

Title:	Simple programming with Logo		
Date :		REF No:	4.24, 4.25, 4,26
Contributors:	Srinath Perur	Std:	IV
		Reviewers:	
Brief Description:	This lesson introduces simple Logo programming.		
Goal:	To introduce simple Logo commands as an extension of the activities done in the last two classes, and to introduce the term program.		
Pre-requisites:	Lessons 4.22 and 4.23		
Duration:	Three classes		
Resources:			

Detailed description:

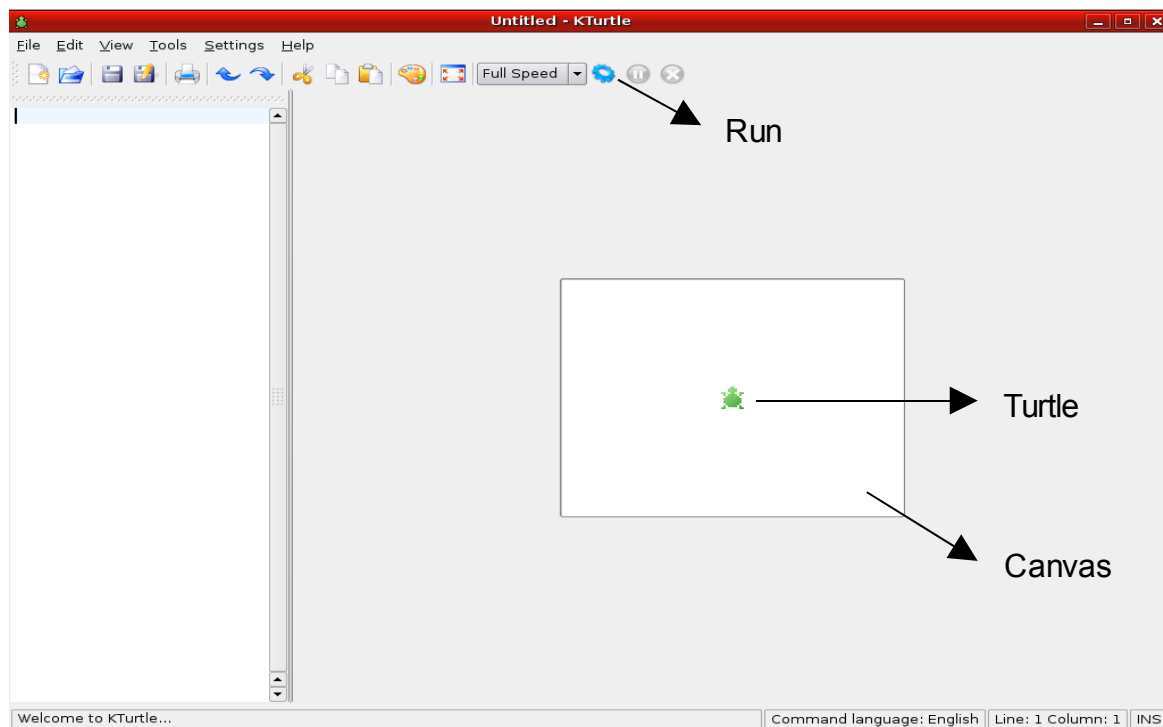
Note: This lesson is meant to introduce simple Logo programming to children. The content in this section is written to introduce teachers to the basics of Logo programming -- a brief description of the conceptual model, practical information on how to open the application, how to enter commands and run them, and so on. The lesson plan in the next section suggests a different approach when children are to be taught. This is because children in the fourth standard may not be familiar with two ideas that are very useful for working with Logo: the coordinate system and angles. The lesson is designed such that some idea of angles is conveyed during the lesson. But care has been taken that none of the examples or exercises require explicit use of the coordinate system.

A computer requires very detailed and precise instructions regarding how to go about performing a task. A list of such instructions taken together is called a **program**. The person who writes a program is called a **programmer**.

When we give instructions to other people, we use a language such as English or Hindi. For giving instructions to a computer, we use a **programming language**. Just like the languages we speak, a programming language consists of words called **keywords** or **commands**, and rules for putting them together so that the computer can understand them. In this lesson we will take a look at how to write simple programs in a language called **Logo**.

Logo

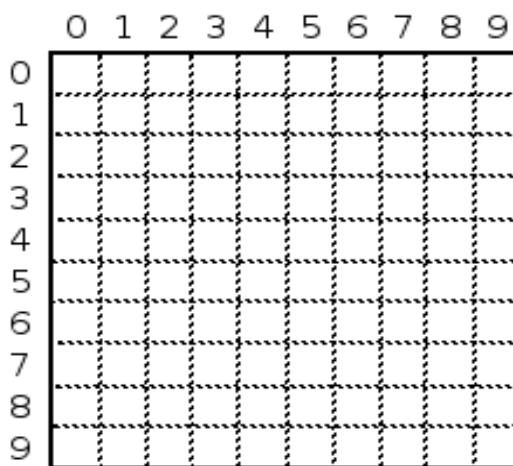
Let us now explore a small part of the programming language **Logo**. On your Edubuntu system, you can click *Applications->Education->KTurtle* to get the following window.



The square in the right pane is called the **canvas**, and there is a **turtle** in the center. In the left pane we can give Logo instructions to the turtle to move forward or backward, or turn by a certain amount. The turtle has a pen, and will leave a trail of its movements on the canvas. Thus we can use the turtle to trace a figure on the canvas. While Logo can be used to write larger programs, here we use only a few keywords that help us to control the movement of the turtle and make it draw.

The Canvas

In the screenshot above, the canvas is the blank region in the right pane in which the turtle draws. The canvas can be thought to be a fine invisible grid made of a large number of squares. (A magnified 10 x 10 grid is shown below for illustration.) The canvas in Kturtle is a 300 x 150 grid by default, but it can be changed as required.



The Turtle

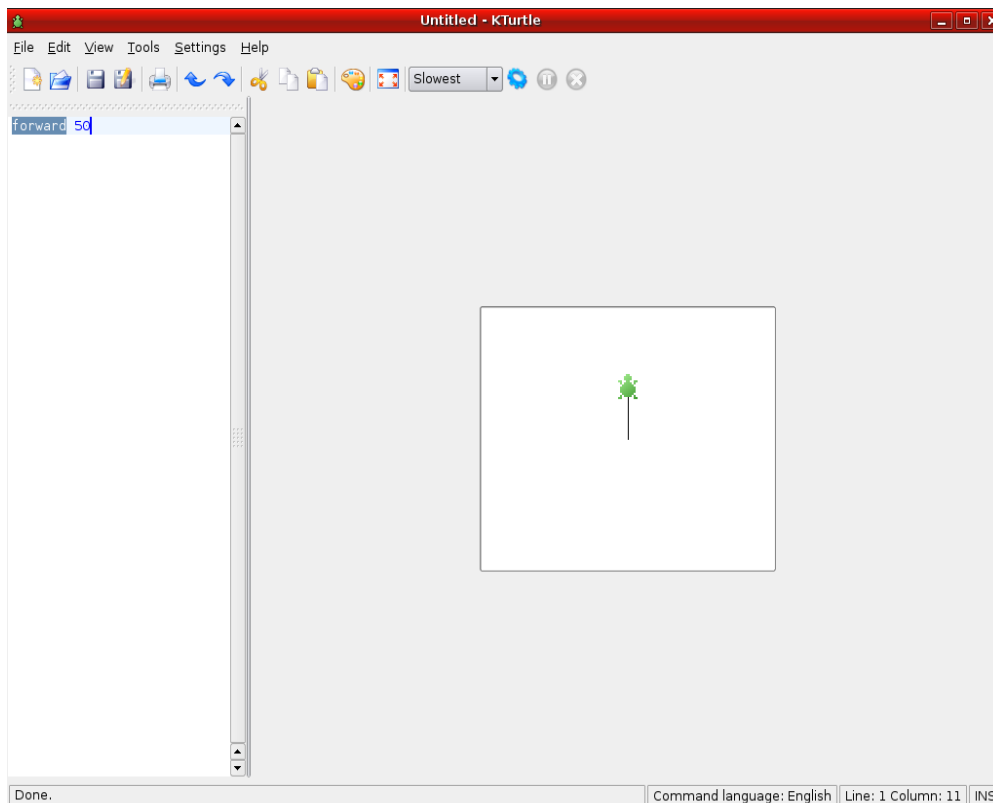
The turtle can be thought of as a cursor on the canvas. It obeys the commands that we type into the left pane of the Kturtle window. When we tell the turtle to move, we also must tell it *how much* to

move. We express this in terms of the number of squares the turtle has to move on the canvas grid. The turtle also leaves behind a trail as it moves, so we can use it to draw on the canvas.

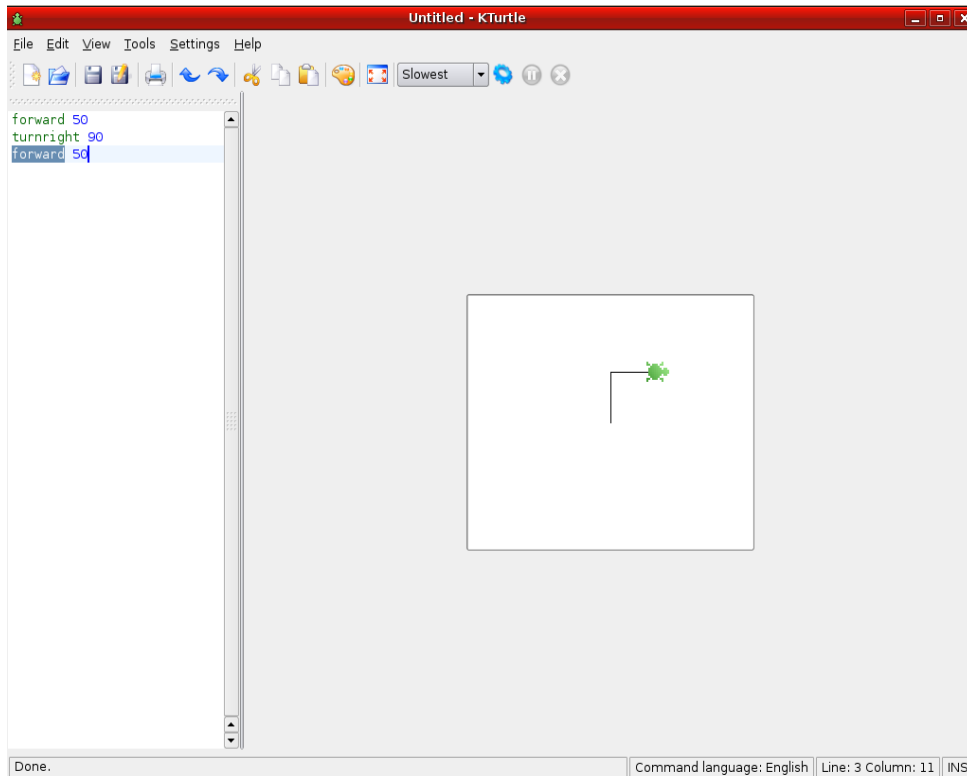
Now let us run a small, one-line program to move the turtle forward by 50 steps. Note that since the canvas grid is very fine, this is not a large distance. To do this:

- Open the K Turtle application as show in the screenshot
- Enter **forward 50** into the left pane
- Click on the Run button indicated in the above screenshot

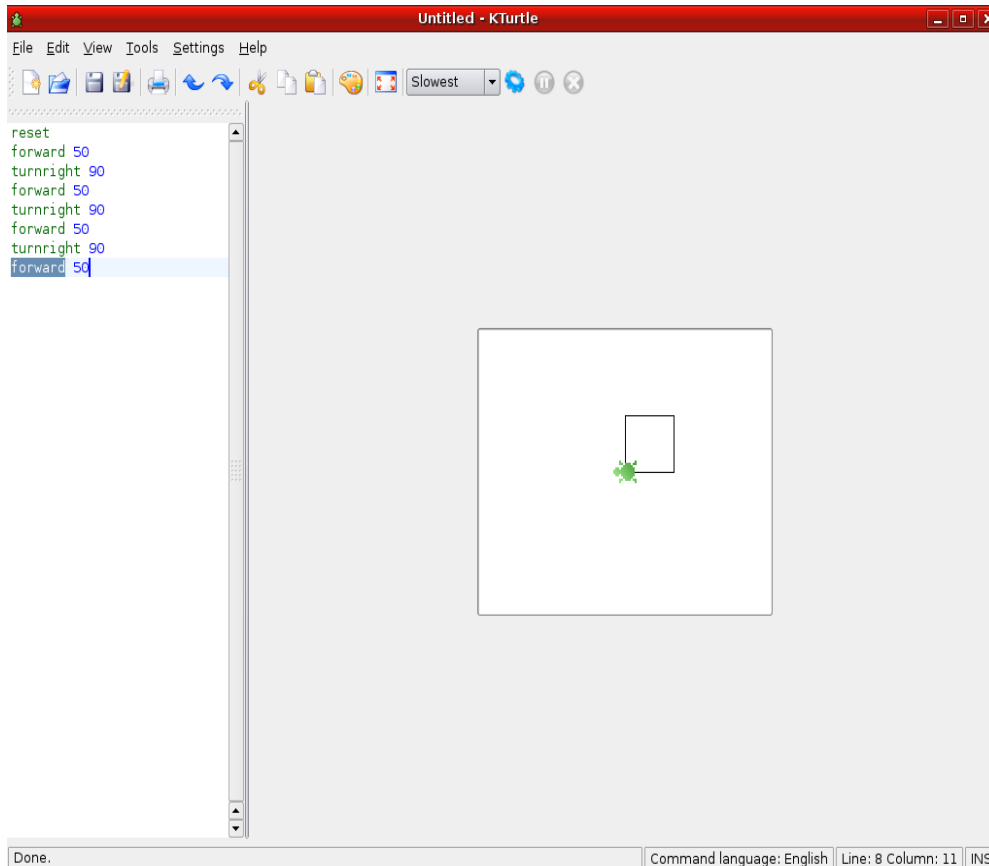
You should find that the turtle moves a short distance forward. You may want to reduce the execution speed to see the movement more clearly. The following screenshot shows the effect of running this simple program.



Next let us say we want the turtle to turn right and then move another 50 steps. We then write in the left pane **turnright 90**, where 90 represents the angle in degrees that the turtle has to turn, and again **forward 50** to get the following.



Now, let us say we want to use what we have learnt to draw a square. We will abandon the current canvas and start afresh by using the **reset** command. The following screenshot shows the program and the square drawn by the turtle.



Simple keywords

So far we have used the **forward**, **turnright**, **reset** and **repeat** keywords. A few more commands are introduced in the lesson plan and exercises. A few simple commands are listed below.

reset	Positions the turtle at the centre of a blank canvas
turnright x	The turtle turns right by x degrees
turnleft x	The turtle turns left by x degrees
center	The turtle moves to the center of the canvas
penup	The turtle lifts its pen and leaves no trail as it moves
pendown	The turtle puts its pen down and leaves a trail as it moves
penwidth x	The width of the trail drawn by the turtle
hide	The turtle becomes invisible
show	The turtle becomes visible
wait x	The turtle waits for x seconds before executing the next instruction

Logo can also be used to accomplish more complex drawings, and other tasks such as text processing and mathematical calculations. We will see some of these in the next lesson. The KTurtle application contains several example programs. Click on *File->Open Examples* in the KTurtle menu and run these examples to see what the programs do. KTurtle comes with an excellent manual with detailed instructions for learning Logo. This can be accessed by hitting F1 or by clicking *Help->KTurtle handbook* on the KTurtle window.

Lesson Plan:

The lesson can start by refreshing the students' memories about the hopping game they played earlier. Just as they directed their friends, now we will be directing a turtle in the computer. The differences between the previous activity and using Logo will have to be specified:

- Here the squares are very small and cannot be seen. They are also much more in number. Therefore for the turtle to move a small distance, we will have to ask it to move many squares. Here give a demo showing how much the turtle moves when the following pairs of commands are entered and run. Also show that the **reset** command moves the turtle back to its original position. Make it a point to set the execution speed to 'slowest' in the drop down menu. Also, run each pair of commands separately for simplicity. You can also ask the class to suggest the distance to be moved.

```
reset  
forward 50
```

```
reset  
forward 100
```

```
reset  
forward 200
```

In the last case, the turtle wraps around the canvas and draws from the other side. Explain that this is because the turtle cannot move outside the canvas.

- By now the students will have noticed that the turtle draws a line as it moves. Explain that this can be used to create drawings on the canvas. This is also a good time to introduce the commands **penup** and **pendown**. Explain that the command **penup** causes the turtle to lift its pen and so not draw as it moves. Demonstrate this by showing the difference between these two programs:

```
reset  
forward 100
```

```
reset  
penup  
forward 100
```

You can start using the word program at this stage. Just explain that a list of instructions given to a computer is called a **computer program**.

Now demonstrate use of the **pendown** command as the opposite of **penup**. Here is a program that draws a dashed line. Explain how it works first before giving a demonstration.

```
reset  
forward 10  
penup  
forward 5  
pendown  
forward 10  
penup  
forward 5  
pendown  
forward 10  
penup  
forward 5  
pendown  
forward 10
```

Introduce the **center** command with an example:

```
reset  
forward 100  
center  
backward 100  
center
```

Earlier in the hopping game, the directions used were **turnright** and **turnleft**. Now, mention that the turtle needs to be told *how much* to turn right or left. Demonstrate these to show how the amount to be turned is indicated from 0 to 90:

```
reset  
forward 100
```

```
reset  
turnright 30  
forward 100
```

```
reset  
turnright 60  
forward 100
```

Demonstrate that turning right or left by 90 takes a full turn in that direction:

```
reset  
forward 100  
center  
turnright 90  
forward 100
```

Just ask the students to remember that turning by 90 will cause a full right turn or left turn.

Note: It is desirable that the demos are done using a projector so that the class can be involved. If a projector is not available, the demos can be done on one computer for small groups of students. It is also important that every student gets some experience in entering a small program and running it.

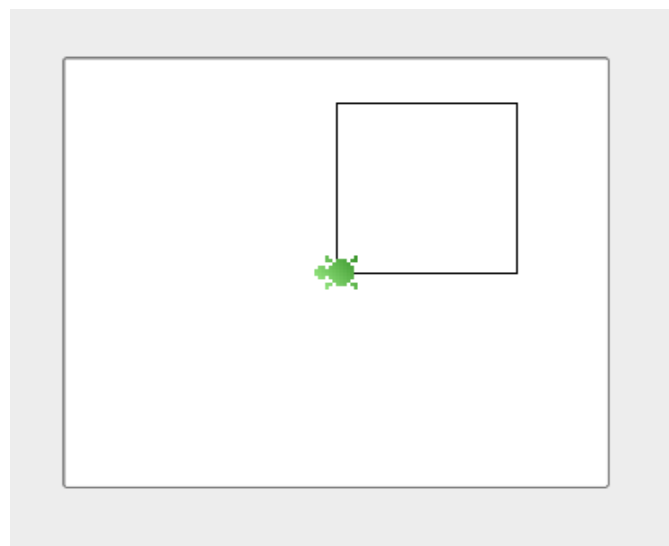
Worksheet

1. A list of instructions given to a computer is called a computer p_____.
2. Match the Logo commands on the left with what it makes the turtle do.

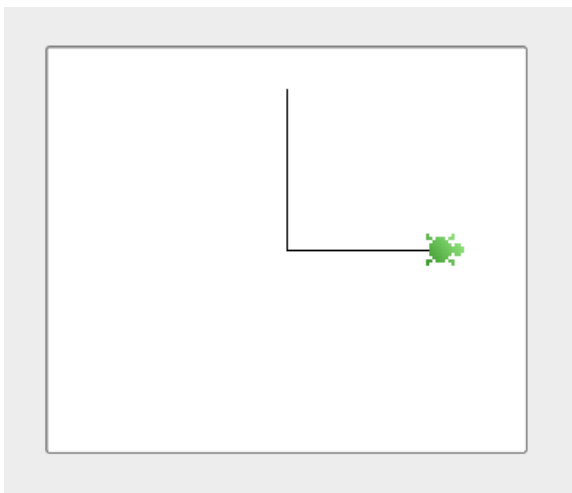
turnright 90
turnleft 30
forward 20
backward 20
turnright 30
turnleft 90

3. Write a Logo program to draw a square with sides of length 100. The first two lines of the program are given. Write the remaining commands and run the program!

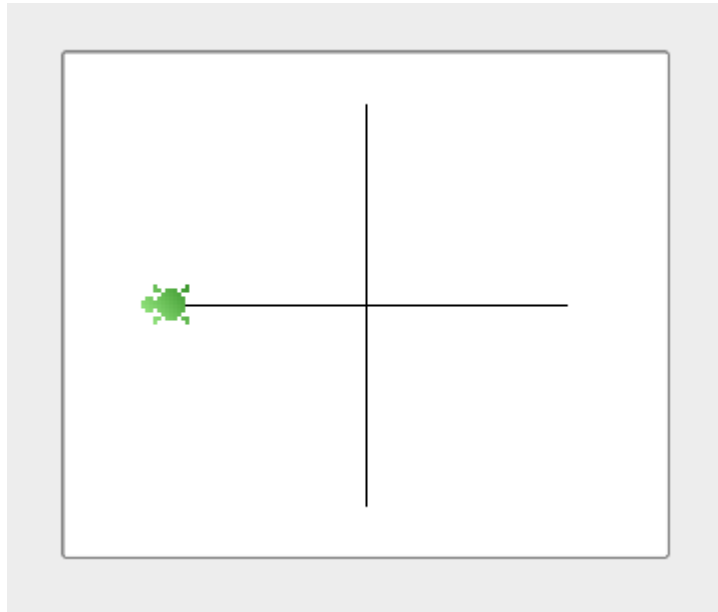
forward 100
turnright 90



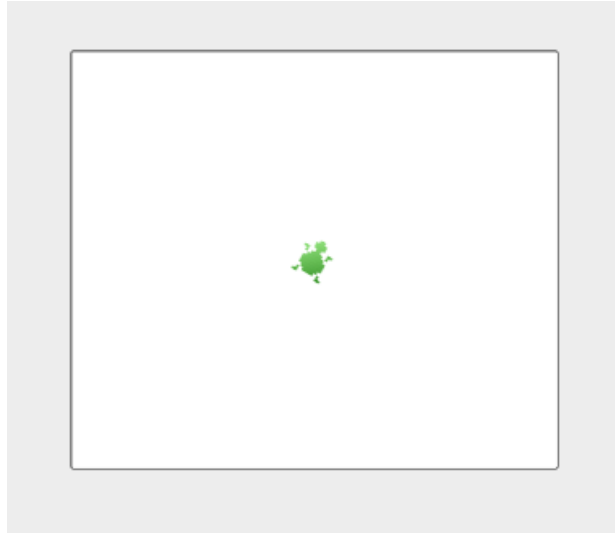
4. Write a Logo program to draw the letter 'L' on the canvas as shown. The two arms of the L are of length 100. (Hint: Move the turtle forward by 100, bring it backward by 100, then turn right and move forward by 100.)



5. Write a Logo program to draw a plus sign on the canvas as shown. (Hint: Continue the method used for drawing the 'L'.)



6. What shape will the turtle draw when the following program is run? Draw it on the empty canvas on the right. (The turtle has already performed the first two commands.)



```
reset  
turnright 30  
backward 140  
turnright 60  
forward 140  
turnright 60  
backward 140
```

7. In this exercise you will learn a new command. Enter this program and see what it does!

```
reset  
forward 100  
print "Hello"  
turnright 90  
forward 100  
print "I am"
```

```
turnright 90  
forward 100  
print "a"  
turnright 90  
forward 100  
print "turtle!"
```

Now add the command **penup** as the second line of the program and run it. What difference do you see? Can you write a program that prints your name on the canvas?

8. This program does something very interesting. Enter it and see what!

```
reset  
penup  
forward 120  
pendown  
penwidth 1  
backward 30  
penwidth 5  
backward 30  
penwidth 10  
backward 30
```

```
penwidth 15  
backward 30  
penwidth 20  
backward 30  
penwidth 25  
backward 30  
penwidth 30  
backward 30  
penwidth 35  
backward 30
```

Now, can you say what does the command `penwidth` does?

9. Write any program you can think of by using the commands you have learnt so far!