

## 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!

## Lessons from Lab

- Look at examples!
  - “I was able to do this in that other program. How did I do that?”
  - On the course schedule page
- Explore!
  - Try things out in interactive mode
  - Then, put the ones that work into a script/program
- Testing!
  - Start with smaller and easy-to-verify tests
  - Test a variety of inputs
- Follow all of the directions!

## Review

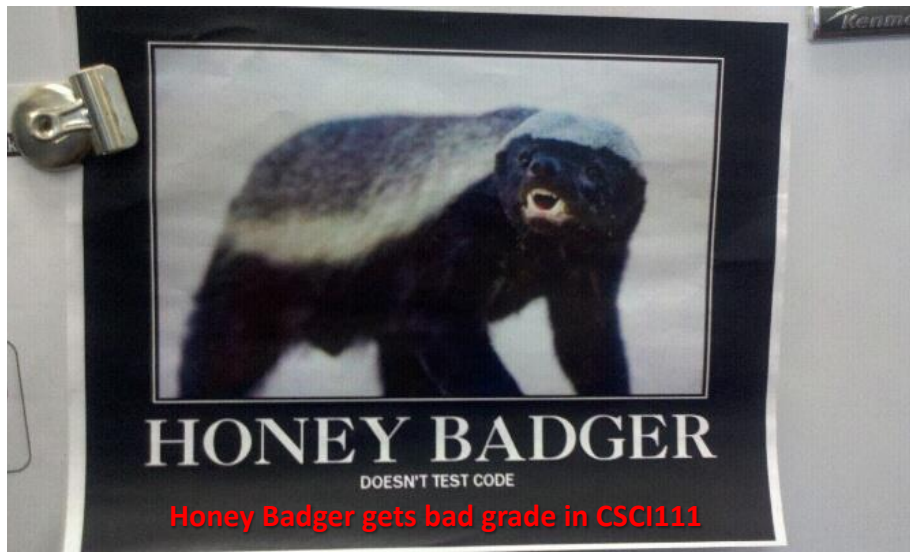
- What program do we use to develop programs?
  - What is the command you execute to start it?
- What is our process for developing programs?
- How can we make our program interactive with a user?

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

## IDLE Review

- Run using `idle3` &

## Testing



## 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: 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

- What are the two types of division?
- How can we find the remainder of a division?

## 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:
  - $6/3 \rightarrow 2.0$
  - $10/3 \rightarrow 3.3333333333333335$
  - $3.0/6.0 \rightarrow 0.5$
  - $10/9 \rightarrow 1.9$

### // Integer Division

- Result is an **int**
- Examples:
  - $6//3 \rightarrow 2$
  - $10//3 \rightarrow 3$
  - $3.0//6.0 \rightarrow 0$
  - $10//9 \rightarrow 1$

## Review: Object-Oriented Programming

- How do we create a new object?
- What is the term for how we give commands to/do operations on objects?
- What is the syntax for calling a method on an object?
- What are two types of methods we talked about?
  - How do they work differently?

## Review

- How do we get access to the code in `graphics.py` in our code?
- What is our typical process for drawing an object?
  - Pattern recognition: We've done this several times now – what is the pattern?
- How can we make a duplicate of a drawable object using the Graphics API?
- How can we find out what we can do to an object?

## Moving a Circle According to the User

- Draw a circle in the upper left-hand corner of the screen
- Tell the user to click somewhere
- Move the circle to where the user clicked

## Lab Overview

- Arithmetic problems
- Graphics API Problems
  - [Update web page](#)