Feedback — Quiz 3b

Help

Thank you. Your submission for this quiz was received.

You submitted this quiz on **Tue 7 Oct 2014 12:22 PM WEST**. You got a score of **100.00** out of **100.00**.

Question 1

When the following code is executed, how many times is `timer_handler` called?

```
import simplegui

def timer_handler():
    …

timer = simplegui.create_timer(10, timer_handler)
timer.start()
```

The body of `timer_handler` isn't given, as it is irrelevant for this question. You may want to finish the code and run it before submitting your answer.

Your Answer		Score	Explanation
(•) Unlimited — It is called repeatedly until you stop the program.	✔	10.00	
( ) 0 — The code hasn't been written correctly.			
( ) 10			
( ) 1			
Total		10.00 / 10.00	

Question 2

You want a timer to create exactly 1000 events. Which of the following solutions are

possible?

Your Answer		Score	Explanation
☐ Specify the number of timer events when creating the timer.	✔	1.00	There is no such option.
☐ In the timer handler, have a local counter for the number of timer calls. In the timer handler, increment the counter. In the timer handler, check the count and possibly stop the timer.	✔	1.00	With a local counter, you'll forget the count between calls.
☐ Have a global counter for the number of timer calls. Outside the timer handler, increment the counter. Outside the timer handler, check the count and possibly stop the timer.	✔	1.00	You can't count the timer calls outside of the handler.
☑ Have a global counter for the number of timer calls. In the timer handler, increment the counter. In the timer handler, check the count and possibly stop the timer.	✔	7.00	
Total		10.00 / 10.00	

Question 3

How do you change the frequency of a running timer, either increasing or decreasing the frequency? E.g., in the code below, we want code at the question marks that changes the timer.

```
…
timer = simplegui.create_timer(1000, timer_handler)
timer.start()
…
???
```

Your Answer		Score	Explanation
○ Create and start the timer again. timer = simplegui.create_timer(300, timer_handler) timer.start()			
◉ You can't. But, you can stop this timer, and start a new one with a different frequency and same handler. timer.stop() timer = simplegui.create_timer(300, timer_handler) timer.start()	✔	10.00	That we use the same variable `timer` is irrelevant. This is a new timer.
○ Just run `create_timer`. It will change the timer. timer = simplegui.create_timer(300, timer_handler)			
○ Just use `set_timer_interval`. timer.set_timer_interval(300)			
Total		10.00 / 10.00	

Question 4

How many timers can you have running at once?

Your Answer		Score	Explanation
○ 1			
◉ Unlimited	✔	10.00	
○ 0			
Total		10.00 / 10.00	

Question 5

The function `time.time()` is used in Python to keep track of time. What unit of time is associated with the value returned by `time.time()`? Hint: Look in the documentation.

Your Answer		Score	Explanation
Hour			
Second	✔	10.00	
Milli-second			
Minute			
Total		10.00 / 10.00	

Question 6

Write a CodeSkulptor program that experiments with the function `time.time()`. (Remember to `import time`.) Determine what date and time corresponds to time zero. Enter the year of that date as a four digit number.

You entered:

```
1970
```

Your Answer		Score	Explanation
1970	✔	10.00	Jan. 1, 1970 GMT
Total		10.00 / 10.00	

Question 7

The Python code below uses a timer to execute the function `update()` 10 times,

computing a good approximation to a common mathematical function. Examine the code, and run it while varying the input value `n`.

What is the common name for what this computes?

```
# Mystery computation in Python
# Takes input n and computes output named result

import simplegui

# global state

result = 1
iteration = 0
max_iterations = 10

# helper functions

def init(start):
    """Initializes n."""
    global n
    n = start
    print "Input is", n

def get_next(current):
    """???  Part of mystery computation."""
    return 0.5 * (current + n / current)

# timer callback

def update():
    """???  Part of mystery computation."""
    global iteration, result
    iteration += 1
    # Stop iterating after max_iterations
    if iteration >= max_iterations:
        timer.stop()
        print "Output is", result
    else:
        result = get_next(result)

# register event handlers

timer = simplegui.create_timer(1, update)

# start program
init(13)
timer.start()
```

Your Answer		Score	Explanation
○ Cosine of *n*			
◉ Square root of *n*	✔	15.00	
○ Square — n^2			
○ Multiplication by 2 — $2n$			
○ Multiplicative inverse — $1/n$			
○ Logarithm base 2			
○ Exponentiation — 2^n			
Total		15.00 / 15.00	

Question Explanation

Such a computation is more typically written using loops, which we haven't introduced yet in this course. However, this example illustrates timers and handler/callback functions and one possible use for them.

Question 8

Given any initial natural number, consider the sequence of numbers generated by repeatedly following the rule:

- divide by two if the number is even or
- multiply by 3 and add 1 if the number is odd.

The Collatz conjecture states that this sequence always terminates at 1. For example, the sequence generated by 23 is:

23, 70, 35, 106, 53, 160, 80, 40, 20, 10, 5, 16, 8, 4, 2, 1

Write a Python program that takes a global variable `n` and uses a timer callback to repeatedly apply the rule above to `n`. Use the code from the previous question as a template. I suggest that your code prints out the sequence of numbers generated by this rule. Run this program for `n` = 217. What is the largest number in the sequence generated by this starting value?

To test your code, starting at `n` = 23 generates a sequence with a maximum value of 160.

You entered:

736

Your Answer		Score	Explanation
736	✔	15.00	
Total		15.00 / 15.00	

Question Explanation

Again, such a computation is more typically written using loops, but this exercise is illustrating the usage of timers and callback functions.

Question 9

CodeSkulptor runs your Python code by converting it into Javascript when you click the "Run" button and then executing this Javascript in your web browser. Open this **example** and run the provided code. If the SimpleGUI frame is spawned as a separate window, you should see an animation of an explosion in the canvas for this frame. If the SimpleGUI frame is spawned as a separate tab on top of the existing window containing the code (as happens in some browser configurations), the animation will "freeze" and a single static image is displayed. (If the SimpleGUI frame spawns as a separate window, you can also cause the animation to freeze by opening a new tab on top of the code window.)

As explained in the FAQ, what is the explanation for this behavior?

Your Answer		Score	Explanation
○ Modern browser don't support running Javascript in multiple windows simultaneously. This situation causes the animation to freeze.			
◉ To save resources, modern browsers only execute the Javascript associated with the topmost tab of a window. The animation freezes since the code tab and its associated Javascript is no longer the topmost tab.	✔	10.00	Correct. If your browser does happen to open a SimpleGUI frame as a new tab on top of the existing code tab, "pull" this tab off of the top of the current window to create a new separate window.

○ Javascript and Python are incompatible languages. As a result, the Python in one tab can't run at the same time as the Javascript in the SimpleGUI frame.	
Total	10.00 / 10.00