An example dialog box for `asksaveasfilename` is shown in Figure 5.3. You can, of course, import both of these functions at once with an import like:

```
from tkinter.filedialog import askopenfilename, asksaveasfilename
```

Both of these functions also have numerous optional parameters so that a program can customize the the resulting dialogs, for example by changing the title or suggesting a default file name. If you are interested in those details, you should consult the Python documentation.

5.10 Chapter Summary

This chapter has covered important elements of the Python string, list, and file objects. Here is a summary of the highlights:

- Strings are sequences of characters. String literals can be delimited with either single or double quotes.
- Strings and lists can be manipulated with the built-in sequence operations for concatenation (`+`), repetition (`*`), indexing (`[]`), slicing (`[:]`), and length (`len()`). A `for` loop can be used to iterate through the characters of a string, items in a list, or lines of a file.
- One way of converting numeric information into string information is to use a string or a list as a lookup table.
- Lists are more general than strings.
  - Strings are always sequences of characters, whereas lists can contain values of any type.
  - Lists are mutable, which means that items in a list can be modified by assigning new values.
- Strings are represented in the computer as numeric codes. ASCII and Unicode are compatible standards that are used for specifying the correspondence between characters and the underlying codes. Python provides the `ord` and `chr` functions for translating between Unicode codes and characters.
- Python string and list objects include many useful built-in methods for string and list processing.

- The process of encoding data to keep it private is called encryption. There are two different kinds of encryption systems: private key and public key.
- Program input and output often involve string processing. Python provides numerous operators for converting back and forth between numbers and strings. The string formatting method (`format`) is particularly useful for producing nicely formatted output.
- Text files are multi-line strings stored in secondary memory. A text file may be opened for reading or writing. When opened for writing, the existing contents of the file are erased. Python provides three file-reading methods: `read()`, `readline()`, and `readlines()`. It is also possible to iterate through the lines of a file with a `for` loop. Data is written to a file using the `print` function. When processing is finished, a file should be closed.

5.11 Exercises

Review Questions

True/False

1. A Python string literal is always enclosed in double quotes.
2. The last character of a string `s` is at position `len(s)-1`.
3. A string always contains a single line of text.
4. In Python `"4" + "5"` is `"45"`.
5. Python lists are mutable, but strings are not.
6. ASCII is a standard for representing characters using numeric codes.
7. The `split` method breaks a string into a list of substrings, and `join` does the opposite.
8. A substitution cipher is a good way to keep sensitive information secure.
9. The `add` method can be used to add an item to the end of a list.
10. The process of associating a file with an object in a program is called “reading” the file.

Multiple Choice

1. Accessing a single character out of a string is called:
   a) slicing b) concatenation c) assignment d) indexing
2. Which of the following is the same as `s[0:-1]`?
   a) `s[-1]` b) `s[:]` c) `s[:len(s)-1]` d) `s[0:len(s)]`
3. What function gives the Unicode value of a character?
   a) `ord` b) `ascii` c) `chr` d) `eval`
4. Which of the following can *not* be used to convert a string of digits into a number?
   a) `int` b) `float` c) `str` d) `eval`
5. A successor to ASCII that includes characters from (nearly) all written languages is
   a) TELLI b) ASCII++ c) Unicode d) ISO
6. Which string method converts all the characters of a string to upper case?
   a) `capitalize` b) `capwords` c) `uppercase` d) `upper`
7. The string "slots" that are filled in by the `format` method are marked by:
   a) `%` b) `$` c) `[]` d) `{}`
8. Which of the following is *not* a file-reading method in Python?
   a) `read` b) `readline` c) `readall` d) `readlines`
9. The term for a program that does its input and output with files is
   a) file-oriented b) multi-line c) batch d) lame
10. Before reading or writing to a file, a file object must be created via
    a) `open` b) `create` c) `File` d) `Folder`

Discussion

1. Given the initial statements:

```
s1 = "spam"
s2 = "ni!"
```

Show the result of evaluating each of the following string expressions.

a) `"The Knights who say, " + s2`
b) `3 * s1 + 2 * s2`
c) `s1[1]`
d) `s1[1:3]`
e) `s1[2] + s2[:2]`
f) `s1 + s2[-1]`
g) `s1.upper()`
h) `s2.upper().ljust(4) * 3`

2. Given the same initial statements as in the previous problem, show a Python expression that could construct each of the following results by performing string operations on s1 and s2.

a) `"NI"`
b) `"ni!spamni!"`
c) `"Spam Ni!  Spam Ni!  Spam Ni!"`
d) `"spam"`
e) `["sp","m"]`
f) `"spm"`

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

a)
```
for ch in "aardvark":
    print(ch)
```
b)
```
for w in "Now is the winter of our discontent...".split():
    print(w)
```
c)
```
for w in "Mississippi".split("i"):
    print(w, end=" ")
```
d)
```
msg = ""
for s in "secret".split("e"):
    msg = msg + s
print(msg)
```
e)
```
msg = ""
for ch in "secret":
    msg = msg + chr(ord(ch)+1)
print(msg)
```

4. Show the string that would result from each of the following string formatting operations. If the operation is not legal, explain why.

   a) `"Looks like {1} and {0} for breakfast".format("eggs", "spam")`

   b) `"There is {0} {1} {2} {3}".format(1,"spam", 4, "you")`

   c) `"Hello {0}".format("Susan", "Computewell")`

   d) `"{0:0.2f} {0:0.2f}".format(2.3, 2.3468)`

   e) `"{7.5f} {7.5f}".format(2.3, 2.3468)`

   f) `"Time left {0:02}:{1:05.2f}".format(1, 37.374)`

   g) `"{1:3}".format("14")`

5. Explain why public key encryption is more useful for securing communications on the Internet than private (shared) key encryption.

Programming Exercises

1. As discussed in the chapter, string formatting could be used to simplify the `dateconvert2.py` program. Go back and redo this program making use of the string-formatting method.

2. A certain CS professor gives 5-point quizzes that are graded on the scale 5-A, 4-B, 3-C, 2-D, 1-F, 0-F. Write a program that accepts a quiz score as an input and prints out the corresponding grade.

3. A certain CS professor gives 100-point exams that are graded on the scale 90–100:A, 80–89:B, 70–79:C, 60–69:D, <60:F. Write a program that accepts an exam score as input and prints out the corresponding grade.

4. An *acronym* is a word formed by taking the first letters of the words in a phrase and making a word from them. For example, RAM is an acronym for “random access memory.” Write a program that allows the user to type in a phrase and then outputs the acronym for that phrase. *Note*: The acronym should be all uppercase, even if the words in the phrase are not capitalized.

5. Numerologists claim to be able to determine a person’s character traits based on the “numeric value” of a name. The value of a name is determined by summing up the values of the letters of the name where “a” is 1, “b” is 2, “c” is 3, up to “z” being 26. For example, the name “Zelle”

would have the value $26 + 5 + 12 + 12 + 5 = 60$ (which happens to be a very auspicious number, by the way). Write a program that calculates the numeric value of a single name provided as input.

6. Expand your solution to the previous problem to allow the calculation of a complete name such as "John Marvin Zelle" or "John Jacob Jingleheimer Smith." The total value is just the sum of the numeric values of all the names.

7. A Caesar cipher is a simple substitution cipher based on the idea of shifting each letter of the plaintext message a fixed number (called the key) of positions in the alphabet. For example, if the key value is 2, the word "Sourpuss" would be encoded as "Uqwtrwuu." The original message can be recovered by "reencoding" it using the negative of the key.

   Write a program that can encode and decode Caesar ciphers. The input to the program will be a string of plaintext and the value of the key. The output will be an encoded message where each character in the original message is replaced by shifting it *key* characters in the Unicode character set. For example, if `ch` is a character in the string and `key` is the amount to shift, then the character that replaces `ch` can be calculated as: `chr(ord(ch) + key)`.

8. One problem with the previous exercise is that it does not deal with the case when we "drop off the end" of the alphabet. A true Caesar cipher does the shifting in a circular fashion where the next character after "z" is "a." Modify your solution to the previous problem to make it circular. You may assume that the input consists only of letters and spaces. *Hint*: Make a string containing all the characters of your alphabet and use positions in this string as your code. You do not have to shift "z" into "a"; just make sure that you use a circular shift over the entire sequence of characters in your alphabet string.

9. Write a program that counts the number of words in a sentence entered by the user.

10. Write a program that calculates the average word length in a sentence entered by the user.

11. Write an improved version of the `chaos.py` program from Chapter 1 that allows a user to input two initial values and the number of iterations,

and then prints a nicely formatted table showing how the values change over time. For example, if the starting values were .25 and .26 with 10 iterations, the table might look like this:

```
index    0.25         0.26
----------------------------
  1     0.731250     0.750360
  2     0.766441     0.730547
  3     0.698135     0.767707
  4     0.821896     0.695499
  5     0.570894     0.825942
  6     0.955399     0.560671
  7     0.166187     0.960644
  8     0.540418     0.147447
  9     0.968629     0.490255
 10     0.118509     0.974630
```

12. Write an improved version of the `futval.py` program from Chapter 2. Your program will prompt the user for the amount of the investment, the annualized interest rate, and the number of years of the investment. The program will then output a nicely formatted table that tracks the value of the investment year by year. Your output might look something like this:

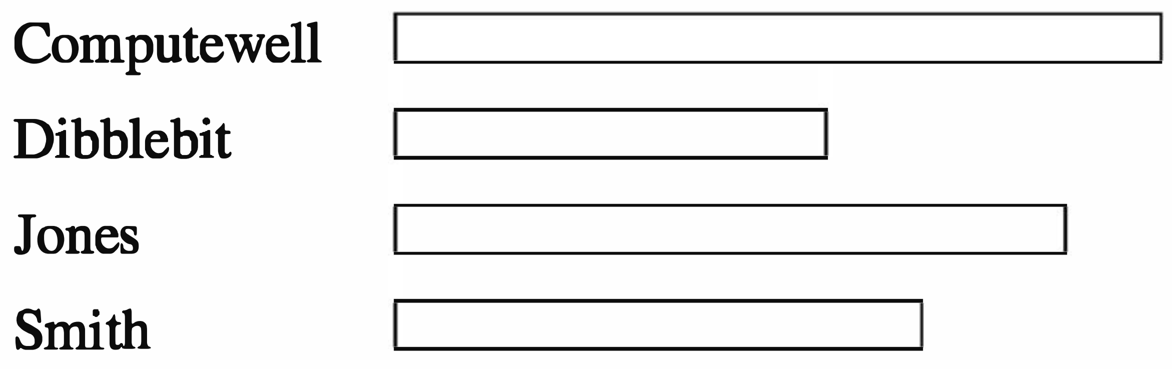
```
Year      Value
----------------
  0      $2000.00
  1      $2200.00
  2      $2420.00
  3      $2662.00
  4      $2928.20
  5      $3221.02
  6      $3542.12
  7      $3897.43
```

13. Redo any of the previous programming problems to make them batch-oriented (using text files for input and output).

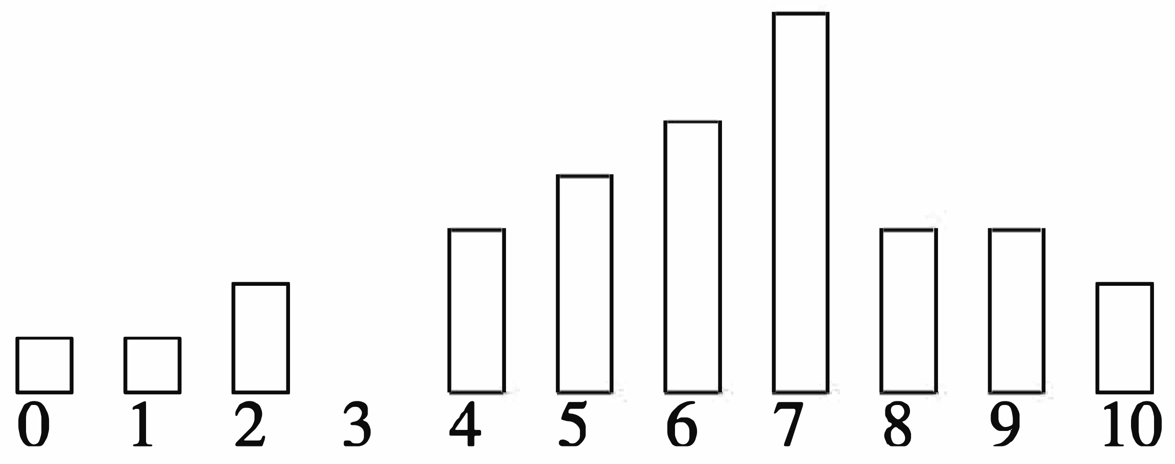
14. Word Count. A common utility on Unix/Linux systems is a small program called "wc." This program analyzes a file to determine the number of

lines, words, and characters contained therein. Write your own version of wc. The program should accept a file name as input and then print three numbers showing the count of lines, words, and characters in the file.

15. Write a program to plot a horizontal bar chart of student exam scores. Your program should get input from a file. The first line of the file contains the count of the number of students in the file, and each subsequent line contains a student's last name followed by a score in the range 0–100. Your program should draw a horizontal rectangle for each student where the length of the bar represents the student's score. The bars should all line up on their left-hand edges. *Hint*: Use the number of students to determine the size of the window and its coordinates. Bonus: label the bars at the left end with the students' names.



16. Write a program to draw a quiz score histogram. Your program should read data from a file. Each line of the file contains a number in the range 0–10. Your program must count the number of occurrences of each score and then draw a vertical bar chart with a bar for each possible score (0–10) with a height corresponding to the count of that score. For example, if 15 students got an 8, then the height of the bar for 8 should be 15. *Hint*: Use a list that stores the count for each possible score. An example histogram is shown below:


0
1
2
3
4
5
6
7
8
9
10