Programming in HP PPL

The HP Prime Programming Language (HP PPL) was designed for general use. It is simple to get started with, but powerful enough for a wide range of applications. HP PPL supports basic programming structures, such as loops and branches of various sorts. It has an extensive set of graphical, input/output, string, matrix and other types of commands. Within any program, you also have access to all the mathematical functions of your HP Prime. Finally, you have access to any App Function from within a program.

Programming Options

You can create an HP PPL program using a number of methods, as shown in the table below. Each method has its benefits and drawbacks. No matter which method you use, the HP Prime Connectivity Kit assures that you can send your program to your HP Prime and your HP Prime Virtual Calculator.

Method	Benefits	Drawbacks
HP Prime	• Mobility: work on your program anytime, any place • Debug: use the built-in check and debugger tools to check your program	• Must use menus to find and enter commands-or use the alpha shift to enter a command letter by letter
HP Prime Virtual Calculator	• Fast: use your PC keyboard to just type in your program • Debug: use the built-in check and debugger tools to check your program	• Must have access to your PC
HP Prime Connectivity Kit	• Fast: use your PC keyboard to just type in your program • Commands, strings, and comments are color-coded for easy identification and reading	• No debugging tools • Must have access to your PC
Your Favorite Word Processor	• Fast: use your PC keyboard to just type in your program	• Must copy and paste the program into the virtual calculator or connectivity kit • No debugging tools • Must have access to your PC

Getting started: user-defined functions

The simplest way of extending the capabilities of your HP Prime via programming is to create user-defined functions. This can be done using two different methods, each with its own variations. In the following examples, we will explore creating, modifying, and deleting user-defined functions.

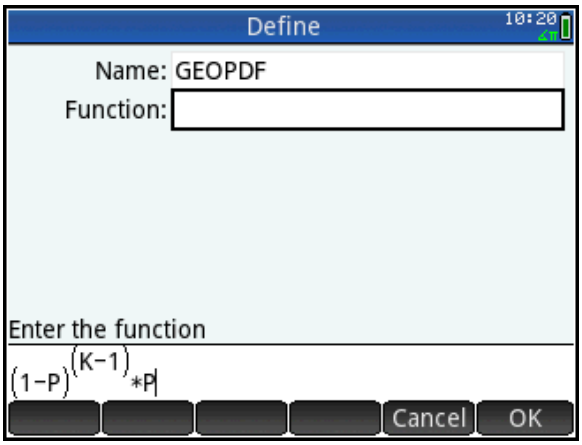
In the setting of a geometric random variable, each trial is independent and has probability of success *p*; the probability density function for the distribution of such a variable computes the probability of the first success occurring on the k^{th} trial. The formula for the geometric probability density function is thus $P(X = k) = (1 - p)^{k-1} \cdot p$. Even though HP Prime has built-in geometric probability density and cumulative density functions, we shall create a user-defined geometric probability density function in two variables. After that, we will edit the function to require only one variable.

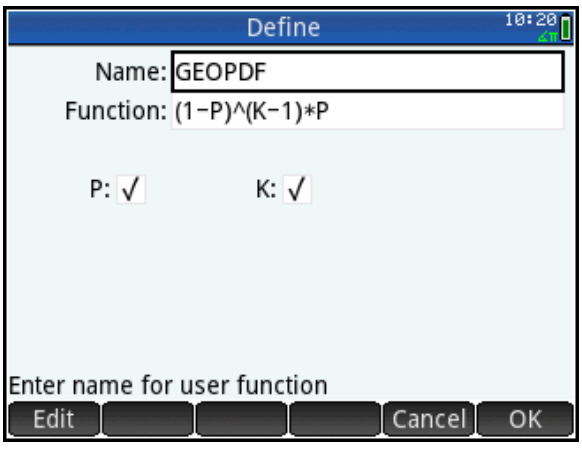
User-defined function GEOPDF(P, K)

In this example, we use the Define dialog box to create a function named GEOPDF.

1. Press Settings to go to the Home view
2. Press Shift (Define) to open the *Define* dialog box
3. The first field in the *Define* dialog box is for the name of the function:
   - Press ALPHA ALPHA to lock the alpha shift
   - Enter "GEOPDF" and tap OK
4. Highlight the *Function* field and enter the formula for the geometric probability density function
   - Enter the formula as shown to the right
   - Tap OK when you are finished

   Note: We use upper-case letters for our variables here





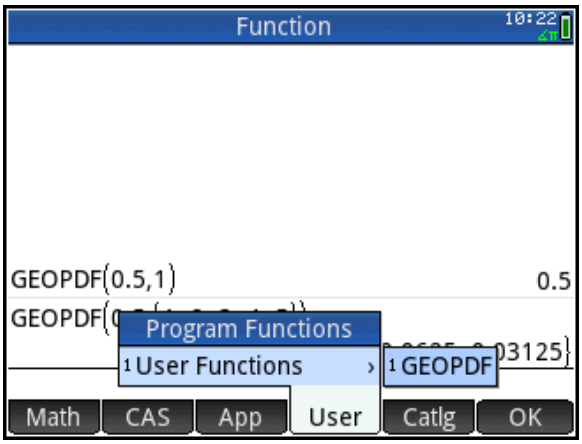
You will now see the two variables (P and K), with check boxes beside each one. The left-to-right order of the variables matches their order in the function; that is our geometric probability density function is actually GEOPDF(P, K).

The check box beside each variable determines whether or not the variable is an input variable. If checked, a variable is an input variable and a value for it must be supplied each time the function is used. If unchecked, the variable is not an input variable; instead of being required as input when the function is used, the current value of the corresponding home real number variable will be used.

If a coin is has equal probabilities for heads and tails, then the probability of heads on the first toss is 0.5. That is GEOPDF(0.5, 1) should return 0.5.

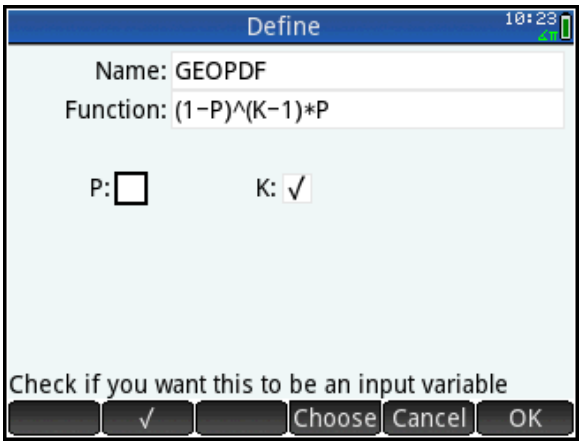
1. To access our new function, press [Mem] to open the Toolbox menus, tap User for the *User* menu, tap *User Functions*, and select *GEOPDF*
2. Complete the command GEOPDF(0.5, 1) and press [Enter]

The result is as expected. The screen shot also shows what happens when you use a list for K.

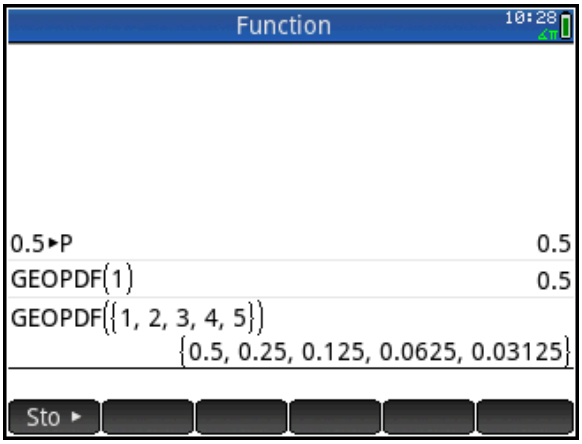


Suppose that, in practice, we find that we rarely change the value of *P* and do many calculations all using the same *P*-value. We can edit the definition of our function so that it takes its *P*-value from the current value of the home variable P instead of supplying that value each time we use the function.

1. Press [Shift] [xtθn] (Define) to open the *Define* dialog box
2. With the Name field highlighted, tap Choose and select *GEOPDF*; the current definition populates the *Define* dialog box
3. Highlight the field for the variable P, tap √ to uncheck it and tap OK
4. In the Home view, store the value 0.5 in P:
   - Type 0.5, tap Sto ▸, press [ALPHA] [EEX] (for P) and press [Enter]



Now our function is simply GEOPDF(K) and takes its probability from the value of the Home real variable P. The figure to the right reprises the earlier examples.



To delete a user-defined function, press [Mem] to open the Toolbox menus, tap User for the *User* menu, and use the cursor keys to navigate to the function you want to delete. With the function highlighted, press [Del].

In the next section, we will continue to examine user-defined functions, but this time we will use the Program Catalog and Program Editor.

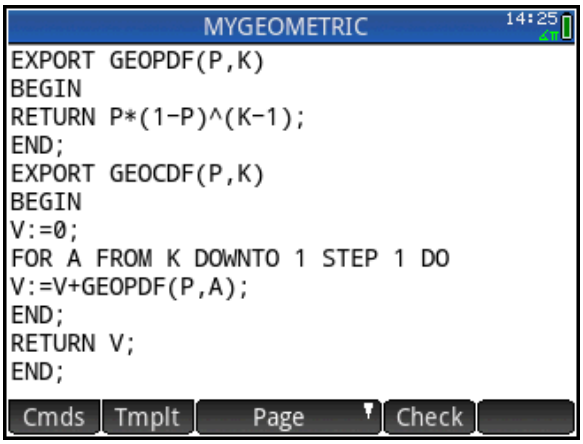
Exported functions in a program: Geometric probability functions

The geometric probability cumulative density function calculates the probability that a success is generated by the K^{th} trial, where each trial is independent and has an equal probability of success *P*. It is thus a sum whose terms are each given by our GEOPDF function. In this example, we will use the network to send you a program named **MYGEOMETRIC** that contains both the probability density and cumulative density functions.

1. Press Shift 1 (Program) to open the Program Catalog
2. Tap on the program name `MYGEOMETRIC`; the Program Editor opens to allow you to edit your new program

The program is shown in the figure to the right. It has three parts:

- The program name, followed by an empty BEGIN…END block structure
- The exported GEOPDF function
- The exported GEOCDF function



Let's take a look at each of these three parts in detail.

```
EXPORT GEOPDF(P,K)
BEGIN
RETURN P*(1-P)^(K-1);
END;
```

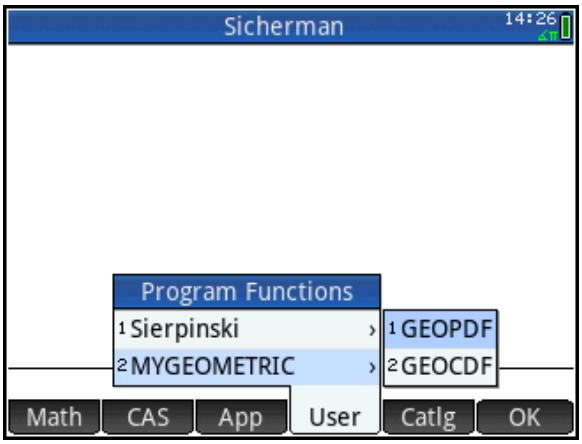
The EXPORT command makes the geometric probability density function GEOPDF(P,X) visible outside of the program. The RETURN statement within the BEGIN…END block defines the function.

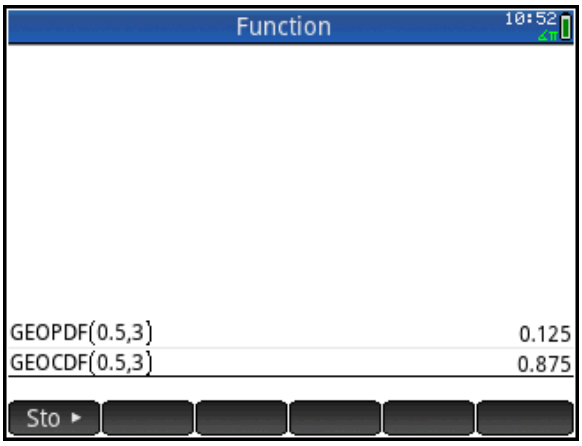
```
EXPORT GEOCDF(P,K)
BEGIN
V:=0;
FOR A FROM K DOWNTO 1 STEP 1 DO
V:=V+GEOPDF(P,A);
END;
RETURN V;
END;
```

The cumulative probability function GEOCDF(P, X) initializes the variable V with a value of zero. The FOR…END loop computes the GEOPDF value for each possible X-value and adds it to V to get a partial cumulative sum for each loop. Once the loop finishes K times, the RETURN command returns the final sum in V.

Return to Home view to use our new functions.

1. Press Settings to open Home view
2. Press Mem B to open the Toolbox menus and tap USER
3. Tap `Geometric` to see our two new functions, as shown in the figure to the right
4. Tap `GEOPDF`; the command is pasted into the Home view command line
5. Complete the command GEOPDF(0.5, 3) and press Enter ≈; it returns the correct result: 0.125
6. Repeat Steps 2 and 3, but this time select `GEOCDF`
7. Complete the command GEOCDF(0.5, 3) and press Enter ≈; it returns the correct result: 0.875

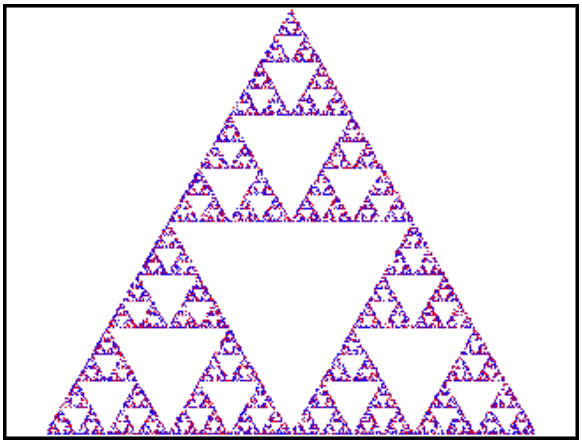




You have now learned two ways to create user-defined functions to extend the capabilities of your HP Prime. Along the way, you have learned how to use the Program Catalog and Editor to create new programs to extend your HP Prime. In the next example, we look at programming in a more general context.

Programming: The Sierpinski Triangle

The program Sierpinski uses the Chaos Game approach to approximating the Sierpinski Triangle. In the Chaos Game, one starts with the three vertices of a triangle (named in order 1, 2, and 3) and a point as initial condition. An integer from 1 to 3 is chosen at random. If 1 is chosen then a point is plotted halfway between the initial condition and Vertex 1. Similarly, you plot the point halfway from the initial condition to Vertex 2 or Vertex 3 if a 2 or 3 is chosen. The new point is then made the initial condition and the Chaos Game continues. Our program draws the first 10,000 points in seconds. The earliest points are red and they decrease in red and increase in blue as the game proceeds.



Program Listing

```
local x1:=160, x2:=299, x3:=21
local y1:=0, y2:=240, y3:=240
local xn:=160, yn:=0;
local a, b, color;

EXPORT Sierpinski()
BEGIN
RECT();
FOR a FROM 1 TO 10000 STEP 1 DO
b:=RANDINT(2)+1;
CASE
IF b=1 THEN xn:=(xn+x1)/2; yn:=(yn+y1)/2;END;
IF b=2 THEN xn:=(xn+x2)/2; yn:=(yn+y2)/2;END;
IF b=3 THEN xn:=(xn+x3)/2; yn:=(yn+y3)/2;END;
END;
color:=RGB(255-a/40,0,a/40);
PIXON_P(IP(xn),IP(yn),color);
END;
WAIT;
END;
```

Comments

Declare the coordinates of the triangle and the initial condition as local variables and declare their vaues. Also declare *a*, *b*, and *color* as local variables (more on these later).

Export the program so it appears in the Toolbox User menu.

RECT() with no argument clears the graphics display.

Local variable *a* is used in a FOR...DO...END loop

Local variable *b* contains the Chaos Game random integer

The CASE...END structure defines the point to plot, based on the random integer value.

The local variable *color* uses the RGB command to set a color for the point.
Turn on the pixel for that point, using the value of *color*.

Pause when finished until a key is pressed.

Notes

We used physical screen coordinates rather than Plot view coordinates. The top left of the display is the point (0, 0) and the bottom right of the display is (320, 240). The point (x1, y1) is at the top middle of the display. The other two points were chosen to be at the bottom of the display and to form a triangle that is close to equilateral. The initial condition was chosen as the top vertex of the triangle.

The RANDINT(N) command generates integers from 0 to N, so we used RANDINT(2)+1 to generate 1, 2, or 3 and store it in the local variable *b*.

The RGB command creates a single hexadecimal integer from three decimal integers from 0 to 255. The first decimal integer is for the red component. Here we used 255-*a*/40 so that the red component would be strong at first and gradually die off. We kept the green component zero. The blue component is the opposite of the red component; it starts off close to zero and increases as the game continues. So early points are red, middle points are purple, and later points are blue.

The command PIXON_P(x, y, color) uses the same physical coordinates that we used for the points. If you prefer to use the Plot view coordinate system, use the command PIXON(x, y, color).

You can change the values of xn and yn to see that the initial condition really does not matter.

Custom HP Apps with programs and notes: Sicherman Dice

Like regular dice, Sicherman dice are cubical in shape and numbered with positive integers. Unlike regular dice, the first Sicherman dices is numbered 1, 2, 2, 3, 3, and 4, while the second Sicherman dice is numbered 1, 3, 4, 5, 6, 8. If 2 Sicherman dice are rolled repeatedly, what does the distribution of the sum of each roll look like?

To simulate rolling regular and Sicherman dice, we created an app called Sicherman that has a program attached to it. Here is the process we followed to create the app:

1. Decide which HP App will be the basis of your new app
    a. We chose the Statistics 1Var app because we wanted to display a histogram of the distribution of the sums
2. Save the HP App under a new name and start it
    a. We chose the name "Sicherman"
3. Open the Program Catalog. The Program Catalog always lists a program with the same name as the current app. Open this program and edit it to add the functionality you need.

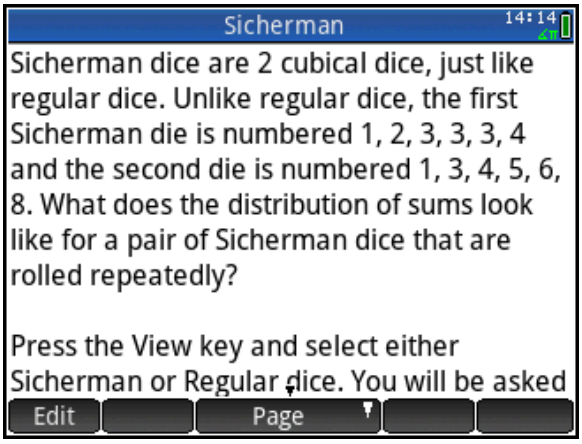
We will now send you the Sicherman App via the Prime Classroom Network. Then we will examine how it works and what the program attached to it contains.

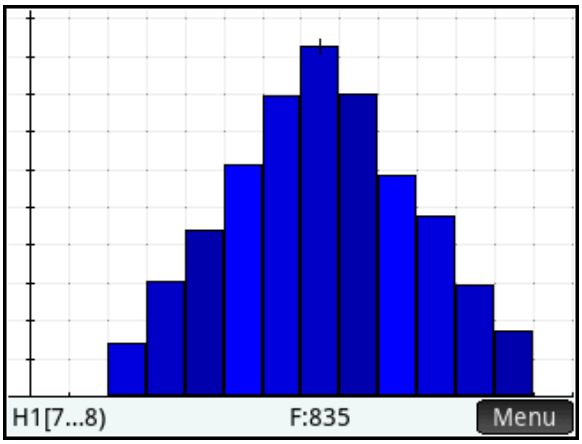
1. Press Apps Info and tap on the icon for the `Sicherman` app

The app is programmed to open in its Info view (more on this later). Press Shift Apps Info (Info) at any time to return to this view. This view describes the app in general terms and directs the student to press View Copy to get started.

1. Press View Copy and select `Regular Dice`. You will be prompted to enter the number of rolls. Type 5000 and press Enter.
2. The simulation will run and then the histogram of the sums of the rolls will be displayed.

As shown in the figure to the right, the distribution is symmetric and centered at 7, as expected.



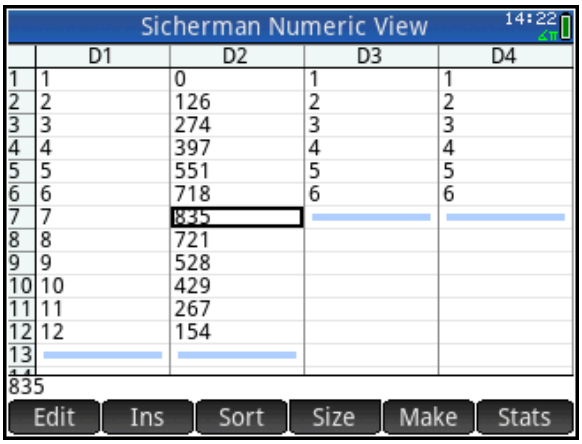


3. Press Num Setup to see the data.

Column D1 contains the possible sums (with 1 retained to make things easier). Column D2 contains the frequencies corresponding to each sum in D1. Columns D3 and D4 represent the two dice used in the simulation.

4. Now press View Copy and select `Sicherman Dice`. Run a simulation of 5000 rolls with these dice.

Are you surprised at the results? Look at Numeric view and compare the data to what is shown for the regular dice in the figure to the right.



We will now look at the Sicherman app program to see how it was constructed and how it behaves.

5. Press Shift 1 Program (Program) to open the Program Catalog
6. Tap on the program name `Sicherman (App)`
7. The Program Editor opens with the Sicherman program listing, shown on the next page with notes on each section.

Program Listing

```
DOIT();
Plot();
START();
EXPORT n, b, a;
VIEW "Sicherman Dice" Sich()
BEGIN
D3:={1,2,2,3,3,4};
D4:={1,3,4,5,6,8};
DOIT();
END;
VIEW "Regular Dice" Regu()
BEGIN
D3:={1,2,3,4,5,6}; D4:={1,2,3,4,5,6};
DOIT();
END;
```

Comments

Declare two subroutines: DOIT() and Plot().

Export the variables *n*, *b*, and *a* outside of the program.

Add an option "Sicherman Dice" to the Views key menu

The program Sich() runs when this option is chosen. The program defines columns D3 and D4 to be the two Sicherman dice and then runs the program DOIT().

Add an option "Regular Dice" to the Views key menu

The program Regu() runs when this option is chosen. The program defines columns D3 and D4 to be two regular dice and then runs the subroutine DOIT().

Program Listing

```
DOIT()
BEGIN
D1:={1,2,3,4,5,6,7,8,9,10,11,12};
D2:={0,0,0,0,0,0,0,0,0,0,0};
INPUT(n,"Number of Rolls","Rolls","Enter
number of rolls",50,50);
FOR a FROM 1 TO n DO
b:=D3(RANDINT(5)+1)+D4(RANDINT(5)+1);
1+D2(b)▸D2(b);
END;
Plot();
END;
Plot()
BEGIN
H1:={"D1", "D2",1,0};
Hwidth:=1;
Hmin:=0;
Hmax:=13;
Xmin:=-0.5;
Xmax:=14;
Ymin:=-0.5;
Ymax:=1.1*MAX(D2);
Xtick:=1;
Ytick:=Ymax/20;
CHECK(1);
STARTVIEW(1,1);
END;
START()
BEGIN
STARTVIEW(6, 1);
END;
```

Comments

The subroutine DOIT() defines D1 to be the possible sums (including 1 for ease) and D2 to be the frequencies of the corresponding sums in D1. Then an input form is created for n, the number of rolls. The FOR…DO…End loop defines *b* to be the sum of two numbers chosen randomly, one from each die. D2(b) is then incremented by one.

When the loop finishes, the Plot() subroutine runs.

The subroutine Plot() defines H1 in Symbolic view. It makes D1 the sample data, D2 the frequencies, and Histogram the plot type with no options.

It sets the bin width to 1.

It sets the Plot view window and makes a guess about where to put the ticks on the y-axis.

It makes sure H1 is checked (active) for graphing and then it displays the histogram in Plot view.

The subroutine START() runs whenever the app is started (by tapping its icon in the App Library or selecting it and tapping Start). It displays the Info view.

Notes

The program Plot() actually runs not only whenever it is called as a subroutine, but also whenever you press Plot Setup while this app is running. Likewise, the program START() runs when the app is selected in the App Library and you press Start. The other app control keys (Symb Setup, Num Setup, etc.) can be tied to subroutines in a similar fashion. This gives you complete control over your custom app.

The VIEW command changes the options available when View Copy is pressed. Each instance of the VIEW command adds exactly one option to the Views menu, in the order in which they are encountered in the program listing. You can reinstate standard Views options (Autoscale, etc.) as well, depending on the base HP app used.

The STARTVIEW command is used to present any view of the current app. View 6 is the Info view.

You noticed that the Sicherman app had a note attached to it, which is displayed when the app starts. The note is in the Info view of the app and is sent whenever the app is transmitted to another HP Prime.

When the Sicherman app is sent to an HP Prime, the Sicherman app program is sent along with it. You must open the Sicherman program on the receiving HP Prime to activate the program.

Thus, all custom HP apps have three files associated with them:

- The app file itself: Sicherman.hpapp
- The app program file: Sicherman.hpappprgrm
- The app note file: Sicherman.hpappnote

These three files are bundled in a single folder that you drop into the Content Pane of HP Connectivity Kit. On the HP Prime Teacher Forum, this directory is wrapped in a zip file. Right-click on this zip file and select `Send` to send the app, along with its note and program, to all connected HP Primes (or select a subset of the connected HP Primes in the Monitor pane and send to just those devices). Be sure to open the zip file to see if there are instructions for the teacher or student, or if there is a short introductory video.

Program Listings

You can use the following program listings to copy and paste the programs into your HP Prime Virtual Calculator. To do this, follow these steps:

1. Press S 1 to open the Program Catalog.
2. Tap New and enter the name of the program exactly as shown (case-sensitive).
3. Tap OK twice.
4. The Program Editor will open. Delete all text that appears in your new program.
5. Copy the program listing as shown. That is, drag to select the program listing and then use Ctrl-C to copy it to the PC clipboard.
6. In the HP Prime Program Editor, click on the HP Prime Virtual calculator Edit menu and select Paste (or use Ctrl-V). The program will be copied to the HP Prime Program Editor.
7. Your new program is ready to use

NOTE: It is always a good idea to tap Check to make sure there are no errors in the program. In some cases, one or more characters may be incorrectly interpreted.

Program: MYGEOMETRIC

```
EXPORT GEOPDF(P,K)
BEGIN
RETURN P*(1-P)^(K-1);
END;
EXPORT GEOCDF(P,K)
BEGIN
V:=0;
FOR A FROM K DOWNTO 1 STEP 1 DO
V:=V+GEOPDF(P,A);
END;
RETURN V;
END;
```

Program: Sierpinski

```
local x1:=160, x2:=299, x3:=21;
local y1:=0, y2:=240, y3:=240;
local xn:=160, yn:=0;
local a, b, color;
EXPORT Sierpinski()
BEGIN
RECT();
FOR a FROM 1 TO 10000 STEP 1 DO
b:=RANDINT(2)+1;
CASE
IF b=1 THEN xn:=(xn+x1)/2; yn:=(yn+y1)/2;END;
IF b=2 THEN xn:=(xn+x2)/2; yn:=(yn+y2)/2;END;
IF b=3 THEN xn:=(xn+x3)/2; yn:=(yn+y3)/2;END;
END;
color:=RGB(255-a/40,0,a/40);
PIXON_P(IP(xn),IP(yn),color);
END;
WAIT;
END;
```


App Program: Sicherman (based on the Statistics 1Var App)

```
DOIT();
Plot();
START();
EXPORT n, b, a;
VIEW "Sicherman Dice" Sich()
BEGIN
D3:={1,2,2,3,3,4};
D4:={1,3,4,5,6,8};
DOIT();
END;
VIEW "Regular Dice" Regu()
BEGIN
D3:={1,2,3,4,5,6};
D4:={1,2,3,4,5,6};
DOIT();
END;

DOIT()
BEGIN
D1:={1,2,3,4,5,6,7,8,9,10,11,12};
D2:={0,0,0,0,0,0,0,0,0,0,0};
INPUT(n,"Number of Rolls","Rolls","Enter number of rolls",50,50);
FOR a FROM 1 TO n DO
b:=D3(RANDINT(5)+1)+D4(RANDINT(5)+1);
1+D2(b)▸D2(b);
END;
STARTVIEW(2,1);
END;

Plot()
BEGIN
H1:={"D1", "D2",1,0,#FF:24h};
Hwidth:=1;
Xmin:=-0.5;
Xmax:=14;
Ymin:=-0.5;
Ymax:=1.1*MAX(D2);
Xtick:=1;
Ytick:=Ymax/20;
CHECK(1);
STARTVIEW(1,1);
END;
START()
BEGIN
STARTVIEW(6,1);
END;
```

