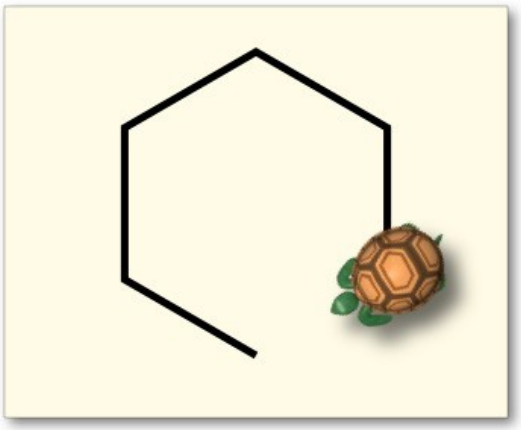
Learning Logo



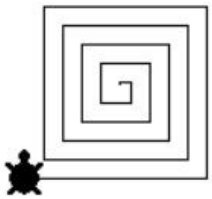
Nacho Iborra

IES San Vicente

Index of contents

1. Introduction

Logo is an educational programming language designed in 1967. Its main aim is to help people learn the basics of programming through simple instructions that produce visual results, thanks to a graphical environment that translates the instructions into drawing movements of a given cursor (commonly a turtle):



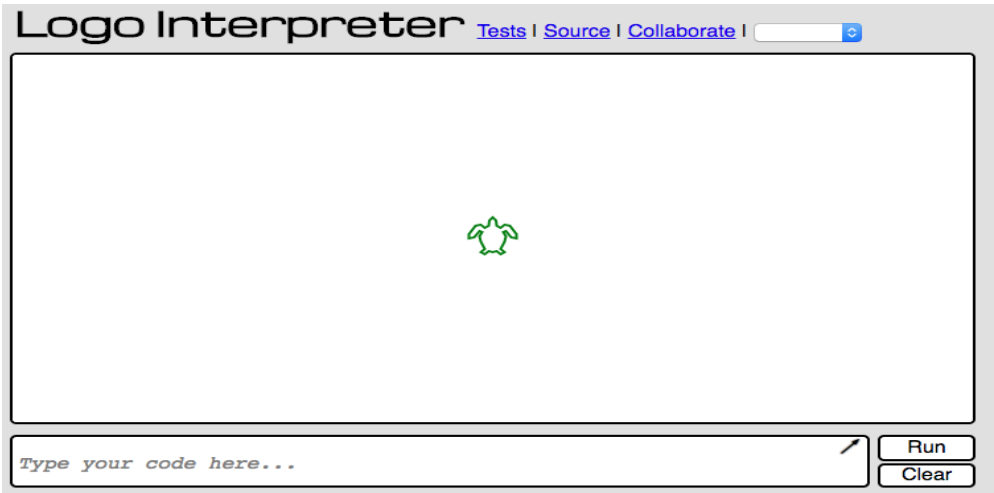
We will see what kind of software do we need to start programming in Logo, and the basic instructions to create some figures.

1.1. Software needed

In order to create our Logo programs, we can choose among some options, such as:

- **WinLogo**: an old Windows application that is still compatible with current Windows versions. You can also find it as *MSWLogo* in several web sites.
- **XLogo**: a Java-based environment, that let us create our programs despite the platform that we are using (Windows, Linux or Mac OSX). This is its official site.
- **UCBLogo**: an old multi-platform Logo programming environment
- **Berkeley Logo**: a multi-platform Logo programming environment developed by the University of Berkeley.
- Online logo interpreters, like this one.

In this document we are going to try the online interpreter. When we go to its web page, we see something like this:



In the lower form we type the instructions to move the turtle, so it's time to learn the basic instruction set.

2. Basic motion instructions

Let's learn the basic motion and drawing instructions to make our turtle move. In the following table you can find a summary of the instruction set (you can use either the abbreviation or the full name), and a description of each one:

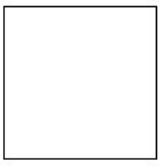
Abbr.	Full name	Description	Example
`fd`	`forward`	Moves the turtle forward the specified amount of steps	`fd 40`
`bk`	`back`	Moves the turtle backwards the specified amount of steps	`bk 30`
`rt`	`right`	Turns turtle to the right the specified number of degrees	`rt 90`
`lt`	`left`	Turns turtle to the left the specified number of degrees	`lt 45`
`cs`	`clearscreen`	Clears the drawing area	`cs`
`st`	`showturtle`	Shows the turtle in the screen	`st`
`ht`	`hideturtle`	Hides the turtle in the screen (useful when drawing accurately, or small figures)	`ht`
`pu`	`penup`	Allows us to move the turtle without drawing anything (useful to move to another part of the screen without drawing)	`pu`
`pd`	`pendown`	The turtle will draw again when moving (if we called to *pu* instruction before)	`pd`

We can type all these instructions in the command line below the canvas:

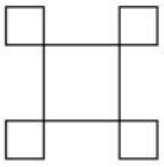
Note that we can type several instructions, separated by whitespaces. When we type "Intro", all the instructions will be applied. Besides, we can use the cursors to go to previous instructions and reload them.

Exercise 1

Use the basic instructions shown above to draw the following figures. Save the correponding instruction set in a separate text file with the name suggested for each figure:



square.txt



squares.txt

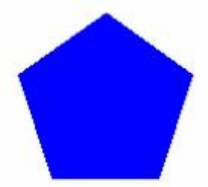
2.1. Dealing with colors and sizes

We can also change the default pencil color and size, and fill closed shapes. Here are the instructions to do this:

Abbr.	Full name	Description	Example
`setpc`	`setpencolor`	Changes the pencil color to the specified color in RGB (between brackets)	`setpc [255 0 0]`
`setpw`	`setpenwidth`	Changes the width (size) of the pencil, making it thicker or thinner	`setpw 10`
`fill`		Fills the current closed area with the current pen color. You need to place the turtle INSIDE a closed area to fill it	`fill`

Exercise 2

Try to create this figure filled with blue color. Save the instructions in a file called *pentagon.txt*.



NOTE: if we sum all the inner angles of a pentagon, we get 540 degrees.

3. Repeating instructions

What if we want to create more complicated structures? Do we have to manually repeat *fd*, *rt* or *lt* instructions indefinitely? Of course, the answer is NO.

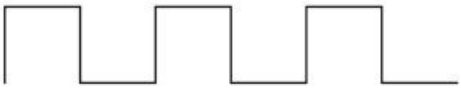
We have a *repeat* instruction that lets us repeat a set of instructions (put between brackets) a given number of times. For instance, we can draw a square of 100 steps side, this way:

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

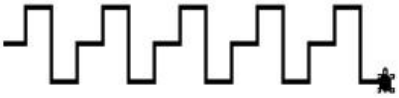
Exercise 3

Use the *repeat* instruction along with the instruction set learnt before to draw these figures. Save the corresponding instructions in the associated text file.

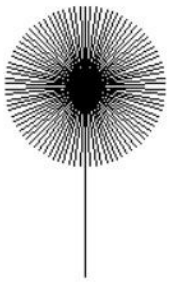
haircomb.txt



upanddown.txt



buildings.txt



flower.txt

4. Using variables

It is also possible to declare variables in Logo, and assign values to them in any part of the code. To declare a variable, we use the following syntax:

```
make "variable_name value
```

For instance:

```
make "size 100
```

Then, if we want to use this variable to move the turtle forward, for instance, we can do it this way:

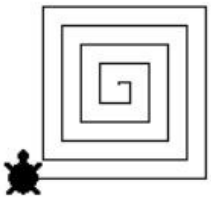
```
fd :size
```

Finally, we can also modify the value of the variable with the same make instruction, and use the old value of the variable. The following instruction adds 20 to the previous value of the same variable:

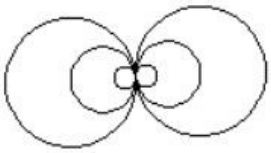
```
make "size :size + 20
```

Exercise 4

Try to use variables and loops to draw these figures:



spiral.txt



circles.txt

NOTE: if you want to draw a circle, you can try moving and rotating a given small quantity. For instance, if you do this:

```
repeat 120 [fd 1 rt 3]
```

you will get a small circle, whereas if you try this:

```
repeat 360 [fd 1 rt 1]
```

you will get a bigger one. Depending on the relationship between the steps forwarded and the degrees rotated, you will get a bigger or smaller circle.