

Unit 4: For Loops and Lists

Skill Builder : Home on the Range()

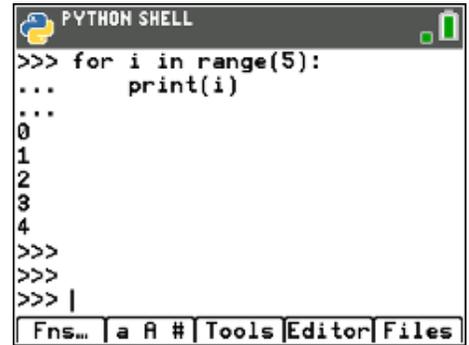
In this lesson, you will create your first of many **for** loops, and examine the many advantages of iterations.

Objectives:

- Explore the **range()** function
- Use **in** to test for membership
- Write a **for** loop

1. Python’s built-in **range()** function is used to generate an arithmetic sequence of *integers*. In this lesson you will use the **range()** function to make a **for** loop.

First, do some code testing using the Python Shell to see what the range() function does in a for statement. The Shell is a good place to test code out without writing a program. You can switch to the shell at any time to try something out.



From the FILE MANAGER, go to **<Files>** and select **<Shell>**.

At the Shell prompt, add the command found on **<Fns...> Ctl**:

```
>>> for i in range(5):
```

You must type the 5 inside the parentheses.

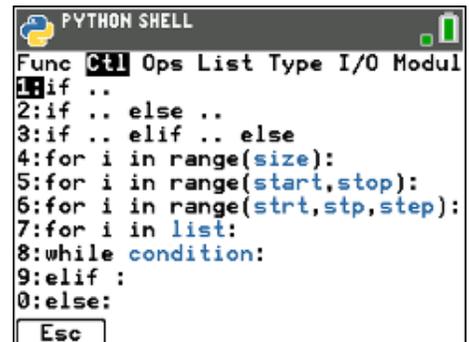
Press **[enter]** and you will see three dots. Python “knows” that you are entering a code block so the cursor is indented.

Enter the **print(i)** function from **<Fns...> I/O** and you must also type the variable **i** inside the parentheses.

Press **[enter]** twice to tell Python that there are no more statements for this block. The Shell will then process the command block and display the range of numbers 0, 1, 2, 3, 4.

*Tip: To “break” (stop) a running program, press the [on] key on the keypad until you see the **KeyboardInterrupt** message.*

2. There are other possible formats for the **range()** arguments on the **Ctl** menu shown in this image:
 - **start** is the starting number
 - **size, stop** or **stp** is the number that ends the loop but **it is not processed**
 - **step** is the increment (step) that is added to the loop variable in each iteration of the loop.
 - All three arguments **must** be integers



3. Write a program (LOOP1) that prints a range of numbers using a **for** statement. Start a new Python file and add the simplest **for** statement:
for i in range(____)

TI-84 PLUS CE PYTHON

STUDENT ACTIVITY

located in <Fns...> Ctl.

You can change the loop variable **i** to any variable that is more appropriate for your program. Inside the **range** parentheses, type an integer greater than zero. We will use 5.

- In the loop body add the print statement:

◆◆ **print(i)**

Remember that this statement is indented to be the body of the loop.

*Notice that we used **5** for the **range()** argument.*

- Run the program. Without a **start** value, the first value printed is 0 and the last value printed is **one less than** your **range** argument. There are 5 numbers printed in this image. If you used a large* **range** argument you will see that many numbers printed, just not the last one. Go Back to <Editor> and enter another number to see.

Without a **step** value, the increment is 1.

Programmers always start counting with 0.

*Tip: Again, to “break” (stop) a running program, press the [on] key on the keypad until you see the **KeyboardInterrupt** message.*

**Not too large: It could take a while.*

- There are two other **for** loop options that use **range()** found on the **Ctl** menu. The image to the right shows each in a sample program but they are incomplete.

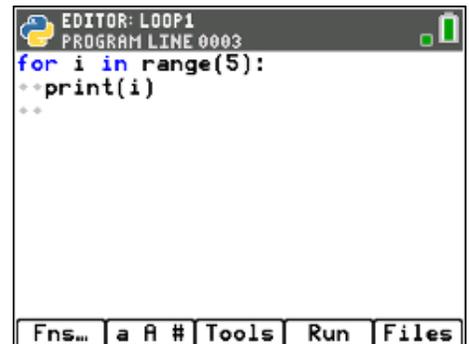
Use the third format to make a “countdown” loop that goes from 10 to 0 by 1s. Try it now, and then go to the next step.



```

EDITOR: LOOP1
PROGRAM LINE 0002
for i in range():
**

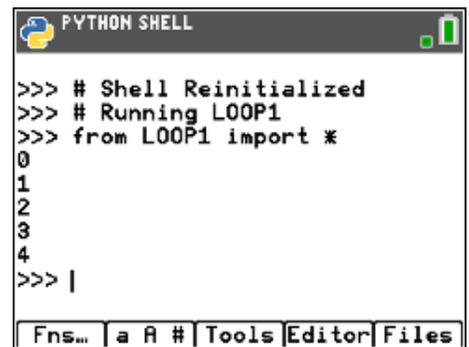
```



```

EDITOR: LOOP1
PROGRAM LINE 0003
for i in range(5):
--print(i)
**

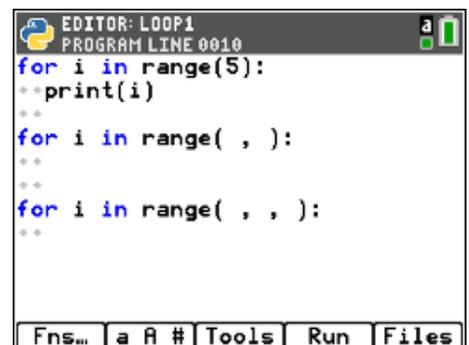
```



```

PYTHON SHELL
>>> # Shell Reinitialized
>>> # Running LOOP1
>>> from LOOP1 import *
0
1
2
3
4
>>> |

```



```

EDITOR: LOOP1
PROGRAM LINE 0010
for i in range(5):
--print(i)
**
for i in range( , ):
**
for i in range( , , ):
**

```



7. When you run the program, did it look something like this?

This loop stops at 1 (*still* 1 short of the stop value) because the program is

```
for i in range(10,0,-1):
    print(i)
```

which stops at 1, not 0.

Again, the *stop* value of a loop is *not* included in the loop body (*block*).

To include 0 in the output make the loop use:

```
range(10, -1, -1)
```

```
PYTHON SHELL
10
9
8
7
6
5
4
3
2
1
>>> |
Fns... a A # Tools Editor Files
```

8. Write a new program (SQUARE) that prints the squares of numbers between (and *including*) two entered numbers by using `input()`.

In a new file, write the two statements to `input` a lower and upper bound of the range (*not shown*).

Next, write the `for` loop using the (*start, stop*) template.

Notice that the upper bound is `upper + 1` to ensure that `upper` is included in the printout.

```
EDITOR: SQUARE
PROGRAM LINE 0002
lower=
upper=
for i in range(lower,upper+1):
    print( )
Fns... a A # Tools Run Files
```

The loop *block* should just print the number and its square (*not shown*).

Compare your program to the one shown in the next step.

9. Use `int(input())` to convert the inputted string value to an *integer*. Using `float()` would cause an error in the `range` function.

The `print` statement prints both `i` and `i*i` separated by a comma.

You could also use `i**2` to square a number.

```
EDITOR: SQUARE
PROGRAM LINE 0002
lower=int(input("lower?"))
upper=int(input("upper?"))
for i in range(lower,upper+1):
    print(i,i*i)
Fns... a A # Tools Run Files
```

10. Run the program: First enter values for lower and upper. Then, the program prints the squares starting with lower and ending with upper. If you enter a larger range of values the Shell will scroll to continue printing.

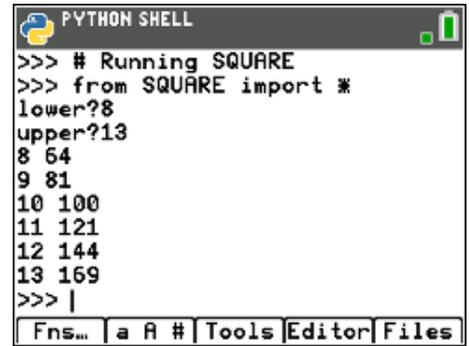
10 Minutes of Code: Python

TI-84 PLUS CE PYTHON

STUDENT ACTIVITY

Tip: To scroll up through the Shell history (backwards, going up the screen) press [2nd] [uparrow].

Try entering various ranges and determine the behavior of this loop. Can you use negative numbers? What happens if upper is smaller than lower?



```
PYTHON SHELL
>>> # Running SQUARE
>>> from SQUARE import *
lower?8
upper?13
8 64
9 81
10 100
11 121
12 144
13 169
>>> |
Fns... | a A # | Tools | Editor | Files
```