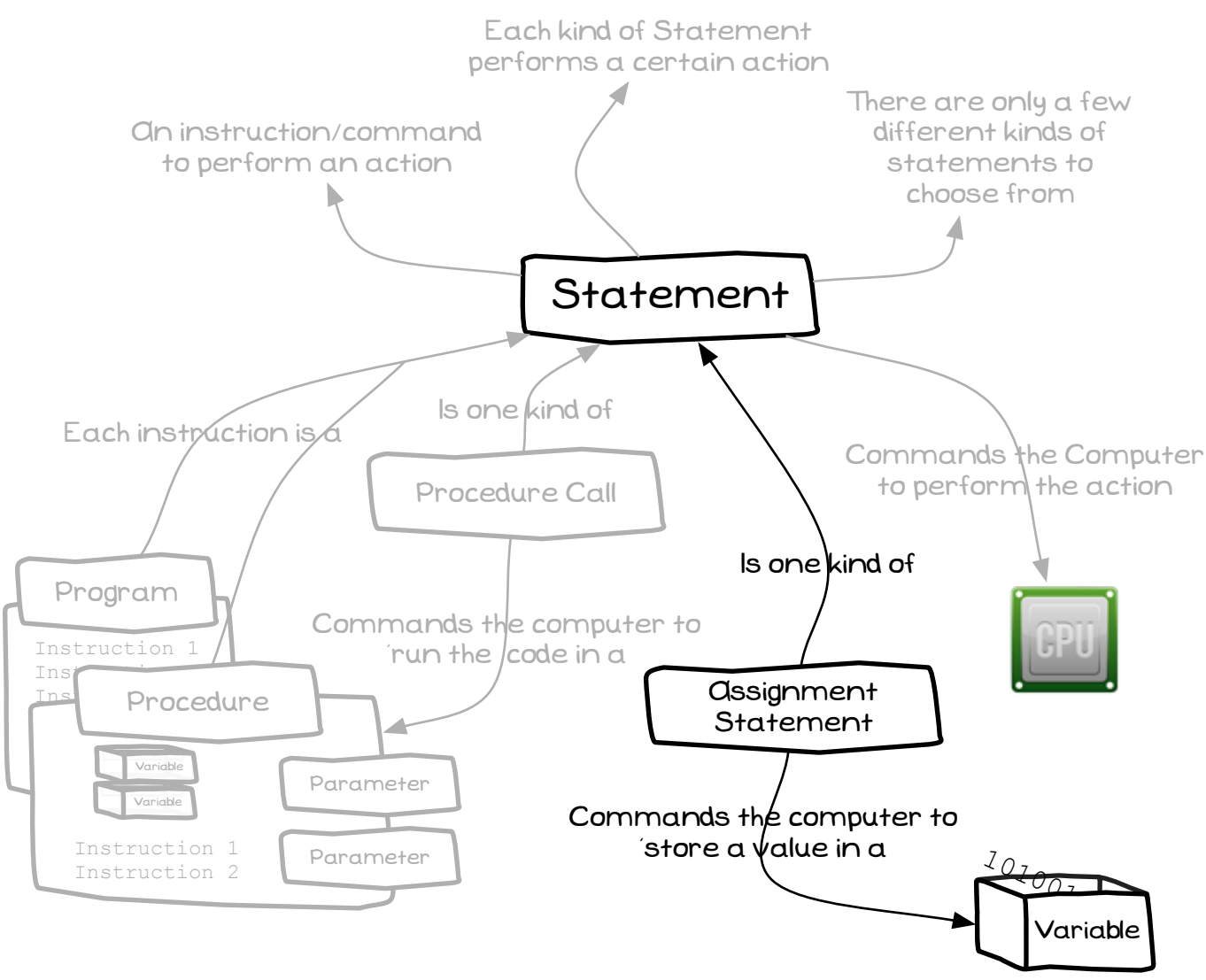

Identifier
Has a value
Has a name
that is an
Occupies a location
in memory
Constant
10100...
The Type determines the
number of bytes used to
store the value in memory
Value is assigned
when it is created,
then it cant be changed
The variable has a Type
that determines how
its value is interpreted
Type
The value can be read
in an expression
Expression

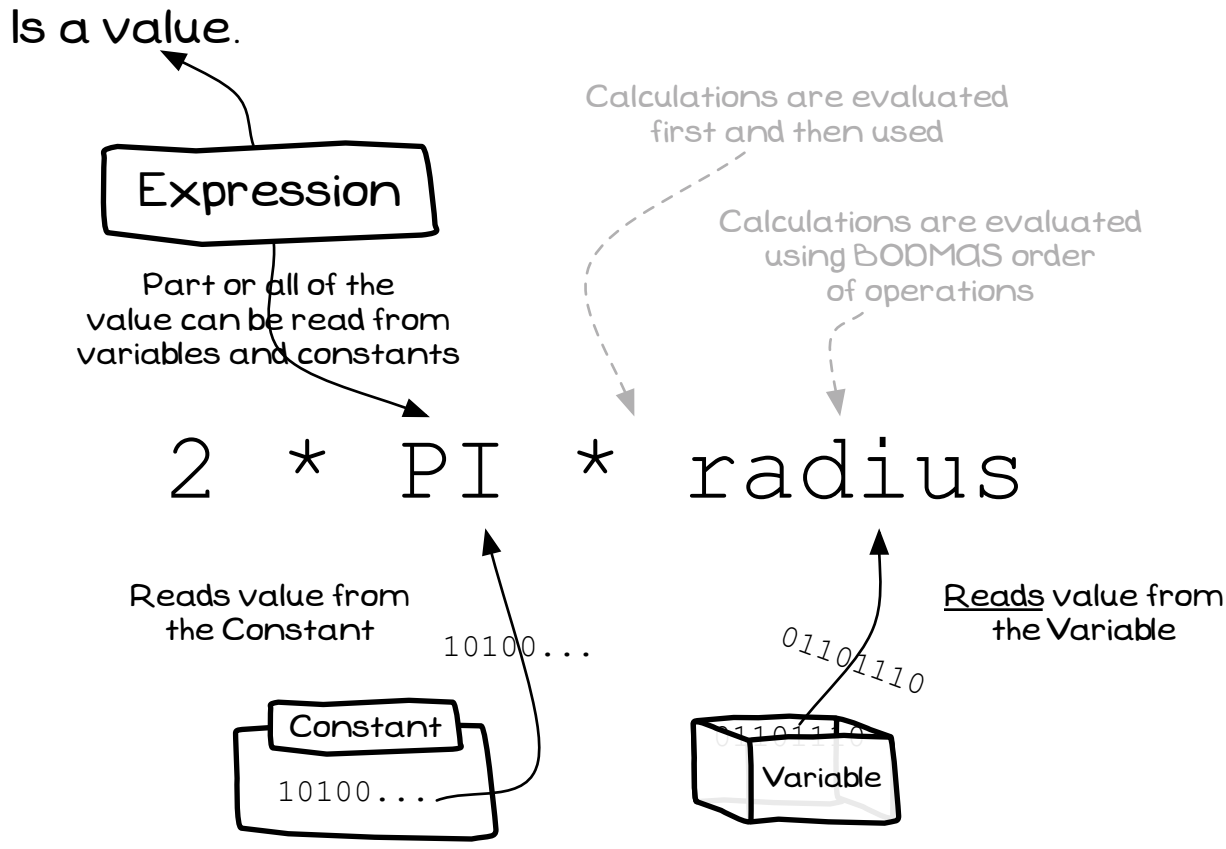

Constant

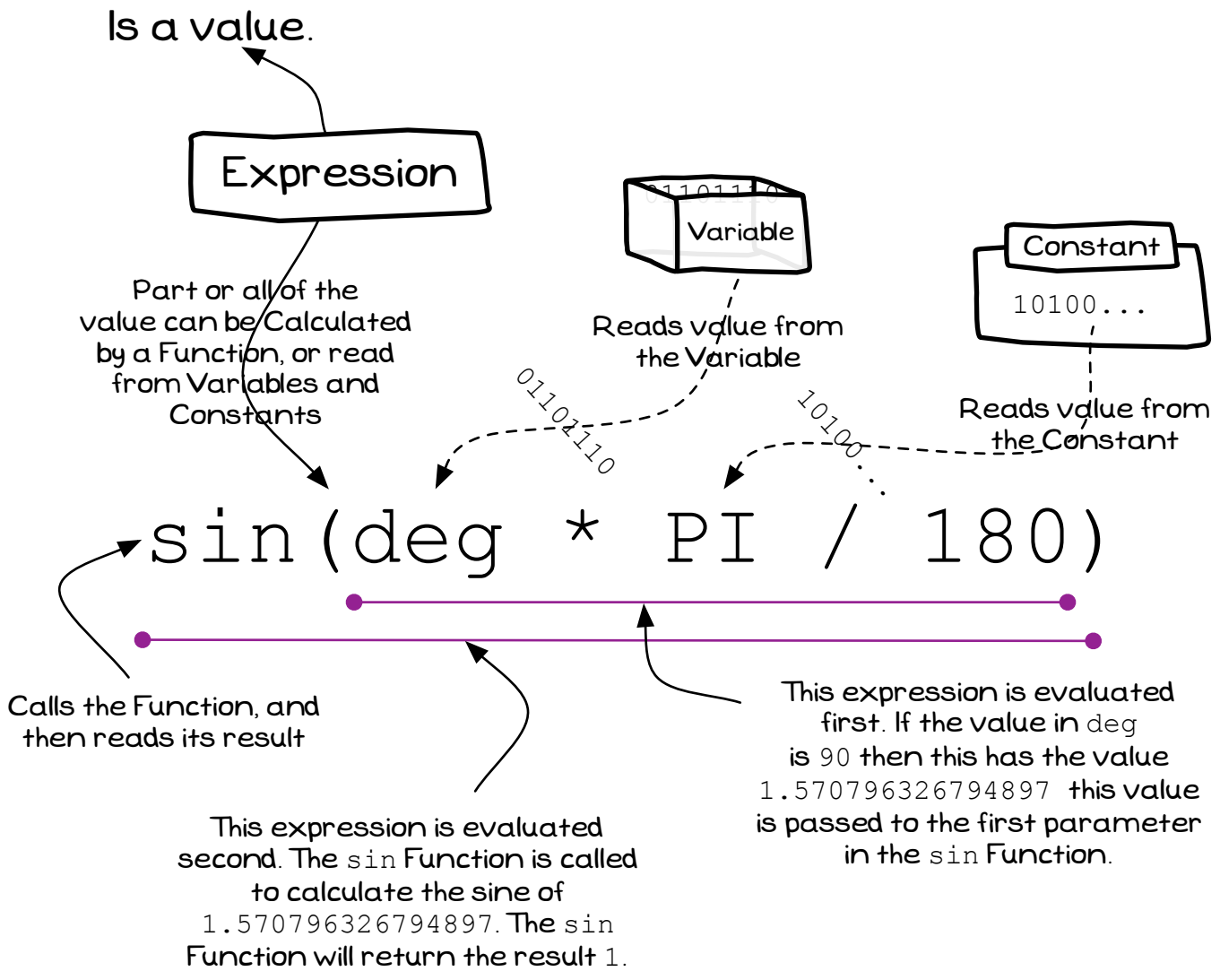

Performs a task
Has a name that is an
Accepts values that it needs to perform its task
Local variables exist entirely within a the Procedure
Procedure
Identifier
Variable
Variable
Variable
Parameter
Parameter
The Procedure's instructions can read and assign values to the Local Variables
Instruction 1
...
Passes in Values
Contains a number of reusable
Starts the
Library
Procedure Call

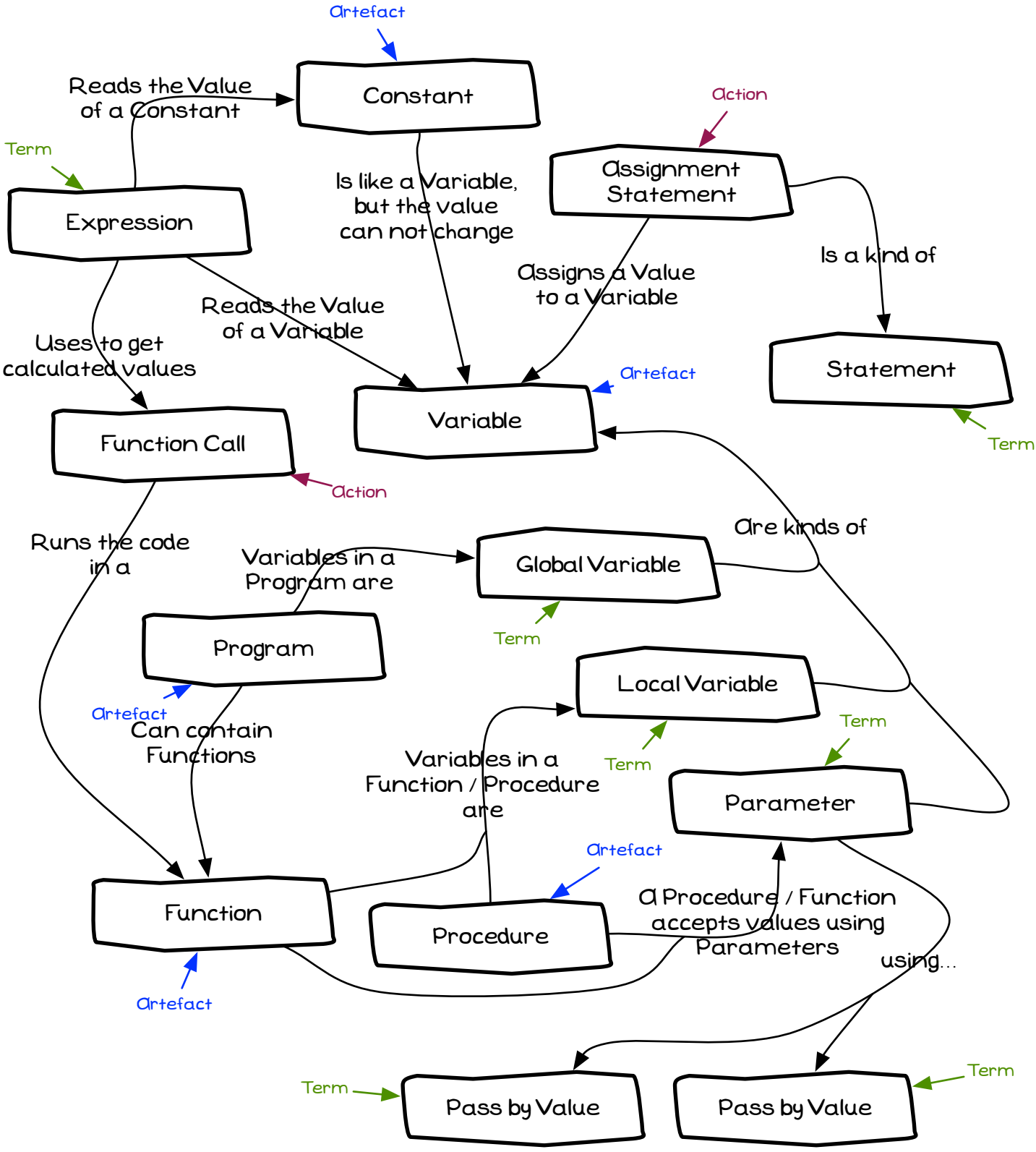

Library
Declares the code the compiler turns into an executable file
May use procedures from Libraries
Has a name that is an
Program
Identifier
A Program may declare its own Procedures.
Constant
10100...
Global Variables exist within the Program
Procedure
Instruction 1
Instruction 2
Instruction 3
Instruction 4
Instruction 5
...
Parameter
Parameter
Variable
Variable
Global Variables can be accessed in any Procedure, and within the Program's code
Instruction 1
Instruction 2
...
Statement
Each instruction is a Statement commanding the computer to perform an action
The Program has an entry point that indicates where the program's instructions start

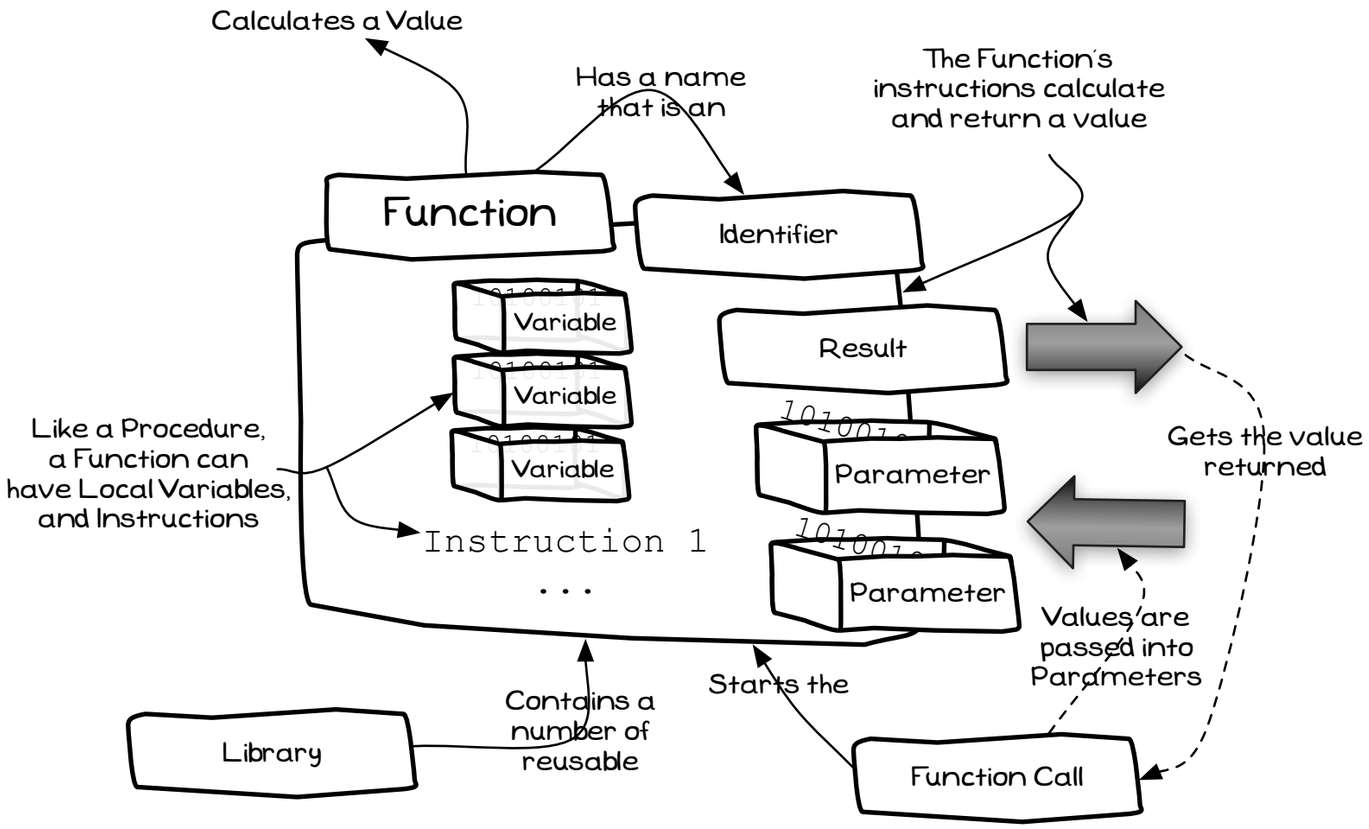

Expression
Value is Calculated
and stored in a Variable
The Value is an Expression
Variable
stored in
7 * 10 + 3
The left hand side of the Assignment is a Variable, a place to store the value
The right hand side of the Assignment is an Expression, the value that is to be stored

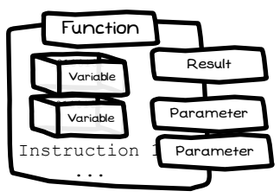

Each kind of Statement performs a certain action
An instruction/command to perform an action
There are only a few different kinds of statements to choose from
Statement
Each instruction is a
Is one kind of
Commands the Computer to perform the action
Procedure Call
Program
Is one kind of
Commands the computer to 'run the 'code in a
Procedure
Assignment Statement
Variable
Variable
Parameter
Parameter
Instruction 1
Instruction 2
Commands the computer to 'store a value in a
Variable

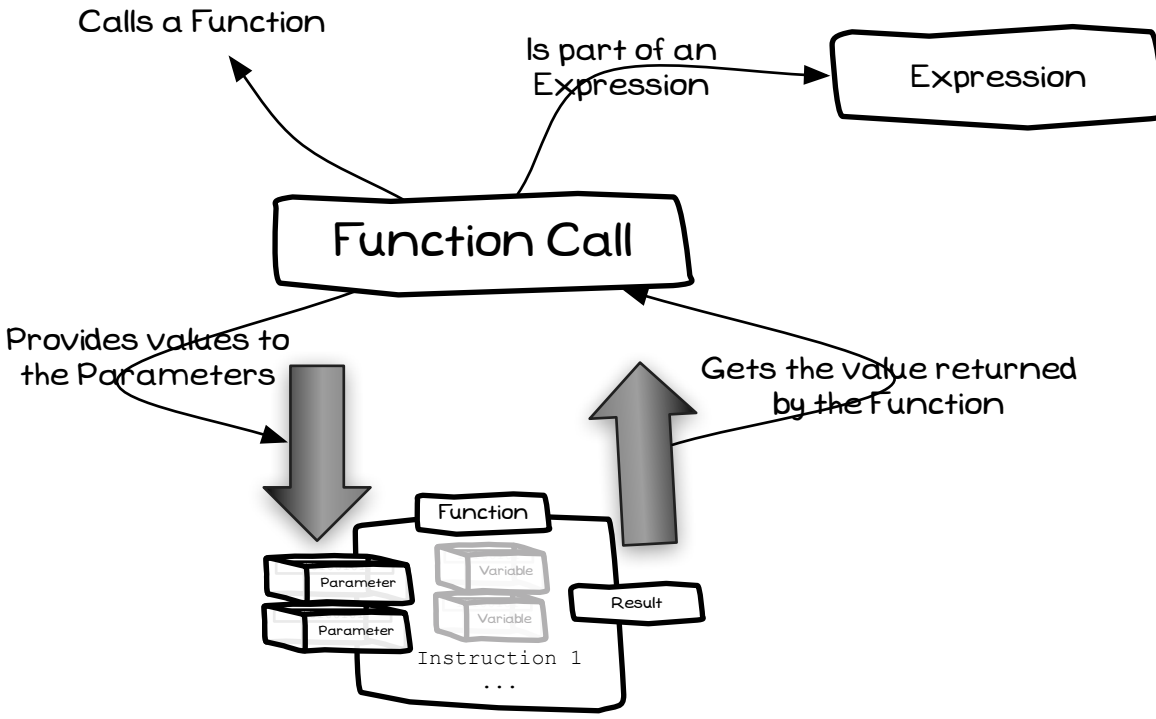

Is a value.
Expression
Part or all of the value can be read from variables and constants
Calculations are evaluated first and then used
Calculations are evaluated using BODMAS order of operations
2 * PI * radius
Reads value from the Constant
10100...
Constant
10100...
01101110
Variable
Reads value from the Variable

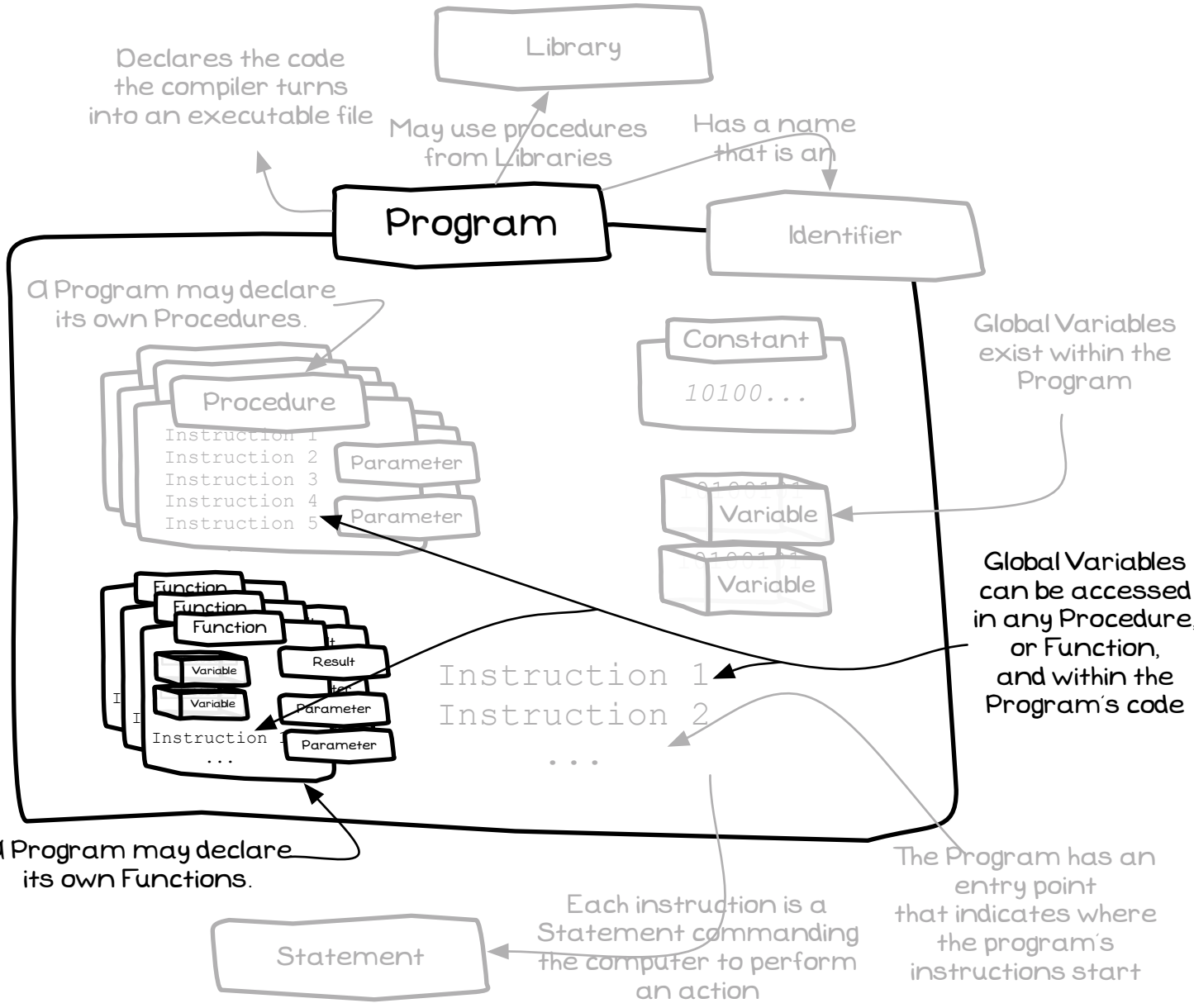

Is a value.
Expression
Variable
Constant
10100...
Part or all of the value can be Calculated by a Function, or read from Variables and Constants
Reads value from the Variable
Reads value from the Constant
01101110
10100...
sin(deg * PI / 180)
Calls the Function, and then reads its result
This expression is evaluated second. The sin Function is called to calculate the sine of 1.570796326794897. The sin Function will return the result 1.
This expression is evaluated first. If the value in deg is 90 then this has the value 1.570796326794897 this value is passed to the first parameter in the sin Function.

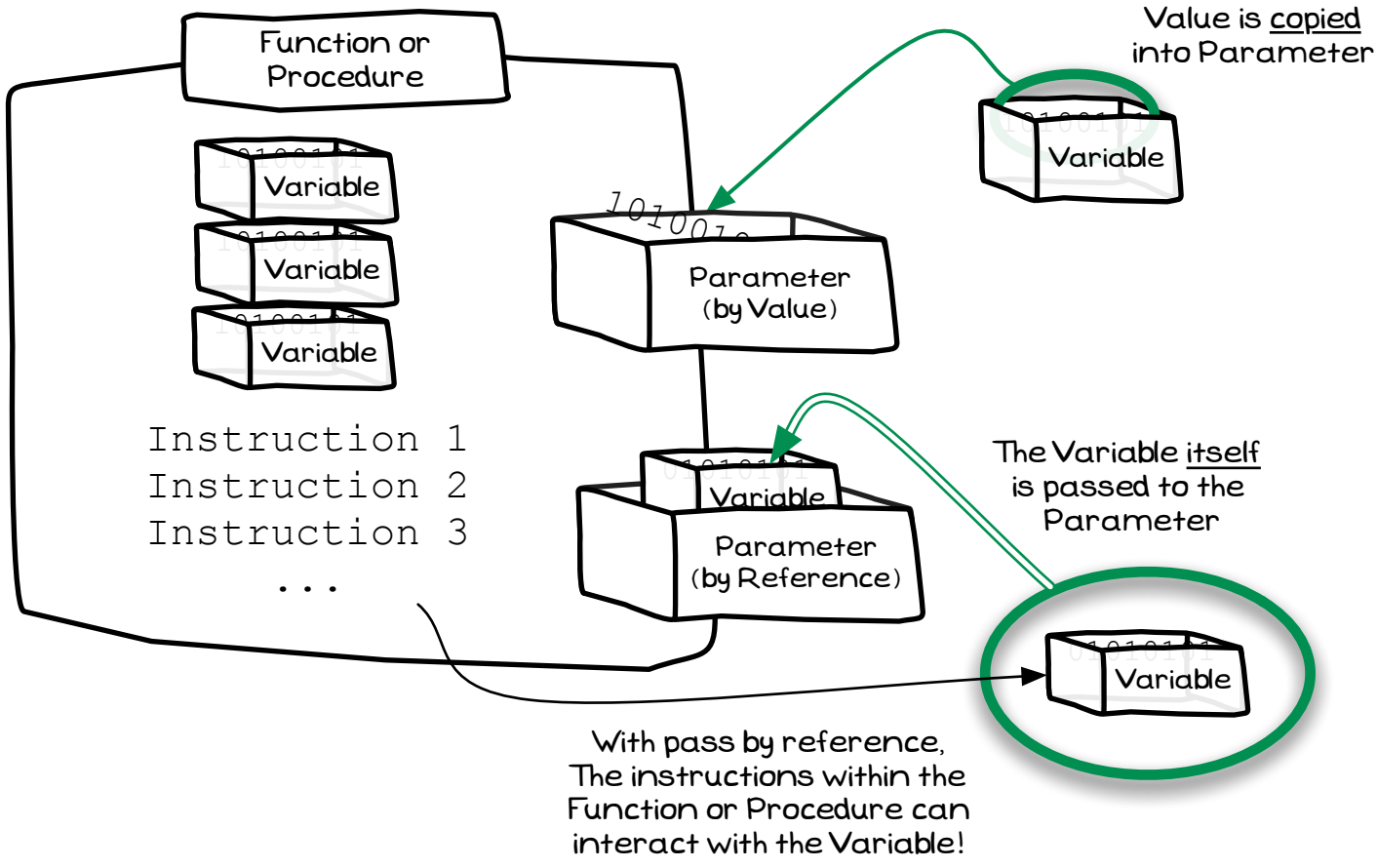

Artefact
Constant
Reads the Value of a Constant
Action
Assignment Statement
Term
Expression
Is like a Variable, but the value can not change
Assigns a Value to a Variable
Is a kind of
Reads the Value of a Variable
Uses to get calculated values
Artefact
Variable
Statement
Term
Function Call
Action
Are kinds of
Runs the code in a
Variables in a Program are
Global Variable
Term
Program
Local Variable
Artefact
Can contain Functions
Term
Term
Variables in a Function / Procedure are
Parameter
Artefact
Function
Procedure
A Procedure / Function accepts values using Parameters
using...
Artefact
Term
Pass by Value
Pass by Value
Term

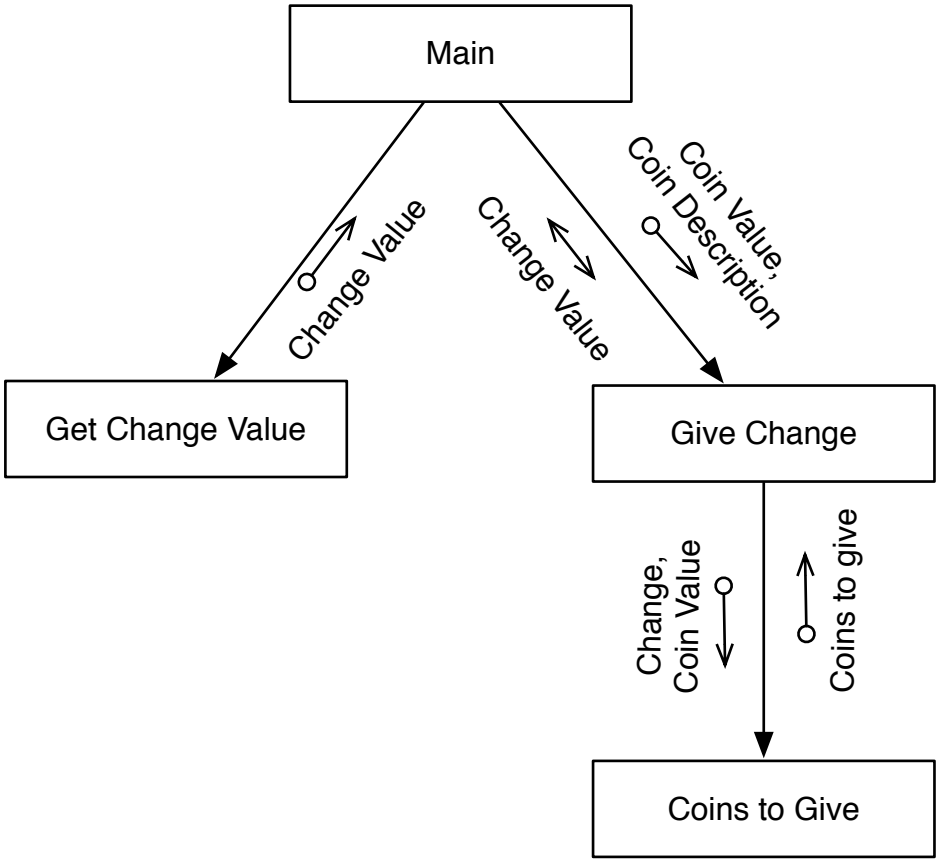

Calculates a Value
Has a name that is an
The Function's instructions calculate and return a value
Function
Identifier
Variable
Variable
Variable
Result
Parameter
Parameter
Like a Procedure, a Function can have Local Variables, and Instructions
Instruction 1
...
Gets the value returned
Values are passed into Parameters
Starts the
Contains a number of reusable
Library
Function Call

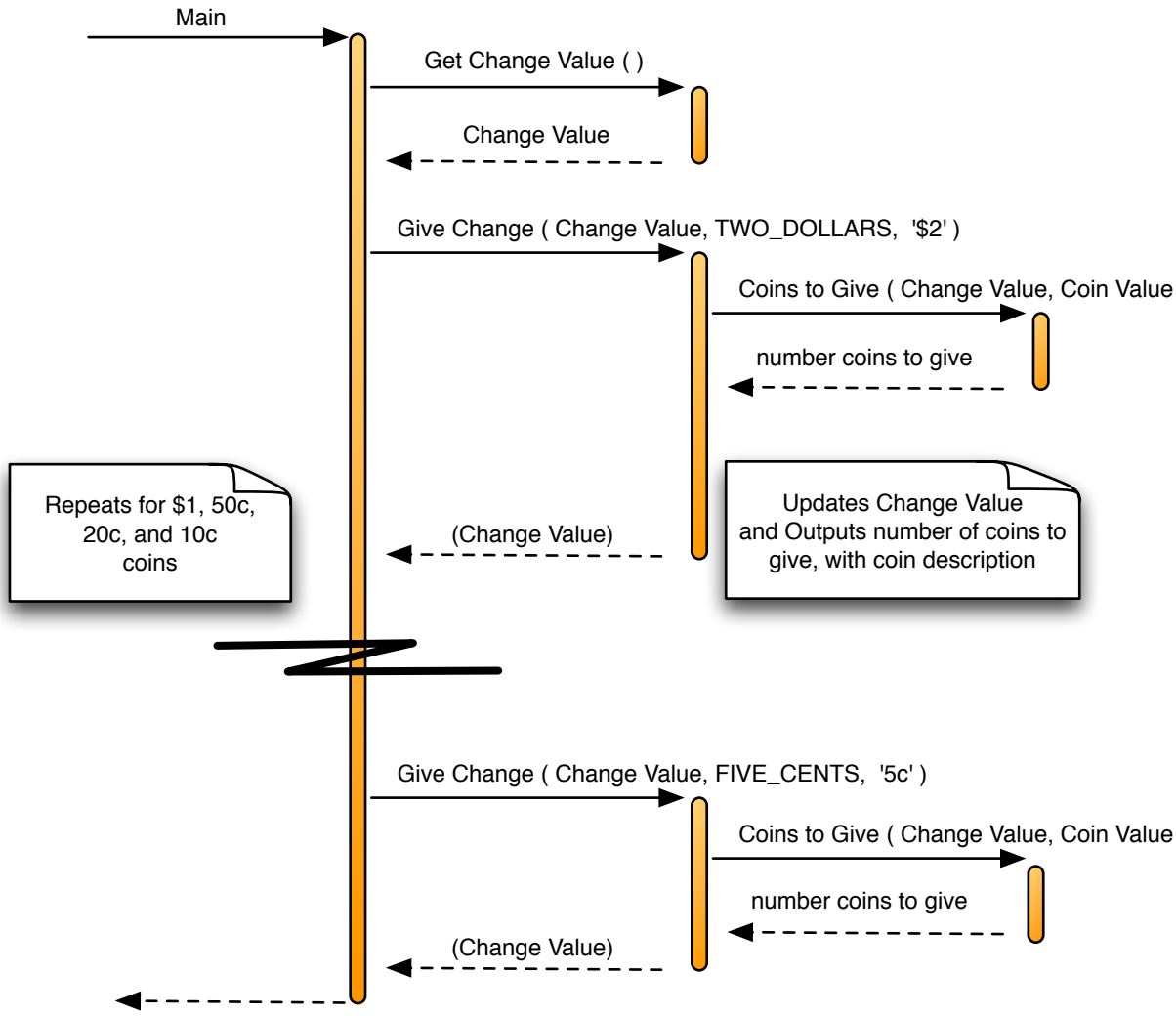

Function
Variable
Variable
Result
Parameter
Parameter
Instruction 1
...

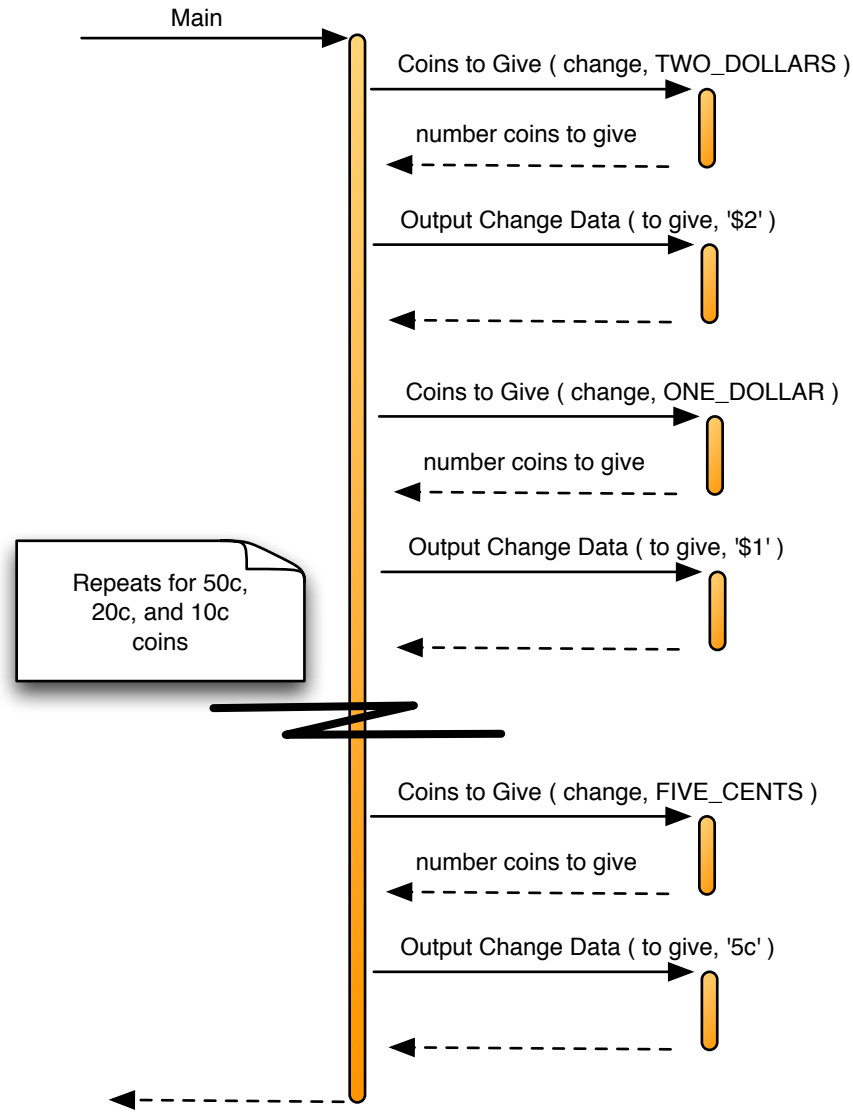

Calls a Function
Is part of an Expression
Expression
Function Call
Provides values to the Parameters
Gets the value returned by the Function
Function
Parameter
Parameter
Variable
Variable
Result
Instruction 1
...

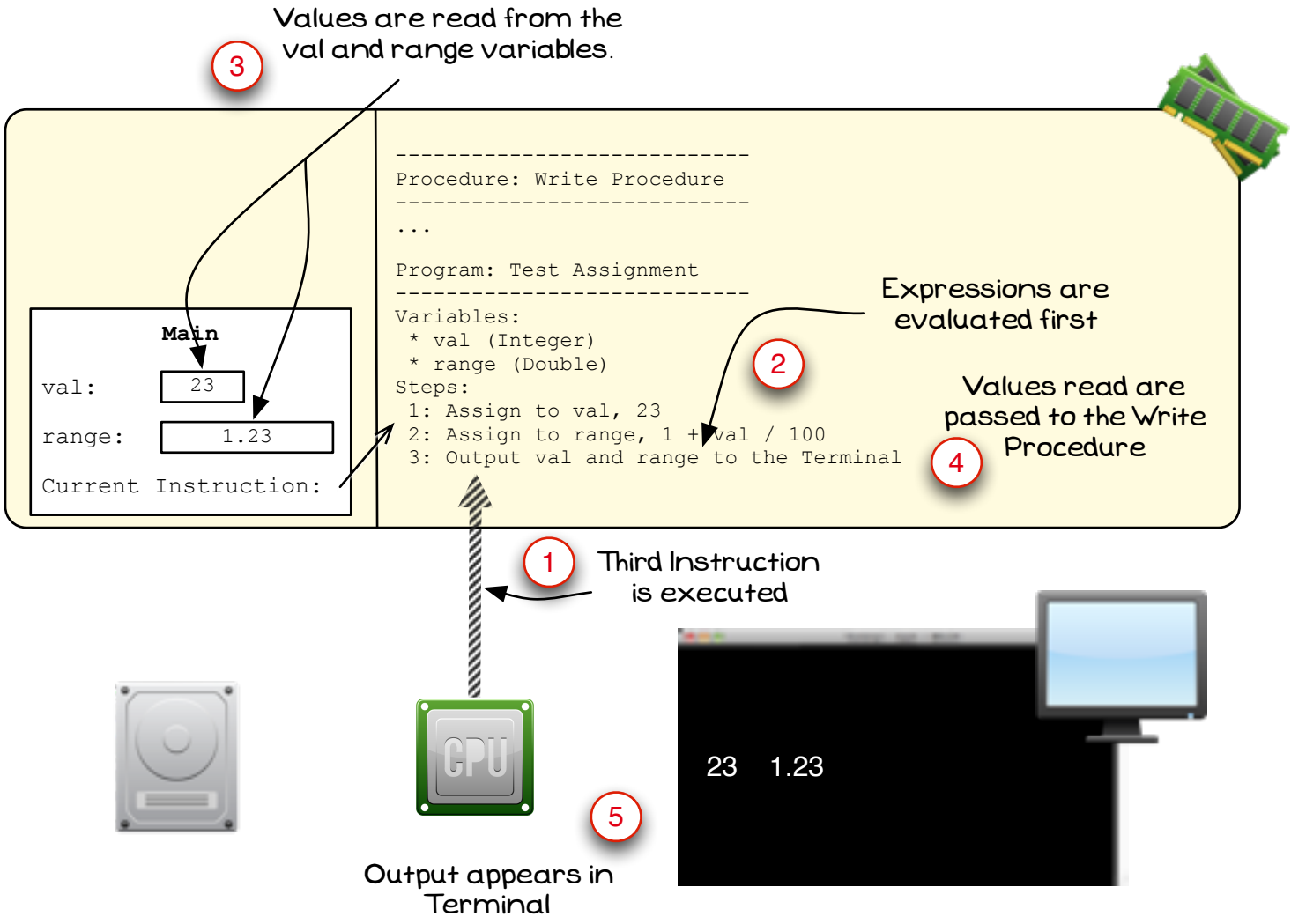

Library
Declares the code the compiler turns into an executable file
May use procedures from Libraries
Has a name that is an
Program
Identifier
A Program may declare its own Procedures.
Procedure
Instruction 1
Instruction 2
Instruction 3
Instruction 4
Instruction 5
Parameter
Parameter
Constant
10100...
Global Variables exist within the Program
Variable
Variable
Function
Function
Function
Variable
Variable
Result
Parameter
Parameter
Instruction 1
Instruction 2
...
Global Variables can be accessed in any Procedure, or Function, and within the Program's code
A Program may declare its own Functions.
The Program has an entry point that indicates where the program's instructions start
Each instruction is a Statement commanding the computer to perform an action
Statement


Function or Procedure
Variable
Variable
Variable
Instruction 1
Instruction 2
Instruction 3
...
Parameter (by Value)
Variable
Parameter (by Reference)
Value is copied into Parameter
Variable
The Variable itself is passed to the Parameter
Variable
With pass by reference, The instructions within the Function or Procedure can interact with the Variable!


Main
Change Value
Change Value
Coin Value,
Coin Description
Get Change Value
Give Change
Change,
Coin Value
Coins to give
Coins to Give


Main
Get Change Value ( )
Change Value
Give Change (Change Value, TWO_DOLLARS, '$2')
Coins to Give (Change Value, Coin Value)
number coins to give
Repeats for $1, 50c, 20c, and 10c coins
(Change Value)
Updates Change Value and Outputs number of coins to give, with coin description
Give Change (Change Value, FIVE_CENTS, '5c')
Coins to Give (Change Value, Coin Value)
number coins to give
(Change Value)

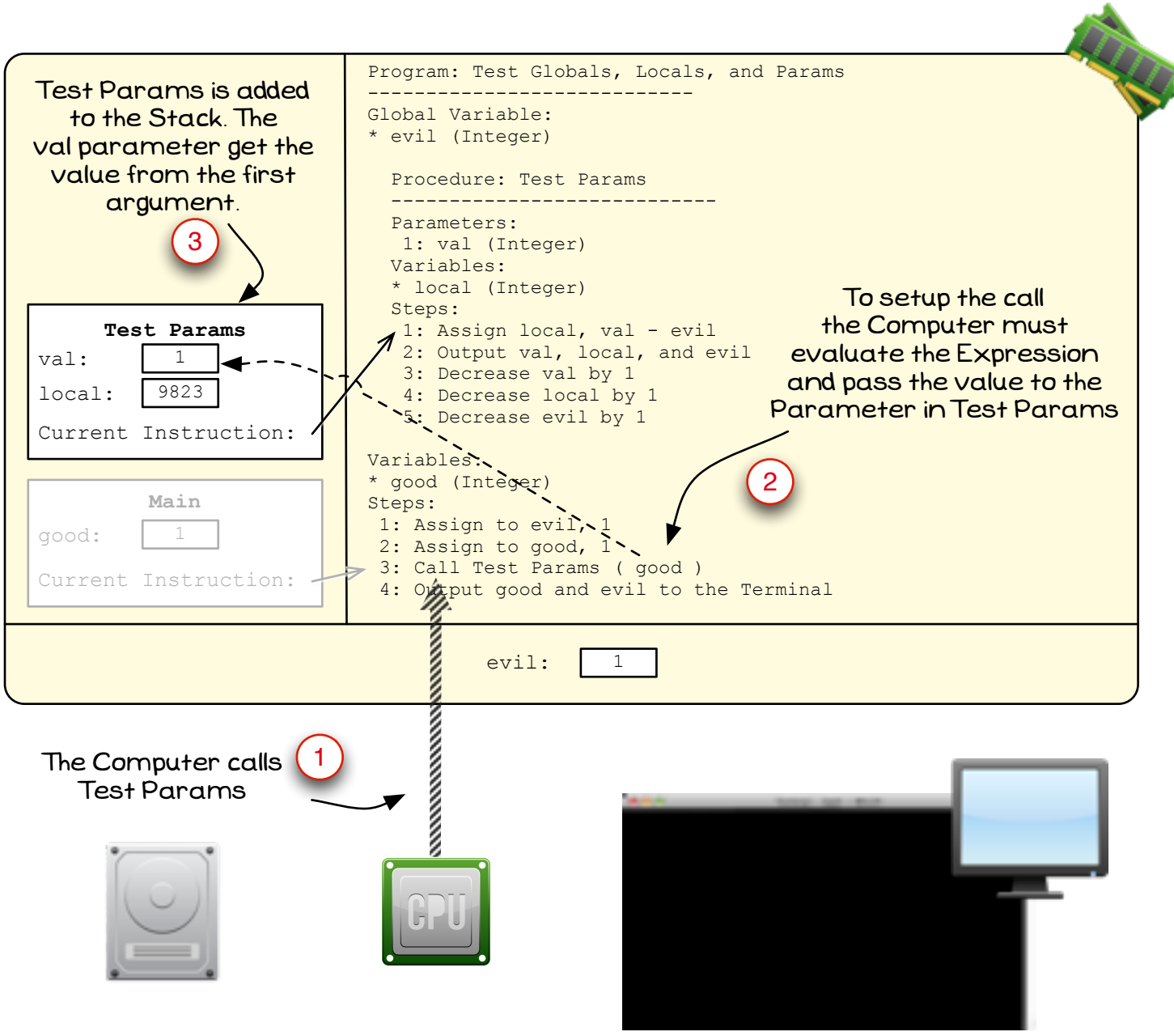

Main
Coins to Give (change, TWO_DOLLARS)
number coins to give
Output Change Data (to give, '$2')
Coins to Give (change, ONE_DOLLAR)
number coins to give
Output Change Data (to give, '$1')
Repeats for 50c, 20c, and 10c coins
Coins to Give (change, FIVE_CENTS)
number coins to give
Output Change Data (to give, '5c')

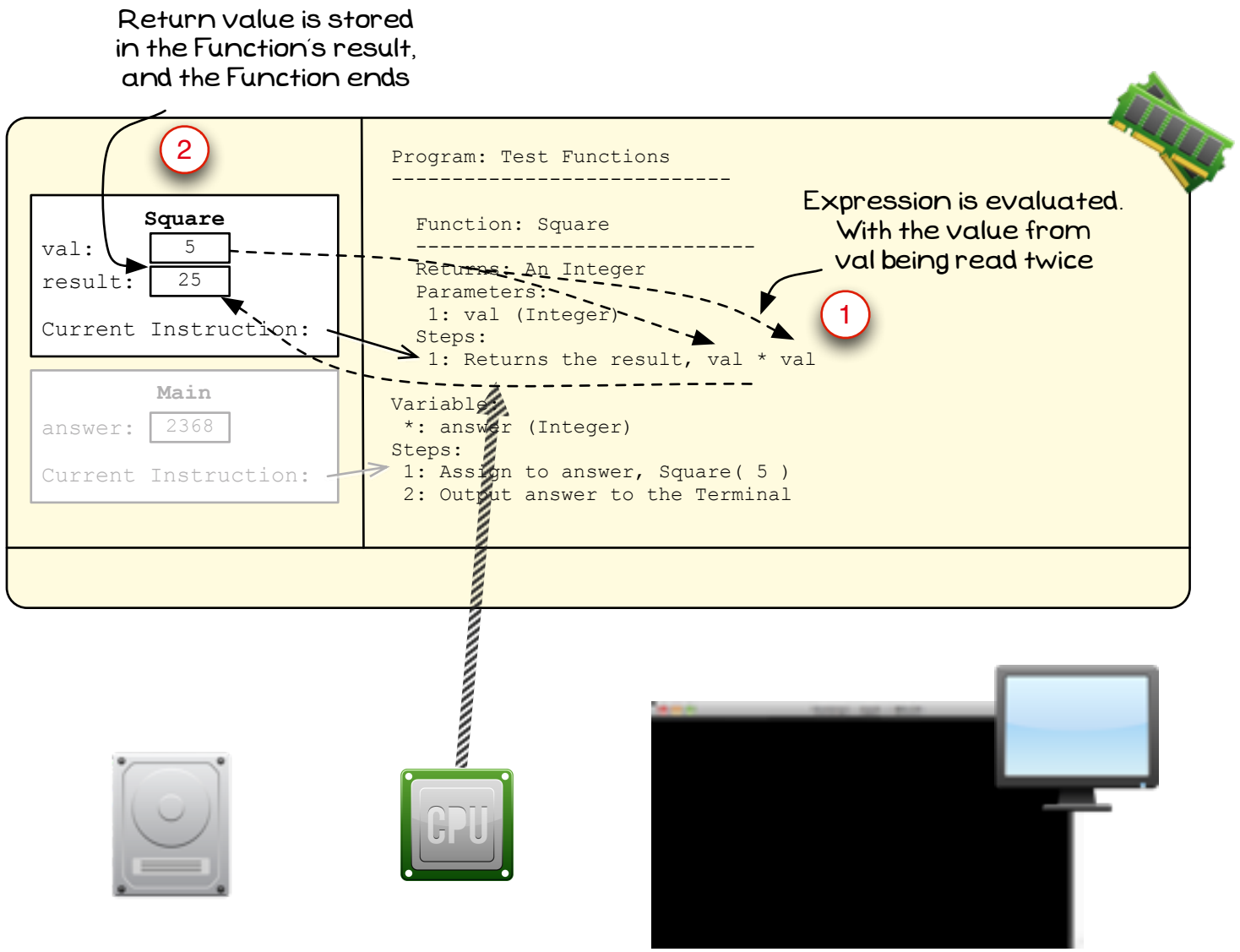

Values are read from the val and range variables.
3
Main
val:
23
range:
1.23
Current Instruction:

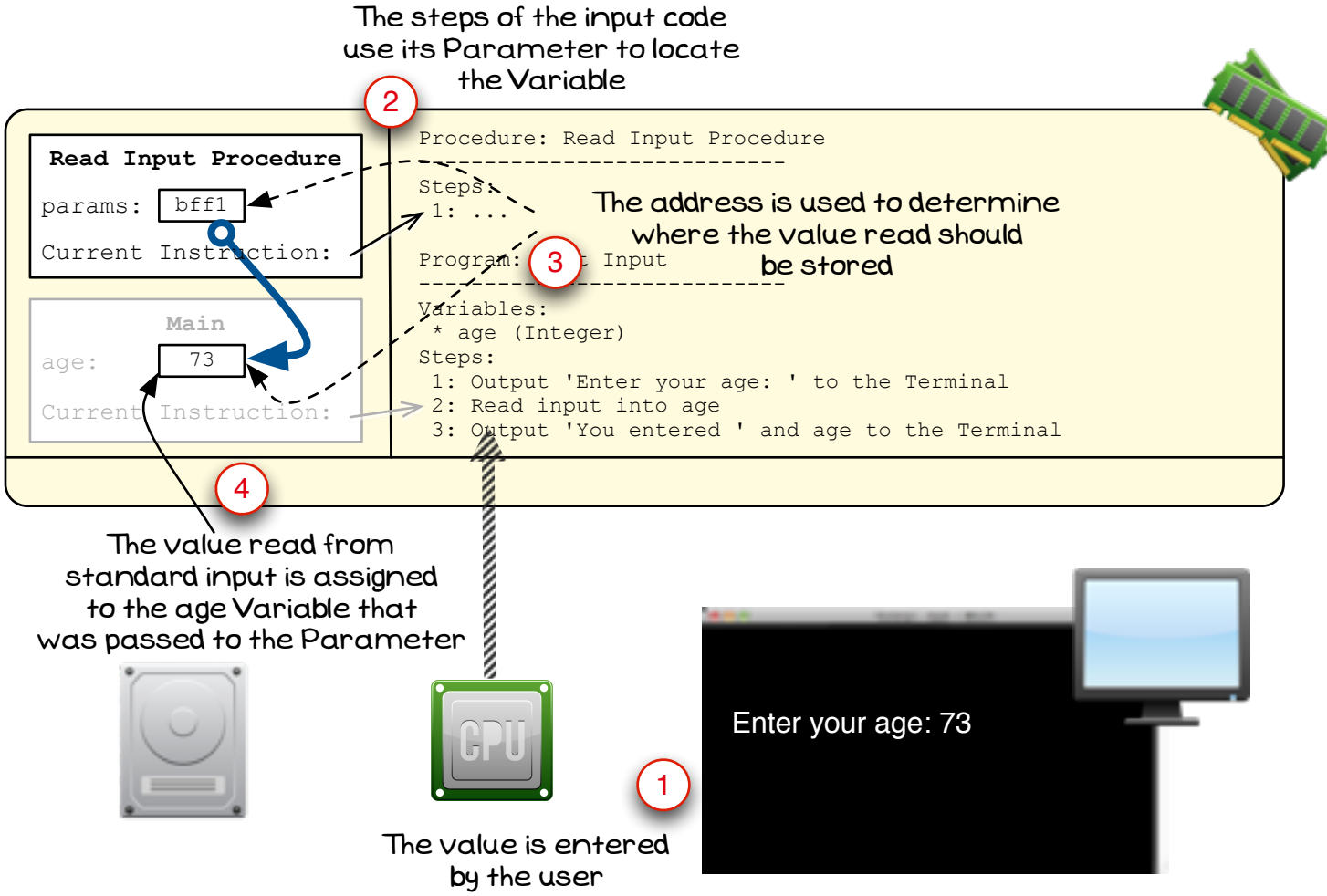
Procedure: Write Procedure

...
Program: Test Assignment

Variables:
* val (Integer)
* range (Double)
Steps:
1: Assign to val, 23
2: Assign to range, 1 + val / 100
3: Output val and range to the Terminal
Expressions are evaluated first
2
Values read are passed to the Write Procedure
4
1
Third Instruction is executed
CPU
5
Output appears in Terminal
23 1.23


Test Params is added to the Stack. The val parameter get the value from the first argument.
3
Test Params
val: 1
local: 9823
Current Instruction:
Main
good: 1
Current Instruction:
Program: Test Globals, Locals, and Params
Global Variable:
* evil (Integer)
Procedure: Test Params
Parameters:
1: val (Integer)
Variables:
* local (Integer)
Steps:
1: Assign local, val - evil
2: Output val, local, and evil
3: Decrease val by 1
4: Decrease local by 1
5: Decrease evil by 1
Variables:
* good (Integer)
Steps:
1: Assign to evil, 1
2: Assign to good, 1
3: Call Test Params (good)
4: Output good and evil to the Terminal
To setup the call the Computer must evaluate the Expression and pass the value to the Parameter in Test Params
2
evil: 1
The Computer calls Test Params
1
CPU


Return value is stored in the Function's result, and the Function ends
2
Square
val:
5
result:
25
Current Instruction:
Main
answer:
2368
Current Instruction:
Program: Test Functions
Function: Square
Returns: An Integer
Parameters:
1: val (Integer)
Steps:
1: Returns the result, val * val
Expression is evaluated. With the value from val being read twice
1
Variables:
*: answer (Integer)
Steps:
1: Assign to answer, Square(5)
2: Output answer to the Terminal
CPU


The steps of the input code use its Parameter to locate the Variable
2
Read Input Procedure
params:
bff1
Current Instruction:
Main
age:
73
Current Instruction:
Procedure: Read Input Procedure
Steps:
1: ...
Program: Input
Variables:
* age (Integer)
Steps:
1: Output 'Enter your age: ' to the Terminal
2: Read input into age
3: Output 'You entered ' and age to the Terminal
The address is used to determine where the value read should be stored
3
4
The value read from standard input is assigned to the age Variable that was passed to the Parameter
CPU
1
Enter your age: 73
The value is entered by the user