Suppose the average of a list of class grades named `grades` needs to be computed. In this case, a for loop can be constructed to iterate over the grades,

```
for k in grades:
    sum = sum + k

print('Class average is', sum/len(grades))
```

However, suppose that the instructor made a mistake in grading, and a point needed to be added to each student's grade? In order to accomplish this, the index value (the location) of each element must be used to update each grade value. Thus, the loop variable of the for loop must iterate over the index values of the list,

```
for k in range(len(grades)):
    grades[k] = grades[k] + 1
```

In such cases, the loop variable `k` is also functioning as an *index variable*. An **index variable** is a variable whose *changing value is used to access elements of an indexed data structure*. Note that the range function may be given only one argument. In that case, the starting value of the range defaults to 0. Thus, `range(len(grades))` is equivalent to `range(0,len(grades))`.

LET'S TRY IT

From the Python Shell, enter the following and observe the results.

```
>>> nums = [10, 20, 30]

>>> for k in range(len(nums)):
    print(nums[k])
???
```

```
>>> for k in range(len(nums)-1, -1, -1):
        print(nums[k])
???
```

An **index variable** is a variable whose changing value is used to access elements of an indexed data structure.

4.3.4 While Loops and Lists (Sequences)

There are situations in which a sequence is to be traversed while a given condition is true. In such cases, a while loop is the appropriate control structure. (Another approach for the partial traversal of a sequence is by use of a for loop containing `break` statements. We avoid the use of `break` statements in this text, favoring the more structured while loop approach.)

Let's say that we need to determine whether the value `40` occurs in list `nums` (equal to [10, 20, 30]). In this case, once the value is found, the traversal of the list is terminated. An example of this is given in Figure 4-11.

Variable `k` is initialized to `0`, and used as an index variable. Thus, the first time through the loop, `k` is `0`, and `nums[0]` (with the value `10`) is compared to `item_to_find`. Since they are not equal, the second clause of the if statement is executed, incrementing `k` to `1`. The loop continues until either the item is found, or the complete list has been traversed. The final if statement determines

```
k = 0
item_to_find = 40
found_item = False

while k < len(nums) and not found_item:
    if nums[k] == item_to_find:
        found_item = True
    else:
        k = k + 1

if found_item:
    print('item found')
else:
    print('item not found')
```

FIGURE 4-11 List Search in Python

which of the two possibilities for ending the loop occurred, displaying either `'item found'` or `'item not found'`. Finally, note that the correct loop condition is `k < len(nums)`, and not `k <= len(nums)`. Otherwise, an "index out of range" error would result.

LET'S TRY IT

Enter and execute the following Python code and observe the results.

```
k = 0
sum = 0
nums = range(100)

while k < len(nums) and sum < 100:
     sum = sum + nums[k]
     k = k + 1

print('The first', k, 'integers sum to 100 or greater')
```

For situations in which a sequence is to be traversed while a given condition is true, a while loop is the appropriate control structure to use.

4.3.5 Let's Apply It—Password Encryption/Decryption Program

The following program (Figure 4-13) allows a user to encrypt and decrypt passwords containing uppercase/lowercase characters, digits, and special characters. This program utilizes the following programming features:

- ➤ for loop
- ➤ nested sequences (tuples)

Example program execution is given in Figure 4-12.

Lines 4–9 perform the initialization needed for the program. Variable `password_out` is used to hold the encrypted or decrypted output of the program. Since the output string is created by appending to it each translated character one at a time, it is initialized to the empty string.

Program Execution ...

```
This program will encrypt and decrypt user passwords

Enter (e) to encrypt a password, and (d) to decrypt: e
Enter password: Pizza2Day!
Your encrypted password is: Njaam2Fmc!
>>>
```

Program Execution ...

```
This program will encrypt and decrypt user passwords

Enter (e) to encrypt a password, and (d) to decrypt: d
Enter password: Njaam2Fmc!
Your decrypted password is: Pizza2Day!
```

FIGURE 4-12 Execution of Password Encryption/Decryption Program

Variable `encryption_key` holds the tuple (of tuples) used to encrypt/decrypt passwords. This tuple contains as elements tuples of length two,

```
encryption_key = (('a', 'm'), ('b', 'h'), etc.
```

The first tuple, `('a', 'm')`, for example, is used to encode the letter `'a'`. Thus, when encrypting a given file, each occurrence of `'a'` is replaced by the letter `'m'`. When decrypting, the reverse is done—all occurrences of letter `'m'` are replaced by the letter `'a'`.

Line 12 contains the program greeting. **Line 15** inputs from the user whether they wish to encrypt or decrypt a password. Based on the response, variable `encrypting` is set to either `True` or `False` (**line 20**).

The program section in **lines 26–47** performs the encryption and decryption. If variable `encrypting` is equal to `True`, then `from_index` is set to `0` and `to_index` is set to `1`, causing the "direction" of the substitution of letters to go from the first in the pair to the second (`'a'` replaced by `'m'`). When encrypting is `False` (and thus decryption should be performed), the direction of the substitution is from the second of the pair to the first (`'m'` replaced by `'a'`).

Variable `case_changer` (**line 33**) is set to the difference between the encoding of the lowercase and the uppercase letters (recall that the encoding of the lowercase letters is greater than that of the uppercase letters). The for loop at **line 38** performs the iteration over the pairs of letters in the encryption key. The first time through the loop, `t = ('a', 'm')`. Thus, `t[from_index]` and `t[to_index]` refer to each of the characters in the pair. Since all characters in the encryption key are in lowercase, when uppercase letters are found in the password, they are converted to lowercase by use of variable `case_changer` (**line 43**) before being compared to the (lowercase) letters in the encryption key. This works because the character encoding of all lowercase letters is greater than the corresponding uppercase version,

```
>>> ord('A')          >>> ord('a')          >>> ord('a') - ord('A')
65                    97                    32
```

A similar approach is used for converting from lowercase back to uppercase. Finally, on **lines 50–53**, the encrypted and decrypted versions of the password are displayed to the user.

```
 1 # Password Encryption/Decryption Program
 2
 3 # init
 4 password_out = ''
 5 case_changer = ord('a') - ord('A')
 6 encryption_key = (('a','m'), ('b','h'), ('c','t'), ('d','f'), ('e','g'),
 7   ('f','k'), ('g','b'), ('h','p'), ('i','j'), ('j','w'), ('k','e'),('l','r'),
 8   ('m','q'), ('n','s'), ('o','l'), ('p','n'), ('q','i'), ('r','u'), ('s','o'),
 9   ('t','x'), ('u','z'), ('v','y'), ('w','v'), ('x','d'), ('y','c'), ('z','a'))
10
11 # program greeting
12 print('This program will encrypt and decrypt user passwords\n')
13
14 # get selection (encrypt/decrypt)
15 which = input('Enter (e) to encrypt a password, and (d) to decrypt: ')
16
17 while which != 'e' and which != 'd':
18     which = input("\nINVALID - Enter 'e' to encrypt, 'd' to decrypt: ")
19
20 encrypting = (which == 'e')  # assigns True or False
21
22 # get password
23 password_in = input('Enter password: ')
24
25 # perform encryption / decryption
26 if encrypting:
27     from_index = 0
28     to_index = 1
29 else:
30     from_index = 1
31     to_index = 0
32
33 case_changer = ord('a') - ord('A')
34
35 for ch in password_in:
36     letter_found = False
37
38     for t in encryption_key:
39         if ('a' <= ch and ch <= 'z') and ch == t[from_index]:
40             password_out = password_out + t[to_index]
41             letter_found = True
42         elif ('A' <= ch and ch <= 'Z') and chr(ord(ch) + 32) == t[from_index]:
43             password_out = password_out + chr(ord(t[to_index]) - case_changer)
44             letter_found = True
45
46     if not letter_found:
47          password_out = password_out + ch
48
49 # output
50 if encrypting:
51     print('Your encrypted password is:', password_out)
52 else:
53     print('Your decrypted password is:', password_out)
```

FIGURE 4-13 Password Encryption/Decryption Program

The substitution occurs in the nested for loops in **lines 35–47**. The outer for loop iterates variable `ch` over each character in the entered password (to be encrypted or decrypted). The first step of the outer for loop is to initialize `letter_found` to `False`. This variable is used to indicate if each character is a (uppercase or lowercase) letter. If so, it is replaced by its corresponding encoding character. If not, it must be a digit or special character, and thus appended as is (**line 47**). The code on **lines 39–41** and **lines 42–46** is similar to each other. The only difference is that since the letters in the encryption key are all lowercase, any uppercase letters in the password need to be converted to lowercase before being compared to the letters in the key.

Self-Test Questions

1. For `nums = [10,30,20,40]`, what does the following for loop output?

```
for k in nums:
    print(k)
```

```
(a) 10      (b) 10      (c) 10
    20          30          30
    30          20          20
    40          40
```

2. For `nums = [10, 30, 20, 40]`, what does the following for loop output?

```
for k in range(1, 4):
    print(nums[k])
```

```
(a) 10      (b) 30      (c) 10
    30          20          30
    20          40          20
                            40
```

3. For `fruit = 'strawberry'`, what does the following for loop output?

```
for k in range(0, len(fruit), 2):
    print(fruit[k], end='')
```

(a) `srwer` **(b)** `tabry`

4. For `nums = [12, 4, 11, 23, 18, 41, 27]`, what is the value of `k` when the while loop terminates?

```
k = 0
while k < len(nums) and nums[k] != 18:
      k = k + 1
```

(a) 3 **(b)** 4 **(c)** 5

ANSWERS: 1. (b), 2. (b), 3. (a), 4. (b)

4.4 More on Python Lists

In this section, we take a closer look at the assignment of lists. We also introduce a useful and convenient means of generating lists that the `range` function cannot produce, called *list comprehensions*.

4.4.1 Assigning and Copying Lists

Because of the way that lists are represented in Python, when a variable is assigned to another variable holding a list, `list2 = list1`, each variable ends up referring to the *same instance* of the list in memory. This is depicted in Figure 4-14.

(b) How many elements would be looked at when the list is traversed (from top to bottom) until the value 19 was found?

Section 4.2

2. Which of the following lists are syntactically correct in Python?
(a) `[1, 2, 3, 'four']` **(b)** `[1, 2, [3, 4]]` **(c)** `[[1, 2, 3]['four']]`

3. For `lst = [4, 2, 9, 1]`, what is the result of each of the following list operations?
(a) `lst[1]` **(b)** `lst.insert(2, 3)` **(c)** `del lst[3]` **(d)** `lst.append(3)`

4. For `fruit = ['apple', 'banana', 'pear', 'cherry']`, use a list operation to change the list to `['apple', 'banana', 'cherry']`.

5. For a list of integers, `lst`, give the code to retrieve the maximum value of the second half of the list.

6. For variable `product_code` containing a string of letters and digits,
(a) Give an if statement that outputs `"Verified"` if `product_code` contains both a `"Z"` and a `"9"`, and outputs `"Failed"` otherwise.
(b) Give a Python instruction that prints out just the last three characters in `product_code`.

7. Which of the following are valid operations on tuples (for tuples `t1` and `t2`)?
(a) `len(t1)` **(b)** `t1 + t2` **(c)** `t1.append(10)` **(d)** `t1.insert(0, 10)`

8. For `str1 = 'Hello World'`, answer the following,
(a) Give an instruction that prints the fourth character of the string.
(b) Give an instruction that finds the index location of the first occurrence of the letter `'o'` in the string.

9. For a nested list `lst` that contains sublists of integers of the form `[n1, n2, n3]`,
(a) Give a Python instruction that determines the length of the list.
(b) Give Python code that determines how many total integer values there are in list `lst`.
(c) Give Python code that totals all the values in list `lst`.
(d) Given an assignment statement that assigns the third integer of the fourth element (sublist) of `lst` to the value 12.

Section 4.3

10. For a list of integers named `nums`,
(a) Write a while loop that adds up all the values in `nums`.
(b) Write a for loop that adds up all the values in `nums` in which the loop variable is assigned each value in the list.
(c) Write a for loop that adds up all the elements in `nums` in which the loop variable is assigned to the index value of each element in the list.
(d) Write a for loop that displays the elements in `nums` backwards.
(e) Write a for loop that displays every other element in `nums`, starting with the first element.

Section 4.4

11. For `list1 = [1, 2, 3, 4]` and `list2 = [5, 6, 7, 8]`, give the values of `list1[0]` and `list2[0]` where indicated after the following assignments.

(a) `list1[0] = 10`				
`list2[0] = 50`	`list1[0]`	______	`list2[0]`	______
(b) `list2 = list1`	`list1[0]`	______	`list2[0]`	______
(c) `list2[0] = 15`	`list1[0]`	______	`list2[0]`	______
(d) `list1[0] = 0`	`list1[0]`	______	`list2[0]`	______

12. Give an appropriate list comprehension for each of the following.

(a) Producing a list of consonants that appear in string variable `w`.

(b) Producing a list of numbers between 1 and 100 that are divisible by 3.

(c) Producing a list of numbers, `zero_values`, from a list of floating-point values, `data_values`, that are within some distance, `epsilon`, from 0.

PYTHON PROGRAMMING EXERCISES

P1. Write a Python program that prompts the user for a list of integers, stores in another list only those values between 1–100, and displays the resulting list.

P2. Write a Python program that prompts the user for a list of integers, stores in another list only those values that are in tuple `valid_values`, and displays the resulting list.

P3. Write a Python program that prompts the user for a list of integers and stores them in a list. For all values that are greater than 100, the string `'over'` should be stored instead. The program should display the resulting list.

P4. Write a Python program that prompts the user to enter a list of first names and stores them in a list. The program should display how many times the letter `'a'` appears within the list.

P5. Write a Python program that prompts the user to enter a list of words and stores in a list only those words whose first letter occurs again within the word (for example, `'Baboon'`). The program should display the resulting list.

P6. Write a Python program that prompts the user to enter types of fruit, and how many pounds of fruit there are for each type. The program should then display the information in the form *fruit*, *weight* listed in alphabetical order, one fruit type per line as shown below,

```
Apple, 6 lbs.
Banana, 11 lbs.
etc.
```

P7. Write a Python program that prompts the user to enter integer values for each of two lists. It then should displays whether the lists are of the same length, whether the elements in each list sum to the same value, and whether there are any values that occur in both lists.

PROGRAM MODIFICATION PROBLEMS

M1. Chinese Zodiac Program: Japanese and Vietnamese Variations

Modify the Chinese Zodiac program in the chapter to allow the user to select the Chinese Zodiac, the Japanese Zodiac, or the Vietnamese Zodiac. The Japanese Zodiac is the same as the Chinese Zodiac, except that "Pig" is substituted with "Wild Boar." The Vietnamese Zodiac is also the same except that the "Ox" is substituted with "Water Buffalo" and "Rabbit" is replaced with "Cat."

M2. Chinese Zodiac Program: Improved Accuracy

The true Chinese Zodiac does not strictly follow the year that a given person was born. It also depends on the month and date as well, which vary over the years. Following are the correct range of dates for each of the Zodiac symbols for the years 1984 to 2007 (which includes two full cycles of the zodiac). Modify