Note that for all string modifications, the variable references the same string until it is reassigned.

Python also provides a `strip` method that "strips off" leading and trailing characters from a string. This is especially useful for stripping off the newline character, `\n`, from the end of a line in text processing if needed,

```
>>> line = 'Hello\n'
>>> print(line)
'Hello'

>>>
```

```
>>> line = 'Hello\n'
>>> print(line.strip('\n'))
'Hello'
>>>
```

The `find`, `replace`, and `strip` methods in Python can be used to search and produce modified strings.

8.3.4 Let's Apply It—Sparse Text Program

Sparse data is data that lacks "density." A diary containing entries only for holidays and special occasions would contain sparse data. Sparse data can often be compressed. Compressed data should retain all (or most) of the original information. To explore this, the program in Figure 8-7 removes all occurrences of the letter 'e' from a provided text file. How much of the compressed text can be understood indicates how much of the information is retained. This program utilizes the following programming features:

- text files
- string methods (replace, strip)

Figure 8-6 shows the file produced by the Sparse Text Program for a passage from *Alice's Adventures in Wonderland.*

The main section of the program begins on **line 32**. The program welcome is displayed on **lines 33–34**. In **lines 37–38**, the file name entered by the user is opened for reading and assigned to file object `input_file`. In **lines 37–40**, a file is opened for writing with the same file name as the input file but with `'e_'` added to the beginning and assigned to the file object `output_file`.

The creation of the modified file is handled by function `createModifiedFile` (**lines 3–28**) called on **line 44**. The function returns a tuple containing how many occurrences of letter `'e'` were removed, and the total character count of the file. The values of the tuple are therefore assigned to variables `num_total_char` and `num_removals`. (Note that Python allows such a multiple assignment on lists, tuples, and even on strings.) The input and output files are closed on **lines 47–48**. Finally, both the number of characters removed and the percentage of the file that the removed characters comprised are displayed (**lines 52–54**).

What remains is function `createModifiedFile`. On **lines 9–11**, variable `empty_str` is initialized to the empty string, and variables `num_total_chars` and `num_removals` are initialized to 0. On **line 16**, variable `orig_line_length` is set to the length of the currently read line (in variable `line`) minus one. This is so that the newline character (`\n`) at the end of the line is not counted.

A Mad Tea-Party (original version)

There was a table set out under a tree in front of the house, and the March Hare and the Hatter were having tea at it: a Dormouse was sitting between them, fast asleep, and the other two were using it as a cushion, resting their elbows on it, and talking over its head. 'Very uncomfortable for the Dormouse,' thought Alice; 'only, as it's asleep, I suppose it doesn't mind.' The table was a large one, but the three were all crowded together at one corner of it: 'No room! No room!' they cried out when they saw Alice coming. 'There's PLENTY of room!' said Alice indignantly, and she sat down in a large arm-chair at one end of the table.

Program Execution ...

```
This program will display the contents of a provided text file with all
occurrences of the letter 'e' removed.

Enter file name (including file extension): alice_tea_party.txt

Thr was a tabl st out undr a tr in front of th hous, and th
March Har and th Hattr wr having ta at it: a Dormous was sitting
btwn thm, fast aslp, and th othr two wr using it as a cushion,
rsting thir lbows on it, and talking ovr its had. 'Vry uncomfortabl
for th Dormous,' thought Alic; 'only, as it's aslp, I suppos it dosn't
mind.' Th tabl was a larg on, but th thr wr all crowdd togthr
at on cornr of it: 'No room! No room!' thy crid out whn thy saw Alic
coming. 'Thr's PLNTY of room!' said Alic indignantly, and sh sat down in
a larg arm-chair at on nd of th tabl

70 occurrences of the letter 'e' removed
Percentage of data lost: 11 %
Modified text in file e_alice_tea_party.txt
```

FIGURE 8-6 Execution of the Sparse Text Program

On **line 20**, variable `modified_line` is then set to the new string produced by the `replace` method applied to the current line. Each occurrence of `'e'` and `'E'` is replaced with the empty string, thus removing these occurrences from the string. Note the consecutive calls to the `replace` method,

```
    modified_line = line.replace('e',empty_str).replace('E',empty_str)
```

This is possible to do when the first method call returns a value (object) for which the second method call can be applied. In this case, `line.replace('e',empty_str)` returns a string (consisting of a copy of the original string with all occurrences of the letter `'e'` removed), which is then operated on by the second instance of the method call to remove all instances of `'E'`.

On **line 21**, the number of characters removed (`num_removals`) is updated. The number removed from the current line is determined by taking the difference between the original line length (without the newline character) and the length of the new modified line minus one, so that the newline character in the modified line is not counted. Finally, the modified line of output to the screen (as well as to the output file) is displayed so the user may observe the results of the file processing as it is occurring. A tuple including the number of total characters in the file and the number of removed characters is returned as the function value.

```python
# Sparse Text Program

def createModifiedFile(input_file, outputfile):

    """For text file input_file, creates a new version in file outputfile
        in which all instances of the letter 'e' are removed.
    """

    empty_str = ''
    num_total_chars = 0
    num_removals = 0

    for line in input_file:

        # save original line length
        orig_line_length = len(line) - 1
        num_total_chars = num_total_chars + orig_line_length

        # remove all occurrances of letter 'e'
        modified_line = line.replace('e',empty_str).replace('E',empty_str)
        num_removals = num_removals + \
                        (orig_line_length - (len(modified_line)-1))

        # simulataneouly output line to screen and output file
        print(modified_line.strip('\n'))
        output_file.write(modified_line)

    return (num_total_chars, num_removals)

# --- main

# program welcome
print("This program will display the contents of a provided text file")
print("with all occurrences of the letter 'e' removed.\n”)

# open files for reading and writing
file_name = input('Enter file name (including file extension): ')
input_file = open(file_name,'r')
new_file_name = 'e_' + file_name
output_file = open(new_file_name,'w')

# create file with all letter e removed
print()
num_total_chars, num_removals = createModifiedFile(input_file, output_file)

# close current input and output files
input_file.close()
output_file.close()

# display percentage of characters removed
print()
print(num_removals, "occurrences of the letter 'e' removed")
print('Percentage of data lost:',
      int((num_removals / num_total_chars) * 100), '%')
print('Modified text in file', new_file_name)
```

FIGURE 8-7 Sparse Text Program

```
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.

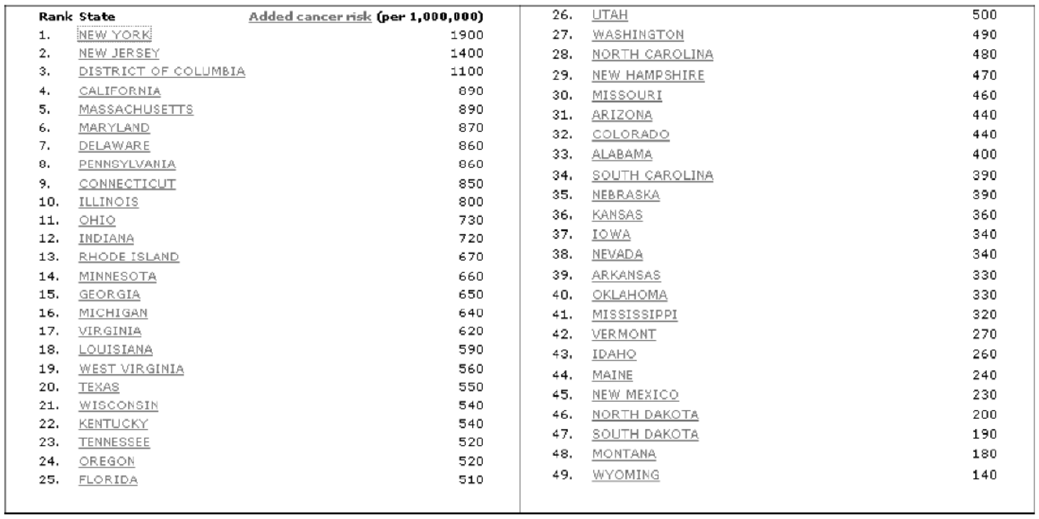
M3. Word Frequency Count Program: Display of Scanned Lines
Modify the Word Frequency Count program in section 8.4.6 so that the text lines being scanned are at the same time displayed on the screen.

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**	– – • •