CMPT 103 – Lab #6

General Information

Python version and IDE:	Python 3.3 / Wing IDE 101
Allocated lab time:	2 hrs and 50 min
Due date:	At the end of the lab period
Lab weight:	3%

Topics

- ✓ Using graphics.py to develop a simple graphical user interface application

Submission

- ✓ **All code file (.py) should be submitted electronically** to your Lab Blackboard site.
- ✓ A portion of the total marks (20%) will be allocated for the programming style. For example, functions should be small; avoid writing duplicate code; names should be meaningful and descriptive; naming conventions should be followed consistently; code should be formatted properly; and comments should be accurate and well written.
- ✓ Comments are **required** for:
  - EACH program indicating the student name and program name.
  - EACH function indicating the function purpose, syntax (example usage of the function), parameters, and return value
  - Any block of code for which the purpose may be unclear (Note: you should always try to write clean code that can be understood easily without comments).

Assignment

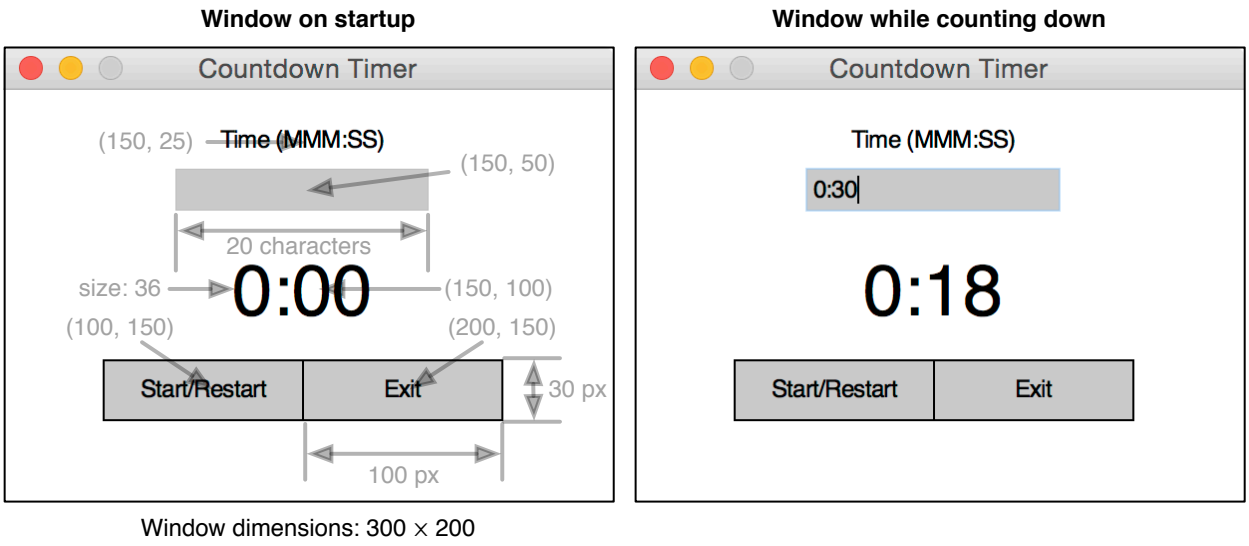
For this lab, please put all functions into a file called `Lab6your_initials.py` (e.g., `Lab6FL.py` where F and L are the first letter of your first name and last name). Please feel free to write helper functions if necessary.

In this lab, you will build a timer application. On startup, the application displays the window as illustrated in Figure 1 (left). Once the user enters a time and presses "Start/Restart", the countdown begins. After the countdown reaches 0:00, the window's background must flash three times. For each flash, the background goes black for 100 milliseconds and white for 100 milliseconds.

You must use the time module (https://docs.python.org/3/library/time.html) in your solution. To access its functions, add the following line to the start of your program:

```
import time
```

You will need to use `time.sleep(float)` and `float = time.time()`. Experiment with both functions in Python and understand them completely before you attempt your solution.



Window dimensions: 300 × 200

Figure 1: Countdown Timer Application

1. [10 marks] Create a function named `add_label` that creates, draws, and returns a `Text` object that can be used as a label within a window. By returning the `Text` object, the caller can later use the `setText(str)` method to update the label's text.

   Syntax: `label = add_label(window, x, y, text)`
   Parameter: `window – GraphWin object: the window`
   `x, y – int values: location for centre of label`
   `text – str object: text to use for label`
   Return value: `label – Text object used for the label`

2. [10 marks] Create a function named `add_button` that creates and draws a button. A button consists of a grey rectangle and a label, and the function must return the `Rectangle` object. The function should make all buttons the same size – according to the figure above.
   For full marks, your function must appropriately call `add_label`.

   Syntax: `button = add_button(window, x, y, text)`
   Parameter: `window – GraphWin object: the window`
   `x, y – int values: location for centre of button`
   `text – str object: text to use for button's label`
   Return value: `button – Rectangle object used for the button`

3. [0 marks] Copy and paste `is_clicked` (from `sample.py`) into your solution.

4. [10 marks] Create a function named `add_entry` that creates, draws, and returns an `Entry` object that can be used to obtain user input within a window. By returning the `Entry` object, the caller can later use `str = obj.getText()` to obtain the user input.

   Syntax: `entry = add_entry(win, x, y, width)`
   Parameter: `win – GraphWin object: the window`
   `x, y – int values: location for centre of entry`
   `width – int value: width (in characters) of object`
   Return value: `entry – Entry object`

5. [10 marks] Write a function named `flash` that accepts two arguments: an object and an integer. The function must "flash" the object the number of specified times. To flash an object, the function must set the background colour to black, delay 100 milliseconds, set the background colour to white, delay 100 milliseconds, and repeat for the specified number of times.

6. [20 marks; 10 marks each] Write a function named `convert_to_seconds` and another named `convert_to_clock`. These functions convert to/from the two representations for a time as shown below.

```
>>> convert_to_seconds("0:00")
0
>>> convert_to_seconds("0:30")
30
>>> convert_to_seconds("1:30")
90
>>> convert_to_seconds("123:45")
7425
>>> convert_to_seconds("fail?")
0
```

```
>>> convert_to_clock(0)
'0:00'
>>> convert_to_clock(30)
'0:30'
>>> convert_to_clock(90)
'1:30'
>>> convert_to_clock(7425)
'123:45'
```

   Hints:
   - a string can be split on the ':' character
   - the integer division operator (`//`) produces an integer result
   - the modulo operator (`%`) evaluates to the remainder
   - `try`/`except` may help with error handling
   - the string method `format` may help you create a string from integers, e.g., `"{},{:05d}".format(123,456)` produces `"123,00456"`; (experiment with this method in Python!)

7. [40 marks] Write a function named `main` that implements the countdown timer as described at the start of this assignment and accounts for the following additional details:
   - After the countdown reaches 0:00, flash the window three times, and then the countdown label may continue to display 0:00.
   - If the user changes the text in the entry box during a countdown, it must not impact the countdown.
   - If the user presses "Start/Restart" during a countdown, the countdown must restart using the time (currently) specified in the entry box. [15 marks]
   - The "Exit" button must close the window without crashing Python.