

Name: _____

Partner: _____

Python Activity 14: Range

What if we want to repeat an action a certain number of times?

Learning Objectives

Students will be able to:

Content:

- Describe the syntax of the **range()** function
- Identify the values in a **range()**
- Predict when to use **range()** versus iterate directly over objects

Process:

- Write code that uses **range()** vs. direct iteration appropriately

Prior Knowledge

- For-each..loops, input, types, * str, \t, list(), sequences

Concept Model: Do not spend more than 2 minutes on CM 1 & 2!

Examine the partially completed code below, and its desired output (to the right):

Concept Model – Python Program

```
height = int(input("How tall to make the triangle? "))
```

```
# Your code goes here
```

CM1. Using *only concepts we've learned so far*, summarize the code we'd have to write to get the output shown to the right:

```
How tall to make  
the triangle? 5  
*  
**  
***  
****  
*****
```

CM2. What might some problems with your approach? (*Hint: Will your solution work for heights other than 5?*)

Critical Thinking Questions:

1. Closely examine the Python program below, which generates the output shown in the Concept Model for all [valid] user inputs:

Range -- Python Program

```
height = int(input("How tall to make the triangle? "))  
for index in range(height):  
    print('*' * (index+1))
```

- a. Circle the **loop variable** in the code.

- b. What are the *values* that the loop variable must represent to produce the displayed output? _____
- c. Underline the **sequence** that the loop is iterating over.
- d. What might the `range(height)` command represent?
-

FYI: A *looping structure* for which you know the number of times it will execute is known as a **count-controlled** loop.

2. Observe the following interactive python session:

```
>>> range(0, 10)
range(0, 10)
>>> type(range(0, 10))
range
>>> list(range(0, 10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> list(range(10))
```

- a. What **type** of object is returned by a call to the built-in function **range()**? _____



- b. What do we have to do to see the values that **range()** contains?
-

- c. What do you expect will happen when the last line of code is executed?
-

3. Observe the following code snippets (assume `a_str` is set to empty string between each):

```
a_str = ''
a. for use_better_names in range(5):
    a_str = a_str + str(use_better_names) + ' '
b. for not_good in range(1,5):
    a_str = a_str + str(not_good) + ' '
c. numIterations = 6
   for b in range(numIterations):
       a_str = a_str + str(b) + ' '
d. numIterations = 6
   for c in range(1, numIterations+1):
       a_str = a_str + str(c) + ' '
e. for a in range(3,20,2):
    a_str = a_str + str(a) + ' '
```

Which of the above questions a-e produce the following outputs, when `a_str` is printed?

0 1 2 3 4 5	a	b	c	d	e	(Circle one)
0 1 2 3 4	a	b	c	d	e	
1 2 3 4	a	b	c	d	e	
3 5 7 9 11 13 15 17 19	a	b	c	d	e	
1 2 3 4 5 6	a	b	c	d	e	



f. After examining the five code fragments in #3, explain how the **range()** built-in function works. Include an explanation of the argument values.

FYI: The Python built-in function - **range()** - is used to define a *sequence* of numbers and can be used in a for..loop to determine the number of times the loop is executed. The syntax is similar to sequence slicing: `range(start num, end num exclusive, optional step)`

4. a. Complete the arguments in the following range function so that the code prints the even numbers between 100 and 200 inclusive.

```
for x in range(_____):
    print(x)
```

b. Complete the arguments in the following range function so that the code prints:
5 4 3 2 1 0.

```
for x in range(_____):
    print(x)
```

5. Examine the following code segment:

```
total = 0
for x in range(5):
    number = int(input("Enter a number: "))
    total = total + number
print("The total is: ", total)
```

a. Why is the variable **total** initialized to 0 in the first line of code?



b. Predict what the following code might do:

```
for x in range(5):
```

c. Explain what the following code does: `total = total + number`


d. How many numbers does the program prompt for? _____

e. What is the **accumulator** in the code segment? _____

f. What does this code do?

6. Observe the following python program:

```
b_str = ''
for i in range(1, 5):
    for j in range(1, 4):
        b_str = b_str + str(i * j) + "\t"
    b_str = b_str + "\n"
```

 a. Examine the code above. What is the output of this program? Trace through the values as they change:

i → range(1,5): [____,____,____,____] j → range(1,4): [____,____,____]
 i j b_str

Before the outer loop:	_____	_____	_____
Iteration 1:	_____	_____	_____
Iteration 2:	_____	_____	_____
Iteration 3:	_____	_____	_____
Iteration 4:	_____	_____	_____
Iteration 5:	_____	_____	_____
Iteration 6:	_____	_____	_____
Iteration 7:	_____	_____	_____
Iteration 8:	_____	_____	_____
Iteration 9:	_____	_____	_____
Iteration 10:	_____	_____	_____
Iteration 11:	_____	_____	_____
Iteration 12:	_____	_____	_____
Final value	_____	_____	_____

Application Questions: Use the Python Interpreter to check your work

1. Write a code segment using a for..loop that *prints* multiples of 5 from 5 to 500, one on a line.

2. It's possible to write a for-each..loop as a for..loop using range(). Convert the following for-each..loop into a for..loop that uses range (..):

```
day = input("What day is today? ")
for d in day:
    print(d + "aturday")
```

3. Use two for..loops with range() to *print* the following output:

```
$
$$
$$$
$$$$
*
**
***
****
```

4. Use a **nested** for..loop to *print* the following output, using range():

```
$
*
$$
**
**
$$$
***
***
***
$$$$
****
****
****
```

5. Write a function, add_matrices, that takes two lists of lists of integers, and *returns* the summed matrix. i.e.,
- $$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} + \begin{bmatrix} 7 & 8 & 9 \\ 1 & 1 & 1 \\ 3 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 8 & 10 & 12 \\ 5 & 6 & 7 \\ 10 & 10 & 10 \end{bmatrix}$$
