Unit 5: Lists, Graphics, and Dynamic Programs | **Skill Builder 2: Lists and Math Boxes**

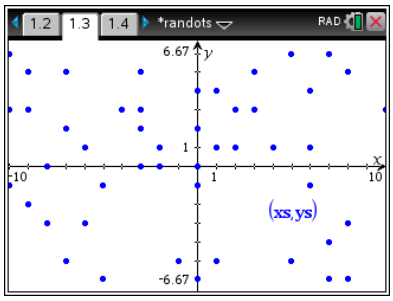
In this lesson, you will learn about controlling individual elements in lists and discover the method of running a program dynamically from a Notes app Math Box.

Objectives:

- Generate random decimal values on a desired interval
- Use the 'interactive' property of Math Boxes in a Notes app to make a program run dynamically

In the previous lesson, we wrote a program that generates two lists of random integers and displays those lists in the form of a scatterplot in a Graphs app.

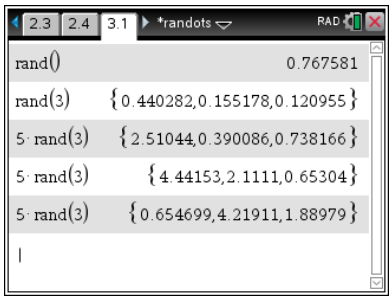
Recall the output of the **randots(n)** program. A sample is shown to the right.



The **randots( )** program was limited to generating only *integral* values for the x- and y-coordinates of the scatterplot so all the points are located on the grid points of the graph. In this activity, we'll extend the program to include non-integer values so that the plot will be more densely packed.

From Integers to Decimals

Rather than use **randInt**(-10, 10, *n*) we will use the **rand( )** function to generate a decimal (between 0 and 1) and then 'scale' the decimal so that the points fill the screen (using the Standard window).



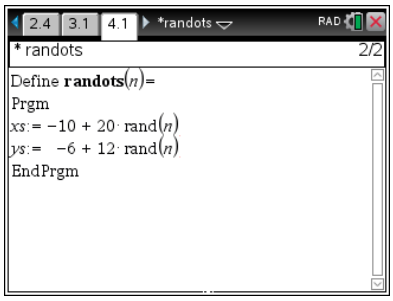
First, let's examine the **rand()** and **rand(*n*)** function outputs in a Calculator app. Notice that without an argument, **rand()** generates a single random decimal between 0 and 1. With an argument of 3, however, it gives a *list* of three decimals between 0 and 1. This random list can then be used as part of an expression to change those values to go beyond the range of 0 to 1. For example, **5*rand(3)** produces a list of three random decimals between 0 and 5.

Teacher Tip: A number times a list multiplies each element of the list by that number:
5 * {1, 2, 3} becomes **{5, 10, 15}**

We use this knowledge to generate a random decimal between -10 and 10. That's a *range* of 20 units. We'll start the expression with **-10**, and add a random value from 0 to 20 using **20*rand( )**.

Teacher Tip: Using **rand( )**, we can generate one random decimal number in the range [A, B] using the formula
n:= A+(B-A)***rand()**
that is, it takes the form n:= *starting_value* + *size_of_range* * **rand()**
There's a lot of rich mathematics in graphical programming.

Modifying the randots(n) Program

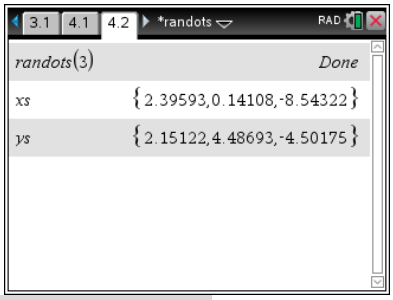


1. In our **randots(n)** program, change the **randInt** functions to an expression using **rand( )**.

   ***xs*:= -10 + 20*rand(*n*)**
   ***ys*:= -6 + 12*rand(*n*)**

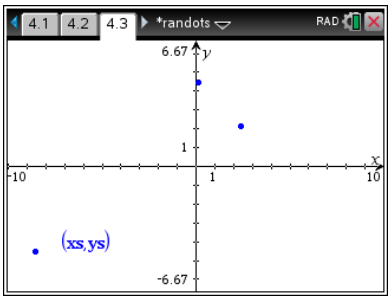
These expressions generate *n* decimal values of **xs** between -10 and 10 and *n* values of **ys** between -6 and 6.

2. Run the program in a Calculator app, and observe the values of *xs* and *ys*.

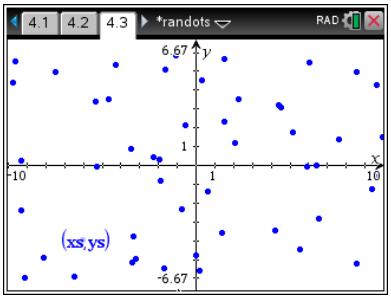


Teacher Tip: When working with a new function in the TI-Nspire™ CX, it helps to study the function in a Calculator app before trying to use it in a program, Also, looking up the function in the Catalog will provide some help with the required and optional arguments.

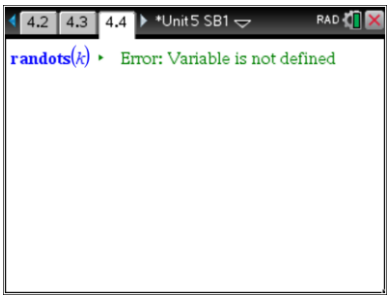
3. Finally, observe a scatterplot of (*xs*,*ys*). We used **randots(3)** for the image to the right.



4. Run the program again with a larger value of the argument. **randots(50)** was used for the image to the right.

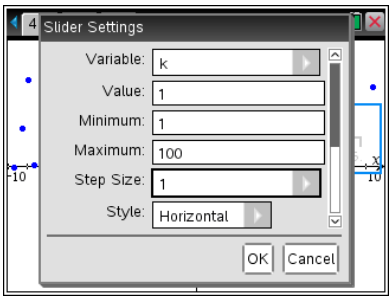


Teacher Tip: The next section introduces another way to run a program using a Math Box in a Notes app. This feature is unique to the TI-Nspire™ CX platform and gives the ability to run a program whenever a global variable used by the program or an argument to the function changes value. The change triggers a rerun of the program dynamically without having to enter it again in a Calculator app**.**

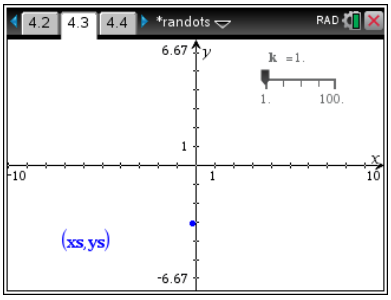


Dynamic Programs

1. To have the program run *dynamically*, add a **Notes** app to the problem.
2. Insert a Math Box into the Notes app by selecting **menu > Insert > Math Box**.
3. Type the program name (or use the **var** key) with a variable (we used *k*) as the argument into the Math Box, and press **enter**.
   - An error message is displayed because the argument is an undefined variable. This will be resolved shortly. Press **enter** or click **OK** to dismiss the dialog box.
   - The actual parameter you use does not have to be the same as the argument that you used in the Program Editor. It represents the value that is 'passed' to the program's argument.



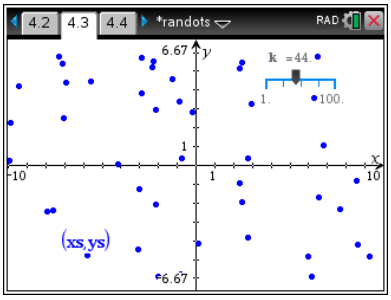
4. In the Graphs app containing the scatter plot, insert a slider (**menu > Actions > Insert Slider**).
5. **Important!** In the Slider Settings box, make the variable name for the slider the same as the argument used in the Math Box. Recall that we used the variable *k*.
6. Set the Value to 1, the Minimum to 1, the Maximum to 100, and the Step Size to 1 as in the image to the right.
7. Press **enter** or click **OK** to place the slider on the Graphs app.
8. Move the slider to a convenient location, and press **enter** or **click** to place it.



You will see that all the dots disappear except for one. This is a result of the program responding to the new value for *k* (1 in our demo) causing the program to respond from the Math Box in the Notes app.

If you look at the Notes app, you'll see that the error message has been replaced by the word 'Done'. The program is running properly now that the variable *k* is defined.

9. In the Graphs app, operate the slider for *k*. Each time the value of *k* changes, it causes the program to rerun, thus generating a whole new set of points for the scatterplot. This 'dynamic' nature of programs residing in 'interactive' Math Boxes is a very powerful feature of the TI-Nspire.



Teacher Tip: When something changes the value of any 'global' value used by a program that is running in a Notes app Math Box, the program is triggered to run. In this example, the slider changes the value of *k* which is the argument to the program in the Notes app Math Box. An entirely new set of points is plotted when the value of k is changed even if the value of *k* is increased or decreased by one.
A ***challenge*** would be to just add *one* point to the current scatterplot when the value of *k* increases by one and remove the last point plotted when it decreases by one.