Lab 3

- Review
  - Lab 2
  - Loops
  - Functions

Lab 2 Feedback

- Getting a little tougher in grading
  - Paying more attention to style (e.g., variable names), efficiency, readability, good output
  - High-level descriptions
  - More strict on adhering to problem specification
  - Demonstrate program **more than once** if gets input from user or outcome changes when run again
    - Find errors before I do!

Testing Discussion

- Consider what inputs could allow you to see different behaviors
  - Example: If only one person splitting the bill
  - What are good test cases for the greatest hits problem?
- Start with at least one test case that is easy to validate

Starting to Know Multiple Ways to Do Same Thing

- Favor the solution with least "conceptual complexity"
  - Approximation: requires fewer characters in a line of code

```
print("The tip is ", total_bill*(percent_tip/100), " dollars")
print("The total cost is ", total_bill +
(total_bill*(percent_tip/100)), " dollars")
print("The total cost per person is ", (total_bill+
(total_bill*(percent_tip/100)))/number_of_people, " dollars")
```

You should be able to understand this code, relatively easily, but it takes time to parse it and know what is happening.

Starting to Know Multiple Ways to Do Same Thing

- Favor the solution with least "conceptual complexity"
  - Approximation: requires fewer characters in a line of code

```
print("The tip is ", total_bill*(percent_tip/100), " dollars")
print("The total cost is ", total_bill +
(total_bill*(percent_tip/100)), " dollars")
print("The total cost per person is ", (total_bill+
(total_bill*(percent_tip/100)))/number_of_people, " dollars")
```

```
cost_tip=total_bill*(percent_tip/100)
print("The tip is", cost_tip, "dollars")

cost_total=total_bill+cost_tip
print("The total cost is", cost_total, "dollars")

cost_per_person=cost_total/number_people
print("The cost per person is", cost_per_person, "dollars")
```

More lines of code but each line is simpler

Text's setText("text") method

- Instead of creating multiple Text objects, just use setText mutator method.
- For example:

```
text = Text( anchorPoint, "original directions")
…
text.setText("new directions")
```

Variable Naming

- Consider which variable name is better:

```
circle = Circle(midPoint, 50)

bodyBottom = Circle(midPoint, 50)
```

Debugging Practices

- Larger, more complex programs → harder to debug
- Debugging practices
  - Trace through the program as if you are the computer
    - Similar to some exam problems
  - Use print statements to display variables' values
  - Or, use Python visualizer to show how variables' values change

Repeating Code

- How do we make code repeat?
- How do we use the range function?
- What questions should we ask when solving a problem that requires repetition?
  - These questions help guide our solution
- What is the ***accumulator design pattern***?
- How do we indicate that a variable will not change during the lifetime of the program?

Review: Accumulator Design Pattern

1. Initialize accumulator variable
2. Loop until done
   - Update the value of the accumulator
3. Display result

Recall our example of adding up the user inputs…

Review: Designing for Change: Constants

- Special variables whose values are defined once and never changed
  - By convention, not enforced by interpreter
- By convention
  - A constant's name is all caps
  - Typically defined at top of program → easy to find, change
- Examples:
  - `NUMBER_OF_INPUTS = 5`

Review

- How do we call functions?
- What are some examples of built-in functions?
- How can we access functions from a module?

Review: More Examples of Built-in Functions

Function Signature	Description
`round(x[,n])`	Return the `float` **x** rounded to **n** digits after the decimal point If no **n**, round to nearest `int`
`abs(x)`	Returns the absolute value of **x**
`type(x)`	Return the type of **x**
`pow(x, y)`	Returns x^y

Interpreter

Review: Animation

- How do we animate our graphics objects?

Problem: Animate Moving to User Click

- Use combinations of the method `move` and the function `sleep`
  - Need to **sleep** so that humans can see the graphics moving
  - Computer would process the **move**s too fast!
- `sleep` is part of the `time` module
  - Takes a `float` parameter representing *seconds* and pauses for that amount of time

`circleShiftAnim.py`

Computational Thinking

- Learning how to think
  - Learning how to learn
  - Learning how to solve problems
- Process
  - Practice!
    - Review slides and examples after class
      - Run them in Python visualizer
  - Finding answers
    - Examples, handouts, textbook, directions, links in directions, previous labs, ...
  - Asking questions
    - We talk you through the process

Drilling good practice early on with smaller problems so that you are well-poised to handle bigger problems!

Lab 3 Overview

- Practice Python programming
  - Loops
  - Constants
  - Functions
  - Animation with Graphics API