Leone Learning Systems, Inc.
Wonder. Create. Grow.

Leone Learning Systems, Inc.
237 Custer Ave
Evanston, IL 60202
Email tj@leonelearningsystems.com

Phone 847 951 0127
Fax 847 733 8812

Essential Primitives

TJ Leone

November 2004

Introduction

This guide accompanies the text *Approaching Precalculus Mathematics Discretely* by Philip G. Lewis.

If you learn all the primitives in this list, you should know as much as you need to get through the exercises. Contact me at tj@leonelearningsystems.com if you find primitives in the book that weren't in the list.

The list

back (bk)
clearscreen (cs)
difference (-)
first
forward (fd)
fput
heading
int
item
last
left (lt)
list
local
lput
make
minus
output (op)
pendown (pd)
penup (pu)
pensize
pos
print
product (*)
quotient (/)
remainder
repeat
right (rt)
round
run
sentence (se)
setheading (seth)
setpensize
setpos
show
sqrt
sum (+)

Understanding the entries

As you have read elsewhere, there are two kinds of Logo procedures: commands and operations. According to Brian Harvey,

"An *operation* is a procedure that computes a value and outputs it. `Sum` and `product` are operations, for example.

"A *command* is a procedure that does *not* output a value but instead has some effect such as printing something on the screen, moving a turtle, or making a sound (Harvey, 1997)." Examples of commands are `print` and `fd`.

The values output by operations can be used as input to commands or other operations. Commands and operations can have zero or more inputs. Operations always have exactly one output. As stated above, commands never have any output.

In the entries below, every Logo primitive given is either a command or an operation. In the following sections, we will examine an example of each.

Command example: BACK

Below is the vocabulary entry for the Logo command BACK.

BACK

BACK dist
BK dist

Moves the turtle backward, i.e., exactly opposite to the direction that it's headed, by the specified distance. (The heading of the turtle does not change.)

dist:(NUMBER) Distance for turtle to travel.

Example:

repeat 4 [back 100 rt 90]

The entries use the following format:

Procedure name.	In this case, the name of the procedure is BACK.
Usage.	In this section, we are given the name of the procedure together with any inputs. If there are alternative spellings for the procedure name, they are given here. In this case, the we see BACK dist and BK dist in this section. The "dist" is the input that BACK needs. The "BK" is an alternate spelling for BACK (so you can use `bk` instead of `back`).
Effect or output.	Commands have effects, i.e., they make something happen. Operations have output. Since BACK is a command, we see a description of its effect ("Moves the turtle backward…").
Description of any output or inputs.	Here we see a description of the input "dist"—it's a NUMBER, and it tells the turtle how much distance to travel.
Example:	Here the procedure name is used in an instruction.

Operation example: DIFFERENCE

Below is the vocabulary entry for the Logo operation DIFFERENCE.

DIFFERENCE

num DIFFERENCE num1 num2
num1 - num2

Outputs the difference of its inputs. Minus sign means infix difference in ambiguous contexts (when preceded by a complete expression), unless it is preceded by a space and followed by a non space.

num:(NUMBER) Result of difference.

num1:(NUMBER) Number to subtract from.
num2:(NUMBER) Number to subtract.

Example:

show 3 - 2
1

Operation example: DIFFERENCE (continued)

Below is an explanation of the sections in the DIFFERENCE entry:

Procedure name.	In this case, the name of the procedure is DIFFERENCE.
Usage.	The line "num DIFFERENCE num1 num2" tells us that DIFFERENCE has one output ("num") and two inputs ("num1" and "num2"). The line "num1 – num2" show an alternate "spelling" for DIFFERENCE, i.e., you can use a minus sign instead of the word DIFFERENCE. Notice that the minus sign is an *infix* operation (goes between its two inputs) while DIFFERENCE is a *prefix* operation (goes before its two inputs).
Effect or output.	"Outputs the difference of its inputs…."
Description of any output or inputs.	"num" is the output. "num1" and "num2" are the inputs.
Example:	Notice that the example uses the infix form (minus sign).

A few weird ones

As stated above, Logo procedures are either operations or commands. However, there are a few procedures that are not easy to categorize by just looking at their vocabulary entries, so I'll go over them here.

SHOW and PRINT

We often associate "output" with printouts or displays of information. However, it is important to understand that the Logo commands SHOW and PRINT *do not produce output*. SHOW and PRINT take input and display the input in the Recall List Box. This is an effect. SHOW and PRINT do not produce anything that can be used as input to other procedures. So they are commands, not operations.

RUN

The procedure RUN can actually be a command *or* an operation, depending on its input. If you look in the Primitives section under RUN, you will see that its input is an instructionlist, which is a list that contains some number of instructions. If the first instruction in that list is an operation, then RUN acts as an operation and outputs the output of that first instruction. If the first instruction is a command, RUN acts as a command.

OUTPUT

This is a command that turns procedures into operations. Its effect is to stop the procedure it's in and pass off its input to Logo as an output of the procedure.

Primitives

BACK

BACK dist
BK dist

Moves the turtle backward, i.e., exactly opposite to the direction that it's headed, by the specified distance. (The heading of the turtle does not change.)

dist:(NUMBER) Distance for turtle to travel.

Example:

```
repeat 4 [back 100 rt 90]
```

CLEARSCREEN

CLEARSCREEN
CS

Erases the graphics window and sends the turtle to its initial position and heading. Like HOME and CLEAN together.

Example:

```
setxy 100 100
clearscreen
```

Primitives

DIFFERENCE

num DIFFERENCE num1 num2
num1 - num2

Outputs the difference of its inputs. Minus sign means infix difference in ambiguous contexts (when preceded by a complete expression), unless it is preceded by a space and followed by a non space.

num:(NUMBER) Result of difference.

num1:(NUMBER) Number to subtract from.
num2:(NUMBER) Number to subtract.

Example:

show 3 - 2
1

FIRST

firstthing FIRST thing

If the input is a word, outputs the first character of the word.
If the input is a list, outputs the first member of the list.
If the input is an array, outputs the origin of the array (that is, the INDEX OF the first element of the array).

firstthing:(THING) First thing of input.

thing:(THING) Existing thing to be extracted from.

Example:

print first [1 2 3]
1
print first "Hello

H

Primitives

FORWARD

FORWARD dist
FD dist

Moves the turtle forward, in the direction that it's headed, by the specified distance (measured in turtle steps).

dist:(NUMBER) Distance for turtle to travel.

Example:

repeat 4 [forward 100 rt 90]

FPUT

newlist FPUT thing list

Outputs a list equal to its second input with one extra member, the first input, at the beginning.

newlist:(LIST) New list formed by inputs.

thing:(THING) Thing to be added to front new list.
list:(LIST) Existing list to be added to.

Example:

show fput 1 [2 3 4]
[1 2 3 4]

Primitives

HEADING

angle HEADING

Outputs an angle, the turtle's heading in degrees. See also SETHEADING.

angle:(NUMBER) Angle, in degrees, of the current heading.

Example:

```
setheading 90
show heading
90
```

INT

int INT num

Outputs its input with fractional part removed, i.e., an integer with the same sign as the input, whose absolute value is the largest integer less than or equal to the absolute value of the input.

int:(INTEGER) Result of int.

num:(NUMBER) Number to int.

Note: Inside the computer numbers are represented in two different forms, one for integers and one for numbers with fractional parts. However, on most computers the largest number that can be represented in integer format is smaller than the largest integer that can be represented (even with exact precision) in floating-point (fraction) format. The INT operation will always output a number whose value is mathematically an integer, but if its input is very large the output may not be in integer format. In that case, operations like REMAINDER that requires an integer input will not accept this number.

Example:

```
show int 8.2
8
show int 8.7
8
```

Primitives

ITEM

item ITEM index thing

If the thing is a word, outputs the index-th character of the word. If the thing is a list, outputs the index-th member of the list. If the thing is an array, outputs the index-th element of the array. An index starts at 1 for words and lists; the starting index of an array is specified when the array is created.

item:(THING) The item extracted from its input.

index:(INTEGER) The index of the item to be extracted.
thing:(THING) Existing thing to be extracted from.

Example:

```
show item 2 [a b c]
b
show item 3 "ABC
c
```

LAST

lastthing LAST thing

If the input is a word, outputs the last character of the word.
If the input is a list, outputs the last member of the list.

lastthing:(THING) Last thing of input.

thing:(THING) Existing thing to be extracted from.

Example:

```
print last [1 2 3]
3
print last "Hello
o
```

Primitives

LEFT

LEFT angle
LT angle

Turns the turtle counterclockwise by the specified angle, measured in degrees (1/360 of a circle).

angle:(NUMBER) Angle, in degrees, for turtle to turn by.

Example:

```
repeat 3 [fd 100 left 120]
```

LIST

list LIST thing1 thing2
list (LIST thing1 thing2 thing3 ...)

Outputs a list whose members are its inputs, which can be any Logo object (word, list, or array).

list:(LIST) Newly formed list of its inputs.

thing1:(THING) First thing to be a member of the output list.
thing2:(THING) Second thing to be a member of the output list.

Example:

```
show (list "This "is "a "List)
```

[This is a List]

```
show list [1 2 3] [a b c]
```

[[1 2 3] [a b c]]

Example2:

```
make "red 100
make "green 100
make "blue 100
show (list :red :green :blue)
```

[100 100 100]

Primitives

LOCAL

LOCAL varname
LOCAL varnamelist
(LOCAL varname1 varname2 ...)

Command that accepts as inputs one or more words, or a list of words. A variable is created for each of these words, with that word as its name. The variables are local to the currently running procedure. Logo variables follow dynamic scope rules; a variable that is local to a procedure is available to any sub procedure invoked by that procedure. The variables created by LOCAL have no initial value; they must be assigned a value (e.g., with MAKE) before the procedure attempts to read their value.

varname:(WORD) Name of the variable you wish to localize.
varnamelist:(LIST) List of names (words) of the variables you wish to localize.
varname1:(WORD) First name of the variable you wish to localize.
varname2:(WORD) Second name of the variable you wish to localize.

Example:

```
to foo
make "bar 1
print :bar
end
foo
1
show :bar
1
to abc
local "xyz
make "xyz 1
print :xyz
end
abc
1
show :xyz

xyz has no value
```

Primitives

LPUT

newlist LPUT thing list

Outputs a list equal to its second input with one extra member, the first input, at the end.

newlist:(LIST) New list formed by inputs.

thing:(THING) Thing to be added to end of new list.
list:(LIST) Existing list to be added to.

Example:

show lput 5 [1 2 3 4]
[1 2 3 4 5]

MAKE

MAKE varname value

Command that assigns the value value to the variable named varname, which must be a word. Variable names are case-insensitive. If a variable with the same name already exists, the value of that variable is changed. If not, a new global variable is created.

varname:(WORD) Name of the variable you wish to assign.
value:(THING) Thing to be assigned to variable.

Example:

make "foo [Hello how are you]

show :foo
[Hello how are you]

Primitives

MINUS

neg MINUS num
- num

Outputs the negative of its input. Minus sign means unary minus if it is immediately preceded by something requiring an input, or preceded by a space and followed by a non space. There is a difference in binding strength between the two forms:

MINUS 3 + 4 means -(3+4)
- 3 + 4 means (-3)+4

neg:(NUMBER) Minus of input.

num:(NUMBER) Number to minus.

Example:

```
show 2 - -3
5
```

OUTPUT

OUTPUT value
OP value

Command that ends the running of the procedure in which it appears. That procedure outputs the value "value" to the context in which it was invoked. Don't be confused: OUTPUT itself is a command, but the procedure that invokes OUTPUT is an operation.

value:(THING) Any Logo thing to output.

Example:

```
to myprog
output [This is the output]
end
show myprog
[This is the output]
```

Primitives

PENDOWN

PENDOWN
PD

Sets the pen's position to DOWN, without changing its mode.

Example:

repeat 10 [fd 10 pu fd 10 pd]

PENUP

PENUP
PU

Sets the pen's position to UP, without changing its mode.

Example:

repeat 10 [fd 10 pu fd 10 pd]

PENSIZE

size PENSIZE

Output pen size information which is a list containing [Width Height]. Width is not used in MSWLogo.

size:(LIST) List of 2 integers representing width and height of the pen.

Example:

```
setpensize [10 20]
show pensize
[20 20]
setpensize [1 1]
show pensize
[1 1]
```

Primitives

POS

pos POS

Outputs the turtle's current position, as a list of two numbers, the X and Y coordinates.

pos:(LIST) List of two numbers representing current X,Y coordinate.

Example:

setpos [100 100]
show pos
[100 100]

PRINT

PRINT thing
PR thing
(PRINT thing1 thing2 ...)
(PR thing1 thing2 ...)

Command that prints the input or inputs to the current write stream (initially the terminal). All the inputs are printed on a single line, separated by spaces, ending with a newline. If an input is a list, square brackets are not printed around it, but brackets are printed around sublists. Braces are always printed around arrays.

thing:(thing) A thing you wish to be printed.

thing1:(thing) First thing you wish to be printed.
thing2:(thing) Second thing you wish to be printed.

Example:

print "Hello
Hello
print [Hello how are you]
Hello how are you

Primitives

PRODUCT

num PRODUCT num1 num2
num (PRODUCT num1 num2 num3 ...)
num1 * num2

Outputs the product of its inputs.

num:(NUMBER) Result of product.

num1:(NUMBER) First number of product.
num2:(NUMBER) Second number of product.

Example:

```
show 2 * 3
6
```

QUOTIENT

num QUOTIENT num1 num2
num (QUOTIENT num2)
num1 / num2

Outputs the quotient of its inputs. The quotient of two integers is an integer if and only if the dividend is a multiple of the divisor. (In other words, QUOTIENT 5 2 is 2.5, not 2, but QUOTIENT 4 2 is 2, not 2.0 -- it does the right thing.) With a single input, QUOTIENT outputs the reciprocal of the input.

num:(NUMBER) Result of quotient.

num1:(NUMBER) Dividend of quotient (this is 1 when not specified).
num2:(NUMBER) Divisor of quotient.

Example:

```
show 6 / 3
2
show 3 / 2
1.5
```

Primitives

REMAINDER

num REMAINDER num1 num2

Outputs the remainder on dividing num1 by num2; both must be integers and the result is an integer with the same sign as num1.

num:(INTEGER) Result of remainder.

num1:(INTEGER) Dividend of remainder.
num2:(INTEGER) Divisor of remainder.

Example:

```
show remainder 6 4
2
show remainder 6 2
0
```

REPEAT

REPEAT num instructionlist

Command that runs the instructionlist repeatedly, num times. See also REPCOUNT.

num:(INTEGER) Number of time to repeat instructionlist.
Instructionlist:(LIST) List of Logo instructions to run repeatedly.

Example:

```
repeat 3 [print (list "This "Is "loop repcount)]
This Is loop 1
This Is loop 2

This Is loop 3
```

Primitives

RIGHT

RIGHT angle
RT angle

Turns the turtle clockwise by the specified angle, measured in degrees (1/360 of a circle).

angle:(NUMBER) Angle, in degrees, for turtle to turn by.

Example:

```
repeat 3 [fd 100 right 120]
```

ROUND

int ROUND num

Outputs the nearest integer to the input.

int:(INTEGER) Result of rounding.

num:(NUMBER) Number to round.

Example:

```
show round 8.2
8
show round 8.7
9
```

Primitives

RUN

RUN instructionlist

Command or operation that runs the Logo instructions in the input list; outputs if the list contains an expression that outputs.

Instructionlist:(LIST) List of Logo instructions to run.

Example:

```
make "thingstodo [print]
make "thingstodo lput "\"Hello :thingstodo
run :thingstodo
Hello
```

SENTENCE

list SENTENCE thing1 thing2
list SE thing1 thing2
list (SENTENCE thing1 thing2 thing3 ...)
list (SE thing1 thing2 thing3 ...)

Outputs a list whose members are its inputs, if those inputs are not lists.
Outputs the members of its inputs, if those inputs are lists.

list:(LIST) Sentence formed by inputs.

thing1:(THING) First thing to be a member of the output sentence.

thing2:(THING) Second thing to be a member of the output sentence.

Example:

```
show (se "A "Sentence "is "simply "a "list "of "words)
[A Sentence is simply a list of words]
```

Primitives

SETHEADING

SETHEADING angle
SETH angle

Turns the turtle to a new absolute heading. The argument is an angle, the heading in degrees clockwise from the positive Y axis. See also HEADING. If you are in PERSPECTIVE mode then, the heading in degrees which is positive from the positive X-Axis to the positive Y-Axis rotating about the Z-Axis.

angle:(NUMBER) Angle, in degrees, to set heading to.

Example:

```
setheading 45
show heading
45
```

SETPENSIZE

SETPENSIZE size

Set hardware-dependent pen characteristics. These commands are not guaranteed compatible between implementations on different machines. The "size" is a list of two members, width and height. MSWLogo only uses the width of them. So just set them both to the same value.

size:(LIST) List of 2 integers representing width and height of the pen.

Example:

```
setpensize [5 5]
```

Primitives

SETPOS

SETPOS pos

Moves the turtle to an absolute X,Y coordinate. The argument is a list of two numbers, the X and Y coordinates. See also POS.

pos:(LIST) List of two numbers representing desired X,Y coordinate.

Example 1:(draw a square)

```
cs
setpos [0 100]
setpos [100 100]
setpos [100 0]
setpos [0 0]
```

Example 2:(the most common logo question)

```
make "x 0
make "y 100
setpos [:x :y]
<will fail>
setpos (list :x :y)

<will work>
```

Why, because the first case IS a list that contains 2 words :x and :y. In the second case a list is BUILT containing the value of :x and :y. You can see this more clearly by using the SHOW command.

```
show [:x :y]
[:x :y]
show (list :x :y)
[0 100]
```

Primitives

SHOW

SHOW thing
(SHOW thing1 thing2 ...)

Command that prints the input or inputs like PRINT, except that if an input is a list it is printed inside square brackets.

thing:(thing) A thing you wish to be shown.
thing1:(thing) First thing you wish to be shown.
thing2:(thing) Second thing you wish to be shown.

Example:

```
show [1 2 3]
[1 2 3]
print [1 2 3]
1 2 3
```

SQRT

num SQRT num1

Outputs the square root of the input, which must be nonnegative.

num:(NUMBER) Result of square root of num.

num1:(NUMBER) Number to take sign of.

Example:

```
show sqrt 4
2
show sqrt 9
3
```

Primitives

SUM

num SUM num1 num2
num (SUM num1 num2 num3 ...)
num1 + num2

Outputs the sum of its inputs.

num:(NUMBER) Result of sum.

num1:(NUMBER) First number to sum.
num2:(NUMBER) Second number to sum.

Example:

show 2 + 3
5

The Authors

The entries in this glossary are from George Mills's MSWLogo Reference Manual. MSWLogo is available at http://www.softronix.com/logo.html.

The introduction and explanation of entries are by TJ Leone.

TJ Leone owns and operates Leone Learning Systems, Inc., a private corporation that offers tutoring and educational software. He has a BA in Math and an MS in Computer Science, both from the City College of New York. He spent two years in graduate studies in education and computer science at Northwestern University, and six years developing educational software there. He is a former Montessori teacher and currently teaches gifted children on a part time basis at the Center for Talent Development at Northwestern University in addition to his tutoring and software development work. His web site is http://www.leonelearningsystems.com.

Reference

Harvey, B. (1997). *Symbolic Computing* (2nd ed. Vol. 1). Cambridge, MA: The MIT Press.