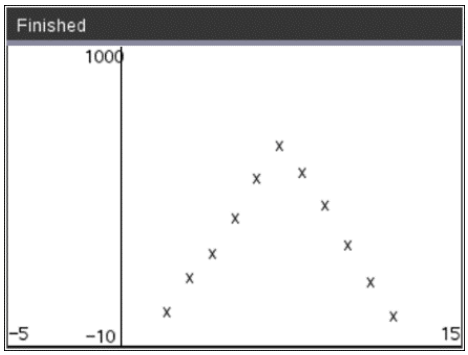
Unit 5: The TI Modules | **Skill Builder 1: The Plot Thickens**

In this lesson, you will create a scatter plot of the dice totals from Unit 4 Application using the **ti_plotlib** module.

Objectives:

- Introduce **ti_plotlib**
- Make a scatter plot
- Adjust the window

In the Unit 4 Application you made a simulation of tossing two dice and logging the experiment's totals in a list. Here you will continue with that program and see how easy it is to make a scatter plot of that data using Python.



Teacher Tip: **ti_plotlib** is based on a popular Python library called matplotlib, a rich library for the graphical visualization of data. **ti_plotlib** implements some of the 'pyplot' functions from matplotlib.

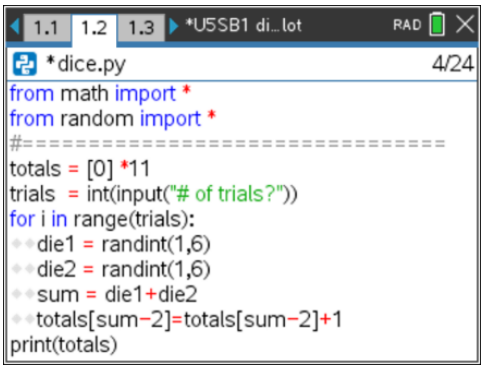
The convention when using matplotlib is to use the statement

import matplotlib.pyplot as plt

and this requires the use of the **plt.** prefix for any of the plotting functions. This ensures that the person reading the program knows that the function is a member of the library. This technique is common when using **matplotlib.pyplot**. The TI-Nspire menus create a similar statement: **import ti_plotlib as plt**

There is a 'Plotting…' template available on the TI-Nspire: **Add Python > New > 'Type'** dropdown that creates a template with this statement inserted. All of the **ti_plotlib** functions will include **plt.** at the front of the function when selected from the menus (even though it is not shown on the menus). Most of the functions also have inline prompts or selection lists.

1. Rather than starting from scratch, load the dice project from the Unit 4 Application. All the necessary code is shown in this image. You may have written more code below the **print(totals)** statement, but you will not need that for this project.



You can make a copy of this program in the document by using:

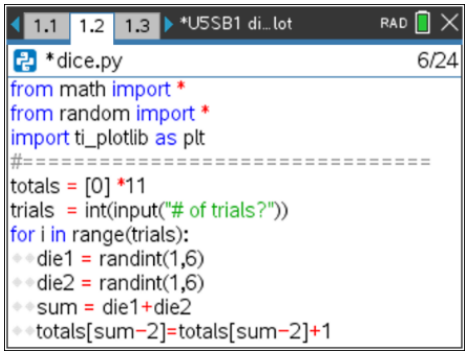
menu > Actions > Create Copy…

If 'Create Copy…' is not available, go back to the program Editor and press **ctrl+B** to store it. There should not be an asterisk (*) in front of the filename at the top of the page (as pictured).

2. To create a graphical plot of the data the program created, we need to import another custom TI module. At the top of your program, below 'from random…', add the following import statement:

   import ti_plotlib as plt

   You can get this entire statement from **menu > TI Plotlib**.

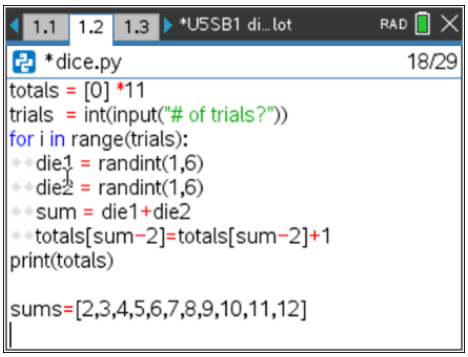


3. Scroll to the bottom of the program (below **print(totals)**).

   To make a scatter plot we need *two* lists, an *xlist* and a *ylist*. **totals** is going to be the *ylist*. For the *xlist*, we use the 11 possible sum values:

   sums = [2,3,4,5,6,7,8,9,10,11,12]

   (There are other, clever ways of making this list but just typing it in is fast and simple.)



Teacher Tip: Another way is to use 'list comprehension': **sums = [i for i in range(2,13)]**

4. Now we can set up and display the scatter plot of (sums, totals).

   Setup statements are taken from **menu > TI Plotlib > Setup**.

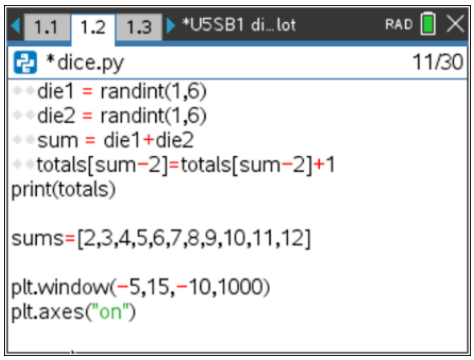
   The two we'll use here are:

   a) **plt.window( )**

   The window settings depend on the data. Use -5,15 for the x-axis and 10,1000 for the y-axis (we plan on tossing a lot of dice).

   b) **plt.axes( )**

   The choices for the "mode" pop up immediately. Select "on".



5. To make the scatter plot, from **menu > TI-Plotlib > Draw** select:

   scatter(*xlist*,*ylist*,"*mark*")

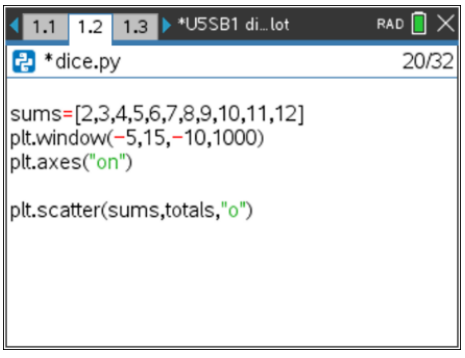
   (**plt.** is added to the front of the function.)

   For *xlist*, type **sums.**

   For *ylist,* type **totals**.

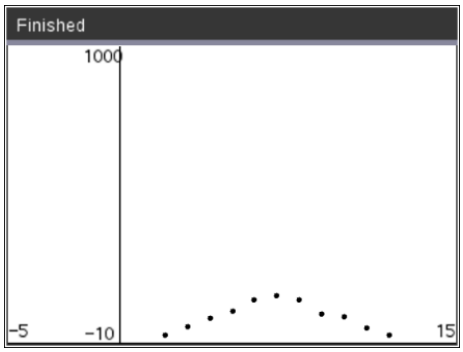
   For *mark*, choose* from the four that are allowed.

   *It is not so simple to choose a different mark once you have chosen



one. There are only four to choose from. They are o, +, x, . (period). If you want to use a different mark, you must type one of these symbols to replace the one you originally chose. You do not get another list to choose from. To see the list again you must paste the function again from the menu.

6. Ready? Run the program with 1000 trials. Do you see something like this image?



Press any key to close the graph.

Try again with 6000 trials.

Can you explain the picture?

You can customize the window for the number of trials.

In the window function change *ymax* to **1.1*max(totals)**.

Note that we *only* added **four** new statements to the program to make the plot!

Remember to save your work!

Teacher Tip: When using statements from **ti_plotlib**, the order of the statements matters because each feature (axes, grid, window, etc.) affects the graphing window. For example, use grid first and then axes because the grid will cover the axes.

This plot illustrates that the sum of 7 is the most common; each side of the hill is a linear function. The height of each point is (approximately) 1/36*trials, 2/36*trials, etc.