## 2.8 Chapter Summary

This chapter has covered a lot of ground laying out both the process that is used to develop programs and the details of Python that are necessary to implement simple programs. Here is a quick summary of some of the key points:

- Writing programs requires a systematic approach to problem solving and involves the following steps:

  1. Problem Analysis: Studying the problem to be solved.
  2. Program Specification: Deciding exactly what the program will do.
  3. Design: Writing an algorithm in pseudocode.
  4. Implementation: Translating the design into a programming language.
  5. Testing/Debugging: Finding and fixing errors in the program.
  6. Maintenance: Keeping the program up to date with evolving needs.

- Many simple programs follow the input, process, output (IPO) pattern.

- Programs are composed of statements that are built from identifiers and expressions.

- Identifiers are names; they begin with an underscore or letter which can be followed by a combination of letter, digit, or underscore characters. Identifiers in Python are case-sensitive.

- Expressions are the fragments of a program that produce data. An expression can be composed of the following components:

  **literals** A literal is a representation of a specific value. For example, `3` is a literal representing the number three.

  **variables** A variable is an identifier that stores a value.

  **operators** Operators are used to combine expressions into more complex expressions. For example, in `x + 3 * y` the operators `+` and `*` are used.

- The Python operators for numbers include the usual arithmetic operations of addition (`+`), subtraction (`-`), multiplication (`*`), division (`/`), and exponentiation (`**`).

- The Python output statement `print` displays the values of a series of expressions to the screen.
- In Python, assignment of a value to a variable is indicated using the equal sign (=). Using assignment, programs can get input from the keyboard. Python also allows simultaneous assignment, which is useful for getting multiple input values with a single prompt.
- The `eval` function can be used to evaluate user input, but it is a security risk and should not be used with input from unknown or untrusted sources.
- Definite loops are loops that execute a known number of times. The Python `for` statement is a definite loop that iterates through a sequence of values. A Python list is often used in a `for` loop to provide a sequence of values for the loop.
- One important use of a `for` statement is in implementing a counted loop, which is a loop designed specifically for the purpose of repeating some portion of the program a specific number of times. A counted loop in Python is created by using the built-in `range` function to produce a suitably sized sequence of numbers.

## 2.9 Exercises

### Review Questions

#### True/False

1. The best way to write a program is to immediately type in some code and then debug it until it works.
2. An algorithm can be written without using a programming language.
3. Programs no longer require modification after they are written and debugged.
4. Python identifiers must start with a letter or underscore.
5. Keywords make good variable names.
6. Expressions are built from literals, variables, and operators.

7. In Python, `x = x + 1` is a legal statement.

8. Python does not allow the input of multiple values with a single statement.

9. A counted loop is designed to iterate a specific number of times.

10. In a flowchart, diamonds are used to show statement sequences, and rectangles are used for decision points.

## Multiple Choice

1. Which of the following is *not* a step in the software development process?
   a) specification b) testing/Debugging
   c) fee setting d) maintenance

2. What is the correct formula for converting Celsius to Fahrenheit?
   a) $F = 9/5(C) + 32$ b) $F = 5/9(C) - 32$
   c) $F = B^2 - 4AC$ d) $F = \frac{212-32}{100-0}$

3. The process of describing exactly *what* a computer program will do to solve a problem is called
   a) design b) implementation c) programming d) specification

4. Which of the following is *not* a legal identifier?
   a) `spam` b) `spAm` c) `2spam` d) `spam4U`

5. Which of the following are *not* used in expressions?
   a) variables b) statements c) operators d) literals

6. Fragments of code that produce or calculate new data values are called
   a) identifiers b) expressions
   c) productive clauses d) assignment statements

7. Which of the following is *not* a part of the IPO pattern?
   a) input b) program c) process d) output

8. The template `for <variable> in range(<expr>)` describes
   a) a general `for` loop b) an assignment statement
   c) a flowchart d) a counted loop

9. Which of the following is the most accurate model of assignment in Python?
   a) sticky-note b) variable-as-box
   c) simultaneous d) plastic-scale

10. In Python, getting user input is done with a special expression called
    a) `for` b) `read` c) simultaneous assignment d) `input`

## Discussion

1. List and describe in your own words the six steps in the software development process.

2. Write out the `chaos.py` program (Section 1.6) and identify the parts of the program as follows:

   - Circle each identifier.
   - Underline each expression.
   - Put a comment at the end of each line indicating the type of statement on that line (output, assignment, input, loop, etc.).

3. Explain the relationships among the concepts: definite loop, `for` loop, and counted loop.

4. Show the output from the following fragments:

   a)
   ```
   for i in range(5):
       print(i * i)
   ```

   b)
   ```
   for d in [3,1,4,1,5]:
       print(d, end=" ")
   ```

   c)
   ```
   for i in range(4):
       print("Hello")
   ```

   d)
   ```
   for i in range(5):
       print(i, 2**i)
   ```

5. Why is it a good idea to first write out an algorithm in pseudocode rather than jumping immediately to Python code?

6. The Python `print` function supports other keyword parameters besides `end`. One of these other keyword parameters is `sep`. What do you think the `sep` parameter does? *Hint*: `sep` is short for separator. Test your idea either by trying it interactively or by consulting the Python documentation.

7. What do you think will happen if the following code is executed?

```
print("start")
for i in range(0):
    print("Hello")
print("end")
```

Look at the flowchart for the `for` statement in this chapter to help you figure this out. Then test your prediction by trying out these lines in a program.

## Programming Exercises

1. A user-friendly program should print an introduction that tells the user what the program does. Modify the `convert.py` program (Section 2.2) to print an introduction.

2. On many systems with Python, it is possible to run a program by simply clicking (or double-clicking) on the icon of the program file. If you are able to run the `convert.py` program this way, you may discover another usability issue. The program starts running in a new window, but as soon as the program has finished, the window disappears so that you cannot read the results. Add an `input` statement at the end of the program so that it pauses to give the user a chance to read the results. Something like this should work:

   ```
   input("Press the <Enter> key to quit.")
   ```

3. Modify the `avg2.py` program (Section 2.5.3) to find the average of three exam scores.

4. Modify the `convert.py` program (Section 2.2) with a loop so that it executes 5 times before quitting. Each time through the loop, the program should get another temperature from the user and print the converted value.

5. Modify the `convert.py` program (Section 2.2) so that it computes and prints a table of Celsius temperatures and the Fahrenheit equivalents every 10 degrees from 0°C to 100°C.

6. Modify the `futval.py` program (Section 2.7) so that the number of years for the investment is also a user input. Make sure to change the final message to reflect the correct number of years.

7. Suppose you have an investment plan where you invest a certain fixed amount every year. Modify `futval.py` to compute the total accumulation of your investment. The inputs to the program will be the amount to invest each year, the interest rate, and the number of years for the investment.

8. As an alternative to APR, the interest accrued on an account is often described in terms of a nominal rate and the number of compounding periods. For example, if the interest rate is 3% and the interest is compounded quarterly, the account actually earns $\frac{3}{4}$% interest every 3 months.

   Modify the `futval.py` program to use this method of entering the interest rate. The program should prompt the user for the yearly rate (`rate`) and the number of times that the interest is compounded each year (`periods`). To compute the value in ten years, the program will loop `10 * periods` times and accrue `rate/period` interest on each iteration.

9. Write a program that converts temperatures from Fahrenheit to Celsius.

10. Write a program that converts distances measured in kilometers to miles. One kilometer is approximately 0.62 miles.

11. Write a program to perform a unit conversion of your own choosing. Make sure that the program prints an introduction that explains what it does.

12. Write an interactive Python calculator program. The program should allow the user to type a mathematical expression, and then print the value of the expression. Include a loop so that the user can perform many calculations (say, up to 100). *Note*: To quit early, the user can make the program crash by typing a bad expression or simply closing the window that the calculator program is running in. You'll learn better ways of terminating interactive programs in later chapters.