CHAPTER EXERCISES

Section 8.1

1. Explain the difference in how numbers are represented in a text file versus how they are represented in a binary file containing numerical values.

2. Explain why binary files do not contain newline characters.

Section 8.2

3. Give an instruction in Python that opens a file named `'datafile.txt'` for reading and assigns identifier `input_file` to the file object created.

4. Give an instruction in Python that opens a file named `'datafile2.txt'` for writing and assigns identifier `output_file` to the file object created.

5. Assume that `input_file` is a file object for a text file open for reading, and `output_file` is a file object for a text file open for writing. Explain the contents of the output file after the following code is executed,

```
empty_str = ''
line = input_file.readline()

while line != empty_str:
    output_file.write(line + '\n')
    line = input_file.readline()
```

Section 8.3

6. Give a for loop that counts all the letter characters in string `line`.

7. For variable `month` which contains the full name of any given month, give an instruction to display just the first three letters of the month.

8. Give an instruction that displays `True` if the letter 'r' appears in a variable named `month`, otherwise displays `False`.

9. Give an instruction for determining how many times the letter 'r' appears in a variable named `month`.

10. For variables `first_name` and `last_name`, give an instruction that displays the person's name in the form *last name*, *first name*.

11. Give an instruction that determines if a variable named `ss_num` contains any non-digit characters, other than a dash.

12. Give an instruction that determines the index of the '@' character in an email address in variable `email_addr`.

13. For variable `date` containing a date of the form 12/14/2012, write a function that produces the same date, but with all slashes characters replaced with dashes.

14. For a variable named `err_mesg` that contains error messages in the form `** error message **`, give an instruction that produces a string containing the error message without the leading and trailing asterisks and blank characters.

Section 8.4

15. Identify the error in the following code,

```
input_file_opened = False

while not input_file_opened:
```

```
try:
    file_name = input('Enter file name: ')
    input_file = open(file_name, 'r')
except:
    print('Input file not found - please reenter')
```

PYTHON PROGRAMMING EXERCISES

P1. Write a Python function called `reduceWhitespace` that is given a line read from a text file and returns the line with all extra whitespace characters between words removed,

'This line has extra   space   characters' → 'This line has extra space characters'

P2. Write a Python function named `extractTemp` that is given a line read from a text file and displays the one number (integer) found in the string,

'The high today will be 75 degrees' → 75

P3. Write a Python function named `checkQuotes` that is given a line read from a text file and returns `True` if each quote characters in the line has a matching quote (of the same type), otherwise returns `False`.

'Today's high temperature will be 75 degrees' → `False`

P4. Write a Python function named `countAllLetters` that is given a line read from a text file and returns a list containing every letter in the line and the number of times that each letter appears (with upper/lower case letters counted together),

'This is a short line' → `[('t', 2), ('h', 2), ('i', 3), ('s', 3), ('a', 1),`
`('o', 1), ('r', 1), ('l', 1), ('n', 1), ('e', 1)]`

P5. Write a Python function named `interleaveChars` that is given *two* lines read from a text file, and returns a single string containing the characters of each string interleaved,

'Hello', 'Goodbye' → `'HGeololdobye'`

P6. Write a program segment that opens and reads a text file and displays how many lines of text are in the file.

P7. Write a program segment that reads a text file named `original_text`, and writes every other line, starting with the first line, to a new file named `half_text`.

P8. Write a program segment that reads a text file named `original_text`, and displays how many times the letter 'e' occurs.

PROGRAM MODIFICATION PROBLEMS

M1. Sparse Text Program: User-Selected Letter Removed
Modify the Sparse Text program in section 8.3.4 so that instead of the letter 'e' being removed, the user is prompted for the letter to remove.

M2. Sparse Text Program: Random Removal of Letters
Modify the Sparse Text program in section 8.3.4 so that instead of a particular letter removed, a percentage of the letters are randomly removed based on a percentage entered by the user.

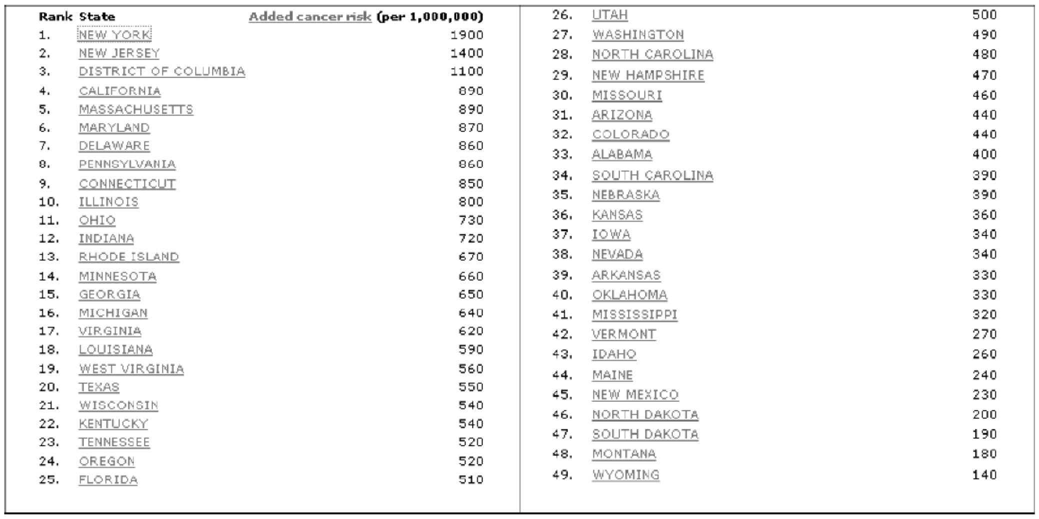
M3. Word Frequency Count Program: Display of Scanned Lines
Modify the Word Frequency Count program in section 8.4.6 so that the text lines being scanned are at the same time displayed on the screen.

M4. Word Frequency Count Program: Counting of a Set of Words
Modify the Word Frequency Count Program so that the user can enter any number of words to be counted within a given text file.

M5. Word Frequency Count Program: Counting of All Words
Modify the Word Frequency Count program so that all the words in a given text file are counted.

M6. Word Frequency Count Program: Outputting Results to a File
Modify the Word Frequency Count program so that the counts of all words in a given text file are output to a file with the same name as the file read, but with the file extension `'.wc'` (for 'word count').

M7. Lung Cancer Correlation Program: Air Pollution and Lung Cancer
Modify the cigarettes and lung cancer correlation program in the Computational Problem Solving section of the chapter to correlate lung cancer with air pollution instead. Use the data from the following ranking of states from highest to lowest amounts of air pollution given below.



Rank	State	Added cancer risk (per 1,000,000)
1.	NEW YORK	1900
2.	NEW JERSEY	1400
3.	DISTRICT OF COLUMBIA	1100
4.	CALIFORNIA	890
5.	MASSACHUSETTS	890
6.	MARYLAND	870
7.	DELAWARE	860
8.	PENNSYLVANIA	860
9.	CONNECTICUT	850
10.	ILLINOIS	800
11.	OHIO	730
12.	INDIANA	720
13.	RHODE ISLAND	670
14.	MINNESOTA	660
15.	GEORGIA	650
16.	MICHIGAN	640
17.	VIRGINIA	620
18.	LOUISIANA	590
19.	WEST VIRGINIA	560
20.	TEXAS	550
21.	WISCONSIN	540
22.	KENTUCKY	540
23.	TENNESSEE	520
24.	OREGON	520
25.	FLORIDA	510
26.	UTAH	500
27.	WASHINGTON	490
28.	NORTH CAROLINA	480
29.	NEW HAMPSHIRE	470
30.	MISSOURI	460
31.	ARIZONA	440
32.	COLORADO	440
33.	ALABAMA	400
34.	SOUTH CAROLINA	390
35.	NEBRASKA	390
36.	KANSAS	360
37.	IOWA	340
38.	NEVADA	340
39.	ARKANSAS	330
40.	OKLAHOMA	330
41.	MISSISSIPPI	320
42.	VERMONT	270
43.	IDAHO	260
44.	MAINE	240
45.	NEW MEXICO	230
46.	NORTH DAKOTA	200
47.	SOUTH DAKOTA	190
48.	MONTANA	180
49.	WYOMING	140

PROGRAM DEVELOPMENT PROBLEMS

D1. Sentence, Word, and Character Count Program
Develop and test a Python program that reads in any given text file and displays the number of lines, words, and total number of characters there are in the file, including spaces and special characters, but not the newline character, `'\n'`.

D2. Variation on a Sparsity Program
Develop and test a program that reads the text in a given file, and produces a new file in which the *first occurrence only* of the vowel in each word is removed, unless the removal would leave an empty word (for example, for the word "I"). Consider how readable the results are for various sample text.

D3. Message Encryption/Decryption Program
Develop and test a Python program that reads messages contained in a text file, and encodes the messages saved in a new file. For encoding messages, a simple substitution key should be used as shown below,

A	F
B	G
C	L
D	R
E	P
. .	. .

Each letter in the left column is substituted with the corresponding letter in the right column when encoding. Thus, to decode, the letters are substituted the opposite way. Unencrypted message files will be simple text files with file extension `.txt`. Encrypted message files will have the same file name, but with file extension `.enc`. For each message encoded, a new substitution key should be randomly generated and saved in a file with the extension '`.key`'. Your program should also be able to decrypt messages given a specific encoded message and the corresponding key.

D4. Morse Code Encryption/Decryption Program
Develop and test a Python program that allows a user to open a text file containing a simple message using only the (uppercase) letters A . . . Z, and saves a Morse code version of the message, that is, containing only the characters dash ("`-`"), dot ("`.`"). In the encoded version, put the encoding of each character on its own line in the text file. Use a blank line to indicate the end of a word, and two blank lines to indicate the end of a sentence. Your program should be able to both convert an English message file into Morse code, and a Morse code file into English. The Morse code for each letter is given below.

A	• -	**N**	- •
B	- • • •	**O**	- - -
C	- • - •	**P**	• - - •
D	- • •	**Q**	- - • -
E	•	**R**	• - •
F	• • - •	**S**	• • •
G	- - •	**T**	-
H	• • • •	**U**	• • -
I	• •	**V**	• • • -
J	• - - -	**W**	• - -
K	- • -	**X**	- • • -
L	• - • •	**Y**	- • - -
M	- -	**Z**	- - • •

11. Give a program segment that prompts the user for two English words, entered in no particular order, and determines if all the letters of first word are contained within the second.

PYTHON PROGRAMMING EXERCISES

P1. Write a Python function called `addDailyTemp` that is given a (possibly empty) dictionary meant to hold the average daily temperature for each day of the week, the day, and the day's average temperature. The function should add the temperature to the dictionary only if does not already contain a temperature for that day. The function should return the resulting dictionary, whether or not it is updated.

P2. Write a Python function named `moderateDays` that is given a dictionary containing the average daily temperature for each day of a week, and returns a list of the days in which the average temperature was between 70 and 79 degrees.

P3. Write a Python function named `getDailyTemps` that prompts the user for the average temperature for each day of the week, and returns a dictionary containing the entered information.

P4. Write a Python function named `getWeekendAvgTemp` that is passed a dictionary of daily temperatures, and returns the average temperature over the weekend for the weekly temperatures given.

P5. Write a Python function named `addVegetable` that is passed a (possible empty) set of vegetable names, and raises a `ValueError` exception if the given vegetable is already in the set, otherwise, the new vegetable should be added and the new set returned.

P6. Write a Python function named `numVowels` that is passed a string containing letters, each of which may be in either uppercase or lowercase, and returns a tuple containing the number of vowels and the number of consonants the string contains.

PROGRAM MODIFICATION PROBLEMS

M1. Word Frequency Count Program: Making Use of Sets
Modify the Word Frequency Count Program in Chapter 8 making use of the set type where possible.

M2. Phone Number Spelling Program: Modifying for Australia
In Australia, phone numbers are of the form 0x-xxxx-xxxx. Modify the Phone Number Spelling program so that it prints out Australian phone numbers with spellings for the last four digits.

M3. Kitchen Tile Visualization Program: Color Name Entry
Modify the Kitchen Tile Visualization Program so that, instead of requiring the user to enter hexadecimal RGB color codes, they are given a subset of twelve color names from which to select (from the colors listed in Figure 9-11).

M4. Kitchen Tile Visualization Program: Randomly Generated Patterns
Modify the Kitchen Tile Visualization Program so that the user can enter up to three different tile colors, as well as the grout color. The program should then display a pattern of tiles that is randomly generated using the three colors specified.

M5. Kitchen Tile Visualization Program: Automatic Complementary Colors
The complementary color of a given color is the "opposite" color (directly opposite on the color wheel). Designers often use complementary colors because of their vibrant contrast. Modify the Kitchen Tile visualization program so that the user enters one tile color, with the complementary color automatically