CS65: Introduction to Computer Science

For Loop
Nested For Loop

Drake
UNIVERSITY

Md Alimoor Reza
Assistant Professor of Computer Science

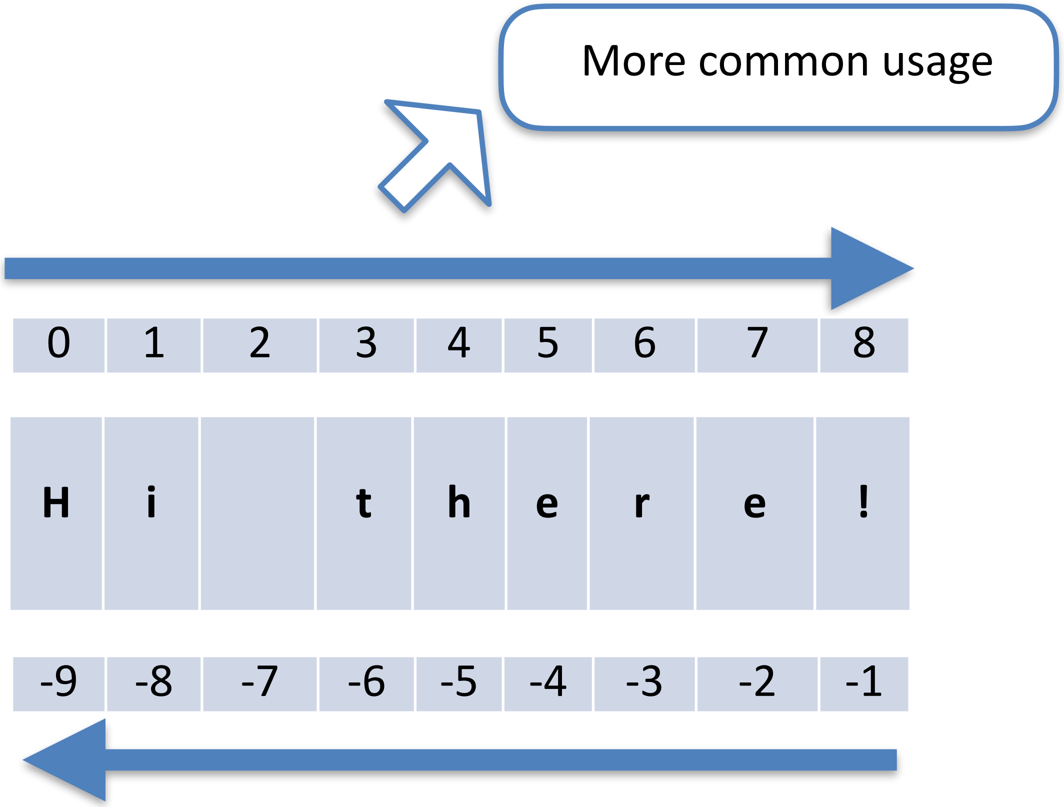
Topics

- Sequence
  - String
  - List

- The **for** loop to solve a repetitive task
  - Value **for** loop
  - Index **for** loop

- Nested **for** loop

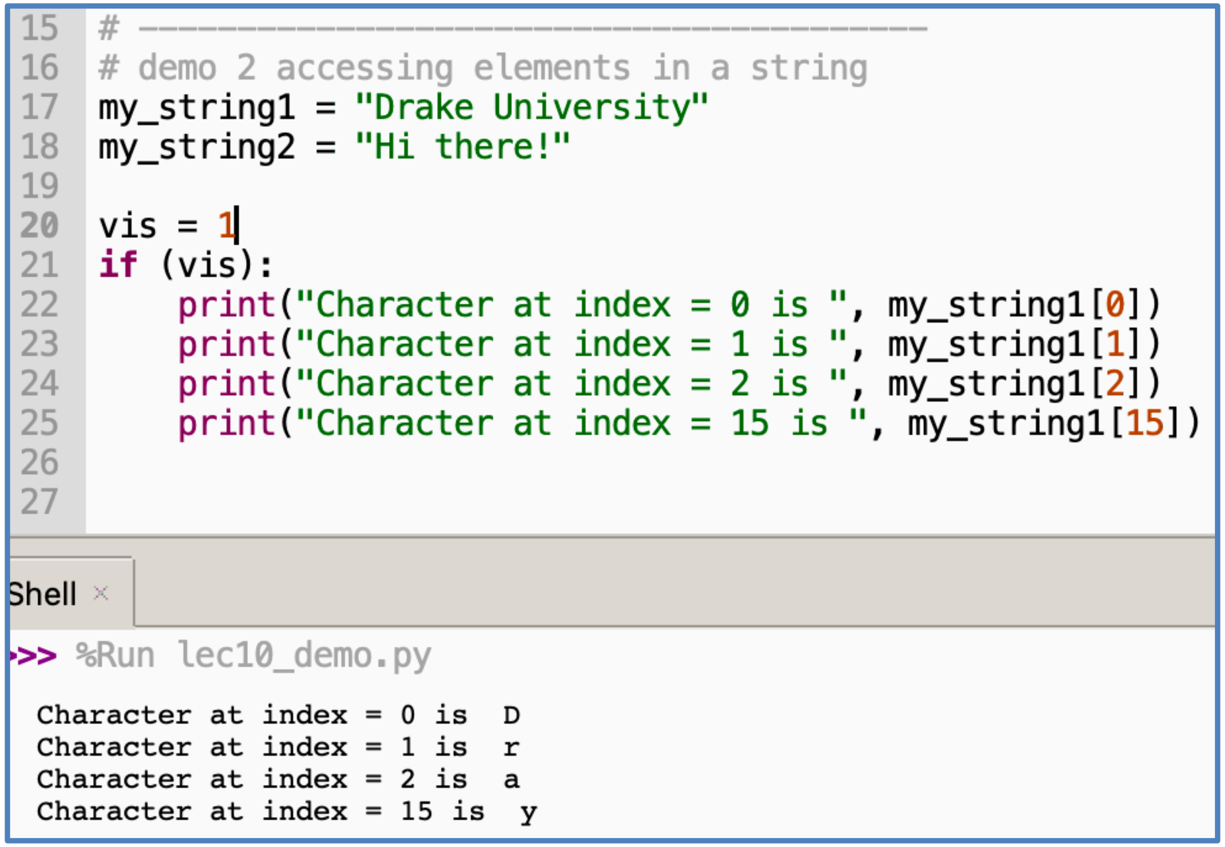
Summary: Indexing



0	1	2	3	4	5	6	7	8
H	**i**		**t**	**h**	**e**	**r**	**e**	**!**
-9	-8	-7	-6	-5	-4	-3	-2	-1

Summary: Accessing items with index

- Use ***variable_name[index]*** access an item in a sequence



```
# ---------------------------------------
# demo 2 accessing elements in a string
my_string1 = "Drake University"
my_string2 = "Hi there!"

vis = 1
if (vis):
    print("Character at index = 0 is ", my_string1[0])
    print("Character at index = 1 is ", my_string1[1])
    print("Character at index = 2 is ", my_string1[2])
    print("Character at index = 15 is ", my_string1[15])
```

Shell

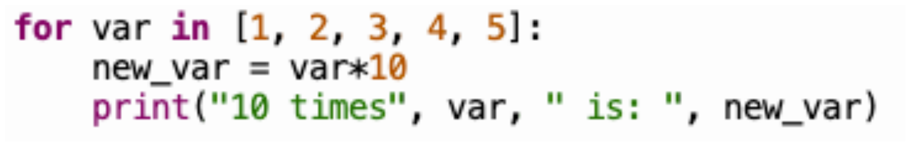
```
>>> %Run lec10_demo.py

 Character at index = 0 is  D
 Character at index = 1 is  r
 Character at index = 2 is  a
 Character at index = 15 is  y
```

Summary: syntax of value **<u>for</u>** loop

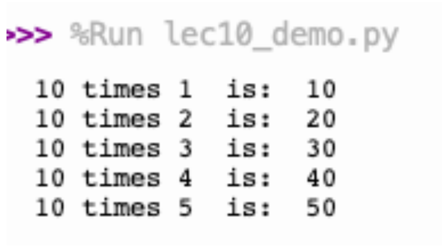
- **for** variable **in** [1, 2, …, 5] **:**
  statements

- Statements will be repeated sequentially from first to last item in a sequence (here it will be repeated 5 times since there are 5 numbers in the List)
  - <u>Iteration 1:</u> <u>variable</u> will be assigned **1**
  - <u>Iteration 2:</u> <u>variable</u> will be assigned **2**
  - …
  - <u>Iteration 5:</u> <u>variable</u> will be assigned **5**

Summary: value **for** loop



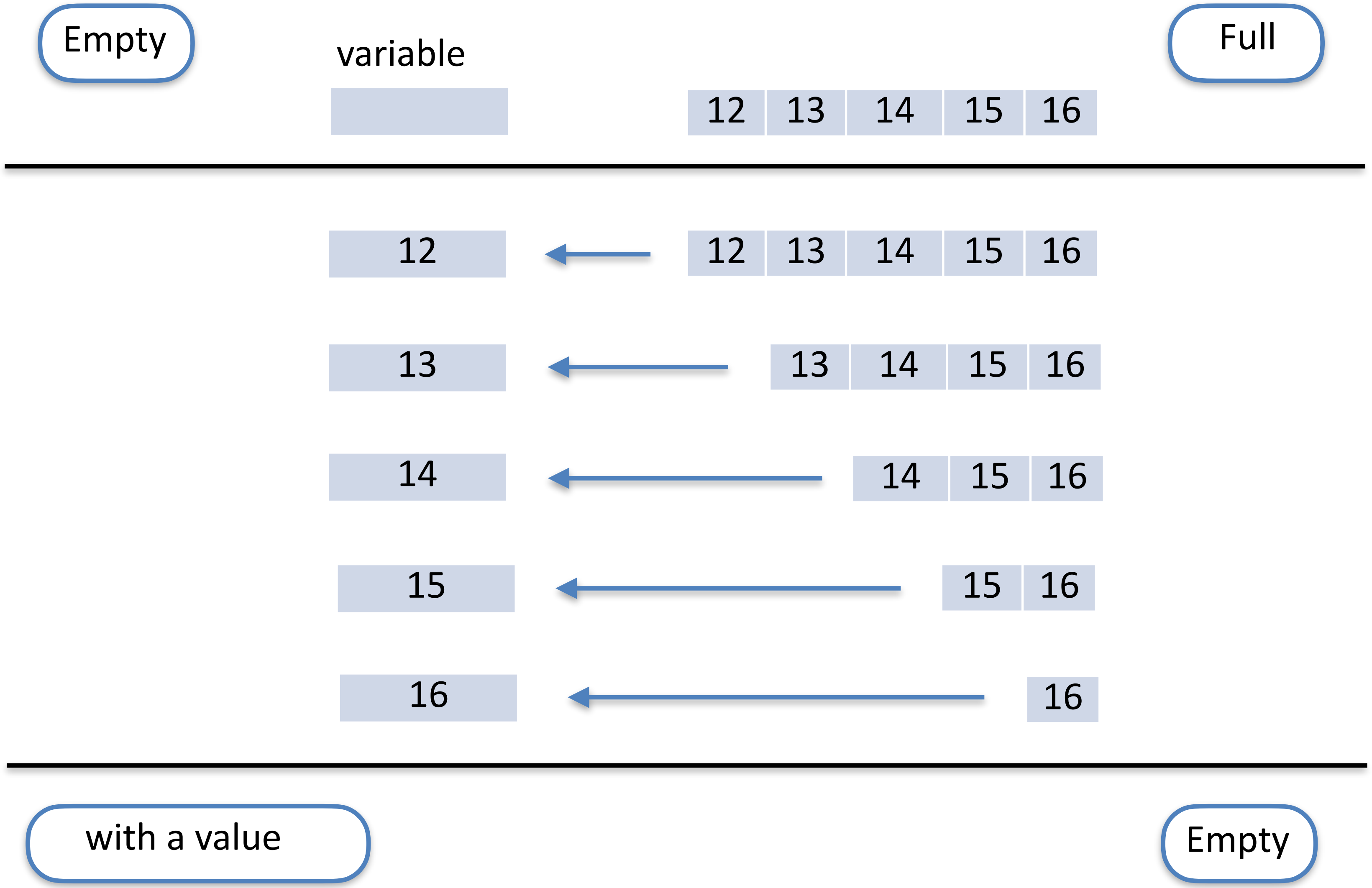
```python
for var in [1, 2, 3, 4, 5]:
    new_var = var*10
    print("10 times", var, " is: ", new_var)
```



```
>>> %Run lec10_demo.py

 10 times 1  is:  10
 10 times 2  is:  20
 10 times 3  is:  30
 10 times 4  is:  40
 10 times 5  is:  50
```

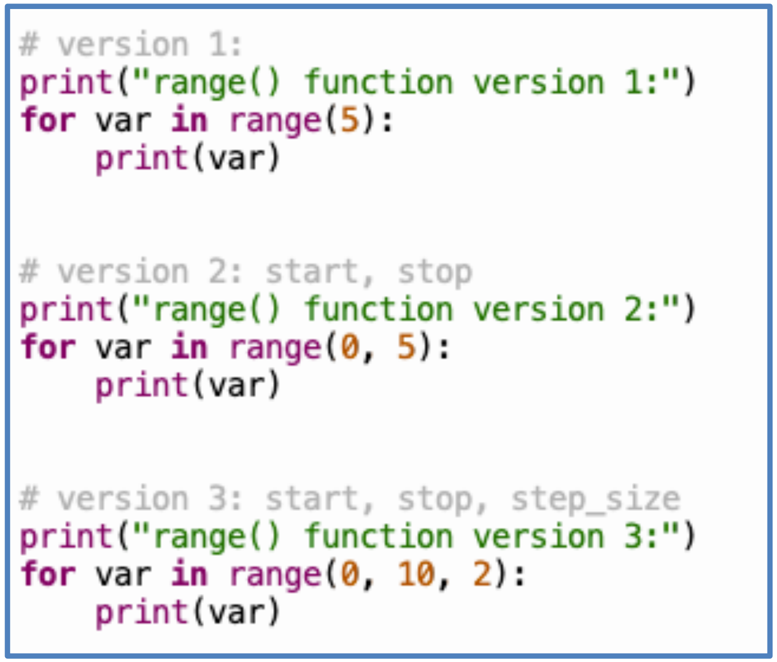
Summary: value for loop visualization

```
for var in [12, 13, 14, 15, 16]:
    print("current num is: ", var)
```



Summary: ***range***() function

- The ***range***() function simplifies the process of for loop writing
- Creates a sequence of numbers on the fly
- These numbers can be used to index the sequence



```python
# version 1:
print("range() function version 1:")
for var in range(5):
    print(var)


# version 2: start, stop
print("range() function version 2:")
for var in range(0, 5):
    print(var)


# version 3: start, stop, step_size
print("range() function version 3:")
for var in range(0, 10, 2):
    print(var)
```

Value **for** loop vs **Index** **for** loop

- So far we have seen the syntax of **value** **for loop**

```
for var in [10, 20, 30, 40, 50] :
        print(var)
```

- There is another form called **index** **for loop**

```
my_list = [10, 20, 30, 40, 50]
length = len(my_list)
for i in range(length) :
        print( my_list[i] )
```

common practice is to name the index variables with **i, j,** or **k**

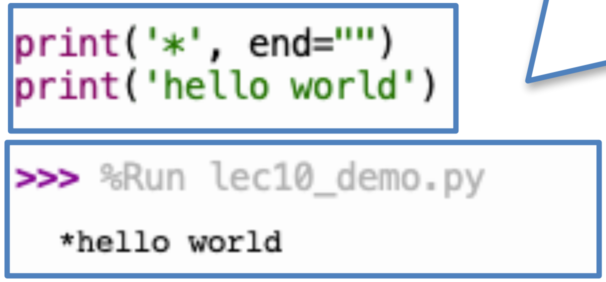
Topics

- Sequence
    - String
    - List
- The **for** loop to solve a repetitive task
    - Value **for** loop
    - Index **for** loop
- Nested **for** loop

Exercises 1

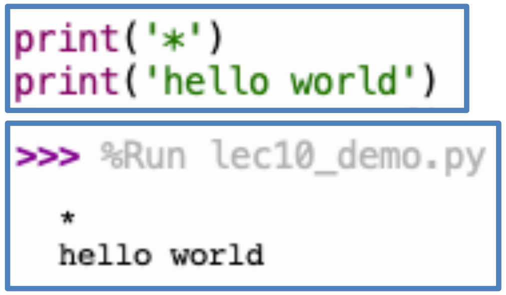
- Write a loop that will print '*' 5 times.
- Write a loop that will print '*' 10 times.
- Write a loop that will print '*' N times (prompt the user to enter this number)

New trick!



```
print('*', end="")
print('hello world')
```

```
>>> %Run lec10_demo.py
 *hello world
```



```
print('*')
print('hello world')
```

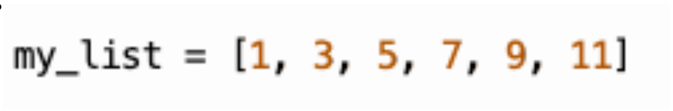
```
>>> %Run lec10_demo.py
 *
 hello world
```

Exercises 2

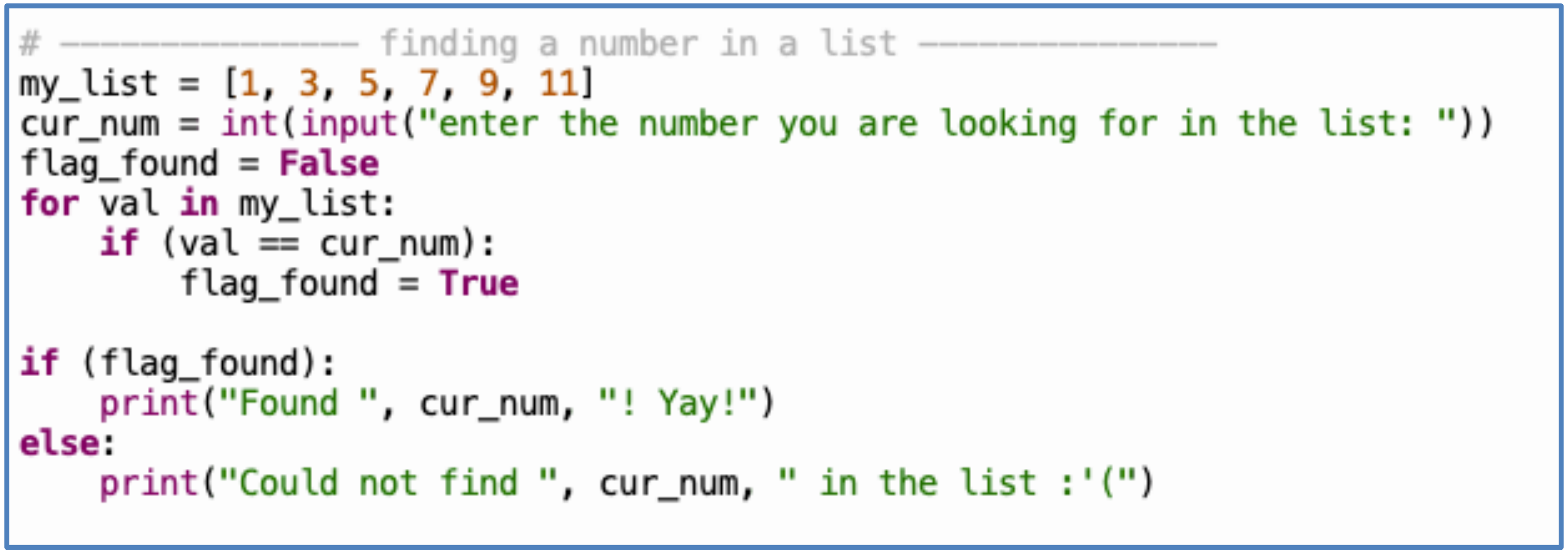
- Prompt the user to enter a number, save it in a variable called max_num
    - eg, max_number = **5**
- Find the sum of all the numbers from **1** to max_num
    - eg, 1 + 2 + 3 + 4 + 5 = 15
    - use **for** loop to do this
- Find the average of these numbers

Exercise 3

- Finding a number (prompt the user to enter that number) in a given list of number



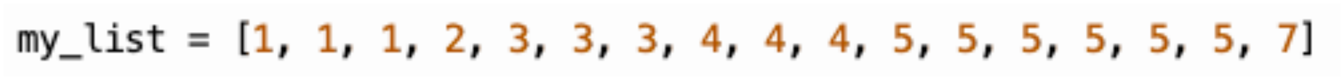
```python
my_list = [1, 3, 5, 7, 9, 11]
```



```python
# ---------------- finding a number in a list ----------------
my_list = [1, 3, 5, 7, 9, 11]
cur_num = int(input("enter the number you are looking for in the list: "))
flag_found = False
for val in my_list:
    if (val == cur_num):
        flag_found = True

if (flag_found):
    print("Found ", cur_num, "! Yay!")
else:
    print("Could not find ", cur_num, " in the list :'(")
```

Exercise 4

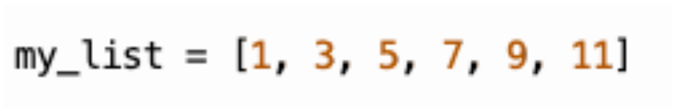
- Counting how many times a number (prompt the user to enter that number) appears in a given list.



```
my_list = [1, 1, 1, 2, 3, 3, 3, 4, 4, 4, 5, 5, 5, 5, 5, 5, 7]
```

Exercise 5

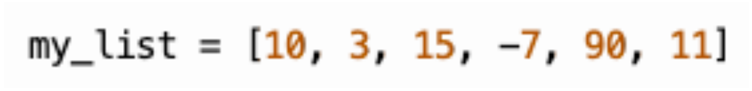
- Finding the **location** of given a number (prompt the user to enter that number) in a given list.



```
my_list = [1, 3, 5, 7, 9, 11]
```

Exercise 6

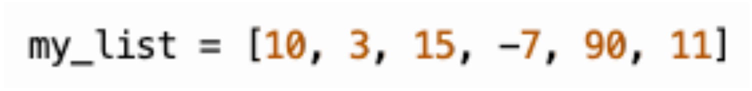
- Finding the **maximum** number in a given list.



```
my_list = [10, 3, 15, -7, 90, 11]
```

Exercise 7

- Finding the **minimum** number in a given list.



```
my_list = [10, 3, 15, -7, 90, 11]
```

Topics

- Sequence
    - String
    - List

- The **for** loop to solve a repetitive task
    - Value **for** loop
    - Index **for** loop

- Nested **for** loop

Nested **for** loops

- Putting one loop inside another
  - The first loop is called the outer loop
  - The second loop is called the inner loop

```
for i in range(3):
    # first line of outer loop
    for j in range(3):
        # first line of inner loop
        print(“i: ”, i, “j: ”, j)
        # …
        # last line of inner loop, go back to beginning
    # …
    # last line of outer loop, go back to the beginning
```

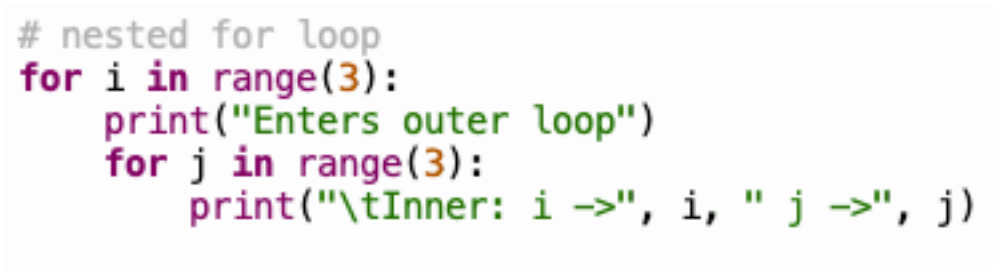
Nested **for** loops

- Putting one loop inside another
  - The first loop is called the outer loop
  - The second loop is called the inner loop

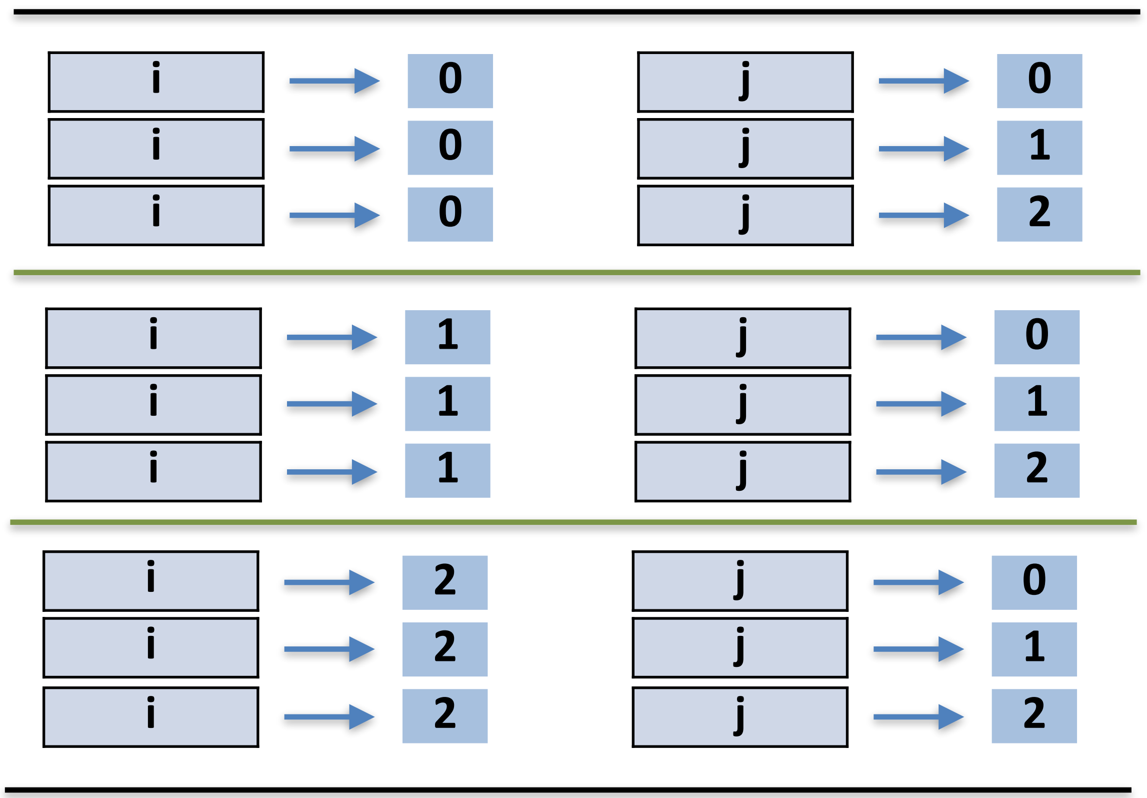
- Here is simpler version:

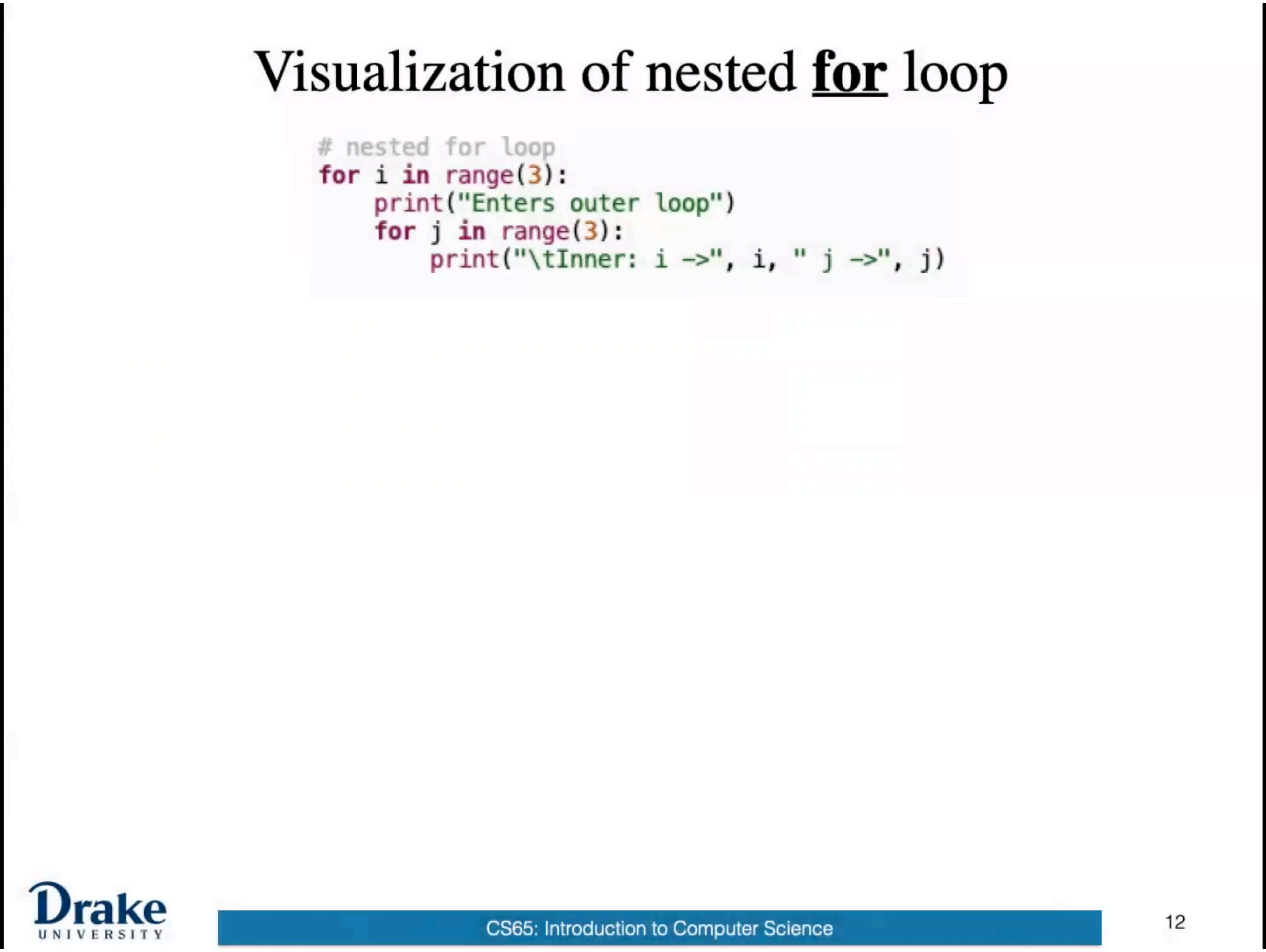
```
for i in range(3):
    for j in range(3):
        print("i: ", i, "j: ", j)
```

Visualization of nested **for** loop



```python
# nested for loop
for i in range(3):
    print("Enters outer loop")
    for j in range(3):
        print("\tInner: i ->", i, " j ->", j)
```

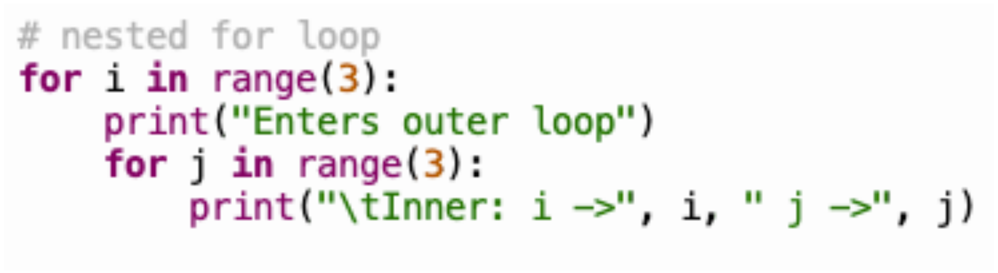




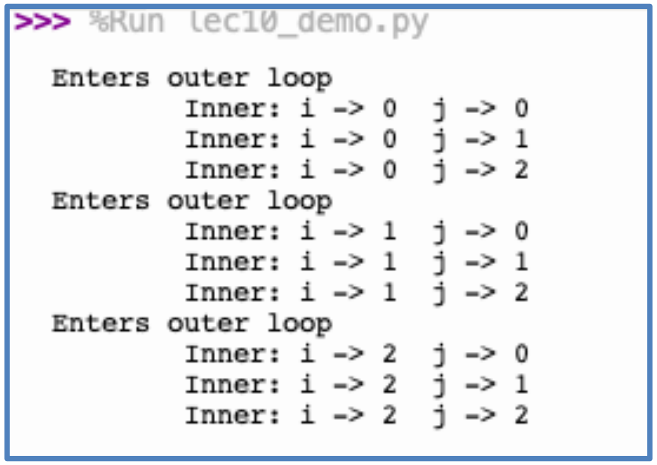
Visualization of nested **for** loop

```
# nested for loop
for i in range(3):
    print("Enters outer loop")
    for j in range(3):
        print("\tInner: i ->", i, " j ->", j)
```

Thonny output: nested **for** loop

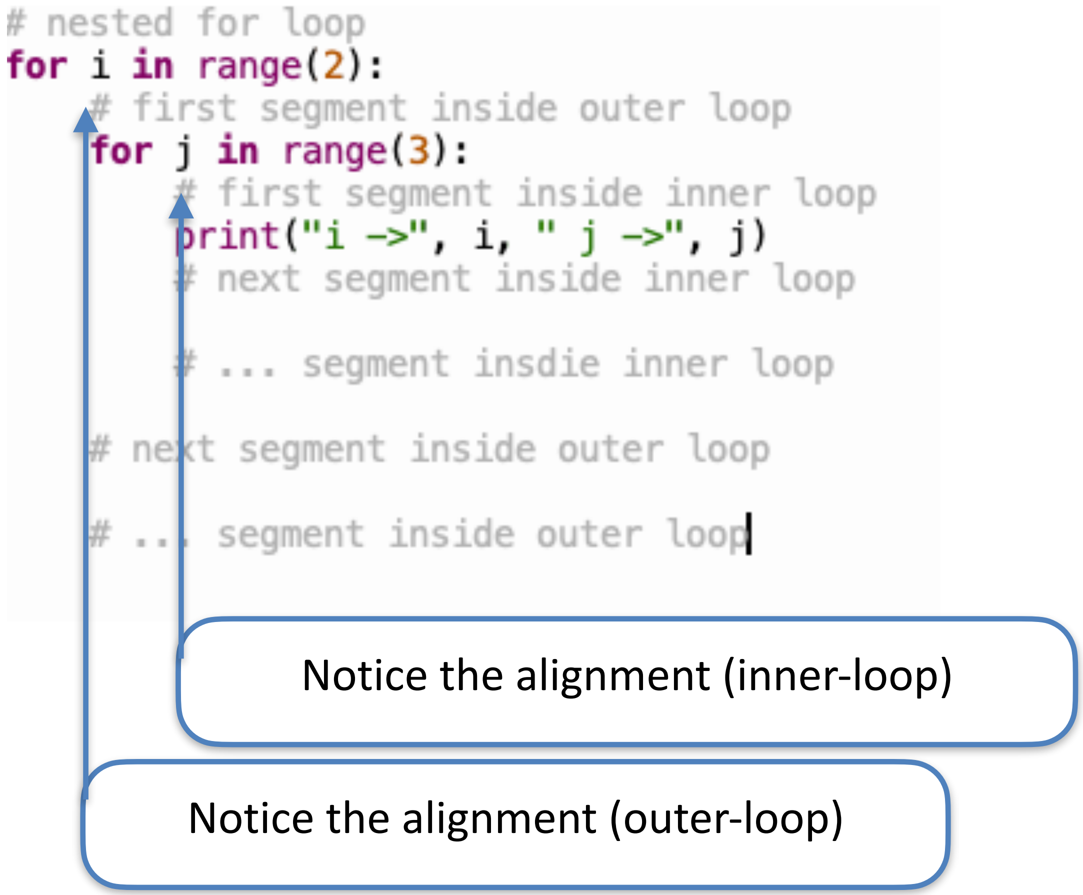


```
# nested for loop
for i in range(3):
    print("Enters outer loop")
    for j in range(3):
        print("\tInner: i ->", i, " j ->", j)
```



```
>>> %Run lec10_demo.py

  Enters outer loop
          Inner: i -> 0   j -> 0
          Inner: i -> 0   j -> 1
          Inner: i -> 0   j -> 2
  Enters outer loop
          Inner: i -> 1   j -> 0
          Inner: i -> 1   j -> 1
          Inner: i -> 1   j -> 2
  Enters outer loop
          Inner: i -> 2   j -> 0
          Inner: i -> 2   j -> 1
          Inner: i -> 2   j -> 2
```

Visualization of nested **for** loop



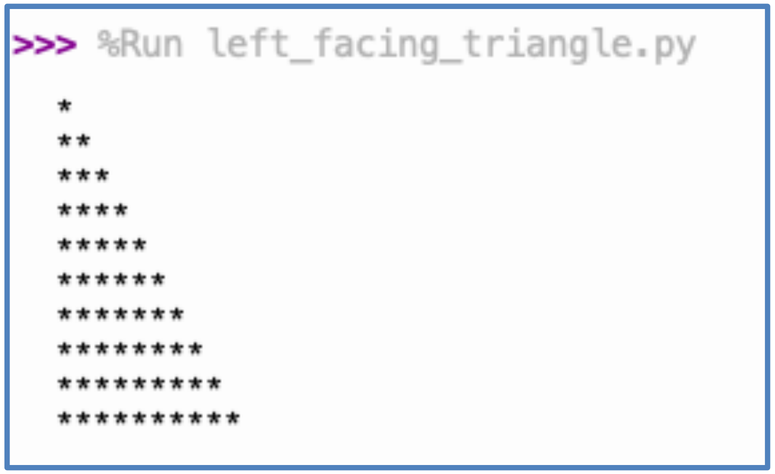
```
# nested for loop
for i in range(2):
    # first segment inside outer loop
    for j in range(3):
        # first segment inside inner loop
        print("i ->", i, " j ->", j)
        # next segment inside inner loop

        # ... segment insdie inner loop

    # next segment inside outer loop

    # ... segment inside outer loop
```

Notice the alignment (inner-loop)

Notice the alignment (outer-loop)

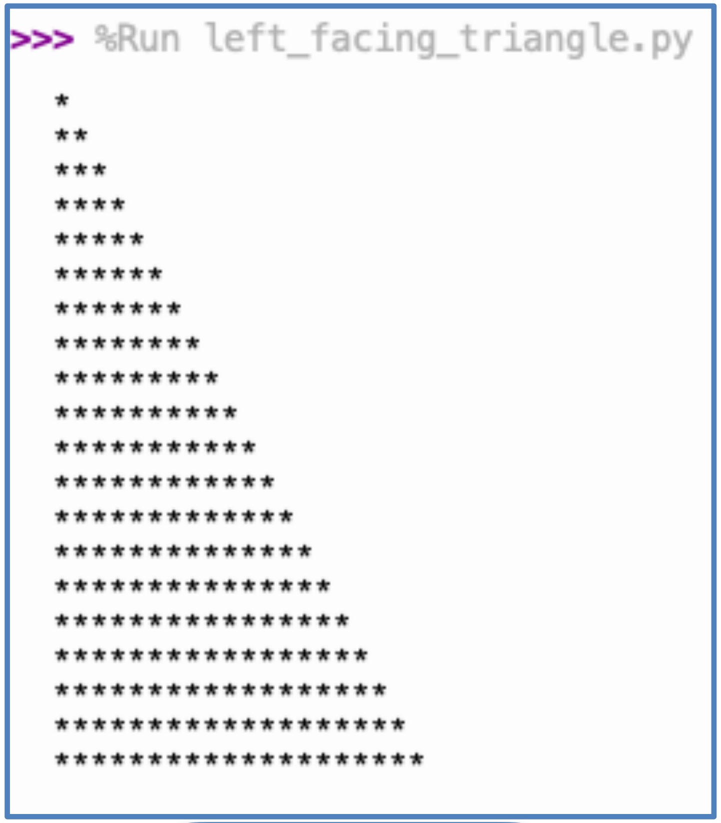
Exercise 9

- Build a left-facing-triangle in the shell output that looks like this:



```
>>> %Run left_facing_triangle.py

 *
 **
 ***
 ****
 *****
 ******
 *******
 ********
 *********
 **********
```

Side length = 10



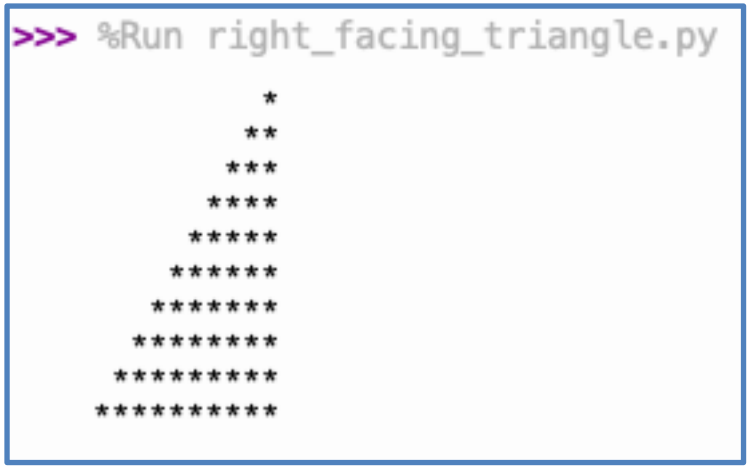
```
>>> %Run left_facing_triangle.py

 *
 **
 ***
 ****
 *****
 ******
 *******
 ********
 *********
 **********
 ***********
 ************
 *************
 **************
 ***************
 ****************
 *****************
 ******************
 *******************
 ********************
```

Side length = 20

- You can use any special character of your choice as a brick, and my favorite is the '*' character :)

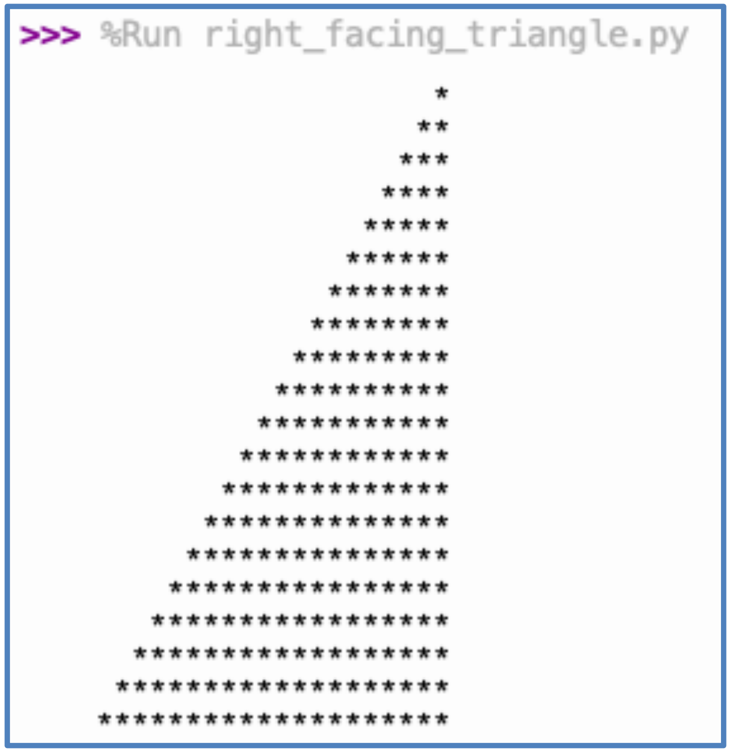
Exercise 10

- Build a right-facing-triangle in the shell output that looks like this:



```
>>> %Run right_facing_triangle.py
         *
        **
       ***
      ****
     *****
    ******
   *******
  ********
 *********
**********
```

Side length = 10

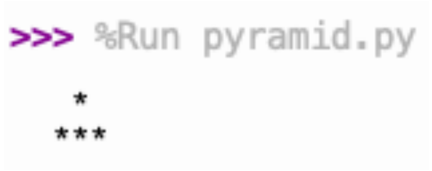


```
>>> %Run right_facing_triangle.py
                   *
                  **
                 ***
                ****
               *****
              ******
             *******
            ********
           *********
          **********
         ***********
        ************
       *************
      **************
     ***************
    ****************
   *****************
  ******************
 *******************
********************
```

Side length = 20

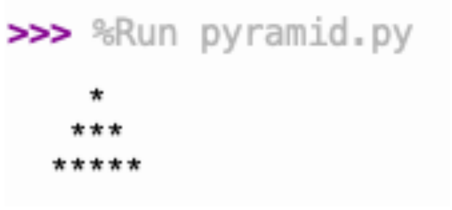
Bonus: more complicated nested **for** loops

- Build a pyramid in the shell output that looks like this:



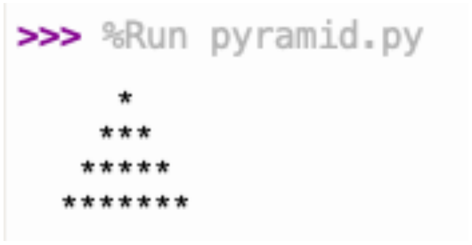
```
>>> %Run pyramid.py
 *
***
```

Base length= 3



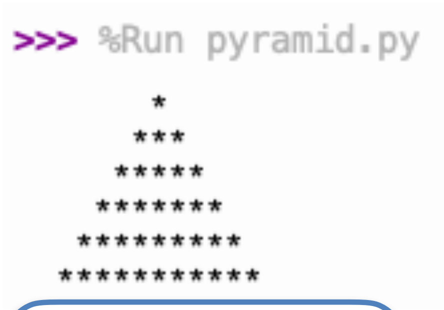
```
>>> %Run pyramid.py
  *
 ***
*****
```

Base length= 5



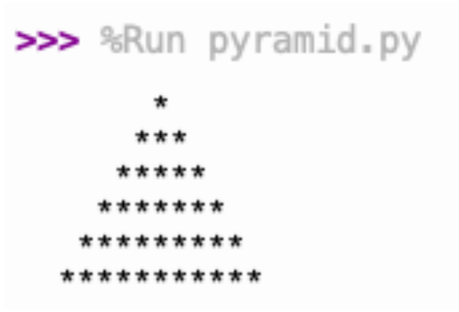
```
>>> %Run pyramid.py
   *
  ***
 *****
*******
```

Base length= 7



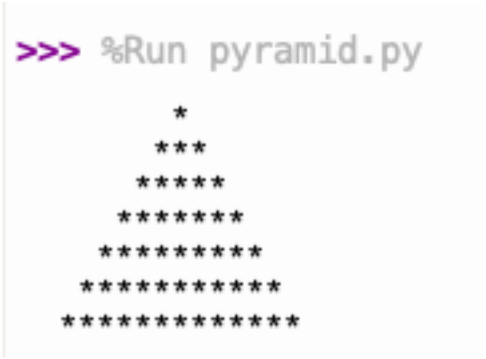
```
>>> %Run pyramid.py
    *
   ***
  *****
 *******
*********
```

Base length= 9



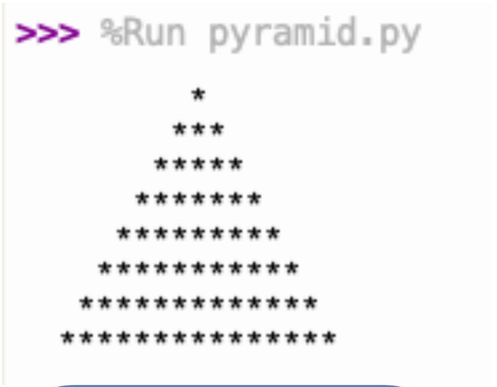
```
>>> %Run pyramid.py
     *
    ***
   *****
  *******
 *********
***********
```

Base length= 11



```
>>> %Run pyramid.py
      *
     ***
    *****
   *******
  *********
 ***********
*************
```

Base length= 13



```
>>> %Run pyramid.py
       *
      ***
     *****
    *******
   *********
  ***********
 *************
***************
```

Base length= 15

Note: Don't expect to see this question in your exam or assignments. Nonetheless, it is a good exercise.

Summary

- **Announcements:**
  - Assignment 2 will be out by today/tomorrow! It will be due in 2 weeks.
  - Next Tuesday (03/08/22), there will be a quiz.
    - Topics:
      - Accessing elements in a String
      - Length of a String
      - while loop
      - for loop