Objectives

- Review
- Lab 2
  - Programming practice

Feedback on Lab 1

- Overall good
- Notes
  - Saved output from each program
    - With user input, try several different good test cases
  - Want *good* output
    - think about what the user wants to see
  - High-level comments
    - Describes what the program does
      - Helps for quick overview when reviewing
  - Electronic submission
    - In directory – looked good!

Review

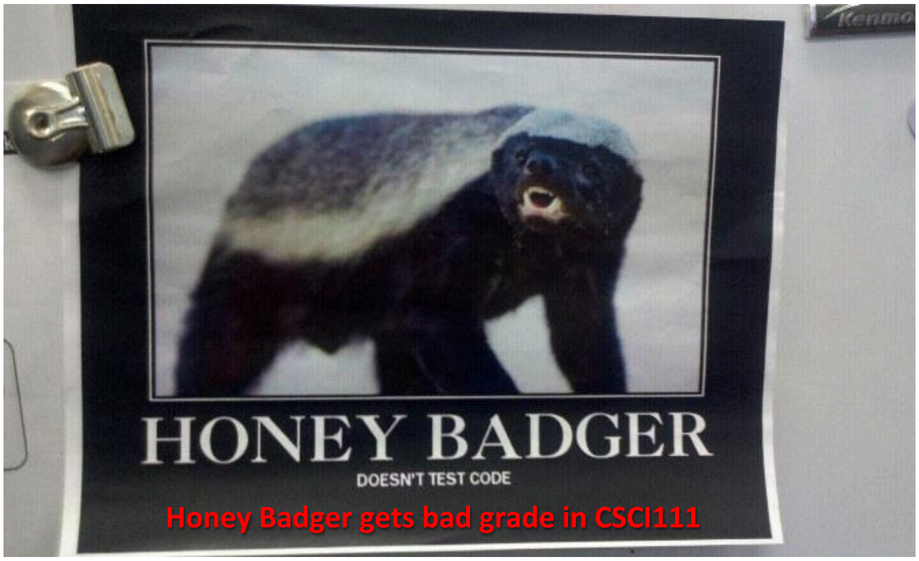
- What program do we use to develop programs?
  - What is the command you execute to start it?
- What is our process for developing programs?
- How can we make our program interactive with a user?

You can install Python/IDLE on your own computers
to practice between labs.

IDLE Review

- Run using `idle3 &`

Testing



Honey Badger gets bad grade in CSCI111

Recommendation

- Get user input last – this is a fairly routine step
- Develop/test without getting input first
  - Speeds up process
- Then, add user input

Review

- What are the two types of division?
- How can we find the remainder of a division?
- How can we make something repeat a certain number of times?

Review: Arithmetic Operations

Symbol	Meaning	Associativity
+	Addition	Left
–	Subtraction	Left
*	Multiplication	Left
/	Division	Left
%	Remainder (“mod”)	Left
**	Exponentiation (power)	Right

Precedence rules: P E - DM% AS

negation

Associativity matters when you have the same operation multiple times

Review: Two Division Operators

/ Float Division

- Result is a `float`
- Examples:
  - ➢ `6/3 → 2.0`
  - ➢ `10/3 → 3.3333333333333335`
  - ➢ `3.0/6.0 → 0.5`
  - ➢ `10/9 → 1.9`

// Integer Division

- Result is an `int`
- Examples:
  - ➢ `6//3 → 2`
  - ➢ `10//3 → 3`
  - ➢ `3.0//6.0 → 0`
  - ➢ `10//9 → 1`

Review: Formalizing Process of Developing Computational Solutions

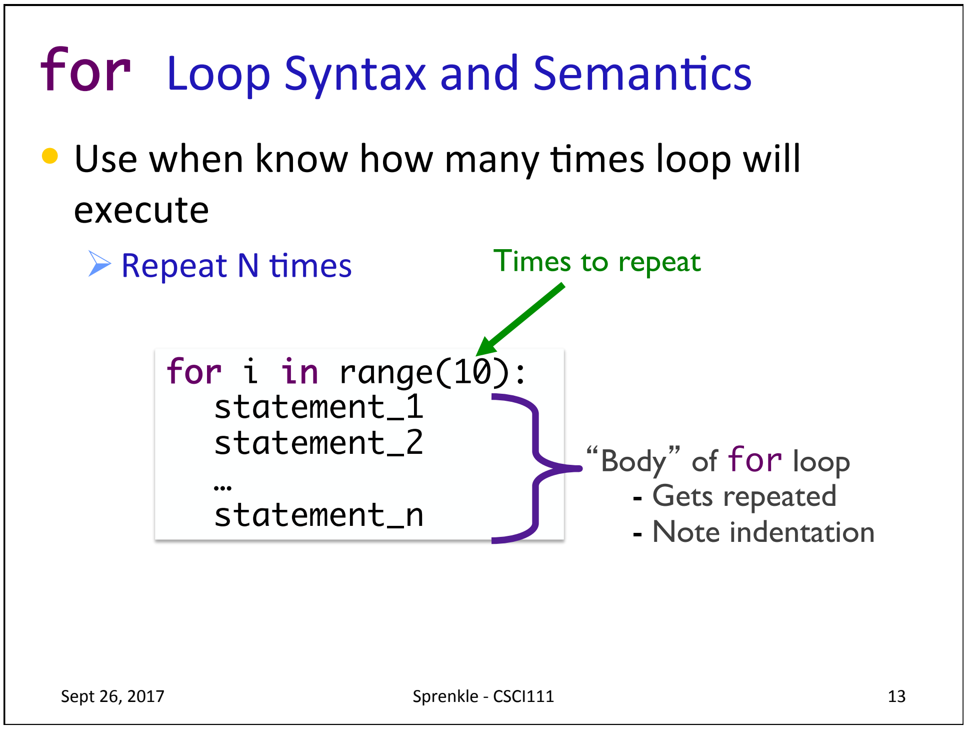
1. Think about the test cases
   a. Input, expected output
2. Create a sketch of how to solve the problem (the algorithm)
3. Fill in the details in Python
4. Test the Python program with *good* test cases
   a. If errors found, **debug** program
   b. Repeat step 3

Good Development Practices

- Design the algorithm
  - Break into pieces
- **Implement** *and* **Test** each piece *separately*
  - Identify the best pieces to make progress
  - Iterate over each step to improve it
- Write comments FIRST for each step
  - Elaborate on what you're doing in comments when necessary

Review

- How do we repeat code?



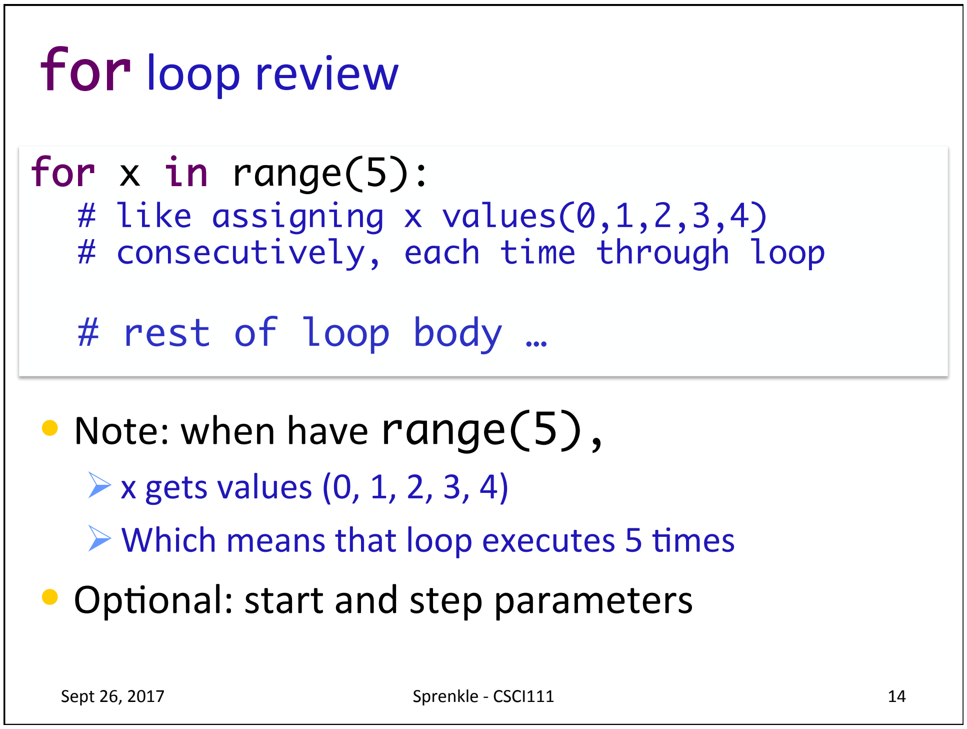
for Loop Syntax and Semantics

- Use when know how many times loop will execute
  - Repeat N times

Times to repeat

```
for i in range(10):
    statement_1
    statement_2
    …
    statement_n
```

“Body” of for loop
- Gets repeated
- Note indentation



for loop review

```
for x in range(5):
   # like assigning x values(0,1,2,3,4)
   # consecutively, each time through loop

   # rest of loop body …
```

- Note: when have `range(5)`,
  - x gets values (0, 1, 2, 3, 4)
  - Which means that loop executes 5 times
- Optional: start and step parameters

Lab Overview

- Arithmetic problems
- Loop problem
- Graphics API Problems
  - Update web page