and operations. When the value gets larger, Python automatically converts to a representation using more bits. Of course, in order to perform operations on larger numbers, Python has to break down the operations into smaller units that the computer hardware is able to handle—similar to the way you might do long division by hand. These operations will not be as efficient (they require more steps), but they allow our Python ints to grow to arbitrary size. And that's what allows our simple factorial program to compute some whopping large results. This is a very cool feature of Python.

3.6 Chapter Summary

This chapter has filled in some important details concerning programs that do numerical computations. Here is a quick summary of some key concepts:

- The way a computer represents a particular kind of information is called a data type. The data type of an object determines what values it can have and what operations it supports.

- Python has several different data types for representing numeric values, including int and float.

- Whole numbers are generally represented using the int data type, and fractional values are represented using floats. All of the Python numeric data types support standard, built-in mathematical operations: addition (+), subtraction (-), multiplication (*), division (/), integer division (//), remainder (%), exponentiation (**), and absolute value (`abs(x)`).

- Python automatically converts numbers from one data type to another in certain situations. For example, in a mixed-type expression involving ints and floats, Python first converts the ints into floats and then uses float arithmetic.

- Programs may also explicitly convert one data type into another using the functions `float()`, `int()`, and `round()`. Type conversion functions should generally be used in place of `eval` for handling numeric user inputs.

- Additional mathematical functions are defined in the math library. To use these functions, a program must first import the library.

- Numerical results are often calculated by computing the sum or product of a sequence of values. The loop accumulator programming pattern is useful for this sort of calculation.
- Both ints and floats are represented on the underlying computer using a fixed-length sequence of bits. This imposes certain limits on these representations. Hardware ints must be in the range $-2^{31} \ldots (2^{31} - 1)$ on a 32-bit machine. Floats have a finite amount of precision and cannot represent most numbers exactly.
- Python's int data type may be used to store whole numbers of arbitrary size. Int values are automatically converted to longer representations when they become too large for the underlying hardware int. Calculations involving these long ints are less efficient than those that use only small ints.

3.7 Exercises

Review Questions

True/False

1. Information that is stored and manipulated by computers is called data.
2. Since floating-point numbers are extremely accurate, they should generally be used instead of ints.
3. Operations like addition and subtraction are defined in the math library.
4. The number of possible arrangements of n items is equal to $n!$.
5. The `sqrt` function computes the squirt of a number.
6. The float data type is identical to the mathematical concept of a real number.
7. Computers represent numbers using base-2 (binary) representations.
8. A hardware float can represent a larger range of values than a hardware int.
9. Type conversion functions such as `float` are a safe alternative to `eval` for getting a number as user input.

10. In Python, `4+5` produces the same result type as `4.0+5.0`.

Multiple Choice

1. Which of the following is *not* a built-in Python data type?
   a) int b) float c) rational d) string

2. Which of the following is *not* a built-in operation?
   a) `+` b) `%` c) `abs()` d) `sqrt()`

3. In order to use functions in the math library, a program must include
   a) a comment b) a loop c) an operator d) an import statement

4. The value of 4! is
   a) 9 b) 24 c) 41 d) 120

5. The most appropriate data type for storing the value of pi is
   a) int b) float c) irrational d) string

6. The number of distinct values that can be represented using 5 bits is
   a) 5 b) 10 c) 32 d) 50

7. In a mixed-type expression involving ints and floats, Python will convert
   a) floats to ints b) ints to strings
   c) both floats and ints to strings d) ints to floats

8. Which of the following is not a Python type-conversion function?
   a) `float` b) `round` c) `int` d) `abs`

9. The pattern used to compute factorials is
   a) accumulator b) input, process, output
   c) counted loop d) plaid

10. In modern Python, an int value that grows larger than the underlying hardware int
    a) causes an overflow b) converts to float
    c) breaks the computer d) uses more memory

Discussion

1. Show the result of evaluating each expression. Be sure that the value is in the proper form to indicate its type (int or float). If the expression is illegal, explain why.

a) `4.0 / 10.0 + 3.5 * 2`

b) `10 % 4 + 6 / 2`

b) `abs(4 - 20 // 3) ** 3`

d) `sqrt(4.5 - 5.0) + 7 * 3`

e) `3 * 10 // 3 + 10 % 3`

f) `3 ** 3`

2. Translate each of the following mathematical expressions into an equivalent Python expression. You may assume that the math library has been imported (via `import math`).

a) $(3+4)(5)$

b) $\frac{n(n-1)}{2}$

c) $4\pi r^2$

d) $\sqrt{r(\cos a)^2 + r(\sin b)^2}$

e) $\frac{y2-y1}{x2-x1}$

3. Show the sequence of numbers that would be generated by each of the following `range` expressions.

a) `range(5)`

b) `range(3, 10)`

c) `range(4, 13, 3)`

d) `range(15, 5, -2)`

e) `range(5, 3)`

4. Show the output that would be generated by each of the following program fragments.

a)
```
for i in range(1, 11):
    print(i*i)
```

b)
```
for i in [1,3,5,7,9]:
    print(i, ":", i**3)
print(i)
```

c)
```
x = 2
y = 10
for j in range(0, y, x):
    print(j, end="")
    print(x + y)
print("done")
```

d)
```
ans = 0
for i in range(1, 11):
    ans = ans + i*i
    print(i)
print (ans)
```

5. What do you think will happen if you use a negative number as the second parameter in the `round` function? For example, what should be the result of `round(314.159265, -1)`? Explain the rationale for your answer. After you've written your answer, consult the Python documentation or try out some examples to see what Python actually does in this case.

6. What do you think will happen when the operands to the integer division or remainder operations are negative? Consider each of the following cases and try to predict the result. Then try them out in Python. *Hint*: Recall the magic formula $a = (a//b)(b) + (a\%b)$.

   a) `-10 // 3`

   b) `-10 % 3`

   c) `10 // -3`

   d) `10 % -3`

   e) `-10 // -3`

Programming Exercises

1. Write a program to calculate the volume and surface area of a sphere from its radius, given as input. Here are some formulas that might be useful:

$$V = 4/3\pi r^3$$

$$A = 4\pi r^2$$

2. Write a program that calculates the cost per square inch of a circular pizza, given its diameter and price. The formula for area is $A = \pi r^2$.

3. Write a program that computes the molecular weight of a carbohydrate (in grams per mole) based on the number of hydrogen, carbon, and oxygen atoms in the molecule. The program should prompt the user to enter the number of hydrogen atoms, the number of carbon atoms, and the number of oxygen atoms. The program then prints the total combined molecular weight of all the atoms based on these individual atom weights:

Atom	Weight (grams / mole)
H	1.00794
C	12.0107
O	15.9994

   For example, the molecular weight of water (H_2O) is: $2(1.00794) + 15.9994 = 18.01528$.

4. Write a program that determines the distance to a lightning strike based on the time elapsed between the flash and the sound of thunder. The speed of sound is approximately 1100 ft/sec and 1 mile is 5280 ft.

5. The Konditorei coffee shop sells coffee at $10.50 a pound plus the cost of shipping. Each order ships for $0.86 per pound + $1.50 fixed cost for overhead. Write a program that calculates the cost of an order.

6. Two points in a plane are specified using the coordinates (x1,y1) and (x2,y2). Write a program that calculates the slope of a line through two (non-vertical) points entered by the user.

$$slope = \frac{y2 - y1}{x2 - x1}$$

7. Write a program that accepts two points (see previous problem) and determines the distance between them.

$$distance = \sqrt{(x2 - x1)^2 + (y2 - y1)^2}$$

8. The Gregorian epact is the number of days between January 1^{st} and the previous new moon. This value is used to figure out the date of Easter. It is calculated by these formulas (using int arithmetic):

$$C = year // 100$$

$$epact = (8 + (C//4) - C + ((8C + 13)//25) + 11(year\%19))\%30$$

Write a program that prompts the user for a 4-digit year and then outputs the value of the epact.

9. Write a program to calculate the area of a triangle given the length of its three sides—a, b, and c—using these formulas:

$$s = \frac{a+b+c}{2}$$

$$A = \sqrt{s(s-a)(s-b)(s-c)}$$

10. Write a program to determine the length of a ladder required to reach a given height when leaned against a house. The height and angle of the ladder are given as inputs. To compute length use:

$$length = \frac{height}{\sin angle}$$

Note: The angle must be in radians. Prompt for an angle in degrees and use this formula to convert:

$$radians = \frac{\pi}{180} degrees$$

11. Write a program to find the sum of the first n natural numbers, where the value of n is provided by the user.

12. Write a program to find the sum of the cubes of the first n natural numbers where the value of n is provided by the user.

13. Write a program to sum a series of numbers entered by the user. The program should first prompt the user for how many numbers are to be summed. The program should then prompt the user for each of the numbers in turn and print out a total sum after all the numbers have been entered. *Hint*: Use an input statement in the body of the loop.

14. Write a program that finds the average of a series of numbers entered by the user. As in the previous problem, the program will first ask the user how many numbers there are. *Note*: The average should always be a float, even if the user inputs are all ints.

15. Write a program that approximates the value of pi by summing the terms of this series: $4/1 - 4/3 + 4/5 - 4/7 + 4/9 - 4/11 + \ldots$ The program should prompt the user for n, the number of terms to sum, and then output the sum of the first n terms of this series. Have your program subtract the approximation from the value of `math.pi` to see how accurate it is.

16. A Fibonacci sequence is a sequence of numbers where each successive number is the sum of the previous two. The classic Fibonacci sequence begins: 1, 1, 2, 3, 5, 8, 13, ... Write a program that computes the nth Fibonacci number where n is a value input by the user. For example, if $n = 6$, then the result is 8.

17. You have seen that the math library contains a function that computes the square root of numbers. In this exercise, you are to write your own algorithm for computing square roots. One way to solve this problem is to use a guess-and-check approach. You first guess what the square root might be, and then see how close your guess is. You can use this information to make another guess and continue guessing until you have found the square root (or a close approximation to it). One particularly good way of making guesses is to use Newton's method. Suppose `x` is the number we want the root of, and `guess` is the current guessed answer. The guess can be improved by using computing the next guess as:

$$\frac{guess + \frac{x}{guess}}{2}$$

Write a program that implements Newton's method. The program should prompt the user for the value to find the square root of (`x`) and the number of times to improve the guess. Starting with a `guess` value of `x/2`, your program should loop the specified number of times applying Newton's method and report the final value of `guess`. You should also subtract your estimate from the value of `math.sqrt(x)` to show how close it is.