

Lab 3

- Review
 - Lab 2
 - Loops
 - Functions

1

Lab 2 Feedback

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

2

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")
```

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

More lines of code but
each line is simpler

```
totalCost = total_bill+tip
print("The total cost is", totalCost, "dollars")
```

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

Jan 31, 2023

5

5

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")
```

```
tip = total_bill*(percent_tip/100)
totalCost = total_bill+tip
cost_per_person = totalCost/number_people
```

Even better because it
groups computation
and printing together

```
print("The tip is", tip, "dollars")
print("The total cost is", totalCost, "dollars")
print("The cost per person is", cost_per_person, "dollars")
```

Jan 31, 2023

Sprenkle - CSC1111

6

6

Variable Naming

- Consider which variable name is better:

```
circle = Circle(midPoint, 50)
```

```
bodyBottom = Circle(midPoint, 50)
```

Coloring an Object

- Consider which statement is more easily understood:

```
circle.setFill("black")
```

VS

```
circle.setFill(rgb_color(0, 0, 0))
```

Takeaways

- Use variable names that are descriptive
 - Code is closer to English, more easily understood
- Keep code “simple” → shorter lines of code, color names

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

Review

Get ready for the day:
Retrieve handouts, Go to slides/examples

- 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?

Jan 31, 2023

Sprenkle - CSC1111

11

11

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...

Jan 31, 2023

Sprenkle - CSC1111

12

12

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

- What are some examples of built-in functions?
- How can we access functions from a module?
- How do we call functions?
 - Built-in functions?
 - Functions from modules?
- What functionality does the `random` module provide?
 - Be specific
- How do we animate?
 - How can we make the animation move faster? Slower?

Review: More Examples of Built-in Functions

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

Animation

- Use combinations of the method `move` and the function `sleep`
 - Need to `sleep` so that humans can see the graphics moving
 - Otherwise, computer processes the `moves` too fast!
- `sleep` is part of the `time` module
 - takes a float representing *seconds* and pauses for that amount of time

Animate Circle Shift Reflection

- Broke the problem down
 1. Move a circle to the position clicked by the user
 2. Animate movement
 - Break the movement into chunks
 - Repeatedly, move one chunk, sleep

Course Objective: Learn to break down problems

Jan 31, 2023

Sprenkle - CSCI111

circleShiftAnim.py

17

17

Some **random** Functions

- `random()`
 - Returns the next random floating point number in the range `[0.0, 1.0)`
- `randint(a, b)`
 - Return a random integer `N` such that $a \leq N \leq b$

```
import random

#random.seed(1)      # module.function()

for x in range(10):
    print(random.random())
```

random_test.py

Jan 31, 2023

18

18

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

Animation Inspiration

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