for the <Enter> key; historically, the <Enter> key is known as a return key. Also, I've structured the modal loop so that it looks very similar to our other key-processing examples. Since we are just waiting for `"Return"`, I could have simplified it to something like this:

```
while win.getKey() != "Return":
    pass
```

In this version, the body of this loop literally does nothing. The loop just keeps checking the condition until <Enter> is pressed and it becomes false. Then the do-nothing loop quits, allowing the program to continue. Either version gets the job done. The second is cleverer, but the first seems more obvious. It's probably best to go with obvious over clever.

The very last line of this function is necessary to ensure that the text entry is truly modal. Any mouse clicks that may have occurred before the <Enter> key was pressed should just be ignored. Since `checkMouse` only returns mouse clicks that have happened since the last call to `checkMouse`, calling the function here at the end has the effect of clearing any click that may have occurred but not yet been checked for.

That's it for this example. I would definitely suggest running and studying the final version of this program, `event_loop3.py`. A couple things you might try are commenting out the `checkMouse` at the end of `handleClick` and seeing if you can create a situation where the program acts weird as a result. Another good exercise would be a modification that allows the user to cancel text entry at any time by hitting the Escape key <Esc>. In effect, `"Escape"` becomes another sentinel for the modal loop, but when used, no `Text` object is created.

## 8.7 Chapter Summary

This chapter has filled in details of Python loops and Boolean expressions. Here are the highlights:

- A Python `for` loop is a definite loop that iterates through a sequence.
- A Python `while` statement is an example of an indefinite loop. It continues to iterate as long as the loop condition remains true. When using an indefinite loop, programmers must guard against the possibility of accidentally writing an infinite loop.

- One important use for an indefinite loop is for implementing the programming pattern interactive loop. An interactive loop allows portions of a program to be repeated according to the wishes of the user.

- A sentinel loop is a loop that handles input until a special value (the sentinel) is encountered. Sentinel loops are a common programming pattern. In writing a sentinel loop, a programmer must be careful that the sentinel is not processed.

- Loops are useful for reading files. Python treats a file as a sequence of lines, so it is particularly easy to process a file line by line using a `for` loop. In other languages, a file loop is generally implemented using a sentinel loop pattern.

- Loops, like other control structures, can be nested. When designing nested loop algorithms, it is best to consider the loops one at a time.

- Complex Boolean expressions can be built from simple conditions using the Boolean operators `and`, `or`, and `not`. Boolean operators obey the rules of Boolean algebra. DeMorgan's laws describe how to negate Boolean expressions involving `and` and `or`.

- Nonstandard loop structures such as a loop and a half can be built using a `while` loop having a loop condition of `True` and using a `break` statement to provide a loop exit.

- Python Boolean operators `and` and `or` employ short-circuit evaluation. They also have operational definitions that allow them to be used in certain decision contexts. Even though Python has a built-in `bool` data type, other data types (e.g., int) may also be used where Boolean expressions are expected.

- GUI programs are generally event driven and implement carefully designed event loops to control user interaction. Interactions are called nonmodal when the user is in control of what happens next and modal when the application dictates what the user must do next.

## 8.8 Exercises

### Review Questions

#### True/False

1. A Python `while` implements a definite loop.
2. The counted loop pattern uses a definite loop.
3. A sentinel loop asks the user whether to continue on each iteration.
4. A sentinel loop should not actually process the sentinel value.
5. The easiest way to iterate through the lines of a file in Python is to use a `while` loop.
6. A `while` is a post-test loop.
7. The Boolean operator `or` returns `True` when both of its operands are true.
8. `a and (b or c) == (a and b) or (a and c)`
9. `not(a or b) == (not a) or not(b)`
10. `True or False`

#### Multiple Choice

1. A loop pattern that asks the user whether to continue on each iteration is called a(n)
   a) interactive loop b) end-of-file loop
   c) sentinel loop d) infinite loop

2. A loop pattern that continues until a special value is input is called a(n)
   a) interactive loop b) end-of-file loop
   c) sentinel loop d) infinite loop

3. A loop structure that tests the loop condition after executing the loop body is called a
   a) pre-test loop b) loop and a half
   c) sentinel loop d) post-test loop

4. A priming read is part of the pattern for a(n)
   a) interactive loop b) end-of-file loop
   c) sentinel loop d) infinite loop

5. What statement can be executed in the body of a loop to cause it to terminate?
   a) `if` b) `input` c) `break` d) `exit`

6. Which of the following is not a valid rule of Boolean algebra?
   a) `(True or x) == True`
   b) `(False and x) == False`
   c) `not(a and b) == not(a) and not(b)`
   d) `(True or False) == True`

7. A loop that never terminates is called
   a) busy b) indefinite c) tight d) infinite

8. Which line would *not* be found in a truth table for `and`?
   a) T T T b) T F T c) F T F d) F F F

9. Which line would *not* be found in a truth table for `or`?
   a) T T T b) T F T c) F T F d) F F F

10. The term for an operator that may not evaluate one of its subexpressions is
    a) short-circuit b) faulty c) exclusive d) indefinite

## Discussion

1. Compare and contrast the following pairs of terms:
   a) definite loop vs. indefinite loop
   b) `for` loop vs. `while` loop
   c) interactive loop vs. sentinel loop
   d) sentinel loop vs. end-of-file loop

2. Give a truth table that shows the Boolean value of each of the following Boolean expressions, for every possible combination of "input" values. *Hint*: Including columns for intermediate expressions is helpful.
   a) `not (`$P$ `and` $Q$`)`

b) `(not` $P$`) and` $Q$

c) `(not` $P$`) or (not` $Q$`)`

d) `(`$P$ `and` $Q$`) or` $R$

e) `(`$P$ `or` $R$`) and (`$Q$ `or` $R$`)`

3. Write a `while` loop fragment that calculates the following values:

a) Sum of the first $n$ counting numbers: $1 + 2 + 3 + \ldots + n$

b) Sum of the first $n$ odd numbers: $1 + 3 + 5 + \ldots + 2n - 1$

c) Sum of a series of numbers entered by the user until the value 999 is entered. *Note*: 999 should not be part of the sum.

d) The number of times a whole number $n$ can be divided by 2 (using integer division) before reaching 1 (i.e., $log_2 n$).

## Programming Exercises

1. The Fibonacci sequence starts $1, 1, 2, 3, 5, 8, \ldots$. Each number in the sequence (after the first two) is the sum of the previous two. Write a program that computes and outputs the $n$th Fibonacci number, where $n$ is a value entered by the user.

2. The National Weather Service computes the windchill index using the following formula:

$$35.74 + 0.6215T - 35.75(V^{0.16}) + 0.4275T(V^{0.16})$$

Where $T$ is the temperature in degrees Fahrenheit, and $V$ is the wind speed in miles per hour.

Write a program that prints a nicely formatted table of windchill values. Rows should represent wind speed for 0 to 50 in 5-mph increments, and the columns represent temperatures from -20 to +60 in 10-degree increments. *Note*: The formula only applies for wind speeds in excess of 3 miles per hour.

3. Write a program that uses a `while` loop to determine how long it takes for an investment to double at a given interest rate. The input will be an annualized interest rate, and the output is the number of years it takes an investment to double. *Note*: The amount of the initial investment does not matter; you can use $1.

4. The Syracuse (also called "Collatz" or "Hailstone") sequence is generated by starting with a natural number and repeatedly applying the following function until reaching 1:

$$syr(x) = \begin{cases} x/2 & \text{if x is even} \\ 3x+1 & \text{if x is odd} \end{cases}$$

For example, the Syracuse sequence starting with 5 is: $5, 16, 8, 4, 2, 1$. It is an open question in mathematics whether this sequence will always go to 1 for every possible starting value.

Write a program that gets a starting value from the user and then prints the Syracuse sequence for that starting value.

5. A positive whole number $n > 2$ is prime if no number between 2 and $\sqrt{n}$ (inclusive) evenly divides $n$. Write a program that accepts a value of $n$ as input and determines if the value is prime. If $n$ is not prime, your program should quit as soon as it finds a value that evenly divides $n$.

6. Modify the previous program to find every prime number less than or equal to $n$.

7. The Goldbach conjecture asserts that every even number is the sum of two prime numbers. Write a program that gets a number from the user, checks to make sure that it is even, and then finds two prime numbers that add up to the number.

8. The greatest common divisor (GCD) of two values can be computed using Euclid's algorithm. Starting with the values $m$ and $n$, we repeatedly apply the formula: `n, m = m, n%m` until $m$ is 0. At that point, $n$ is the GCD of the original $m$ and $n$. Write a program that finds the GCD of two numbers using this algorithm.

9. Write a program that computes the fuel efficiency of a multi-leg journey. The program will first prompt for the starting odometer reading and then get information about a series of legs. For each leg, the user enters the current odometer reading and the amount of gas used (separated by a space). The user signals the end of the trip with a blank line. The program should print out the miles per gallon achieved on each leg and the total MPG for the trip.

10. Modify the previous program to get its input from a file.

11. Heating and cooling degree days are measures used by utility companies to estimate energy requirements. If the average temperature for a day is below 60, then the number of degrees below 60 is added to the heating degree days. If the temperature is above 80, the amount over 80 is added to the cooling degree days. Write a program that accepts a sequence of average daily temperatures and computes the running total of cooling and heating degree days. The program should print these two totals after all the data has been processed.

12. Modify the previous program to get its input from a file.

13. Write a program that graphically plots a regression line—that is, the line with the best fit through a collection of points. First ask the user to specify the data points by clicking on them in a graphics window. To find the end of input, place a small rectangle labeled "Done" in the lower-left corner of the window; the program will stop gathering points when the user clicks inside that rectangle.

    The regression line is the line with the following equation:

$$y = \bar{y} + m(x - \bar{x})$$

    where

$$m = \frac{\sum x_i y_i - n\bar{x}\bar{y}}{\sum x_i^2 - n\bar{x}^2}$$

    $\bar{x}$ is the mean of the $x$-values, $\bar{y}$ is the mean of the $y$-values, and $n$ is the number of points.

    As the user clicks on points, the program should draw them in the graphics window and keep track of the count of input values and the running sum of $x$, $y$, $x^2$, and $xy$ values. When the user clicks inside the "Done" rectangle, the program then computes the value of $y$ (using the equations above) corresponding to the $x$ values at the left and right edges of the window to compute the endpoints of the regression line spanning the window. After the line is drawn, the program will pause for another mouse click before closing the window and quitting.

14. Write a program that converts a color image to grayscale. The user supplies the name of a file containing a GIF or PPM image, and the program loads the image and displays the file. At the click of the mouse, the program converts the image to grayscale. The user is then prompted for a file name to store the grayscale image in.

You will probably want to go back and review the `Image` object from the graphics libarary (Section 4.8.4). The basic idea for converting the image is to go through it pixel by pixel and convert each one from color to an appropriate shade of gray. A gray pixel is created by setting its red, green, and blue components to have the same brightness. So `color_rgb(0,0,0)` is black, `color_rgb(255,255,255)` is white, and `color_rgb(127,127,127)` is a gray "halfway" between. You should use a weighted average of the original RGB values to determine the brightness of the gray. Here is the pseudocode for the grayscale algorithm:

```
for each row in the image:
    for each column in the image:
        r, g, b = get pixel information for current row and column
        brightness = int(round(0.299r + 0.587g + 0.114b))
        set pixel to color_rgb(brightness, brightness, brightness)
    update the image # to see progress row by row
```

*Note*: The pixel operations in the `Image` class are rather slow, so you will want to use relatively small images (*not* 12 megapixels) to test your program.

15. Write a program to convert an image to its color negative. The general form of the program will be similar to that of the previous problem. The negative of a pixel is formed by subtracting each color value from 255. So the new pixel color is `color_rgb(255-r, 255-g, 255-b)`.

16. Modify the `event_loop3` program to use the <Esc> key as described in the text. When the user types into an `Entry` box, hitting <Esc> should cause the `Entry` to disappear and discard whatever text may have been typed in the box.