Code Clinic Tips

This page include helpful tips based on our experience in helping students in the "Code Clinic" (*interactivepython@online.rice.edu*). Be sure and take a look at these tips if you get stuck.

Tip #1

Make sure that `format()`**returns** (not prints) the formatted time as a string. Also, be sure to call format with the global timer counter in your draw handler to output the formatted time on the canvas. Remember: return a string in the definition of format, call format correctly in the draw handler.

We have also prepared a simple testing template for `format()` that you can use to make sure that your implementation of `format()` works correctly. Just cut and paste your implementation of `format()` into the template and run.

http://www.codeskulptor.org/#examples-format_template.py

We have built an experimental automated unit testing application for testing your `format` function. Simply click on the link below, enter a CodeSkulptor URL for your `format` code, and then submit to Owltest. Please be patient since the app takes a few seconds to load and run.

http://codeskulptor.appspot.com/owltest/?urlTests=iipp_f13.stopwatch_tests.py&urlPylintConfig=skip

Tip #2

Excellent student post (by Amir Elnemr) from previous session on computing digits in `format`

Here is the format function required logic and operations for anyone who is struggling with it. I tried my best to explain everything so I apologize if some parts seem redundant or too simple for you.

The problem: Given a number of tenths of seconds format a string output in the format A:BC.D where: A = the amount of minutes in that number B = the amount of tens of seconds C = the amount of seconds in excess of tens of seconds D = the amount of the remaining tenths of seconds

If a number doesn't have a value a value of 0 should be printed out as a place holder. Explanation: 9 seconds should be printed out as 09 seconds where 0 is represented by B and 9 is represented by C and if the amount of time is less than 1 min a 0 should be displayed in the minutes place represented by A

The solution: A simple formula using integer division and/or modulo operations can be used for each number (A to D) to determine its value. Recall that integer division drops any remainders in the quotient and modulo gives back the remainder of the division. in python the integer division operator is // and the modulo operator is % Examples: 5 // 2 = 2 (while 5 / 2 = 2.5 integer division drops the extra .5) 5 % 2 = 1 (because 5 / 2 = 2 with a remainder of 1)

Using this knowledge here is how you can get the value of each number (A to D): A (the amount of minutes) the given number is in tenth of seconds so

1. We need to divide it by 10 to be in seconds and
2. Divide that by 60 to be in minutes. (1 and 2 combined- divide by 600) So we can get A by dividing the tenths of seconds by 600 and dropping the remainder (using integer division)

B (the amount of tens of seconds)

1. We need to get the amount of whole seconds first and drop the remainder tenths of seconds with an easy integer division by 10
2. The number we have now can be split into two parts; The amount of minutes and the amount of seconds that are less than one minute. We can divide that number of seconds by 60 to give us the number of minutes but what we're really concerned with is the second part; the remainder of that division so we can use a modulo operation (the amount of seconds % 60) to get that remainder
3. Now that we have the amount of seconds how can we get the tens of seconds in that number? let's say the number is 34 we can simple divide it by 10 to get 3 and drop the remainder 4 with an integer division by 10

C (the amount of seconds in excess of tens of seconds) The same as B except in step 3 using the same 34 seconds example we don't need the 3 this time; instead we need the 4 We can simply accomplish this by using a modulo operation instead of the integer division by 10 to get the remainder (the number of seconds that are less than 10 seconds)

D (the amount of the remaining tenths of seconds) This is the simplest of all. The given amount of tenths of seconds can be split into two parts; the number of whole seconds and the number of tenths of seconds that are less than one second. using the same logic in B and C you can easily come up with the formula.

Useful tip: Integer division will yield 0 where normal division will yield a decimal point number less than 1

After all the values are determined what's left is to return them from the function as a string with the format of A:BC.D . so all you need is to return the concatenation of the string equivalent of those values and the pretty colon and point.

Tip #3

Important

As you build more and more complicated code, you should focus on building your code incrementally. By that, I mean: write a little bit of code, test/debug it, write some more code, test/debug it, etc. Writing 20-30 lines of code and THEN trying to debug it with lots of potential errors is a very bad idea. If you keep your program always in a close-to-working state, debugging it is MUCH easier. Again, I suggest following the mini-process development process.

Tip #4

Subtle bug that may arise if your code is poorly structured

We've had several people ask about a subtle bug where their code runs correctly the vast majority of the time, but occasionally awards a "success" on a time ending in `"1"`. I 've seen this error in the Code Clinic. It's a subtle one that points out the trickiness of event-driven programming. Here is what I saw and how to fix it:

- The draw handler calls `format` which, in addition to returning a string also assigns it to a global variable. In the error case, the string ends in "0".
- The timer handler ticks and increments the `counter` to end in 1.
- The stop button handler is called and stops the timer. However, instead of correctly checking the counter, it incorrectly checks the last character of the formatted string which is "0". Thus, the handler records an incorrect success.
- The draw handler then calls `format` with the `counter` ending in 1 and produces a formatted time ending in "1".

So, to fix this error, make your test for a success be something of the form : `(counter % 10) == 0`.