Module 05

For Loops

Emily Zhao
CSCI–UA.0002

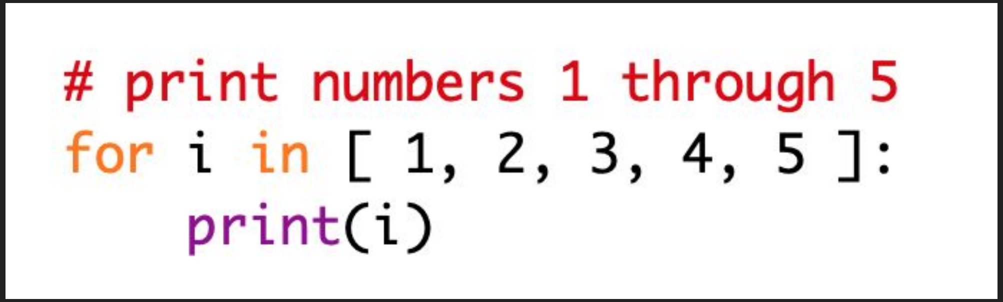
Agenda

— Review Module 5 + Quiz 5
— Practice Problems

Tell me everything you know about for loops

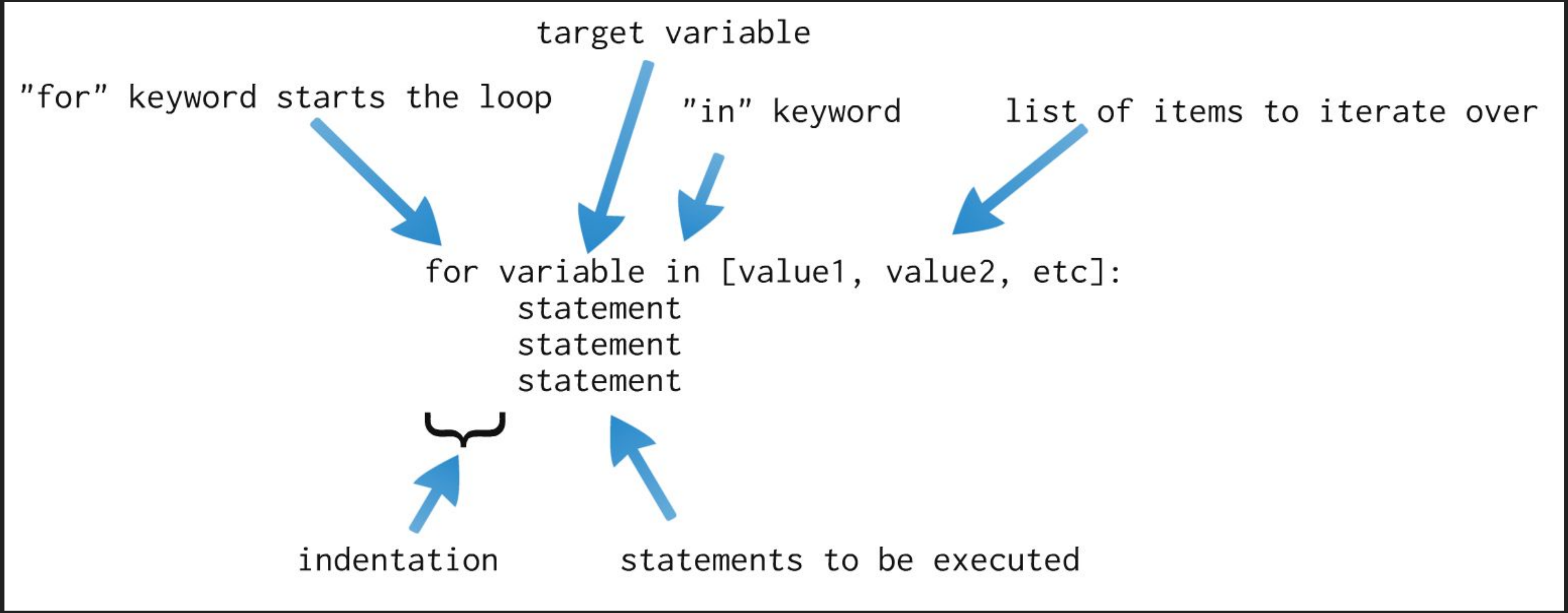
The “for” loop

- The “for” loop will iterate once for each item defined in the list passed to it when the loop begins
- Lists in Python are defined by the square bracket characters “[“ and “]”. Items are separated by a comma.
- The first time a “for” loop iterates the target variable will assume the value of the first item in the list
- The second time a “for” loop iterates the target variable will assume the value of the second item in the list
- This continues until you reach the end of the list



```python
# print numbers 1 through 5
for i in [ 1, 2, 3, 4, 5 ]:
    print(i)
```

The “for” loop

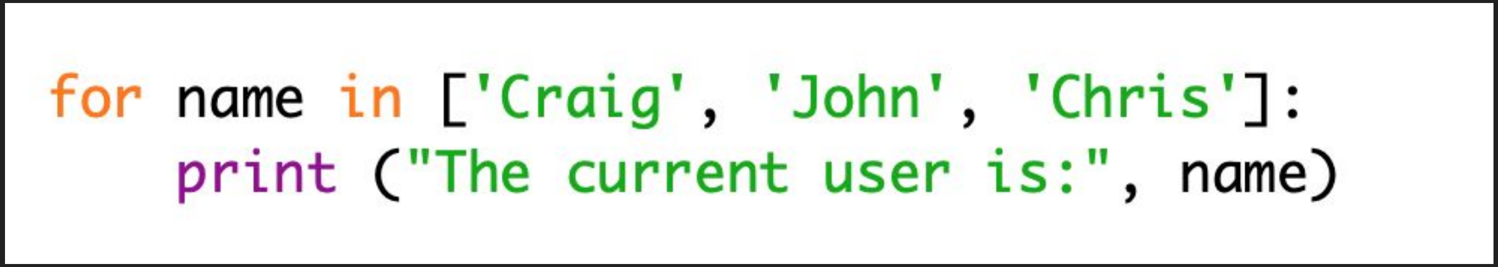

target variable
"for" keyword starts the loop
"in" keyword
list of items to iterate over
indentation
statements to be executed

```
for variable in [value1, value2, etc]:
    statement
    statement
    statement
```

True or False: You can name the target variable anything you want.

Lists

- We will talk more about lists near the end of the semester.
- With that said, lists can contain collections of different data:



```python
for name in ['Craig', 'John', 'Chris']:
    print ("The current user is:", name)
```

```
> The current user is: Craig
> The current user is: John
> The current user is: Chris
```

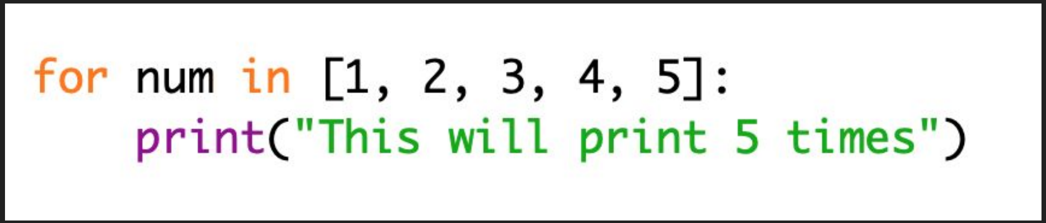
Strings

```python
for letter in "Emily":
    print(letter)
```

```
> E
  M
  I
  L
  Y
```

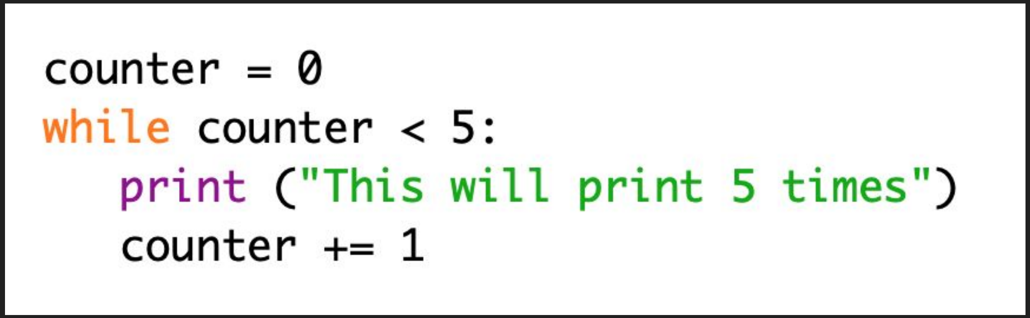
Count Controlled vs Condition Controlled

A **count** controlled loop is a repetition structure that iterates a specific number of times



```
for num in [1, 2, 3, 4, 5]:
    print("This will print 5 times")
```

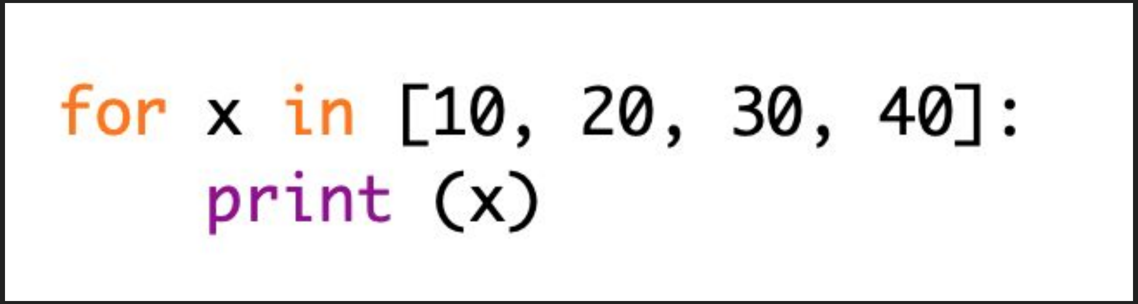
In contrast, a **condition** controlled loop iterates a variable number of times – we control the # of iterations through our Boolean condition



```
counter = 0
while counter < 5:
   print ("This will print 5 times")
   counter += 1
```

Every count-controlled loop can be written as a while loop

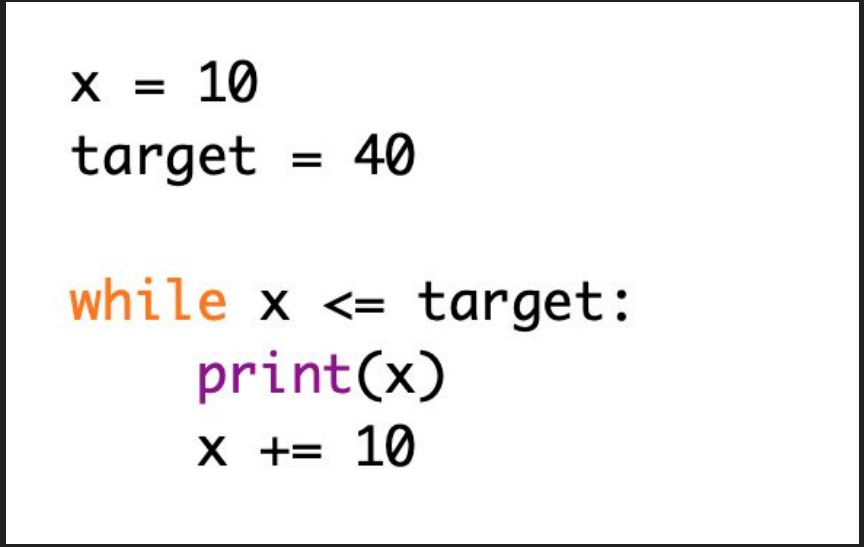
Rewrite the following loop as a “while” loop



```python
for x in [10, 20, 30, 40]:
    print (x)
```

Goal:
print the multiples of 10 from a
starting value (10) to a target value (40)

Boolean expression to test:
Have we reached our target value yet?
Until we do, increment our starting number by 10



```python
x = 10
target = 40

while x <= target:
    print(x)
    x += 10
```

`range()`

`range()`

`range(`*`stop`*`)`

`range(`*`start, stop, step`*`)`

- Returns a sequence of numbers, starting from 0 by default, and increments by 1 by default, and stops before a specified number.
- In its simplest form, it takes a single integer, the number at which it stops before.
- The range() function returns an “iterable”, which is a Python data type similar to a list.

What’s the output?

```
range(5)            → [0, 1, 2, 3, 4]
range(1, 5)         → [1, 2, 3, 4]
range(5, 10)        → [5, 6, 7, 8, 9]
range(0, 10, 2)     → [0, 2, 4, 6, 8]
range(1, 10, 2)     → [1, 3, 5, 7, 9]
range(10, 0, -3)    → [10, 7, 4, 1]
range(0, 10, -3)    →
```

How many different ways can you write it?

Print every multiple of 3 from 0 to 100.

- While loop
- For loop
- Conditionals

Print every multiple of 3 between a starting input and ending input. Are there less ways to do this?

Programming Challenge: FizzBuzz

A classic interview question for computer programming jobs

- Write a program that prints the numbers from 1 to 100
  - For the multiples of 3, print “Fizz” instead of the number
  - For the multiples of 5, print “Buzz” instead of the number
  - For numbers which are multiples of both 3 and 5, print “FizzBuzz”

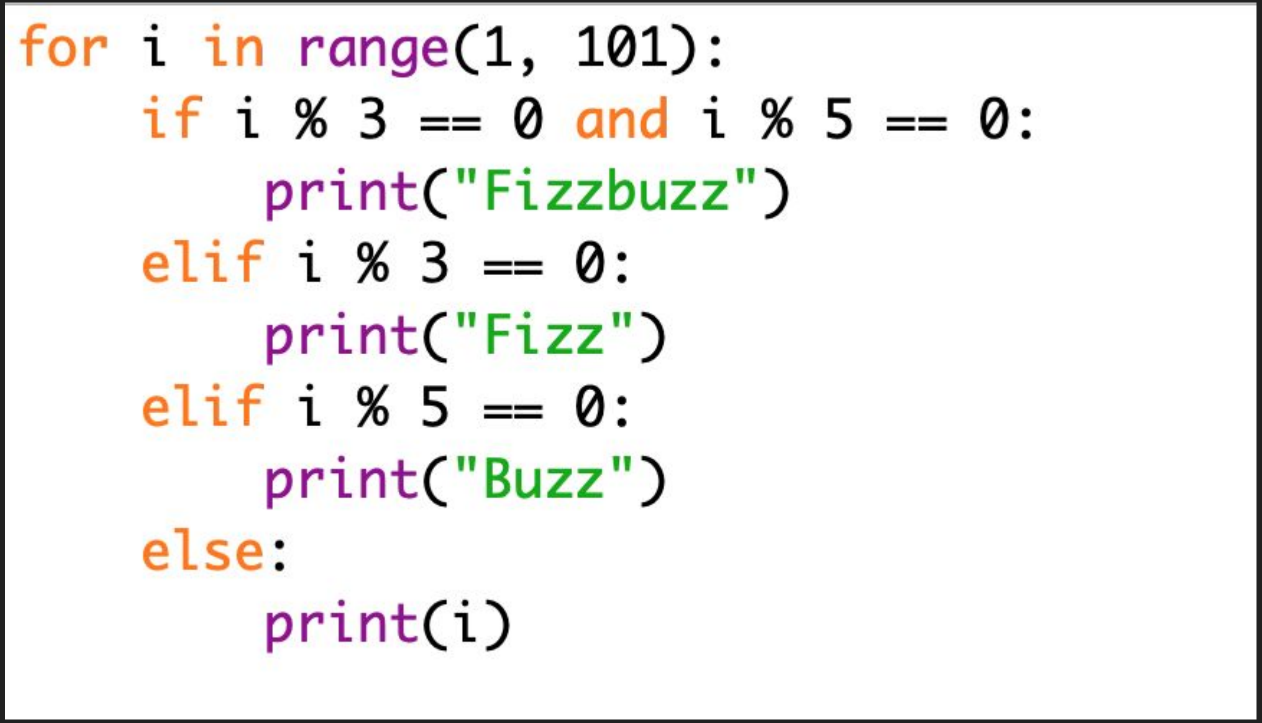
```
> 1, 2, Fizz, 4, Buzz, Fizz, 7, 8, Fizz,
  10, 11, Fizz, 13, 14, FizzBuzz…
```

FizzBuzz (thought process)

- How do I print the numbers 1 through 100?
- How do I check a number’s divisibility by 3, 5, or both?
  - What operator do I have to use?
- Once I’ve found a way to check, which condition should I check first?
- How do I go about structuring my code?

Logic:
Print corresponding string depending on divisibility
Otherwise, just print the number

FizzBuzz [SOLUTION]



```python
for i in range(1, 101):
    if i % 3 == 0 and i % 5 == 0:
        print("Fizzbuzz")
    elif i % 3 == 0:
        print("Fizz")
    elif i % 5 == 0:
        print("Buzz")
    else:
        print(i)
```

Programming Challenge: Stair Steps

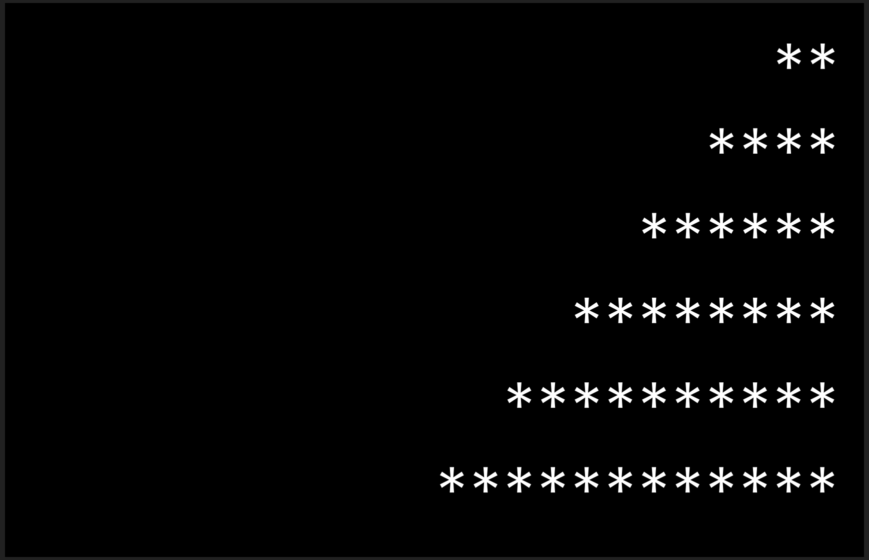
Write a program using a for loop that prints out the following pattern of characters. (If you're stuck, print the output first without for loops. Is there a pattern?)

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

Extension: ask the user for starting starting amount and ending amount of asterisks, making sure that the ending amount is greater than the starting amount. You can also ask the user for the "step" value at which they would like to increase the asterisks by.

Programming Challenge: Stair Steps

Extension: Right align the output



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

Programming Challenge: Stair Steps [Solution pt. 1]

```
**

****

******

********

**********

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

What's the pattern?

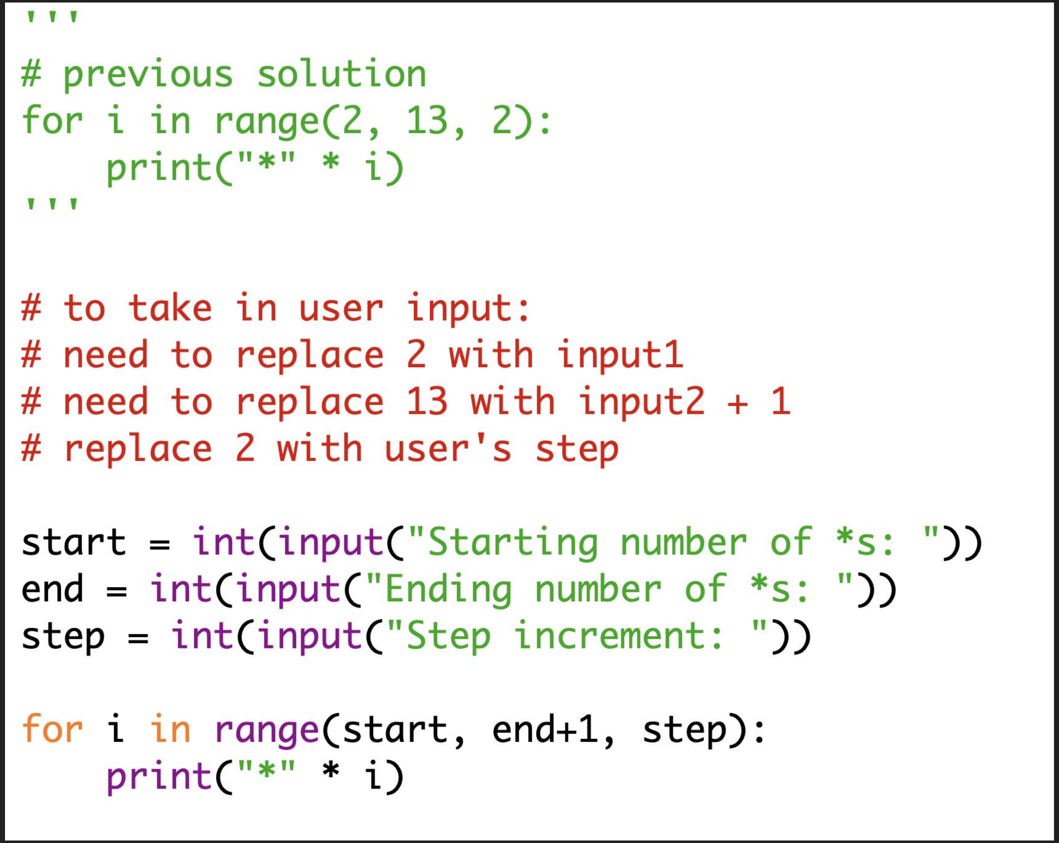
```python
# hard-code it first

print("*" * 2)
print("*" * 4)
print("*" * 6)

# looks like I need to generate 2, 4, 6...
# I can do that with range
# start at 2, end at 12+1, step by 2
# range(2, 12+1, 2)

for i in range(2, 13, 2):
    print("*" * i)
```

Programming Challenge: Stair Steps [Solution pt. 2]

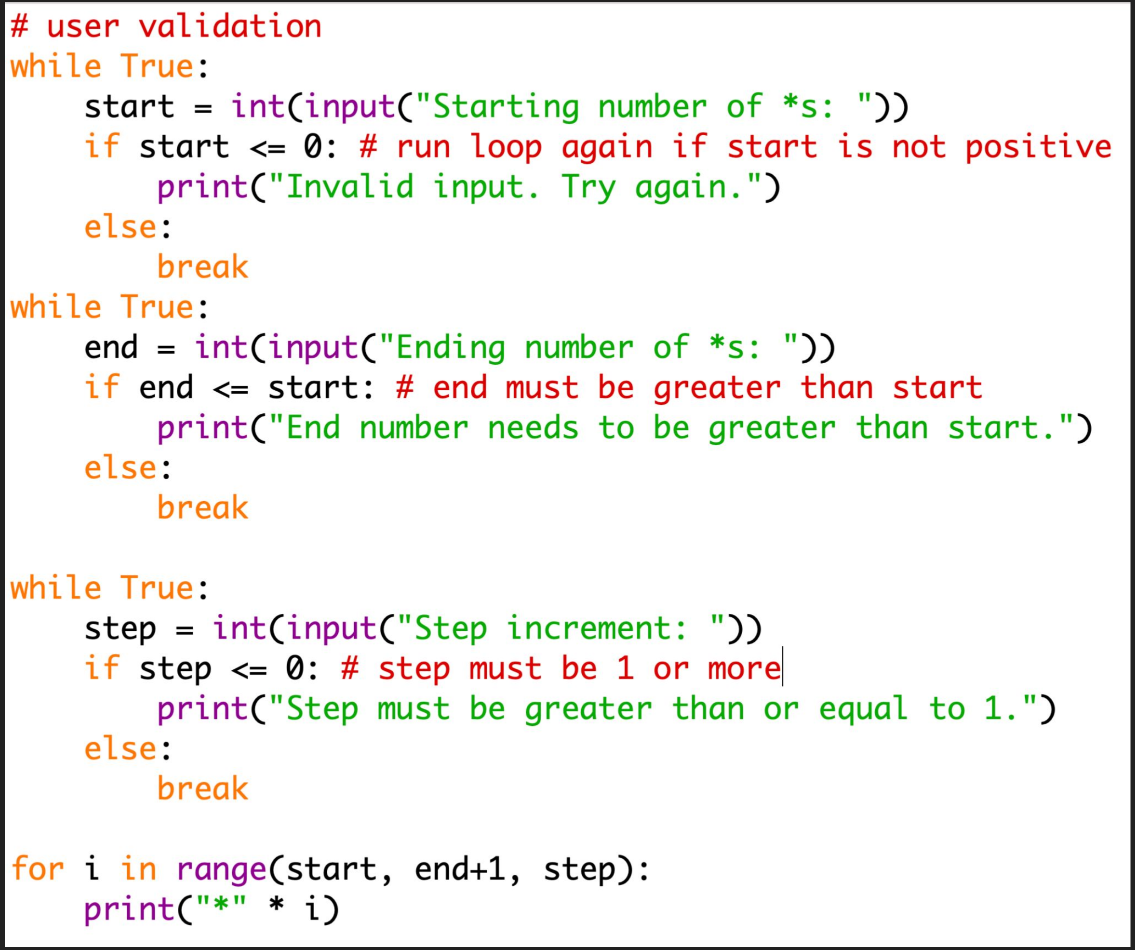


```python
'''
# previous solution
for i in range(2, 13, 2):
    print("*" * i)
'''

# to take in user input:
# need to replace 2 with input1
# need to replace 13 with input2 + 1
# replace 2 with user's step

start = int(input("Starting number of *s: "))
end = int(input("Ending number of *s: "))
step = int(input("Step increment: "))

for i in range(start, end+1, step):
    print("*" * i)
```

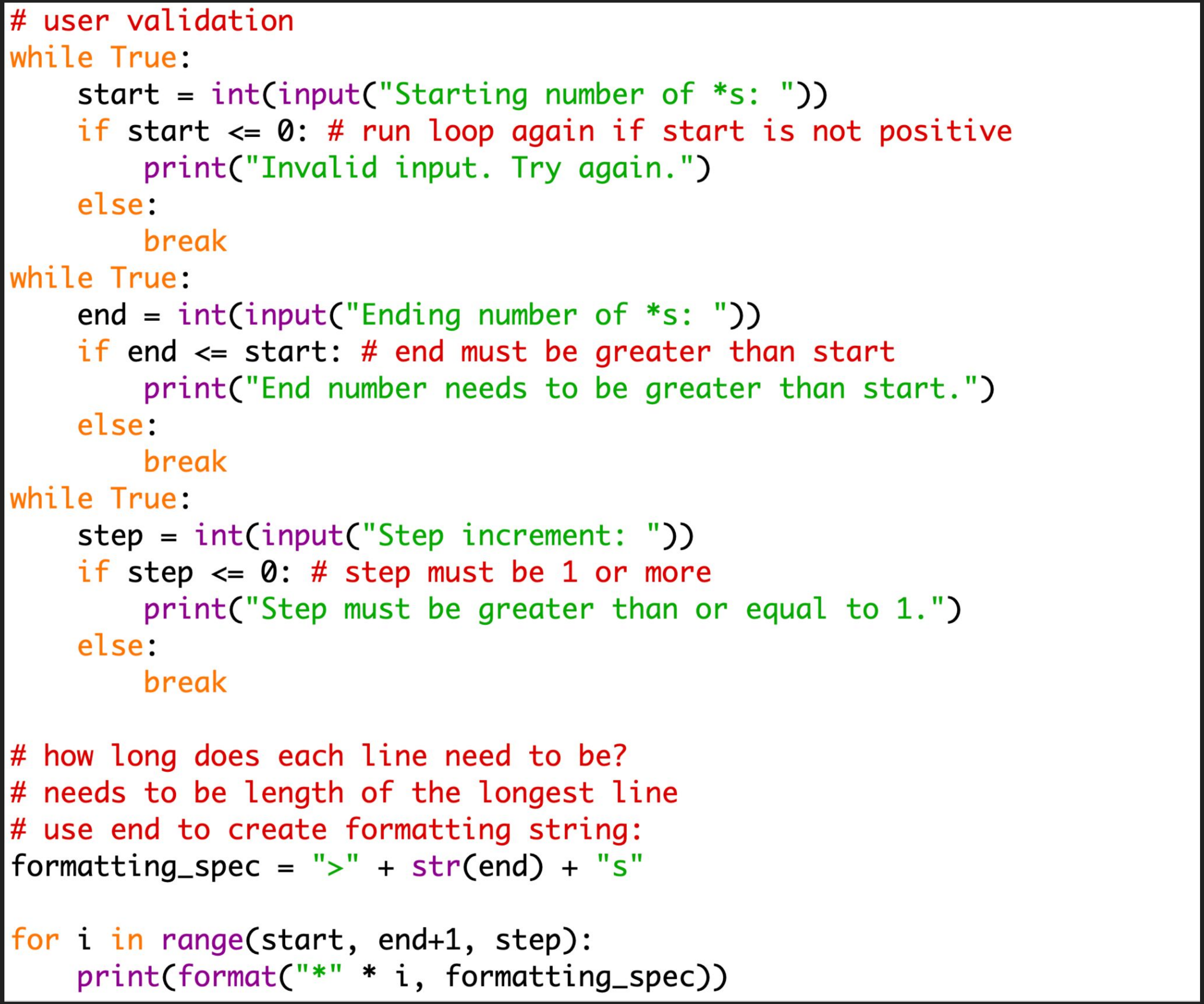


```python
# user validation
while True:
    start = int(input("Starting number of *s: "))
    if start <= 0: # run loop again if start is not positive
        print("Invalid input. Try again.")
    else:
        break
while True:
    end = int(input("Ending number of *s: "))
    if end <= start: # end must be greater than start
        print("End number needs to be greater than start.")
    else:
        break

while True:
    step = int(input("Step increment: "))
    if step <= 0: # step must be 1 or more
        print("Step must be greater than or equal to 1.")
    else:
        break

for i in range(start, end+1, step):
    print("*" * i)
```

Programming Challenge: Stair Steps [Solution pt. 3]



```python
# user validation
while True:
    start = int(input("Starting number of *s: "))
    if start <= 0: # run loop again if start is not positive
        print("Invalid input. Try again.")
    else:
        break
while True:
    end = int(input("Ending number of *s: "))
    if end <= start: # end must be greater than start
        print("End number needs to be greater than start.")
    else:
        break
while True:
    step = int(input("Step increment: "))
    if step <= 0: # step must be 1 or more
        print("Step must be greater than or equal to 1.")
    else:
        break

# how long does each line need to be?
# needs to be length of the longest line
# use end to create formatting string:
formatting_spec = ">" + str(end) + "s"

for i in range(start, end+1, step):
    print(format("*" * i, formatting_spec))
```

Nested For Loops

Nested Loops

- A nested loop can be described as a "loop inside of a loop"
- It's the same idea as nested selection statements ("if" statements inside other "if" statements)

Nested Loops

- The innermost loop will iterate through all its iterations for every single iteration of an outer loop
- Inner loops complete their iterations faster than outer loops
- To get the total number of iterations of a nested loop, multiply the number of iterations of all the loops

Programming Challenge: Clock Simulator

Write a program that prints out every possible time value for a single day

- Print out the hours and minutes
    - 0:0
    - 0:1
    - ...
    - 23:59

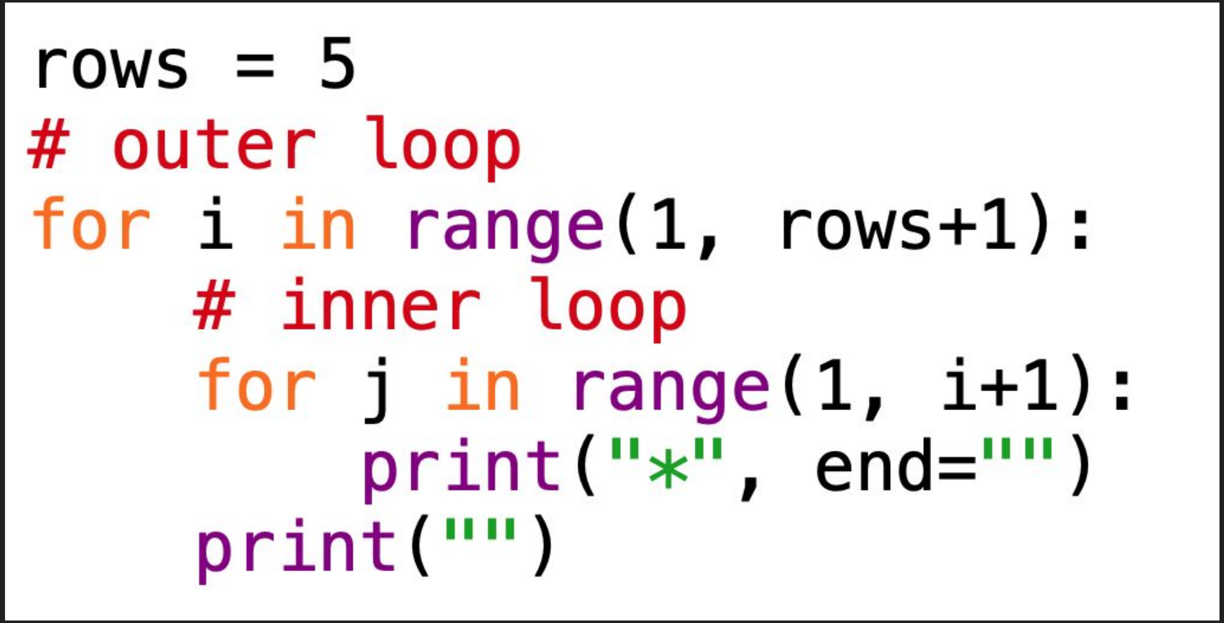
Extension:

- Can you get your output to have leading 0s?
- Print out the seconds, too.
- Example: 01:03:22

Programming Challenge: Clock Simulator [SOLUTION]

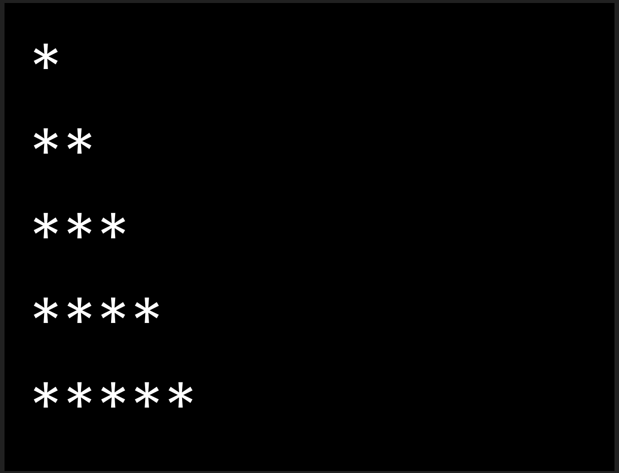
```python
for h in range(0,25):
    for m in range(0,60):
        for s in range(0, 60):
            print(format(h, "02d"), format(m, "02d"), format(s, "02d"), sep= ":")
```

What’s the output?



```python
rows = 5
# outer loop
for i in range(1, rows+1):
    # inner loop
    for j in range(1, i+1):
        print("*", end="")
    print("")
```

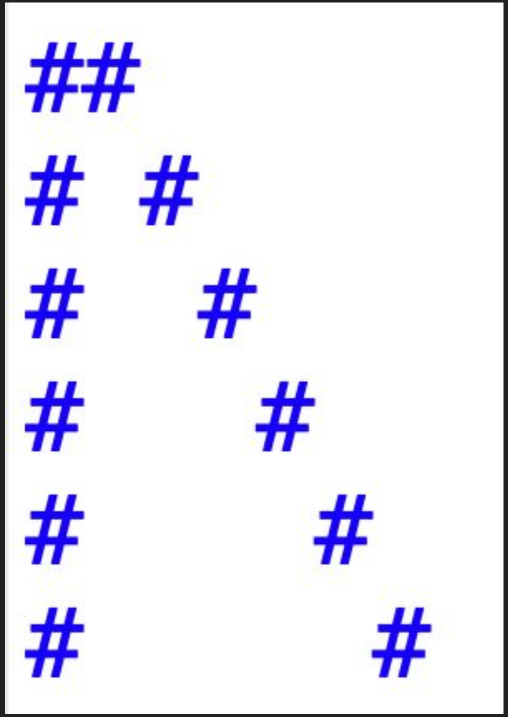
→



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

Programming Challenge: Graphical Pattern

Write a program that prints the following pattern:



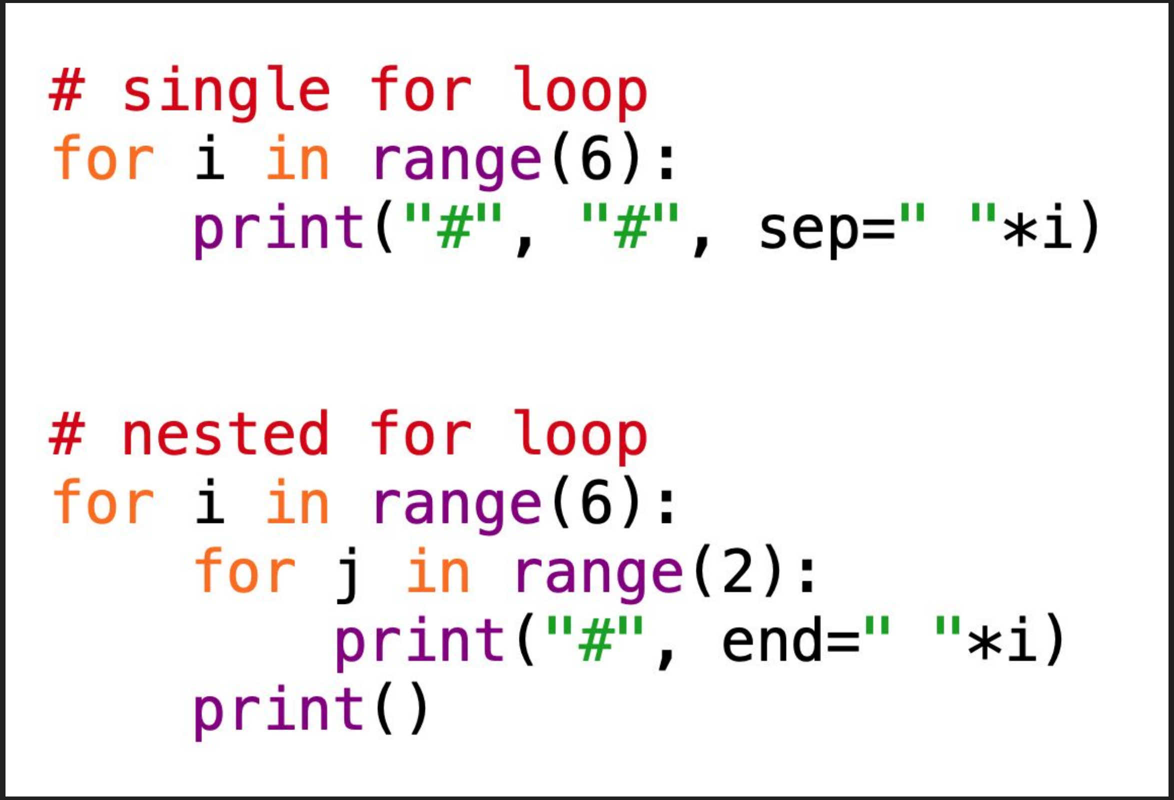
```
##
# #
#  #
#   #
#    #
#     #
```

- What's your initial feeling? Single for loop or nested?
- Try writing it using both
- Which one do you like better?
- Is there something one can do that the other can't?

Logic:

- How do I print two #s with a for loop?
- How do I print 6 lines of just single #s?
- What is the correlation of the spaces between the #s and the line number?

Programming Challenge: Graphical Pattern



```python
# single for loop
for i in range(6):
    print("#", "#", sep=" "*i)


# nested for loop
for i in range(6):
    for j in range(2):
        print("#", end=" "*i)
    print()
```

Programming Challenge: Drawing Shapes

- Ask the user how many sides they would like their shape to be (up to 8) or specify random if they don’t care. (Assume they won’t type an integer greater than 8)
- Then ask if they would like it drawn or named
  - If they want it drawn, draw it using turtle
  - If they want it named, tell them what it is (triangle, square, pentagon, etc…)

Homework

- Assignment #4 (due tonight)
- Self-Paced Learning Module #6 (due next Mon)
- Quiz #5 (due next Mon)