Unit 5: The TI Modules — Skill Builder 2: Data Sharing

In this lesson, you will transfer data between the Python environment and the TI-84 Plus CE Python graphing calculator environment.

Objectives:

- Work with store_list(), recall_list()
- Export a list to the calculator for further analysis.

The Python App is a separate system included in the TI-84 Plus CE Python graphing calculator. The most obvious change is the inclusion of lowercase characters.

But these are two separate “worlds.” The calculator world knows nothing about the data in a Python program, and the Python world knows nothing about the variables in the calculator.

It’s as if Python is a separate entity in the calculator. But there are special TI-developed functions in Python that let you transfer data between the two worlds (like teleporting).

Teacher Tip: Transferring data from a Python program to the TI-84 Plus CE Python lets you graph, plot and analyze the data using the TI-84 Plus CE tools with which you are already familiar.

Storing Python data to a TI-84 Plus CE Python variable (and back) is a great way to preserve values from one run to the next. If your “game” program maintains a high score, store the high score in a list and send it to the TI-84 Plus CE Python. The next time you run the program you can recall the list from the TI-84 Plus CE Python and extract the high score.

There is a constraint on list sizes: **store_list()** and **recall_list()** can only be used withs lists of 100 elements or less. If the list contains more than 100 elements you will see the error message:

A clever programmer will figure out a way to overcome this constraint.

1. In the last lesson, you wrote a program to generate data from dice tossing. We can use that program now to demonstrate the transfer of data from the Python world to the calculator world.

   Recall the original DICE program you wrote in **Unit 4, Application**. The name is DICE, and the complete code is shown in this image.

2. Make a copy of this program in the **<Fns…> Files** FILE MANAGER. Select the "DICE" file, and select **<Manage> Replicate Program**. Ours is now named DICEC.

Teacher Tip: Using all letters in a filename is easier than using letters and digits.

3. In your new duplicated program, you will use a function that is part of the **ti_system** module, so at the top of your program add the statement:

   from ti_system import *

4. At the end of the program, when the **totals** list is complete, add this new function that sends the contents of the list to the calculator's RAM:

   store_list(" ",)

   found on **<Fns…> Modul> ti_system**.

   See the next step for completing the statement.

5. The **store_list(“ “ ,)** function requires two arguments. Both of them are the names of lists:
   - Inside the quotes (the first argument) type the name of a TI-84 Plus CE Python list*
   - After the comma, type the name of the Python list in this program to be transferred; in this case: **totals**

   *Note: The TI-84 Plus CE Python list name must conform to the TI-84 Plus CE naming conventions for lists:
   - All UPPERCASE
   - Five characters or less
   - Must begin with a letter
   - Can be one of the six built-in lists (L1…L6). Use a number from “1” to “6” in the quotes.
   - So, to meet these constraints, our TI-84 Plus CE Python list is called **TOTLS.**

6. *Run the program. If you see something unusual then there is probably something wrong in your code. Nothing new should happen on the screen, but …*

7. Quit Python (press **[2nd] [mode]**) and answer **<Ok>** or press **[enter]**).

8. From a blank line on the home screen, press **[list]** (**[2nd] [stat]**) to see the names of lists in your device. Below the built-in lists L_1 … L_6, you see your named lists and **TOTLS** is among them (in alphabetical order). *Your list names will likely look different than what is shown here.* Select this list and, on the home screen, press **[enter]** to see its contents.

   Now that you have the data in your TI-84 Plus CE Python graphing calculator world you can perform graphical and statistical analysis on it using the calculator tools with which you are already familiar.

We will demonstrate a graphical example next.

9. To make a scatter plot of this data you also need a list of sums. Press **[stat] Edit** and enter the numbers 2 …12 into one of the lists. We use L_1.

10. Set up a **[statplot]** found above **[2nd] [y=]** of TOTLS versus L_1 as shown. Turn **Plot1 On.** The **Type:** will be scatter plot, the first icon. To get the list name **L_1** as the Xlist press **[2nd] [1].** To get the list name **TOTLS** as the Ylist press **[list]**, scroll down to the name and press **[enter]**.

    Verify that your settings match the screen here. Then …

11. Press **[zoom] ZoomStat** to see the scatter plot. Yours should be similar to this but not exactly the same.

    So, with little effort you can program a simulation using Python, transfer the data to your TI-84 Plus CE Python, and perform all the graphical and statistical analysis in your calculator using the tools you already know about. ☺

Teacher Tip: The list in this demonstration only contains 11 elements, so the size limit for lists is not an issue. Remember: There is a 100-element limit on list transfers between Python and the TI-84 Plus CE Python worlds. Both Python and TI-84 Plus CE Python lists can contain more than 100 elements. It's just the *transfer functions* that limits the list size to 100 elements.
What would a clever programmer do to overcome this constraint? Break the large list into 100-element chunks. Send the chunks over into separate lists. Then combine them into one large list on the other side, kind of like the internet which sends data in "packets." Look up "packet switching" online.