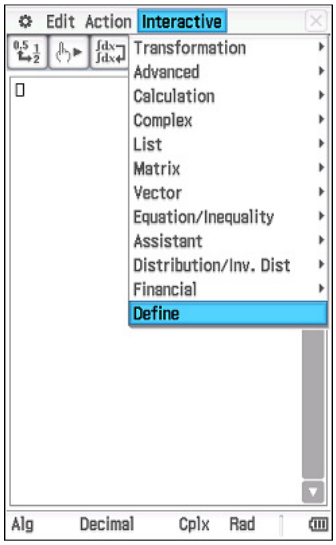
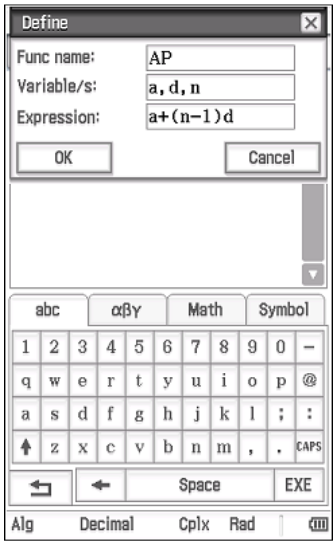
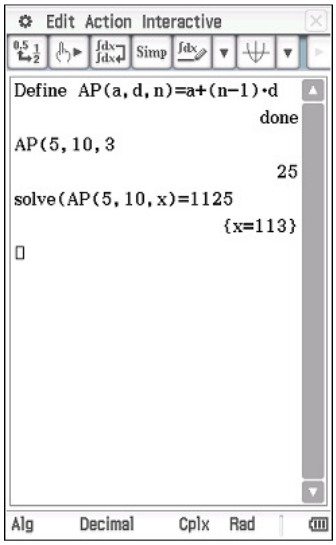
ClassPad II - How to add your own functions and programs

1. Define new functions in Main

This is useful for simple functions which use a formula to return a single value, such as the n^{th} term of an AP. Open the Main application. Use the ***abc*** tab to enter the function name, avoiding names reserved by ClassPad. Switch to the ***Var*** tab for variables and expression.



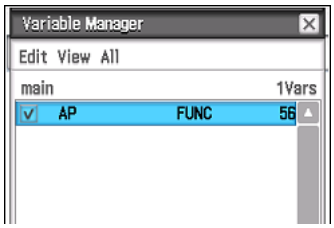


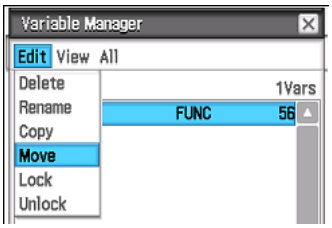


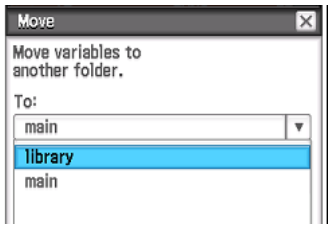
Tip:
A user-defined function can only contain a single mathematical expression and must not contain any commands. You may want to consult the user manual: As an example, it is OK to use binomialCDf, but not binomialCD, as the former is a function but the latter is a command!

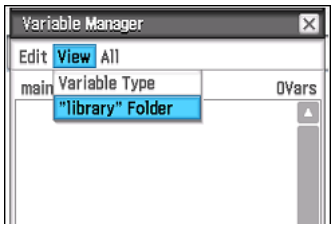
If a function doesn't behave as expected, try defining it again. If you use an existing user defined function name, ClassPad will simply overwrite the old function with the new one.

Once created in Main, a good idea is to use the Variable Manager to shift functions into the library folder, so that the function can be accessed from within eActivities, as well as the rest of the calculator.





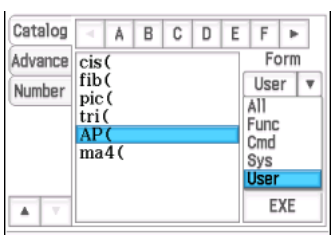


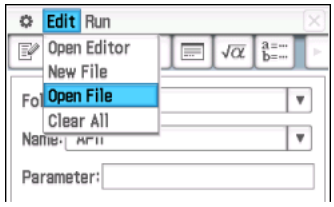


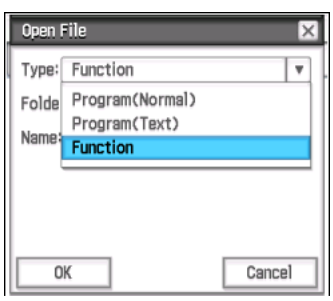
To delete a function, follow the above steps to move the function, but choose Delete rather than Move.

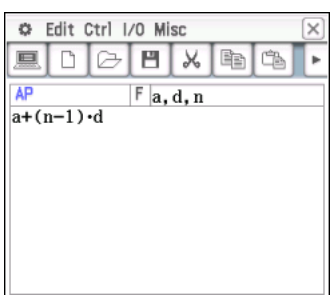
Videos 043, 044, 071 explore defining functions at www.classpad.com.au

User defined functions can easily be found in the Catalog by selecting the User option. They can also be edited in the Program application.







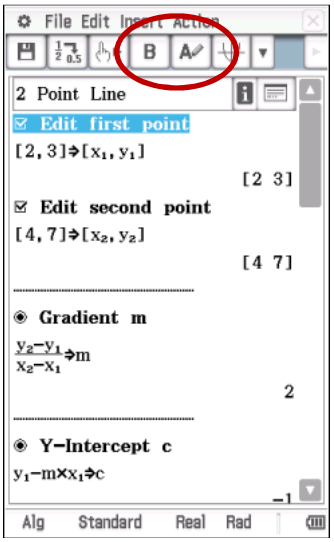


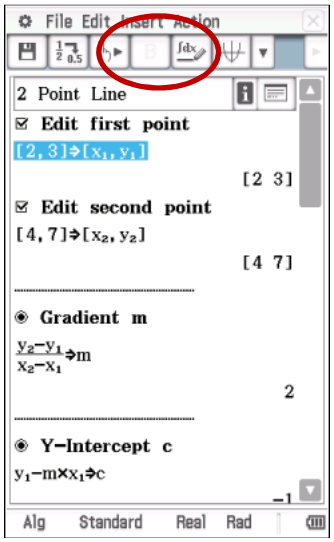
Make any corrections or edits to your function and then tap Edit, Save File.

2. Create an '*open program*' in eActivity

'*Open*' programs allow the user to see all the steps involved in producing the result and allow for easy modification on the fly. Programs can be written either (A) in the main body of an eActivity; or (B) in a Main strip inserted into the main body of an eActivity.

A. Writing a program in the main body has the advantage that we can add line by line comments that explain each step in our program. However, the downside is that each program will require its own eActivity.
Open the eActivity application and start a new eActivity - tap File, New.





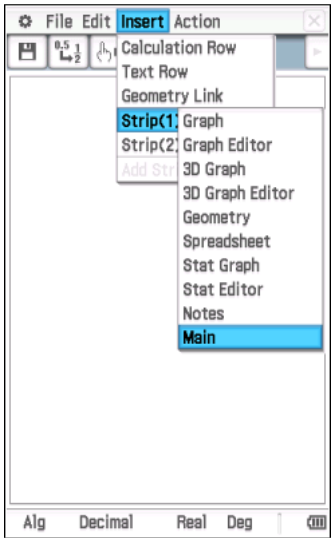
An example is shown at left.

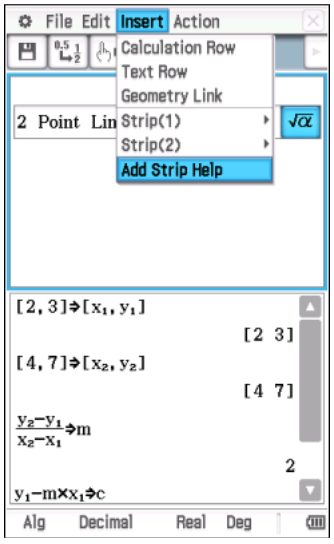
Note how to store multiple variable values using just one line: $[2,3] \Rightarrow [x_1, y_1]$ stores the values of 2 and 3 as x_1 and y_1 respectively.

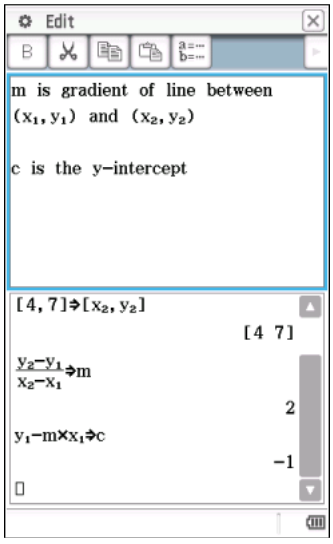
Also note the difference between a text line, as selected in first image, and a calculation line, as selected in second image. Use ∫dx and A/ to toggle between the two types.

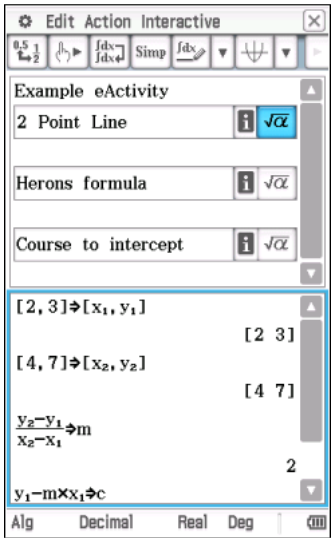
Bold selected text using B.

B. Writing a program using a Main strip allows us to store multiple programs within a single eActivity. Comments can be added by inserting Strip Help from the Insert menu.









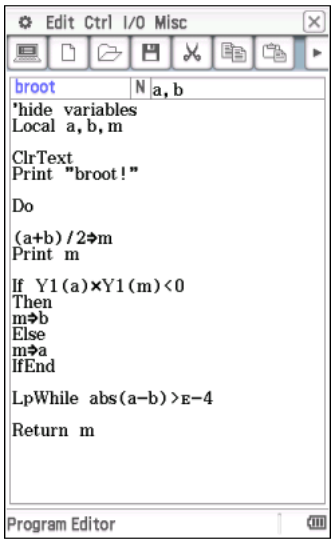
To create a variable such as x_1 use an x from the **abc**/abc tab and the subscript from the **abc**/Math tab.

Use File-Save to save a copy of your eActivity, choosing a suitable filename of up to 20 characters.

Videos 631, 632 and 633 explore these types of eActivities at www.classpad.com.au
A handout on creating eActivities is available from the Helpsheets page at charliewatson.com/classpad

3. Create a program in Program.

For this example, we'll use the bisection method to find a root, between a and b, of the function Y1 in Graph and Table. (NB This method assumes that Y1(a)×Y1(b)< 0 for a solution to exist.)
Open the Program application, tap Edit, New File, enter a name such as ***broot*** and tap OK.



Notes

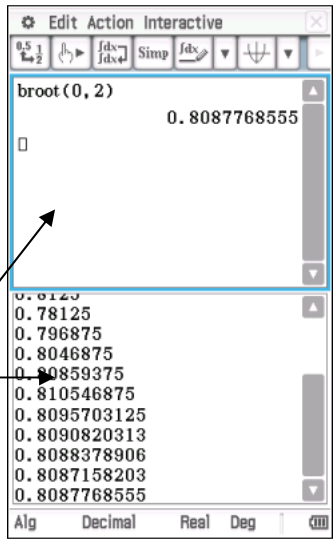
- This program has two input variables, a and b
- Lines starting with an apostrophe are ignored
- Start a new line by pressing the EXE key

Local variables are used only within program and then cleared.

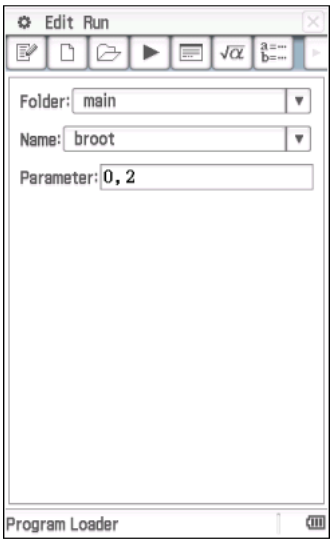
ClrText clears the result screen.

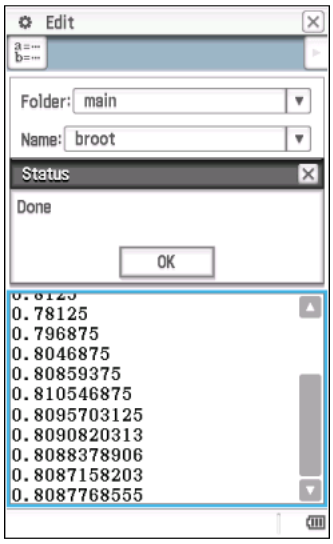
Print writes values and text to the result screen.

Return writes variables to the main screen, when program is run in main.



Once your code has been entered, tap Edit, Save File and then tap . Check that the name of the current program is broot, enter 0 and 2 as the parameters (a and b) and tap . If all goes well, the Status box appears with the message 'Done' and you can tap OK. Tap ⊠ to return to the program application.



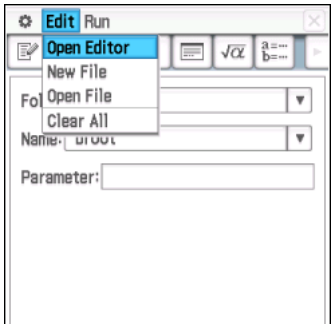


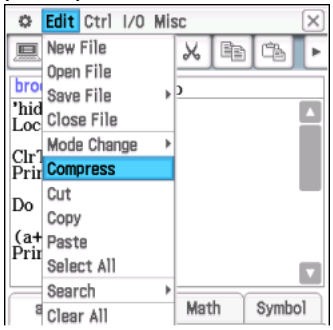
This program has no error trapping and the output is very basic. If ClassPad gets stuck in an endless loop hold down the Clear button.

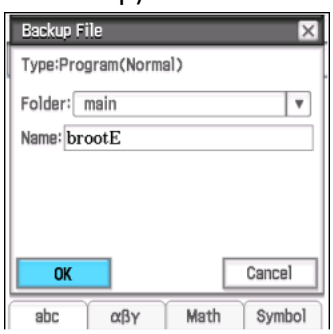
Consult the program chapter in the ClassPad manual for methods to handle more complex text output, including functions to handle strings.

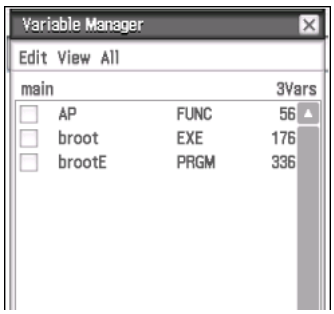
Videos 984 and 985 explore programming at www.classpad.com.au

Tap Edit, Open Editor to modify or debug your program. Choose Compress to reduce file size and prevent further editing (you will be prompted to save an editable backup).



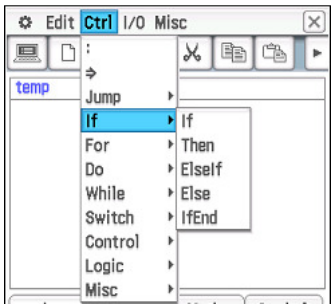


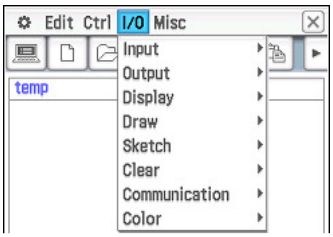




In the variable manager, inside the main folder, you should find broot and brootE amongst the variables. Delete, move and rename your programs in this folder.

Additional programming tips - Ctrl and I/O Menus





If ... Then ... Else ... IfEnd
Use for branching and decision making.

```
temp N a,b
ClrText
Print "Difference is"
If a>b
Then
Print a-b
Else
Print b-a
IfEnd
```

Program takes two values and prints out the difference.

Else is optional.

Can also include ElseIf statement.

For ... Next
Use for repeating fixed number of times

```
temp N n
Local i,n,s
ClrText
Print "Sum of squares is"
0⇒s
For 1⇒i To n
s+i^2⇒s
Next
Print s
```

Program sums the squares of all integers from 1 to n.

Can add 'Step *k*' to increment *i* in steps of *k*.

Do ... LpWhile
Use for looping until a condition is met.

```
temp N n
Local n,x
ClrText
Print "Smallest number is"
0⇒x
Do
x+1⇒x
LpWhile x^2<n
Print x
```

Programs finds the smallest number which has a square greater than n.

Beware of writing an endless loop! Press the Clear button to break out of such a program.

While ... WhileEnd
Use for looping, but checks for condition at start of loop.

```
temp N n
Local n,s
ClrText
Print "Sum of roots is"
0⇒s
While n≥1
s+√(n)⇒s
n-1⇒n
WhileEnd
Print s
```

Program sums the roots of integers from 1 to n.

Again, beware of endless loops.

Input
Use for obtaining a single number input from user.

```
temp N
Local m,n
Input n,"Integer","Factors"
ClrText
For 1⇒m To n
If n/m=int(n/m)
Then:Print m:IfEnd:Next
```

Program displays all factors of any integer.

Note use of colons to enter multiple commands on a single line.

GetPen and **GetKey**
Use to get location where screen is tapped or code of key pressed.

```
temp N
Local a,b:ClrText
Print "Tap the screen"
GetPen a,b
Do:GetPen a,b:LpWhile a=0
Print a
Print b
Print "Press a key"
GetKey a
Do:GetKey a:LpWhile a=0
Print a
```

Program prints the pixel coordinates of where screen tapped and then code of key tapped (see manual for code listing).

Note use of GetPen and GetKey before entering Do:LpWhile loop to flush any stored values

PrintNatural and **Message**
Pause and display result.

```
temp N n
Local n
ClrText
PrintNatural n,"Fraction"
Message "Finished","OK"
```

In Main, check the setup is Standard rather than Decimal. Run the program by entering a decimal number. The program displays the decimal as a fraction and then the message "Finished".

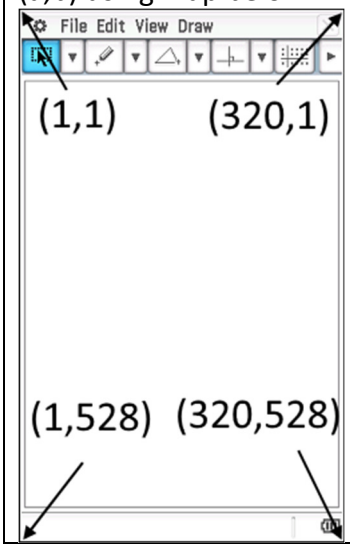
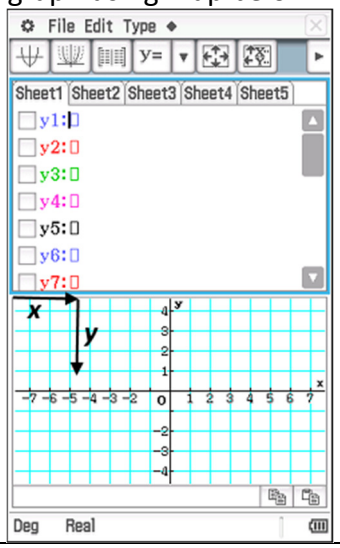
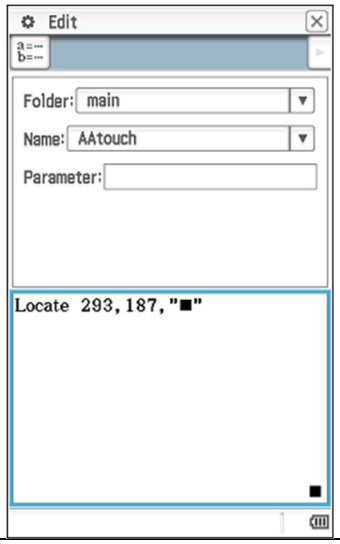
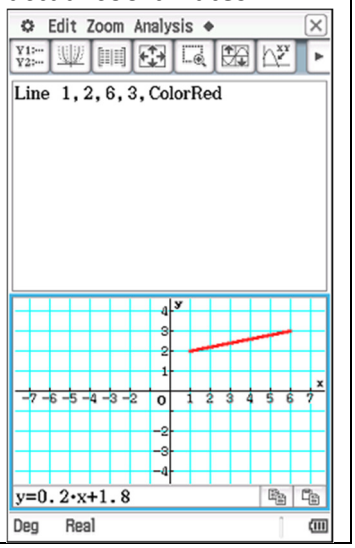
InputFunc
Use for obtaining a function from the user.

```
temp N
Local f,n,soln
InputFunc f(x),"f(x)","F"
ClrText
For -2⇒n To 2
Print f(n):Next
solve(f(x)=0,x)⇒soln
Print soln
```

Program displays value of function for x=-2 to 2 and then finds roots.

Strings after f(x) are prompt and title for inputbox

Screen dimensions - most drawing functions refer to actual coordinates on graph screen, but some use pixels (across, down) from top left-hand corner of either whole screen, graph window or program output window.

GetPen a,b will return (a,b) using map below	**Text** x,y,"?" will plot ? on graph using map below	**Locate** a,b,"?" will plot ? as shown below	Most functions use actual coordinates
			

Plot x,y,color	Displays pointer and plots point at (x,y). Color optional.
PlotOn x,y,color	Plots point at (x,y). PlotOff x,y deletes this point.
Line x_1,y_1,x_2,y_2,color	Draws line from first to second point
Distance x_1,y_1,x_2,y_2	Calculates distance from first to second point
Horizontal y,color	Draws horizontal line through y-coordinate
Vertical x,color	Draws vertical line through x-coordinate
Circle x,y,r,color	Draws circle centre (x,y) with radius r
PxlOn a,b,color	Plots 1x1 pixel at (a,b) pixels from TLHC. PxlOff a,b erases.
Text a,b,?,color	Writes text at (a,b) pixels form TLHC, where ? is number, variable or "Text"
Colours	*Use Color# where # is Black, Blue, Red, Magenta, Green, Cyan or Yellow*

GRAPH and TABLE

ViewWindow	Set using ViewWindow xMin,xMax,xScale,yMin,yMax,yScale ViewWindow alone sets default values, same as ClrGraph Get current values using inbuilt variables xMin, xMax, etc
DispText	Use to re-display output window following display of graph, table or other window.
Define y1(x)=2x+1	Use to write function to y1-y20 in Graph and Table
GTSelOn/Off n	Select/Deselect graph yn for graphing
DrawGraph	Draws currently selected graph(s)

STATISTICS

Create lists	{1,2,3,4,5}=>list1, {2,4,7,8,10}=>list2, etc
Individual list items	list2[3] would return 7 using above list2
Calculate stats	OneVariable list1,1
Retrieve stats	Use symbols and print: Print σ_x, Print $\hat{x}$
Set up stat graphs	StatGraph 1,On,Scatter,list1,list2,1,Square *(marker can be Square/Cross/Ldot/Dot)*
DrawStat	Draws all selected stat graphs
Perform regression	LinearReg list1,list2
DispStat	Displays last regression coefficients. Get values using aCoef, bCoef, rCorr, etc

STRINGS

InputStr s3,p,t	Prompt user to enter a string to store in s3 using prompt p and title t
StrJoin s1,s2,s3	Joins s1 and s2 together and store in s3
StrLeft s1,n,s3	Takes n left characters
StrMid s1,n,s3,m	Takes m characters starting from nth *(m optional)*
StrRight s1,n,s3	Takes n right characters
StrLen s1,s3	Length of string
StrLwr s1,s3	Converts to lowercase
StrUpr s1,s3	Converts to uppercase
StrSrc s1,s2,s3,n	Looks for s2 in s1, starting at n^{th} character *(n optional)*
strToExp s1	Converts s1 to an expression and executes
ExpToStr e1,s3	Converts expression e1 into string and save in s3

NumToStr v1,"Fix0",s3 Converts variable value to string in format Fix0 to Fix9