If you get an error message that seems to indicate that the Python interpreter is on the wrong track, it is a good idea to check for spelling and capitalization.

1.7 Problem Solving: Algorithm Design

You will soon learn how to program calculations and decision making in Python. But before we look at the mechanics of implementing computations in the next chapter, let's consider how you can describe the steps that are necessary for finding a solution to a problem.

You may have run across advertisements that encourage you to pay for a computerized service that matches you up with a romantic partner. Think how this might work. You fill out a form and send it in. Others do the same. The data are processed by a computer program. Is it reasonable to assume that the computer can perform the task of finding the best match for you?

Finding the perfect partner is not a problem that a computer can solve.

Suppose your younger brother, not the computer, had all the forms on his desk. What instructions could you give him? You can't say, "Find the best-looking person who likes inline skating and browsing the Internet". There is no objective standard for good looks, and your brother's opinion (or that of a computer program analyzing the digitized photo) will likely be different from yours. If you can't give written instructions for someone to solve the problem, there is no way the computer can magically find the right solution. The computer can only do what you tell it to do. It just does it faster, without getting bored or exhausted. For that reason, a computerized match-making service cannot guarantee to find the optimal match for you.

Contrast the problem of finding partners with the following problem:

> You put $10,000 into a bank account that earns 5 percent interest per year. How many years does it take for the account balance to be double the original?

Could you solve this problem by hand? Sure, you could. You figure out the balance as follows:

year	interest	balance
0		10000
1	10000.00 x 0.05 = 500.00	10000.00 + 500.00 = 10500.00
2	10500.00 x 0.05 = 525.00	10500.00 + 525.00 = 11025.00
3	11025.00 x 0.05 = 551.25	11025.00 + 551.25 = 11576.25
4	11576.25 x 0.05 = 578.81	11576.25 + 578.81 = 12155.06

You keep going until the balance is at least $20,000. Then the last number in the year column is the answer.

Of course, carrying out this computation is intensely boring to you or your younger brother. But computers are very good at carrying out repetitive calculations quickly and flawlessly. What is important to the computer is a description of the steps for finding the solution. Each step must be clear and unambiguous, requiring no guesswork. Here is such a description:

```
Start with a year value of 0, a column for the interest, and a balance of $10,000.
```

year	interest	balance
0		10000

```
Repeat the following steps while the balance is less than $20,000
    Add 1 to the year value.
    Compute the interest as balance x 0.05 (i.e., 5 percent interest).
    Add the interest to the balance.
```

year	interest	balance
0		10000
1	500.00	10500.00
14	942.82	19799.32
15	989.96	20789.28

```
Report the final year value as the answer.
```

Of course, these steps are not yet in a language that a computer can understand, but you will soon learn how to formulate them in Python. This informal description is called **pseudocode**.

Pseudocode is an informal description of a sequence of steps for solving a problem.

There are no strict requirements for pseudocode because it is read by human readers, not a computer program. Here are the kinds of pseudocode statements that we will use in this book:

- Use statements such as the following to describe how a value is set or changed:

```
total cost = purchase price + operating cost
Multiply the balance value by 1.05.
Remove the first and last character from the word.
```

- You can describe decisions and repetitions as follows:

```
If total cost 1 < total cost 2
While the balance is less than $20,000
For each picture in the sequence
```

Use indentation to indicate which statements should be selected or repeated:

```
For each car
    operating cost = 10 x annual fuel cost
    total cost = purchase price + operating cost
```

Here, the indentation indicates that both statements should be executed for each car.

- Indicate results with statements such as:

```
Choose car 1.
Report the final year value as the answer.
```

The exact wording is not important. What is important is that pseudocode describes a sequence of steps that is

- Unambiguous
- Executable
- Terminating

An algorithm for solving a problem is a sequence of steps that is unambiguous, executable, and terminating.

The step sequence is *unambiguous* when there are precise instructions for what to do at each step and where to go next. There is no room for guesswork or personal opinion. A step is *executable* when it can be carried out in practice. Had we said to use the actual interest rate that will be charged in years to come, and not a fixed rate of 5 percent per year, that step would not have been executable, because there is no way for anyone to know what that interest rate will be. A sequence of steps is *terminating* if it will eventually come to an end. In our example, it requires a bit of thought to see that the sequence will not go on forever: With every step, the balance goes up by at least $500, so eventually it must reach $20,000.

An algorithm is a recipe for finding a solution.

A sequence of steps that is unambiguous, executable, and terminating is called an **algorithm**. We have found an algorithm to solve our investment problem, and thus we can find the solution by programming a computer. The existence of an algorithm is an essential prerequisite for programming a task. You need to first discover and describe an algorithm for the task that you want to solve before you start programming (see Figure 8).

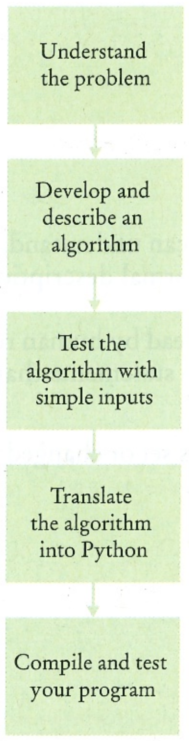

Understand the problem
Develop and describe an algorithm
Test the algorithm with simple inputs
Translate the algorithm into Python
Compile and test your program

Figure 8 The Software Development Process

SELF CHECK

20. Suppose the interest rate was 20 percent. How long would it take for the investment to double?

21. Suppose your cell phone carrier charges you $29.95 for up to 300 minutes of calls, and $0.45 for each additional minute, plus 12.5 percent taxes and fees. Give an algorithm to compute the monthly charge from a given number of minutes.

22. Consider the following pseudocode for finding the most attractive photo from a sequence of photos:

```
Pick the first photo and call it "the best so far".
For each photo in the sequence
    If it is more attractive than the "best so far"
        Discard "the best so far".
        Call this photo "the best so far".
The photo called "the best so far" is the most attractive photo in the sequence.
```

Is this an algorithm that will find the most attractive photo?

23. Suppose each photo in Self Check 22 had a price tag. Give an algorithm for finding the most expensive photo.

24. Suppose you have a random sequence of black and white marbles and want to rearrange it so that the black and white marbles are grouped together. Consider this algorithm:

```
Repeat until sorted
    Locate the first black marble that is preceded by a white marble, and switch them.
```

What does the algorithm do with the sequence ○●○●●? Spell out the steps until the algorithm stops.

25. Suppose you have a random sequence of colored marbles. Consider this pseudocode:

```
Repeat until sorted
    Locate the first marble that is preceded by a marble of a different color, and switch them.
```

Why is this not an algorithm?

Practice It Now you can try these exercises at the end of the chapter: R1.15, R1.17, P1.4.

HOW TO 1.1 Describing an Algorithm with Pseudocode

This is the first of many "How To" sections in this book that give you step-by-step procedures for carrying out important tasks in developing computer programs.

Before you are ready to write a program in Python, you need to develop an algorithm—a method for arriving at a solution for a particular problem. Describe the algorithm in pseudocode: a sequence of precise steps formulated in English.

Problem Statement You have the choice of buying two cars. One is more fuel efficient than the other, but also more expensive. You know the price and fuel efficiency (in miles per gallon, mpg) of both cars. You plan to keep the car for ten years. Assume a price of $4 per gallon of gas and usage of 15,000 miles per year. You will pay cash for the car and not worry about financing costs. Which car is the better deal?

Step 1 Determine the inputs and outputs.

In our sample problem, we have these inputs:

- `purchase price1` and `fuel efficiency1`, the price and fuel efficiency (in mpg) of the first car
- `purchase price2` and `fuel efficiency2`, the price and fuel efficiency of the second car

We simply want to know which car is the better buy. That is the desired output.

Step 2 Break down the problem into smaller tasks.

For each car, we need to know the total cost of driving it. Let's do this computation separately for each car. Once we have the total cost for each car, we can decide which car is the better deal.

The total cost for each car is `purchase price + operating cost`.

We assume a constant usage and gas price for ten years, so the operating cost depends on the cost of driving the car for one year.

The operating cost is `10 x annual fuel cost`.

The annual fuel cost is `price per gallon x annual fuel consumed`.

The annual fuel consumed is `annual miles driven / fuel efficiency`. For example, if you drive the car for 15,000 miles and the fuel efficiency is 15 miles/gallon, the car consumes 1,000 gallons.

Step 3 Describe each subtask in pseudocode.

In your description, arrange the steps so that any intermediate values are computed before they are needed in other computations. For example, list the step

```
total cost = purchase price + operating cost
```

after you have computed `operating cost`.

Here is the algorithm for deciding which car to buy:

```
For each car, compute the total cost as follows:
    annual fuel consumed = annual miles driven / fuel efficiency
    annual fuel cost = price per gallon x annual fuel consumed
    operating cost = 10 x annual fuel cost
    total cost = purchase price + operating cost
If total cost1 < total cost2
    Choose car1.
Else
    Choose car2.
```

Step 4 Test your pseudocode by working a problem.

We will use these sample values:

Car 1: $25,000, 50 miles/gallon
Car 2: $20,000, 30 miles/gallon

Here is the calculation for the cost of the first car:

```
annual fuel consumed = annual miles driven / fuel efficiency = 15000 / 50 = 300
annual fuel cost = price per gallon x annual fuel consumed = 4 x 300 = 1200
operating cost = 10 x annual fuel cost = 10 x 1200 = 12000
total cost = purchase price + operating cost = 25000 + 12000 = 37000
```

Similarly, the total cost for the second car is $40,000. Therefore, the output of the algorithm is to choose car 1.

WORKED EXAMPLE 1.1 Writing an Algorithm for Tiling a Floor

Problem Statement Make a plan for tiling a rectangular bathroom floor with alternating black and white tiles measuring 4 × 4 inches. The floor dimensions, measured in inches, are multiples of 4.

Step 1 Determine the inputs and outputs.

The inputs are the floor dimensions (length × width), measured in inches. The output is a tiled floor.

Step 2 Break down the problem into smaller tasks.

A natural subtask is to lay one row of tiles. If you can solve that task, then you can solve the problem by laying one row next to the other, starting from a wall, until you reach the opposite wall.

How do you lay a row? Start with a tile at one wall. If it is white, put a black one next to it. If it is black, put a white one next to it. Keep going until you reach the opposite wall. The row will contain `width / 4` tiles.

Step 3 Describe each subtask in pseudocode.

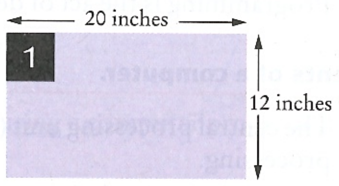
In the pseudocode, you want to be more precise about exactly where the tiles are placed.

```
Place a black tile in the northwest corner.
While the floor is not yet filled, repeat the following steps:
    Repeat this step width / 4 – 1 times:
        Place a tile east of the previously placed tile. If the previously placed tile was white, pick a black one;
            otherwise, a white one.
    Locate the tile at the beginning of the row that you just placed. If there is space to the south, place a tile of
    the opposite color below it.
```

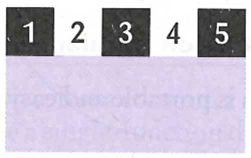
Step 4 Test your pseudocode by working a problem.

Suppose you want to tile an area measuring 20 × 12 inches.

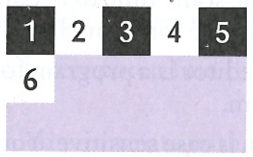
The first step is to place a black tile in the northwest corner.



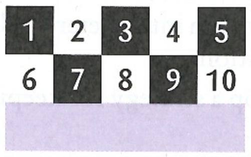
Next, alternate four tiles until reaching the east wall. (`width / 4 – 1 = 20 / 4 – 1 = 4`)



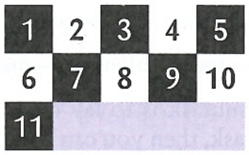
There is room to the south. Locate the tile at the beginning of the completed row. It is black. Place a white tile south of it.



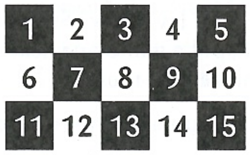
Complete the row.



There is still room to the south. Locate the tile at the beginning of the completed row. It is white. Place a black tile south of it.



Complete the row.



Now the entire floor is filled, and you are done.

CHAPTER SUMMARY

Define "computer program" and programming.

- Computers execute very basic instructions in rapid succession.
- A computer program is a sequence of instructions and decisions.
- Programming is the act of designing and implementing computer programs.

Describe the components of a computer.

- The central processing unit (CPU) performs program control and data processing.
- Storage devices include memory and secondary storage.

Describe the benefits of the Python language.

- Python is portable and easy to learn and use.

Become familiar with your Python programming environment.

- Set aside some time to become familiar with the programming environment that you will use for your class work.
- A text editor is a program for entering and modifying text, such as a Python program.
- Python is case sensitive. You must be careful about distinguishing between upper- and lowercase letters.
- The Python interpreter reads Python programs and executes the program instructions.
- Develop a strategy for keeping backup copies of your work before disaster strikes.

Describe the building blocks of a simple program.

- A comment provides information to the programmer.
- A function is a collection of instructions that perform a particular task.
- A function is called by specifying the function name and its arguments.
- A string is a sequence of characters enclosed in a pair of single or double quotation marks.

Classify program errors as compile-time and run-time errors.

- A compile-time error is a violation of the programming language rules that is detected when the code is translated into executable form.
- An exception occurs when an instruction is syntactically correct, but impossible to perform.
- A run-time error is any error that occurs when the program compiles and runs, but produces unexpected results.

Write pseudocode for simple algorithms.

- Pseudocode is an informal description of a sequence of steps for solving a problem.
- An algorithm for solving a problem is a sequence of steps that is unambiguous, executable, and terminating.

REVIEW QUESTIONS

■ **R1.1** Explain the difference between using a computer program and programming a computer.

■ **R1.2** Which parts of a computer can store program code? Which can store user data?

■ **R1.3** Which parts of a computer serve to give information to the user? Which parts take user input?

■■■ **R1.4** A toaster is a single-function device, but a computer can be programmed to carry out different tasks. Is your cell phone a single-function device, or is it a programmable computer? (Your answer will depend on your cell phone model.)

■ **R1.5** Which programming languages were mentioned in this chapter? When were they invented? By whom? (Look it up on the Internet.)

■■ **R1.6** On your own computer or on a lab computer, find the exact location (folder or directory name) of

a. The sample file `hello.py`, which you wrote with the editor.

b. The Python program launcher `python` or `python.exe`.

■■ **R1.7** What does this program print?

```
print("39 + 3")
print(39 + 3)
```