Unit 4: For Loops and Lists

Skill Builder 2: Welcome to Lists

This lesson introduces you to Python lists, which are similar to, but not the same as, TI-84 Plus CE graphing calculator lists.

Objectives:

- Create a list in the Shell
- Write a program to make a list
- List functions: **.append()**
- Add elements to a list
- List tools on the menu
- Analyze a list: mean

1. Our first venture into the world of lists takes place in a short program to make and display a list.

   In a new program, see **<Fns…> List** for the major list functions shown in the image, like the square brackets **[ ]**, **list()**, **.append()** and others.

2. The assignment statement

   a = list(range(5, 10))

   stores the numbers from 5 to 9 into the variable **a** as a list.

   range( ) provides the numbers and **list( )** arranges them into a list.

   Find **list( )** on **<Fns…> Lists**.

   Find **range( )** by pressing **[2nd] [0]** to access the **catalog** and press the **[R]** key to jump to the "R" section of the catalog.

3. The assignment statement only stores the list in the variable. To see the contents of the list use **print(a)**.

   Run the program to see the result.

   To refer to an element use the syntax **a[3]** (square brackets) to see the fourth element, 8, as shown in the Shell. The square brackets, [ ], are on the keypad (**[2nd] [x]** and **[2nd] [-]**) *and* on the **<a A # screen>** *and* on **<Fns…> Lists**.

Teacher Tip: TI-84 Plus CE lists begin with index 1. Python lists begin with index 0. This difference may take getting used to if you are accustomed to TI-Nspire™ list notation. There are other ways to make a list, including:

a = [0] * 5 which makes a 5-element list whose values are all zero: [0, 0, 0, 0, 0]. Weird, eh? That's Python for you! Lots of funny shortcuts. Try entering **"Hello" * 5** at a Shell prompt. For Python sequences (a list or a string of characters), the ***** operator is a "repetition" operator.

4. The elements of the list are accessible by their *index,* but the first element of the list has *index* 0.

The list **a** has 5 elements, 5, 6, 7, 8, 9, and the *indices* are 0, 1, 2, 3 and 4. There is no element a[5] in a 5-element list.

List **a** looks like this:

Index:	0	1	2	3	4
Element:	5	6	7	8	9

The elements can be any Python data type, including lists themselves.

Both the *index* and the element can both be represented by variables as you will see next.

5. Write a program that makes use of a list of numbers that we will enter using **input**. But first, we will tell the program how many numbers there will be.

In a new Python file, use an **input()** statement to enter the number of numbers (we used the variable **amt**).

6. Create an empty list using the assignment statement:

nums = [ ]

This list will be used to store all the numbers entered.

You can easily type the two square brackets using **[2nd] [x]** and **[2nd] [-]** or select them from **<Fns…> Lists** or from the **<a A #>** screen.

Tip: If you use a plural like **nums** *for a list name, it helps you remember that it contains many values.*

7. Add a **for** loop to enter the numbers.

   Use the loop index **i**. The range value is **amt**, the variable that holds the number of numbers to enter.

   See the next step for the **for** *block*.

8. **print** the index value **i**. This represents the index of the list or the "Element number."

   Then write an **input** statement to enter the number. Use a **float()** function to allow a decimal **value**:

   value = float(input("enter value: "))

   float() can be found under **<Fns…> Type**.

Teacher Tip: It is possible to include the element number (i) in the input prompt

value = float(input("Element number " + str(i) + ": "))

but this requires introducing the *str( )* function (to convert the number i to a string) and the concept of string *concatenation*.

The input prompt *cannot* have more than one argument, so

value = float((input("Element number", i ":")

This does not work! There are too many arguments.

9. To add each **value** to the list **nums**, use the Python function:

   nums.append(value)

   Type the list variable **nums**, then get **.append()** on **<Fns…> Lists**.

   Use the variable **value** as the item to append to the list.

This is the last statement of the loop *block*. Press backspace (**del**) to the beginning of the next line to write the summary code.

Teacher Tip: The "dot" notation used in **nums.append()** is necessary here because **.append()** is a member function (method) of the list class built into Python. This is common in object-oriented languages like Python, and we use these methods even if we are not writing OOP programs ourselves.

Another technique for adding elements to a list is:

nums = nums + [value] or **nums += [value]**

10. Print the list with the single statement:

    print(nums)

    This would be a good time to test your program. When you run the program, enter the number of numbers you plan to enter and then enter each value one at a time. The **print** statement tells you which element you are entering, but remember that the count starts with 0.

    After entering the numbers, you should see the list displayed in square brackets like this: [45, 43, 89, 25]. (Your numbers will be different.)

11. There are many useful list functions in Python.

    See **<Fns…> Lists** again.

    Scroll all the way down.

    Note: Some of these functions begin with a period (.) and others do not. A dotted function must be preceded by a list name like **nums.append()**. *Undotted functions take a list name as an argument such as* **min(nums)**.

12. Get the **sum()** function from the **<Fns…> Lists** menu. To calculate the arithmetic mean of the list, use the formula:

    mean = sum(nums) / amt

Add a **print( )** statement to display the mean.

What additional summary information can you display?

Is there a "geometric mean" as well?

Teacher Tip: Lists play a major role in programming and there are many more functions and techniques for list processing in Python. And in the TI-84 Plus CE Python's **ti_system** module there are functions for transferring Python lists and real values to and from variables.