

Unit 4: for loops and lists

Skill Builder 1: 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 a sequence of numbers. In this lesson you will use the **range()** function to make a **for** loop.

Open any Python Shell. If you do not have a Shell in your document, press **ctrl+doc > Add Python > Shell**.

Make a range using the assignment statement:

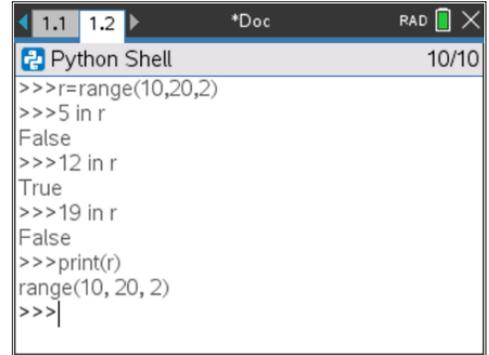
```
r = range(10,20,2)
```

range() is found on **menu > Built-ins > Lists**.

Now use the **in** operator to test certain values, like **5 in r**.

Note that the results are either **True** or **False**.

Try some other values, as pictured. **range()** does not actually contain the numbers but rather creates them when needed. Printing a **range()** function just results in the **range()** function itself.



```
Python Shell 10/10
>>>r=range(10,20,2)
>>>5 in r
False
>>>12 in r
True
>>>19 in r
False
>>>print(r)
range(10, 20, 2)
>>>|
```

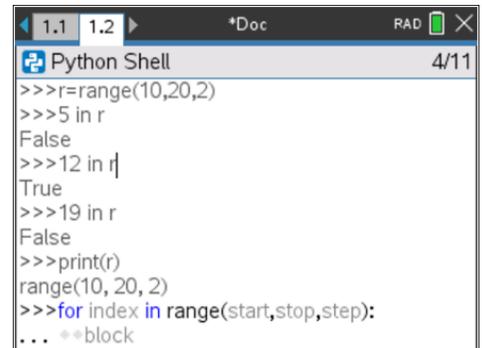
2. Still in the Shell (since this is a good place to try things out without writing a program):

If you type **r enter** or **print(r) enter**, all you see is **range(...)**.

To see all the values in the **range()** use the **for** statement:

Press **menu > Built-ins > Control** to select:

```
for index in range(start, stop, step)
```



```
Python Shell 4/11
>>>r=range(10,20,2)
>>>5 in r
False
>>>12 in r|
True
>>>19 in r
False
>>>print(r)
range(10, 20, 2)
>>>for index in range(start,stop,step):
... **block
```

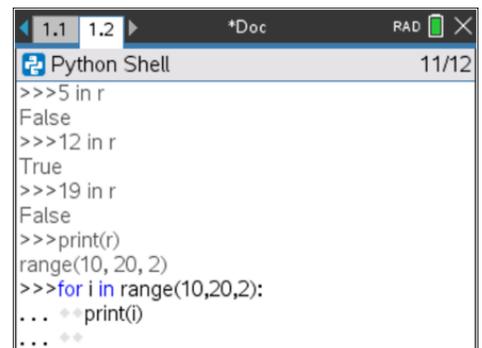
3. Change the prompts (press **tab** to jump from one prompt to the next):

index change to **i**

start, stop, step change to **10,20,2**

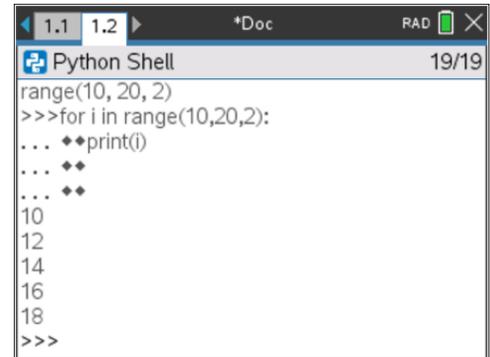
block change to **print(i)**

See the image to the right. Notice the indenting of the **print()** function.



```
Python Shell 11/12
>>>5 in r
False
>>>12 in r
True
>>>19 in r
False
>>>print(r)
range(10, 20, 2)
>>>for i in range(10,20,2):
... **print(i)
... **
```

- Press **enter** a couple of times until the command is processed. You should see the numbers 10, 12, 14, 16, and 18 printed. *20 is not printed.*



```

Python Shell 19/19
range(10, 20, 2)
>>>for i in range(10,20,2):
...  **print(i)
...  **
...  **
10
12
14
16
18
>>>
    
```

- Write a program using a **for** statement. Start a new Python file and add the simplest **for** statement
for index in range(size)
 located in **menu > Built-ins > Control**.

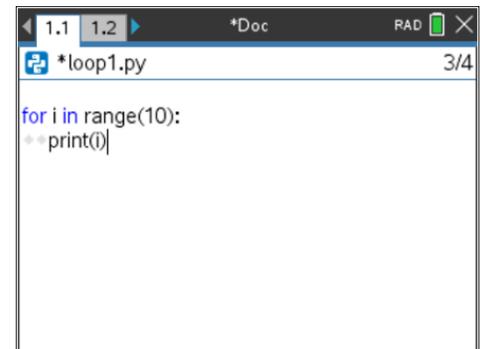
Replace the three prompts with your code as before (size is just one value). Try it yourself...



```

*loop1.py 2/4
for index in range(size):
  ==block
    
```

- See if your code is similar to that shown in the image.
i can be any variable, the *size* can be any positive integer, and the *block* can be any code. Try **print("hello")** instead.



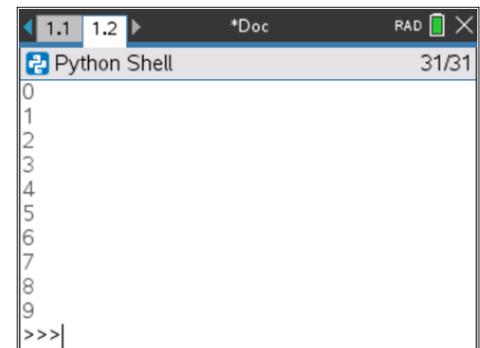
```

*loop1.py 3/4
for i in range(10):
  ==print(i)
    
```

- Run the program. Your output should print some integers.

Without a start value, the loop begins with 0. The loop stops *one short* of the size (we used 10). **size** is the upper limit of the range and is not part of the range values.

Without a step value, the step size is 1.



```

Python Shell 31/31
0
1
2
3
4
5
6
7
8
9
>>>|
    
```



10 Minutes of Code - Python

TI-NSPIRE™ CX II TECHNOLOGY

UNIT 4: SKILL BUILDER 1

STUDENT ACTIVITY

- There are two other **for** loop options that use **range()** found on the **Control** menu.

Can you make a 'countdown' loop that goes from 10 to 0 by 1's? Try it now and then go to the next step...

```

1.1 1.2 *Doc RAD X
*loop1.py 11/11
for i in range(10):
  print(i)
for index in range(start,stop):
  block
for index in range(start,stop,step):
  block
  
```

- Did you try something like this?
This loop stops at 1 (*still* 1 short of the stop value).

Tip: This split screen was achieved by pressing **ctrl+4** (Group) in the Editor. The Shell history is cleared; to run the program again press **ctrl+R** in the Editor.

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

```

1.1 *Doc RAD X
loop1.py 15/19 Python Shell 23/23
10
9
8
7
6
5
4
3
2
1
>>>
  
```

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

In a new document write the two statements to **input** a lower and upper bound of the range.

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

The loop *block* should just print the number and its square.

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

```

1.2 2.1 2.2 *Doc RAD X
*squares.py 8/15
# Print the squares of numbers from
lower=
# to
upper=
for index in range(start,stop):
  block
  
```

- Did you make the same mistake we did? The two final values, the upper value and its square (10 and 100, respectively), are not included in the printout. 😞

The **range()** should be:

range(lower, upper + 1)

Did you use **num*num** or **num**2**?

Remember to save your work!

```

1.1 1.2 2.1 *Doc RAD X
squares.py 8/9 Python Shell 16/16
# Print the squares of numbers
lower=int(input("Lower? "))
# to
upper=int(input("Upper? "))
for num in range(lower, upper):
  print(num, num*num)
>>>#Running squares
>>>from square
Lower? 5
Upper? 10
5 25
6 36
7 49
8 64
9 81
>>>
  
```