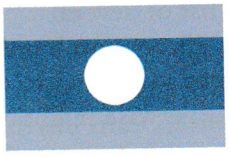
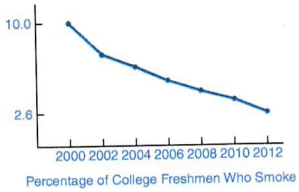
CHAPTER 6 KEY TERMS AND CONCEPTS	EXAMPLES
	[Run, possible outcome] **tan** **['tan', 'red']** **['blue', 'gray', 'red', 'tan']** **4**
6.3 Turtle Graphics **Turtle graphics** are drawn with a pen that can be thought of as being attached to the tail of a robotic turtle. The turtle responds to commands from the *turtle* module. The turtle can be instructed to raise or lower the pen, use a specified color, rotate in place, move to a designated point, move forward or backward for a specified distance, draw a dot, and display text. When the pen is lowered, the pen draws while the turtle moves. If the set of statements that draw an enclosed region are preceded by **t.begin_fill()** and followed by **t.end_fill()**, the inside of the region will have the color specified by a statement of the form **t.fillcolor(colorName)**. Section 6.3 defines functions that draw rectangles, lines, dots, stars, and text with specified locations, sizes, and colors. These functions simplify writing programs that draw flags and charts.	**import turtle** **t = turtle.Turtle()** **t.hideturtle()** **t.up() # raise the pen** **# move to (10,20) without drawing** **t.goto(10,20)** **# draw red dot of diameter 6 with** **# center at (10,20)** **t.dot(6, "red")** **t.down() # lower the pen** **t.pencolor("blue")** **# draw blue line from (10,20) to** **# (30,40)** **t.goto(30,40)** **# display hi to right of (30,40)** **t.write("hi")**   Percentage of College Freshmen Who Smoke
6.4 Recursion A **recursive function** is a function that calls itself, where successive calls reduce a computation to smaller computations of the same type until a **base case** with a trivial solution is reached.	**def factorial(n):** ** if n == 1: # base case** ** return 1** ** else:** ** return n * factorial(n - 1)**

CHAPTER 6 PROGRAMMING PROJECTS

1. **Guess My Number** Write a robust program that randomly selects a number from 1 through 100 and asks the user to guess the number. At each guess the user should be told if the guess is proper, and if so, whether it is too high or too low. The user should be told of the number of guesses when finally guessing the correct number. See Fig. 6.28.

```
I've thought of a number from 1 through 100.
Guess the number: 50
Too low
Try again: 123
Number must be from 1 through 100.
Try again: sixty
You did not enter a number.
Try again: 60
Too high
Try again: 56
Correct. You took 5 guesses.
```

FIGURE 6.28 Possible outcome of Programming Project 1.

2. Analyze a Poker Hand Write a program using the file **DeckOfCardsList.dat** that randomly selects and displays five cards from the deck of cards and determines which of the following seven categories describes the hand: four-of-a-kind, full house (three cards of one rank, two cards of another rank), three-of-a-kind, two pairs, one pair, or ranks-all-different. See Fig. 6.29. (***Hint:*** Determine the number of different ranks in the hand and analyze each of the four possible cases.)

```
K♥, K♦, 2♦, K♣, 5♠
three-of-a-kind
```

FIGURE 6.29 Possible outcome of Programming Project 2.

3. Analyze a Bridge Hand Write a program using the file **DeckOfCardsList.dat** that randomly selects and displays 13 cards from the deck of cards and gives the suit distribution. See Fig. 6.30.

```
10♥, 3♥, J♣, 2♣, 10♦, K♣, 2♥, 6♦, 6♣, 4♣, 7♦, 6♠, 4♦
Number of ♣ is 5
Number of ♦ is 4
Number of ♥ is 3
Number of ♠ is 1
```

FIGURE 6.30 Possible outcome of Programming Project 3.

4. American Flag The width (w) of the official American flag is 1.9 times the height (h). The blue rectangular canton (referred to as the "union") has width $\frac{2}{5}w$ and height $\frac{7}{13}h$. Write a program that draws an American flag. See Fig. 6.31. The colorful insert pages contain a picture of the flag with its true colors.

5. Permutations A reordering of the letters of a word is called a *permutation* of the word. A word of n different characters has $n!$ permutations where $n! = n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot 2 \cdot 1$. For instance, the word *python* has 6! or 720 permutations. Some of its permutations are *pythno*, *ypntoh*, *tonyhp*, and *ontphy*. Write a program that requests a word without repeated characters as input and then displays all the

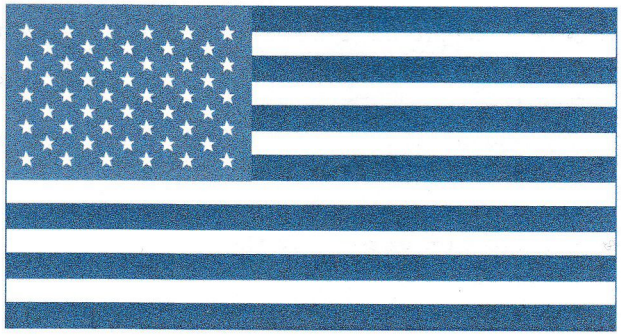


FIGURE 6.31 Outcome of Programming Project 4.

permutations of the word. See Fig. 6.32. (***Hint:*** Suppose the word has six characters. Consider the characters of the word one at a time. Then display the words beginning with that character and followed by each of the 5! permutations of the remaining characters of the word.)

```
Enter a word: ear
ear era aer are rea rae
```

FIGURE 6.32 Possible outcome of Programming Project 5.

6. Pascal's Triangle The triangular array of numbers in Fig. 6.33 is called *Pascal's triangle*, in honor of the seventeenth century mathematician Blaise Pascal. The n^{th} row of the triangle gives the coefficients of the terms in the expansion of $(1 + x)^n$. For instance, the 5^{th} row tells us that

$$(1 + x)^5 = 1 + 5x + 10x^2 + 10x^3 + 5x^4 + 1x^5$$

```
                    1                       Row
                  1   1                     0
                1   2   1                   1
              1   3   3   1                 2
            1   4   6   4   1               3
          1   5  10  10   5   1             4
                                            5
```

FIGURE 6.33 Pascal's Triangle.

With the coefficients arranged in this way, each number in the triangle is the sum of the two numbers directly above it (one to the left and one to the right). For example, in row four, 1 is the sum of 1 (the only number above it), 4 is the sum of 1 and 3, 6 is the sum of 3 and 3, and so on. Since each row can be calculated from the previous row, recursion can easily be used to generate any row of Pascal's triangle. Write a program that prompts the user for a nonnegative integer n and then displays the numbers in the n^{th} row of the triangle. See Fig. 6.34.

```
Enter a nonnegative integer: 6
Row 6: 1 6 15 20 15 6 1
```

FIGURE 6.34 Possible outcome of Programming Project 6.