Unit 4: for loops and lists | **Skill Builder 3: Throwing in Some Randomness**

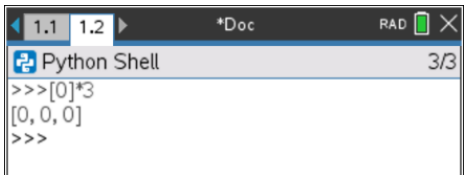
In this lesson, you will make a list of random numbers to investigate patterns.

Objectives:

- Use **randint(a,b)**
- Create a list of random integers
- Analyze properties of the list

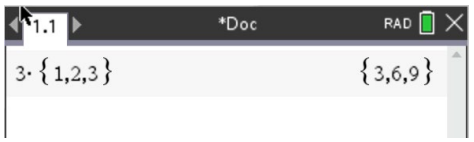
Introduction:

Lists (or arrays) are a convenient way to store many different values in a single variable. Python is very flexible with the contents of a list (it can hold different types of values, but this is rare) and has several ways of making a list.



One expression worth noting is **[0] * 3** which 'replicates' the element 3 times.

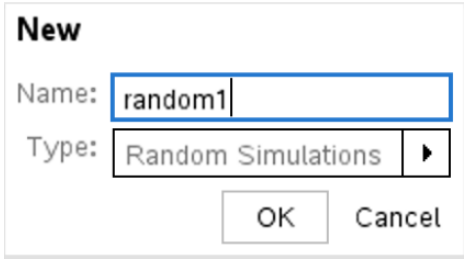
What does **3 * [1,2,3]** produce? The same expression in the Calculator app gives a completely different result as shown.



Use caution when working with lists in Python because the result might not be what you are accustomed to!

We will stick with the basics in this lesson…

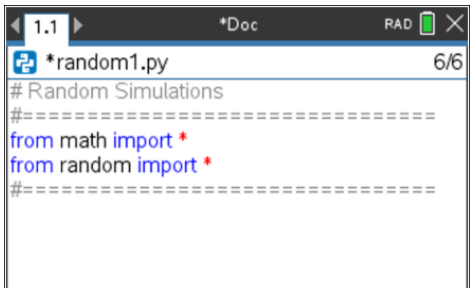
1. For this project, when creating the new Python program, choose 'Random Simulations' from the **Type:** dropdown list.



2. The *Random Simulations* template provides you with two modules: **math** and **random.** You've seen that **math** includes functions like sqrt(), trig functions, exp() and log(), and others.

   The **random** module contains functions that work with random numbers.

   One useful function is **randint(a, b)** which returns a **rand**om **int**eger between and including the values of **a** and **b**. The arguments can be numbers, variables or expressions that yield a number.



3. First, create an empty list:

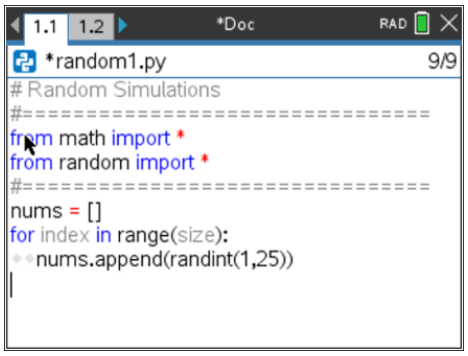
   nums=[ ] *(with nothing between the brackets)*

   Use a **for** loop to make a list of 100 random numbers, each selected from the range 0 to 25 (*you* supply the *index* variable and range *size*).

   The loop *block* is:

   nums.append(randint(1,25))

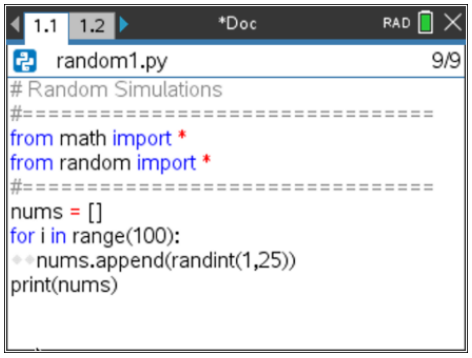
   Pay attention to the two right parentheses at the end of this statement: a common syntax error.



4. Add a print statement to the program (after the **for** loop) to print the list after the numbers have been created:

   print(nums)

   Run the program now to make sure it works. Pressing **ctrl+R** in the Shell will re-run the program and you will see a different set of numbers in each run.

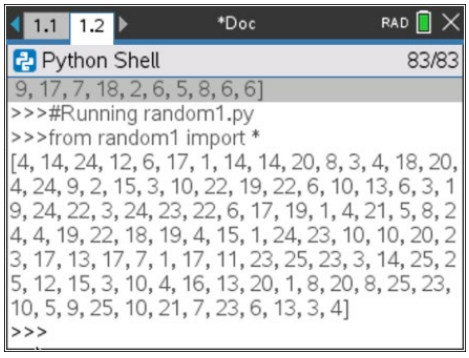


5. Here's a sample run.

   Add code to your program to determine the average of the numbers. Try it yourself first. (Remember the last lesson!)

   Display the minimum and maximum values in the list. (Hint: See **menu > Built-ins > Lists.**)

   Can you sort the elements?



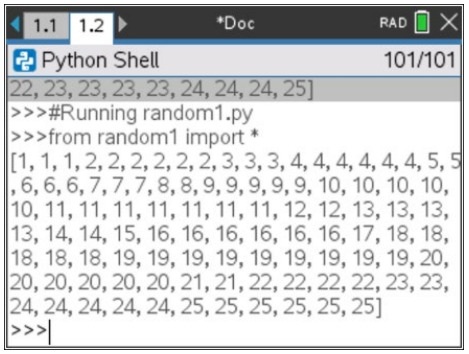
6. About sorting…

   Python has two tools for sorting a list:

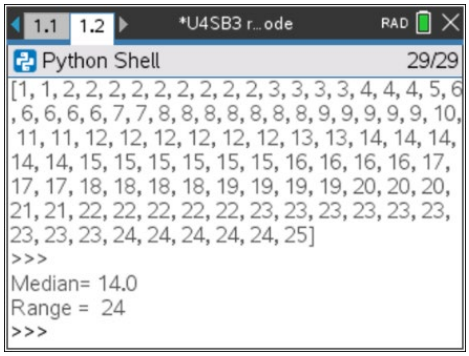
   nums.sort() arranges the elements into *ascending* order.

   sorted(n) *returns* a list that is sorted but *does not change the original list*.

   Use **nums2 = sorted(n)** to keep the original list and make a new, sorted list named **nums2**.



An *optional* exercise:



7. After sorting the list, it is now possible to determine the **median** (middle) value and the **range** (difference between minimum and maximum) of the data set.

   Add statements to your program to display these values. Note: If the length (size) of the list is an *even* number, then the median is the average of the two 'middle' numbers.

   Use the **len(<*list*>)** function found on **menu > Built-ins > Lists** to get the length of the list.

   Can you also determine the Q1(first quartile) and Q3 (third quartile) values?