10. Programming with TI InterActive!

introduction

This file is in a different format from the other materials. Instead of having a number of student tasks, help, hint and other files there is just a single document explaining the way in which programs can be developed and used within TI InterActive!.

function definitions

Review of function definitions

The basis of programming is the use of the Math Box to define a function. Here are a few examples of simple ways to define and evaluate a function.

$x^2 \rightarrow f(x)$	$f(2)$	$f(p)$
"Done"	4	p^2
$g(x) := x^3$	$g(2)$	$g(n)$
"Done"	8	n^3
Define $h(x) = x^4$	$h(2)$	$h(g(f(y)))$
"Done"	16	y^{24}

A function can also be 'piecewise' defined. Here are a few examples.

The first uses the 'when' function and the input is entered as **p(x):=when(x>0,x,-x)**.

$$p(x) := \begin{cases} x & x > 0 \\ -x & \text{else} \end{cases}$$

"Done"

$p(2)$ 2 $p(-2)$ 2

This can also be entered using the {()} menu symbol from the Math Palette and selecting the symbol. The dialog box allows you to select the number of pieces.

$$q(x) := \begin{cases} 0 & x < 1 \\ 1 & x < 2 \text{ and } x \geq 1 \\ 2 & x > 2 \end{cases}$$

"Done"

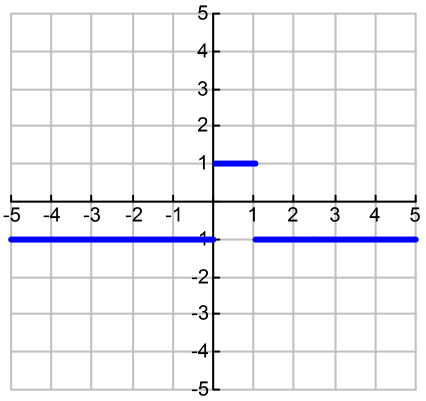
$q(-2)$ 0 $q(1.5)$ 1 $q(3)$ 2

Another way to produce multi-part definitions is to add a number of `when' statements together:

$$r(x) := \begin{cases} -1 & x < 0 \\ 0 & \text{else} \end{cases} + \begin{cases} 1 & x \geq 0 \text{ and } x < 1 \\ 0 & \text{else} \end{cases} + \begin{cases} -1 & x > 1 \\ 0 & \text{else} \end{cases}$$

"Done"

The graph of r(x) clearly shows the 3 pieces.



The piecewise structure can also be inserted using the menus of the Math Palette:

Math → Algebra → Piecewise function.

If you follow a different menu path you will find the program structure commands:

Tools → Program Logic → Control.

Before we can use these commands we need to learn about an alternative way to define a function.

multi-statements

Multi-statement functions

The following function expects a positive integer x as input and returns different values depending on whether x is positive (x/2) or negative (3x+1).

This example illustrates the use of the If..Then..Else..EndIf structure.

Note that:

a. the function can only be set up using the 'Define' structure,
b. the Input must be set to Plain Text,
c. statements are separated by two colon symbols (::),
d. the body of the definition is enclosed in a Func...EndFunc structure,
e. a value to be assigned to the function is set using Return.

$$\text{Define } s(x) = Func :: If \text{ int}\left(\frac{x}{2}\right) = \frac{x}{2} \; Then :: \text{Return } x/2 :: \text{Else} :: Return \; 3 \cdot x + 1 :: EndIf :: EndFunc$$

$s(5)$	$s(4)$	$s(s(s(s(s(117)))))$
16	2	22

loops

Functions with loops

In order to produce a tidy looking program it is helpful to break the program into separate lines.

Remember to use the Plain Text mode of input and to use CTRL-Enter to begin a new line. You can create variables within a program as well as access previously defined variables. If you do not need the internal variables to be accessible outside the program then define them to be 'Local'.

The next example computes the factorial of a given positive integer.

```
Define fac(x)=Func
::Local f,c
::1→f
::For c,1,x
::f*c→f
::EndFor
::Return f
::EndFunc
    "Done"
```

```
fac(5)
120
```

Using programs to extend TII's functionality

This short introduction concludes with the development of a tool to help with data-handling using lists. It will be used to take a list of data of a variate recorded at regular intervals and compute a new list of data containing the moving averages of the data over a given number of intervals. List L1 contains data on 36 monthly house sales for an estate agent (realtor) taken from the book "Statistics" by Roger Fentem, Collins Educational ISBN 0-00-322371-X.

```
Define movave(list1,m) = Func
::Local n,c,r
::dim(list1)→n
::seq(c,c,1,n)→list2
::{0}→list3::{0}→list4
::For c,1,n-m+1
::mean(seq(list1[r],r,c,m+c-1))→list3[c]
::(m-1)/2+c→list4[c]
::EndFor
::Return approx(list3)
::EndFunc
    "Done"
```

L1
15.
14.
13.
14.
13.
14.
17.
15.
18.
19.
16.
15.
16.
13.
14.

A MathBox is used to create a program to define a function called 'movave'. This function has two arguments; the first argument 'list1' is the original data list while the second argument 'm' defines the number of months to use when making the moving averages.

The function returns the list 'list3' of moving averages of data from 'list1' taken 'm' at a time. As a side effect lists 'list2' and 'list4' are created - 'list2' is a count of the number of elements for the data set in `list1', and 'list4' contains the adjusted count numbers for corresponding moving averages.

The program is entered in plain text input mode.

Programming commands are found in the MathBox's Tool menu. Each line of the program ends with Ctrl+Enter. Each line begins with a pair of colons "::".

```
movave(L1, 5)
{13.8, 13.6, 14.2, 14.6, 15.4, 16.6, 17., 16.6, 16.8, 15.8, 14.8, 14.4, 14.2, 14.4, 15.6, 16., 16.8, 17.8, 18.,
17.6, 17.8, 17.2, 16.6, 16., 16., 16.4, 17.4, 18.6, 19.6, 20., 20., 18.8}
```

Since 'list2' and 'list4' were not declared as 'Local' they are still available:

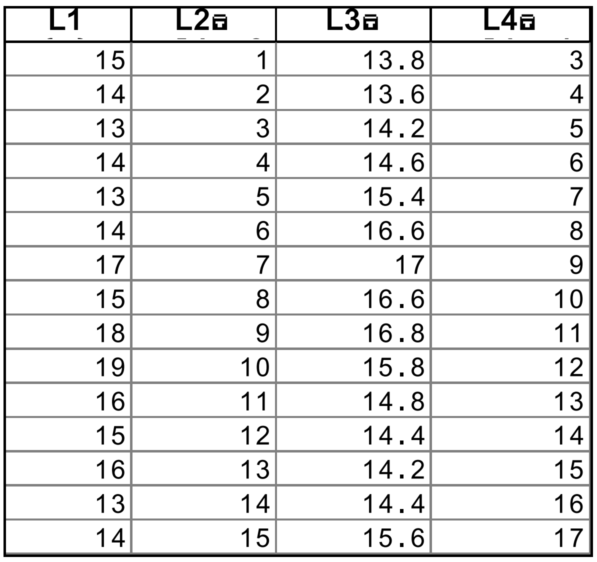
list2

$\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36\}$

list4

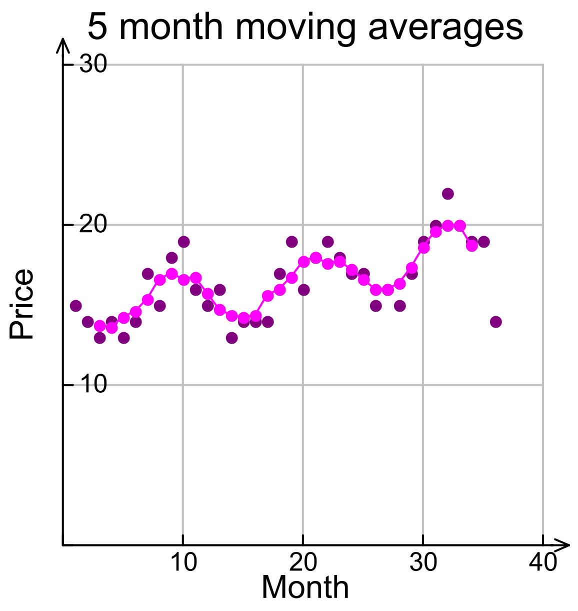
$\{3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34\}$

This function and lists can be used to define list L2, L3, L4 from which we can plot the original data and the moving avarages.



L1	L2	L3	L4
15	1	13.8	3
14	2	13.6	4
13	3	14.2	5
14	4	14.6	6
13	5	15.4	7
14	6	16.6	8
17	7	17	9
15	8	16.6	10
18	9	16.8	11
19	10	15.8	12
16	11	14.8	13
15	12	14.4	14
16	13	14.2	15
13	14	14.4	16
14	15	15.6	17

Now we can draw the scatterplot of L1 against L2, together with the xyline of L3 against L4. By changing the value of $m = 5$ we can see how the moving averages smooth data as m increases.



<u>Note</u>: It would be good for the program to be able to set up the statistical plots directly, but we do not have access to the Xmin, Xmax etc. variables. It would also be beneficial to have the option of smaller size dots for data points.