

Lecture 11:

LOOPS

CSC111: Introduction to CS through Programming

R. Jordan Crouser

Assistant Professor of Computer Science

Smith College

Debrief of first few coding assignments (1/2)

- Overall: really solid!
- Strengths:
 - nicely formatted submissions
 - informative variable names
 - great use of comments (particularly for leaving “notes” and making sections for individual components)
- Still working on:
 - balancing matching specs vs. being creative
 - sketching out a plan **before** you start coding
- Some tips:
 - use blank lines to separate “logical chunks” of code (like paragraphs)
 - Python automatically ignores blank lines! (no need to comment)

Debrief of first few coding assignments (2/2)

- Reminder:

Collaboration and Academic Integrity

Students are strongly encouraged to form study groups and to collaborate on assignments and labs. The following information is required for all submitted work:

1. The names of all collaborating students be listed at the top of the submission.
If you worked alone, please state: "*I did not collaborate with anyone on this assignment.*"
2. A "**References**" section, with in-line citations to any resources you used.
Citations should include page numbers (if a printed resource) or a direct URL (if an online resource). If you did not use any resources in completing the assignment, please state: "*I did not utilize any external resources in completing this assignment.*"

Debrief of first few coding assignments (2/2)

- Reminder:

Collaboration and Academic Integrity

Students are strongly encouraged to form study groups and to collaborate on assignments and labs. The following information is required for all submitted work:

1. The names of all collaborating students be listed at the top of the submission.
If you worked alone, please state: "*I did not collaborate with anyone on this assignment.*"
2. A "**References**" section, with in-line citations to any resources you used.
Citations should include page numbers (if a printed resource) or a direct URL (if an online resource). If you did not use any resources in completing the assignment, please state: "*I did not utilize any external resources in completing this assignment.*"

Overview

✓ Announcements

- Monday: Loops
 - `for...in` (looping through items in a list)
 - the `range()` function (getting a list of numbers)
 - `while` (looping until something happens)
- Lab: Loops
- Wednesday: Life Skill #2/3: Debugging & Documentation
- Friday: Dictionaries and Sets

Reality: online classes can be rough



Activity: Beats



Discussion

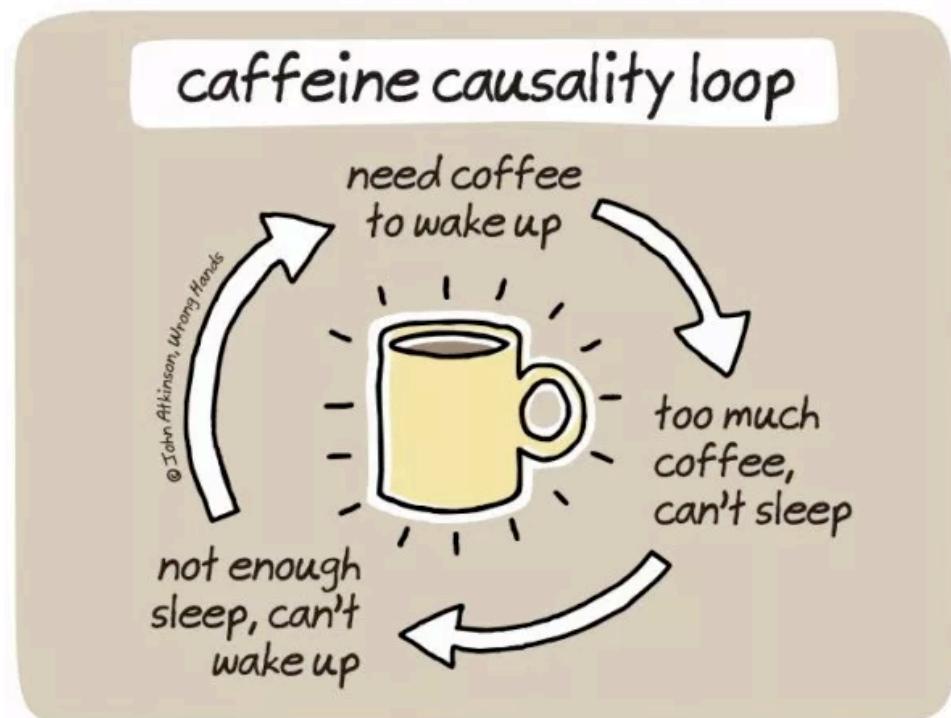
What kinds of **repetition** did
you notice?



Loops: a familiar idea



<https://xkcd.com/1411/>



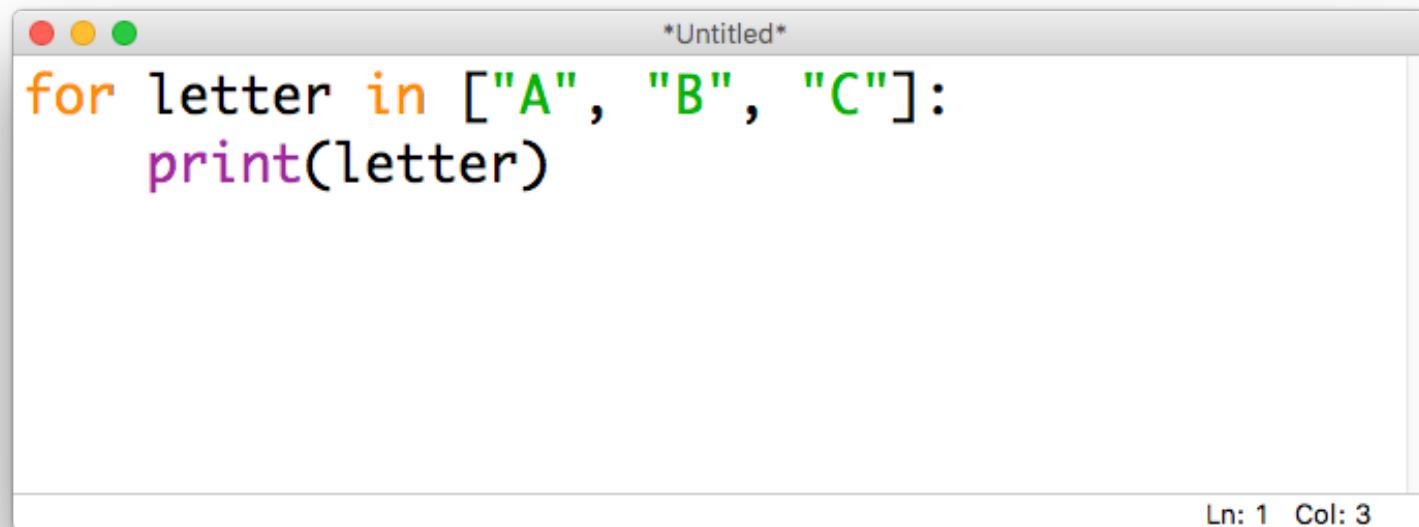
© John Atkinson, Wrong Hands

Loops in computer programming

- **Goal:** simplify the description of repeated blocks of code
(i.e. make it shorter/easier to understand by highlighting **what's being repeated** and **for how long**)
- **Three approaches:**
 - run for **each item** in a list
 - run a specific **number of times**
 - run **until** some condition is met

for...in loops

- In Python, we use the keywords **for** and **in** to loop through a list



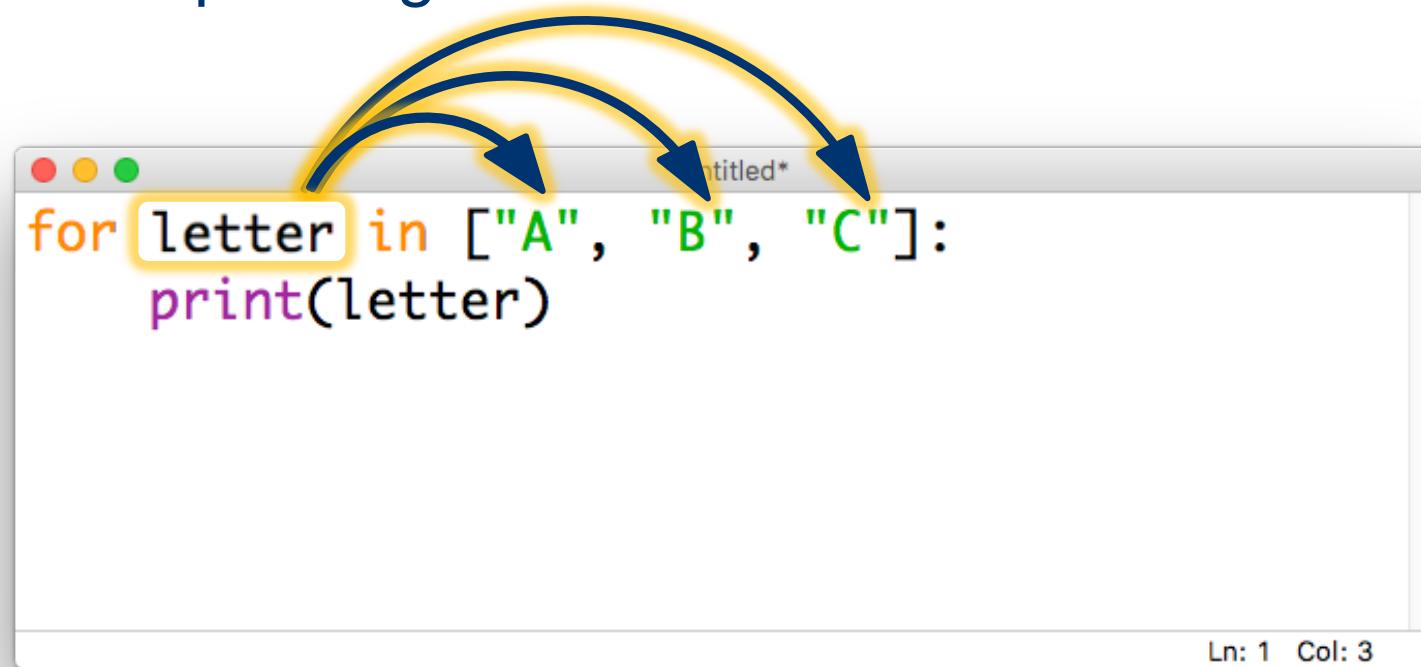
The image shows a screenshot of a Mac OS X terminal window. The window title is "*Untitled*". Inside the terminal, there is a single line of Python code:

```
for letter in ["A", "B", "C"]:  
    print(letter)
```

In the bottom right corner of the terminal window, there is a status bar displaying "Ln: 1 Col: 3".

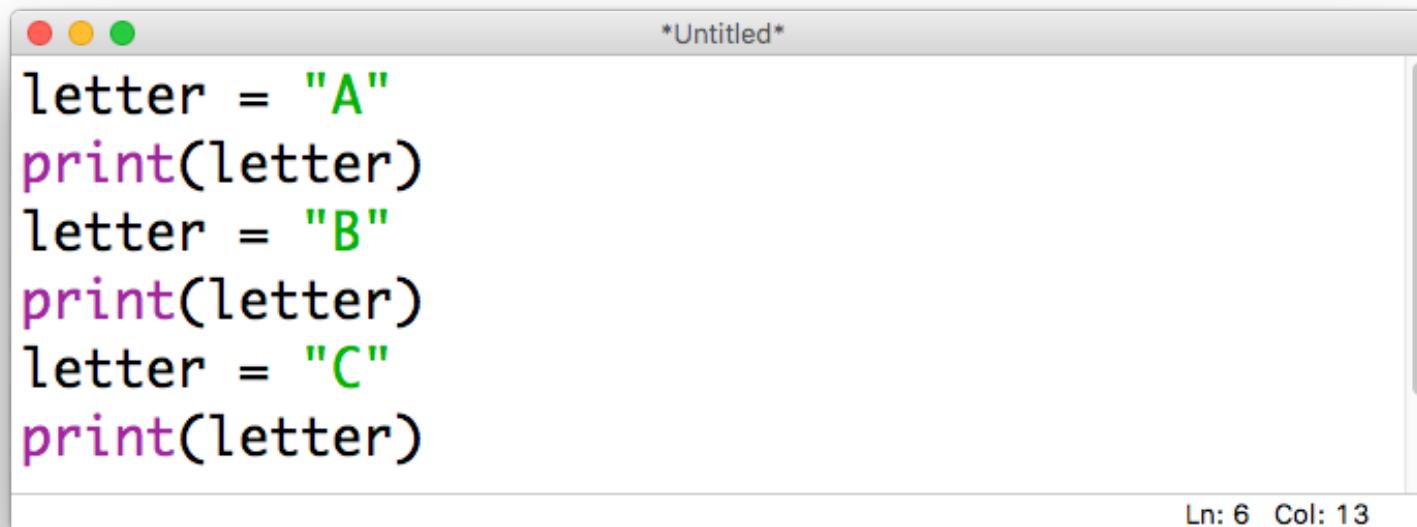
for...in loops

- We can think of this in terms of where the variable **letter** is pointing:



for...in loops: unpacked

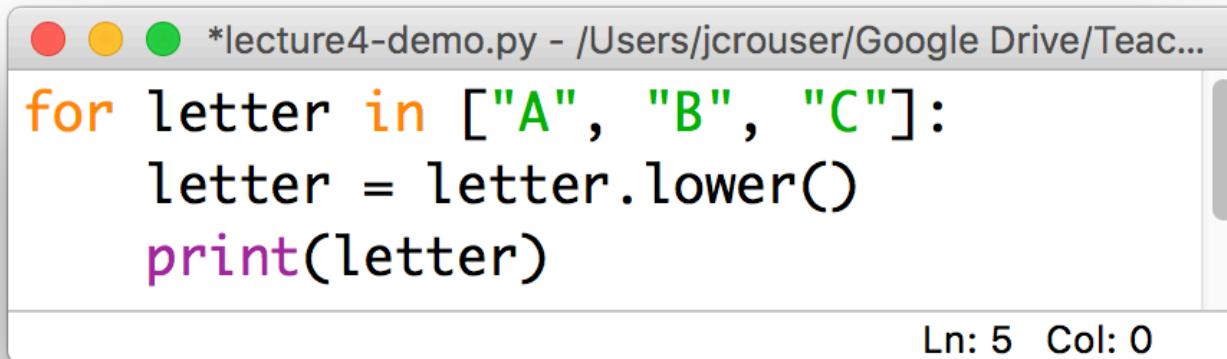
- We could accomplish the same thing by writing it out as **three separate assignments**:



```
letter = "A"
print(letter)
letter = "B"
print(letter)
letter = "C"
print(letter)
```

for...in loops: a common “gotcha”

- Python will allow you to **modify a list** while you’re looping through it:



A screenshot of a terminal window titled "*lecture4-demo.py - /Users/jcrouser/Google Drive/Teac...". The window contains the following Python code:

```
for letter in ["A", "B", "C"]:
    letter = letter.lower()
    print(letter)
```

The status bar at the bottom right shows "Ln: 5 Col: 0".

- This is generally a **bad idea** (more on why later)
 - it's fine to format the **values**, etc.
 - just don't **overwrite** the originals!

Demo: compute a sum

Use a **for** loop to compute
the **sum** of a list of numbers

- **Step 1:** pseudocode
- **Step 2:** python

Looping n times

- **Bad news:** there isn't a way to say “run this loop n times” in Python – we'll have to find a way around that
- If we want a **for...in** loop to run a specific # of times, can “trick” it using a list of numbers that's the right size

common
practice:
use the
letter **i**
to denote
the “index”

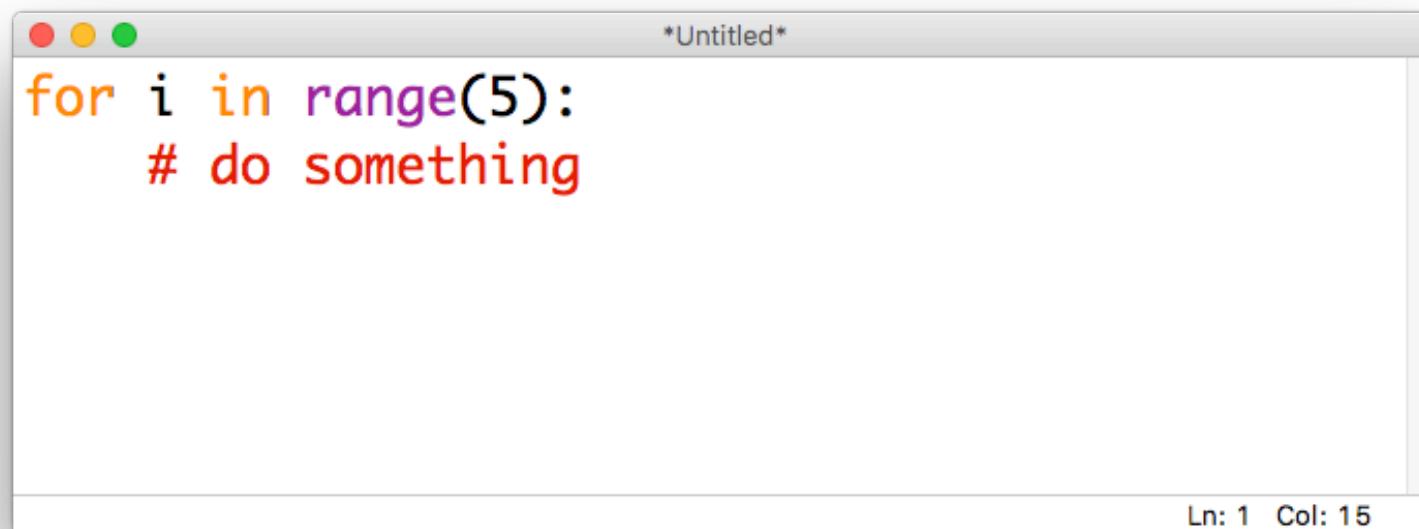


```
*Untitled*
for i in [1,2,3,4,5]:
    print("Hello!")
```

Ln: 2 Col: 19

The **range()** function

- The **range()** function lets us generate lists of integers
- Given **one** integer **a**, **range(a)** will generate a list starting at 0 and going up to (but not including) **a**
- For example, if we want a loop to run **5** times:



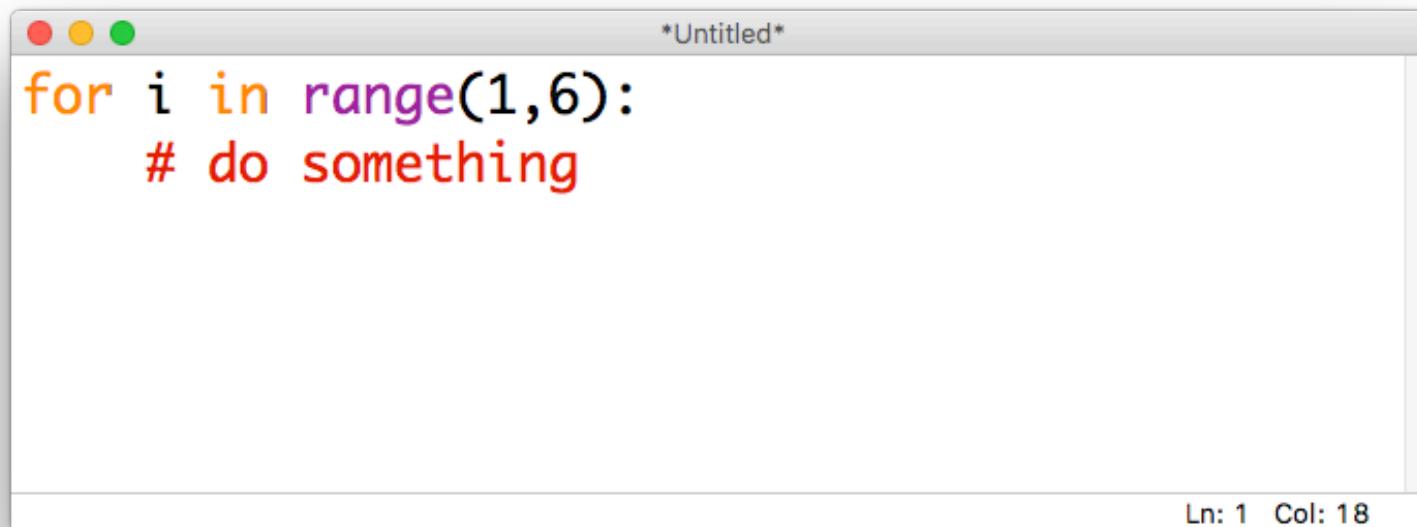
A screenshot of a code editor window titled "*Untitled*". The code in the editor is:

```
for i in range(5):
    # do something
```

The code uses color-coded syntax highlighting where `for`, `i`, and `range` are in blue, and the loop body is in red. The status bar at the bottom right shows "Ln: 1 Col: 15".

The **range()** function

- Given **two** integers **a, b**, **range(a, b)** will generate a list starting at **a** and going up to (but not including) **b**
- E.g., if we want to loop over the integers from 1 to 5:



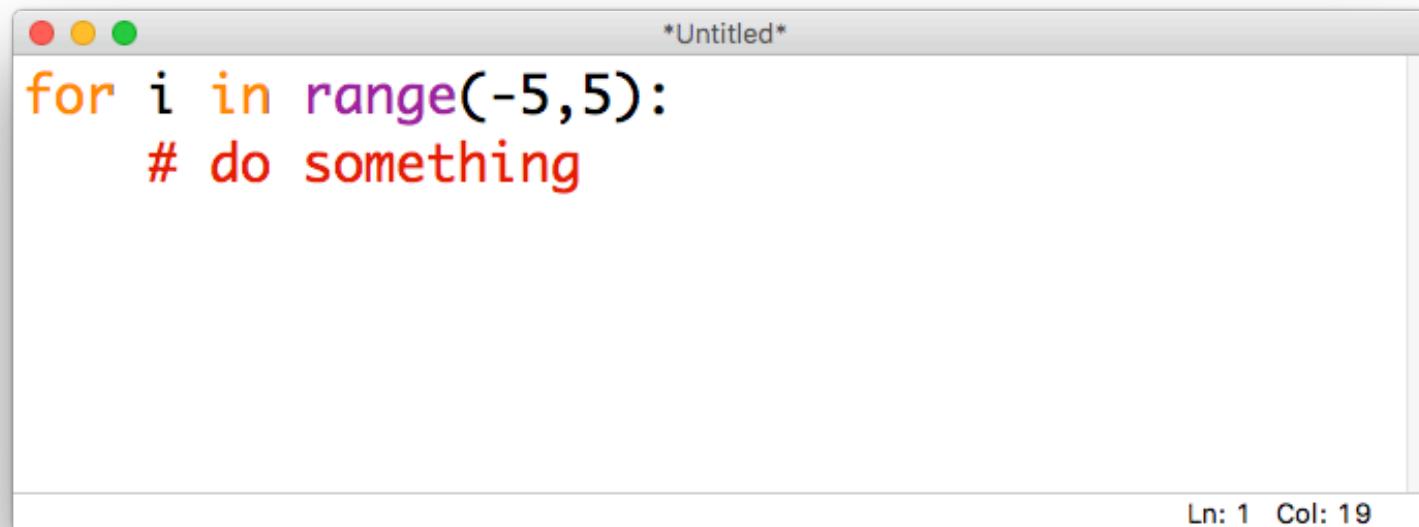
A screenshot of a code editor window titled '*Untitled*'. The code in the editor is:

```
for i in range(1,6):
    # do something
```

The code uses color-coded syntax highlighting where `for`, `i`, and `range` are in blue, and the numbers `1` and `6` are in green. The comment `# do something` is in red. The status bar at the bottom right of the editor shows 'Ln: 1 Col: 18'.

The **range()** function

- These values can be **positive** or **negative** (but for now, the second integer should be **larger** than the first)
- E.g., if we want to loop over the integers from -5 to 5:



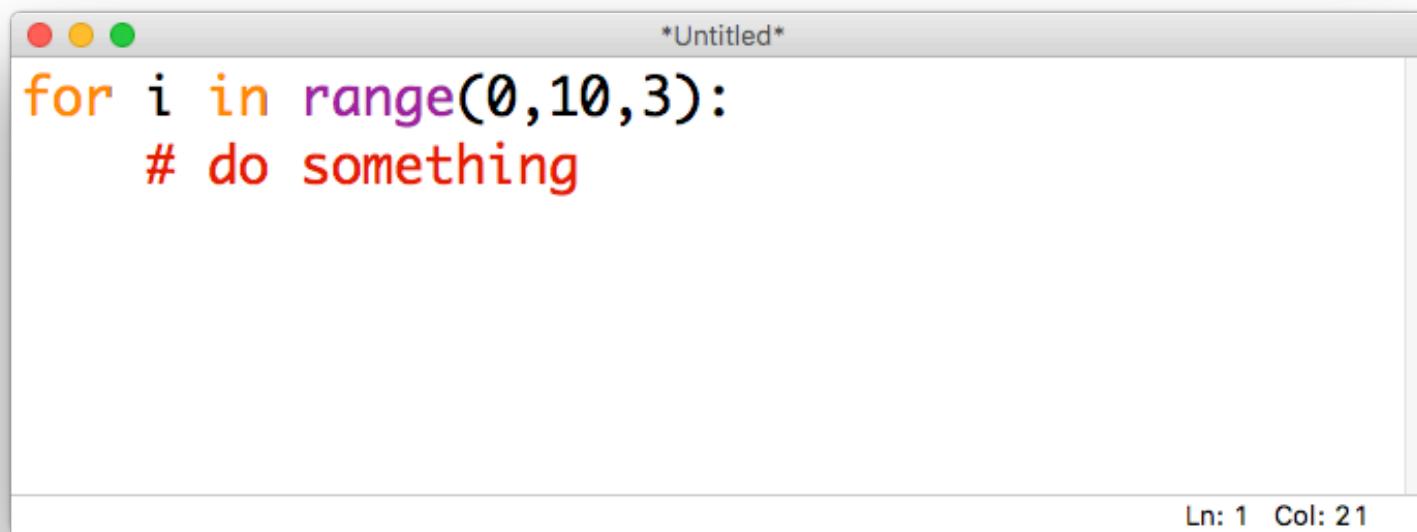
A screenshot of a code editor window titled '*Untitled*'. The code in the editor is:

```
for i in range(-5,5):
    # do something
```

The code uses color-coded syntax highlighting where `for`, `i`, and `range` are in blue, and the range arguments are in purple. The comment `# do something` is in red. The status bar at the bottom right of the editor shows "Ln: 1 Col: 19".

The **range()** function

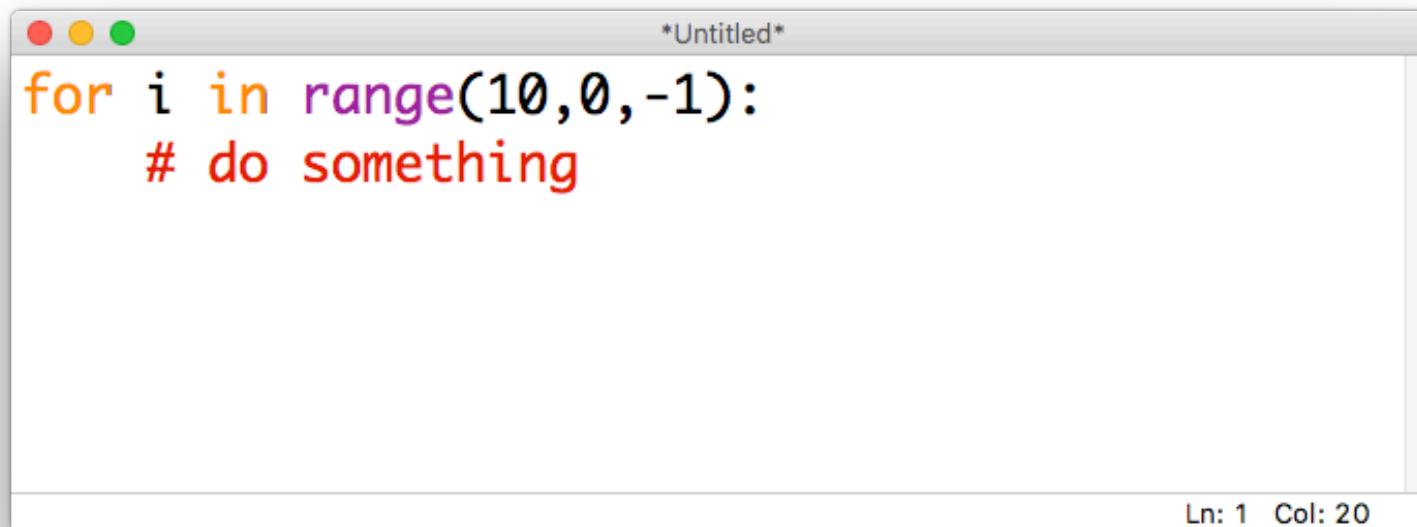
- Given **three** integers **a**, **b**, **c**, calling **range(a,b,c)** will generate a list starting at **a** and going up to (but not including) **b** with step size **c**
- E.g., if we want the integers from 0 to 9, counting by 3s:



```
*Untitled*
for i in range(0,10,3):
    # do something
Ln: 1 Col: 21
```

The **range()** function

- If we want to count down instead of up, we can set **b < a** and use a negative step size
- E.g., if we want to count down from 10 to 1:



A screenshot of a code editor window titled "*Untitled*". The code in the editor is:

```
for i in range(10, 0, -1):
    # do something
```

The code uses color-coded syntax highlighting. The status bar at the bottom right of the editor shows "Ln: 1 Col: 20".

15-Minute Exercise: convert °F to °C

Use a **for** loop and the **range()** function to generate a **conversion table** of temperatures from °F to °C ranging from 100°F to –30°F in increments of 10°F

Tips:

- use the formula $^{\circ}C = (^{\circ}F - 32) * 5 / 9$
- "`{0:.1f}`".format(c) will round to 1 decimal place

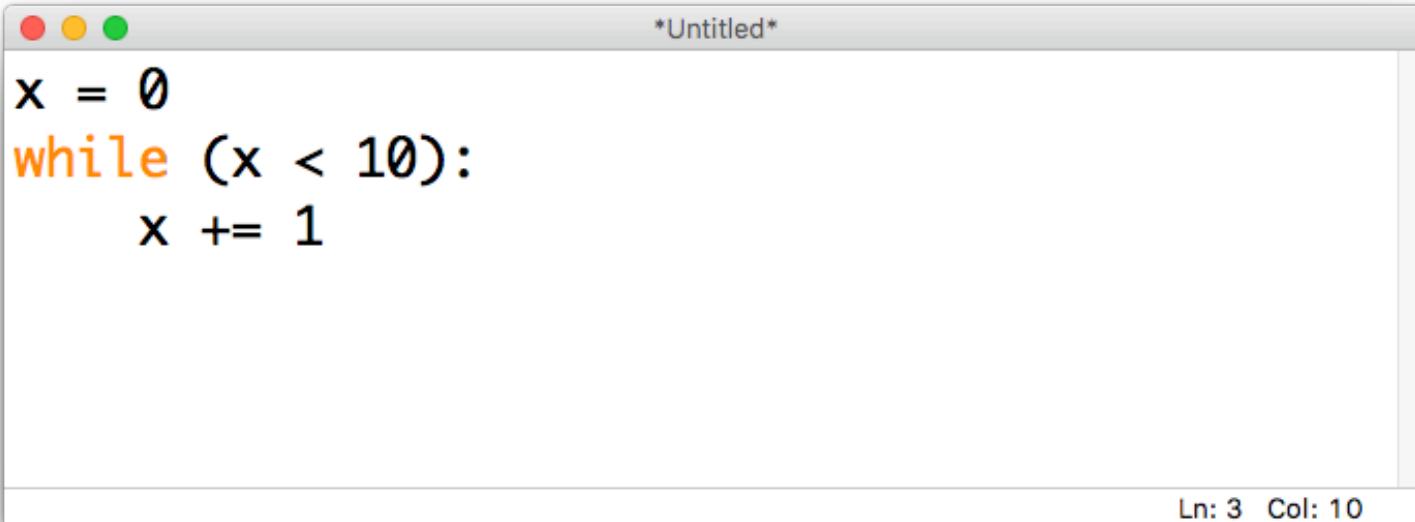
Discussion

What did you come up with?



while loops

- We may sometimes want a program to continue doing the same thing **until something happens**
- In Python we can do this with a **while** loop, which is paired with a conditional (True/False) statement



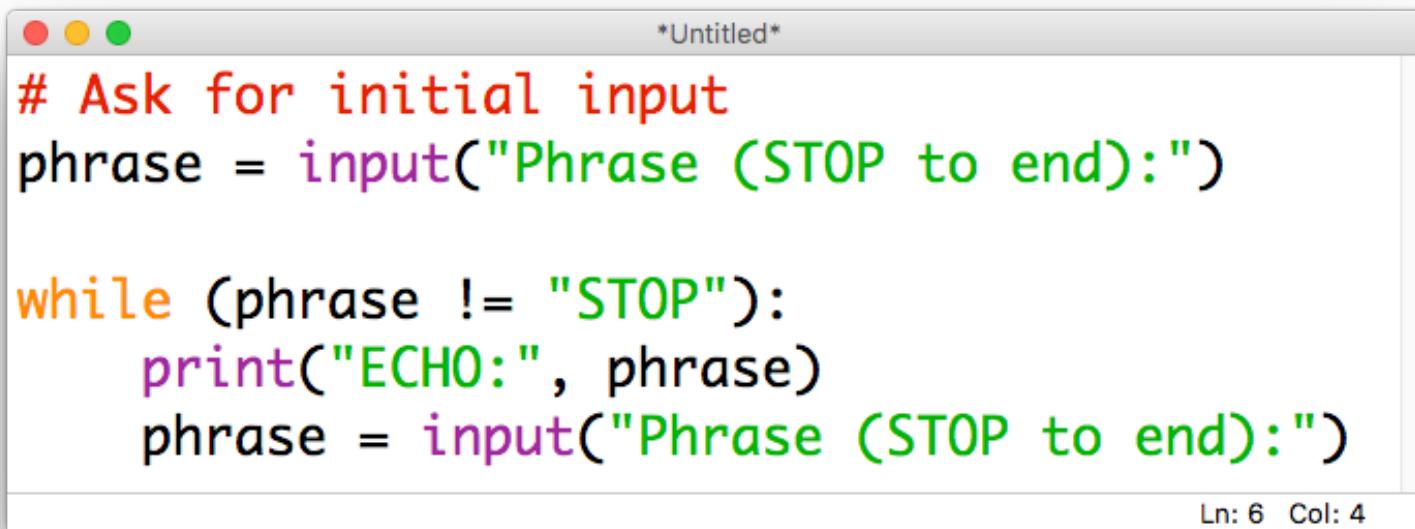
The image shows a screenshot of a Mac OS X terminal window. The window title is "Untitled". Inside the terminal, there is a single line of Python code:

```
x = 0
while (x < 10):
    x += 1
```

In the bottom right corner of the terminal window, there is a status bar displaying "Ln: 3 Col: 10".

while loops

- **while** loops can be especially useful when combined with the **input()** function
- For example, we may want to continue asking for input until the user tells us they are done:



```
# Ask for initial input
phrase = input("Phrase (STOP to end):")

while phrase != "STOP":
    print("ECHO:", phrase)
    phrase = input("Phrase (STOP to end):")
```

Ln: 6 Col: 4

Demo: compute a sum pt.2

Modify our previous demo program to
use a **while loop** to compute
the **sum** of a series of numbers
entered by the user

(continue until the user enters a blank)

Checking in

What's **one thing** you learned
in today's class?



Assignment #3: Magic 8 Ball



Assignment #3: Magic 8 Ball

- In this assignment, you will write a python program that **simulates a Magic 8 Ball**
- Your program should:
 - Take in a question from the user
 - Print a randomly-selected response
 - Determine if the user wants to ask another question
 - If so, repeat!
- Submit your repl on Moodle by **11:55pm on Thursday**

Up next

- ✓ Monday: Loops
 - ✓ `for...in` (looping through items in a list)
 - ✓ the `range()` function (getting a list of numbers)
 - ✓ `while` (looping until something happens)
- Lab: Loops
- Wednesday: Life Skill #2/3: Debugging & Documentation
- A3 due Thursday at 11:55pm
- Friday: Dictionaries and Sets