CSCI 220L

Lab 10

Learning Objectives

- Write conditional logic.

Activities

Create a file `lab10.py` containing the following functions. Include a short docstring in each function.

1. FizzBuzz

   FizzBuzz is a classic programming interview question that tests whether an applicant has a basic knowledge of loops and conditionals. A FizzBuzz program prints the numbers 1 to 100, but every multiple of 3 is replaced with the word “Fizz” and every multiple of 5 is replaced with the word “Buzz”. Any number that is a multiple of both 3 and 5 should be replaced by “FizzBuzz”.

   Write a void function `fizzbuzz()` that performs the FizzBuzz program as described above. However, it should ask the user to enter a number between 20 and 100 as the maximum number. If the user enters a number below 20, display the message “Too low!” instead. If the number is above 100, the message should be “Too high!”.

2. Your Own Testing Framework

   The test cases you wrote last week were a step in the right direction, but they are still less than ideal since you have to scan through all the console output to find incorrect results. What if your tests only displayed output for incorrect results?

   (a) Write a void function `test(name, result, expected)` that checks if *result* is equal to *expected*, and if not displays a failure message containing the test name, the expected result, and the actual result. For example, you might write a test case like so:

   ```
   test("Mean of 3 ints", mean([5, 10, 15]), 10)
   ```

   If this test fails because your function returned 11 instead of 10, you might get a message like:

   ```
   Failed: Mean of 3 ints (expected 10, got 11)
   ```

   (b) Write a void function `run_tests()` that will contain your test cases for the remaining exercises. For example:

   ```
   def run_tests():
       """Runs all of this lab's test cases."""
       print("Beginning tests...")

       # Add your test cases here:
       test("Mean of 3 ints", mean([5, 10, 15]), 10)

       print("Testing complete!")
   ```

   This function will start out nearly empty (since you don’t have any tests to run yet), but you will add to it in the following problem.

3. Logic Practice

   Complete the following functions. For each one, add at least two test cases to `run_tests()`.

   (a) `odd(x)`, which returns true for odd numbers and false for even numbers.

   (b) `eligible(weight, wins)`, which returns true if a player with the given weight and win count is eligible as a starter on the wrestling team. Players are eligible if either (1) they have at least 20 wins, (2) they weigh at least 200 lbs., or (3) they weigh between 150 and 160 lbs. and have at least 5 wins.

   (c) `letter(score)`, which returns the letter grade for a given score (ignoring + and –). The grading scale is A (90+), B (80–89), C (70–79), D (60–69), F (59 or less).

   (d) `digit_pair(x, y, z)`, which returns true if at least two of the arguments have the same final digit. You may assume the arguments are whole numbers.

4. Overlapping Circles

   Write a void function `circles()` that creates a GUI with the following features:

   (a) It allows the user to draw 5 circles from mouse clicks (one click for the circle's center and another on its circumference to get the circle's radius).

   (b) Each circle's color is determined by its size. A circle with a radius below 30 is red. Any larger circle is green.

   (c) A message is printed on the console for each pair of overlapping circles, such as "Circle 1 overlaps with Circle 2". You can display the message as soon a circle is placed or wait until the end. Make sure not to display any duplicate overlaps (e.g. "1 overlaps with 2" and "2 overlaps with 1" should not both appear).

   Below are some suggested helper functions. If any of your functions start to get too long, consider breaking it up into smaller ones to keep your program maintainable and modular. You might also consider adding test cases for additional functions you write.

   - `distance(p1, p2)`, which returns the distance between two points using the distance formula.
   - `color_circle(circle)`, which colors the circle according to the description above.
   - `overlap(cir1, cir2)`, which returns true if the two circles overlap, or false otherwise. How can you tell if two circles overlap? Try sketching out a diagram and think about the connection between their radii and the distance between their center points.

Upload `lab10.py` to the OAKS dropbox before the deadline. Make sure you have most of the exercises completed before your lab meeting.