Unit 4: for loops and lists — Skill Builder 2: Welcome to Lists

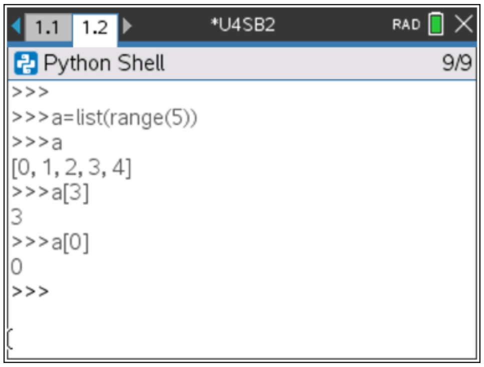
This lesson introduces you to Python lists, which are similar to, but not the same as TI-Nspire lists.

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 the Python Shell. In any Shell app see **menu > Built-ins > Lists** for the major list functions like **list()**, **.append()**, and others.
The assignment statement
a = list(range(0,5,1))
or
a = list(range(5))
stores the numbers from 0 to 4 into the variable **a** as a list.
range() provides the numbers and **list()** arranges them into a list. Both functions are found on **menu > Built-ins > Lists**.
Pressing **enter** only stores the list in the variable.
To see the list, just type the variable **a** then press **enter**.
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: 0, 1, 2, 3, and 4 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:



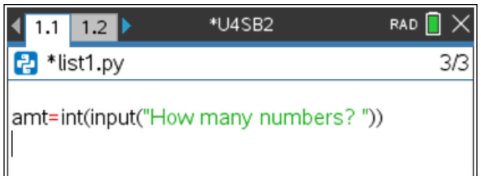
```
>>>
>>>a=list(range(5))
>>>a
[0, 1, 2, 3, 4]
>>>a[3]
3
>>>a[0]
0
>>>
```

Index:	0	1	2	3	4
Element:	0	1	2	3	4

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

Teacher Tip: TI-Nspire 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.

2. Write a program that makes use of a list of numbers that we will enter.
But first, 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**).



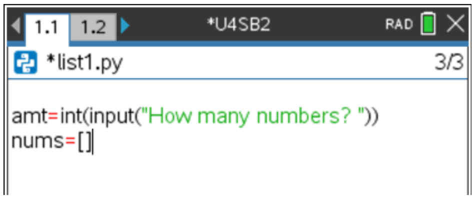
```
amt=int(input("How many numbers? "))
```

3. 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 **ctrl+(** or select them from **menu > Built-ins > Lists**.

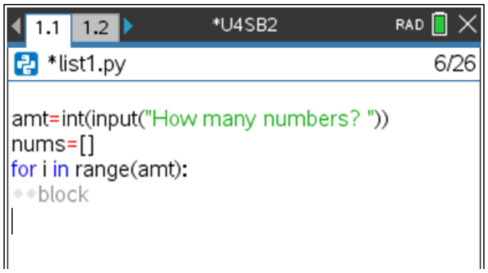


```
amt=int(input("How many numbers? "))
nums=[]
```

4. 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.

   For the *block*…

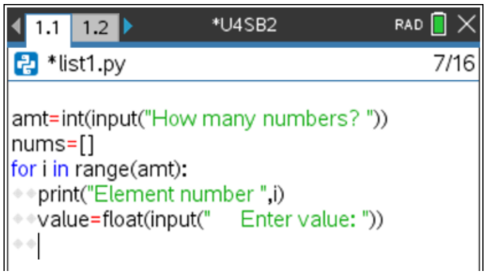


```
amt=int(input("How many numbers? "))
nums=[]
for i in range(amt):
    block
```

5. …**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: "))



```
amt=int(input("How many numbers? "))
nums=[]
for i in range(amt):
    print("Element number ",i)
    value=float(input("     Enter value: "))
```

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 *concatenation* of strings.

The input prompt cannot have more than one argument, so

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

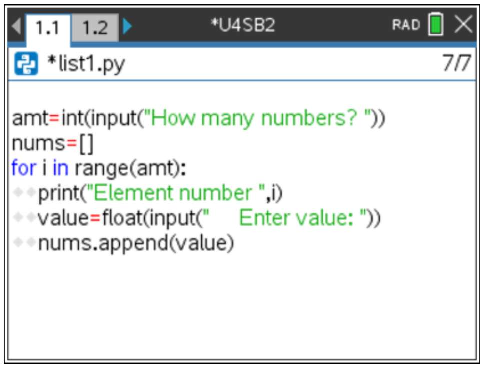
DOES NOT WORK! There are too many arguments.

6. To add the values to the list, use the Python function:

   nums.append(value)

   This is the last statement of the loop *block*.

   Press backspace (**del**) to the beginning of the next line to write the summary code.



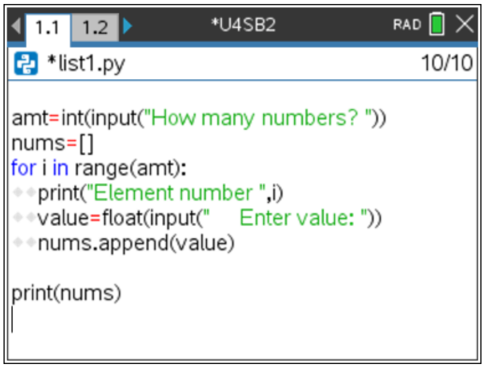
```
amt=int(input("How many numbers? "))
nums=[]
for i in range(amt):
    print("Element number ",i)
    value=float(input("     Enter value: "))
    nums.append(value)
```

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 way to combine or add to a list is: nums=nums + [value] which ‘adds’ (appends) the list [value] to the end of the list [nums]. This is very *different* from TI-Nspire behavior.

7. Print the list with the single statement:

   print(nums)

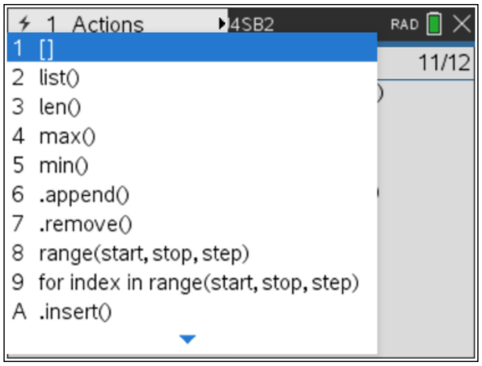
   This would be a good time to test your program. When you run the program, enter the number of numbers to enter and then enter each value one at a time. The print statement tells you which element you are entering but remember that the count starts with 0.
   After entering the numbers, you should see the list displayed in square brackets like this: [45, 43, 89, 25] (Your numbers will be different.)



```
amt=int(input("How many numbers? "))
nums=[]
for i in range(amt):
    print("Element number ",i)
    value=float(input("    Enter value: "))
    nums.append(value)

print(nums)
```

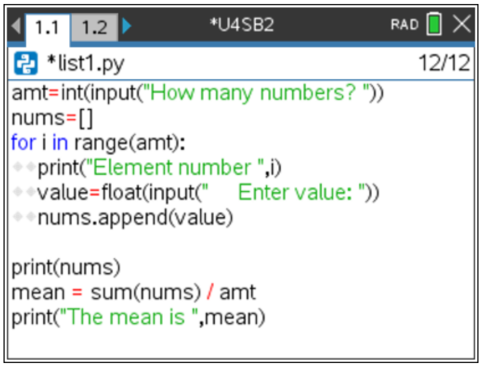
8. There are many useful list functions in Python.
   See **menu > Built-ins > Lists** again.
   Scroll all the way down.



9. Get the **sum()** function from the **Lists** menu. To calculate the mean of the list, use the formula:

   mean = sum(nums) / amt

   What additional summary information can you display?
   Is there a ‘geometric mean’ as well?



```
amt=int(input("How many numbers? "))
nums=[]
for i in range(amt):
    print("Element number ",i)
    value=float(input("    Enter value: "))
    nums.append(value)

print(nums)
mean = sum(nums) / amt
print("The mean is ",mean)
```

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