Chapter 12

Text Files

There is a ton of interesting data to be found on the internet stored in text files. In this chapter we will learn how to work with data stored in text files.

12.1 Reading from files

Suppose we have a text file called `example.txt` whose contents are shown below, and we want to read its contents into Python. There are several ways to do so. We will look at two of them.

```
Hello.
This is a text file.
Bye!
```

1. The first way to read a text file uses a list comprehension to load the file line-by-line into a list:

   ```
   lines = [line.strip() for line in open('example.txt')]
   ```

   The list `lines` is now

   ```
   ['Hello.', 'This is a text file.', 'Bye!']
   ```

   The string method `strip` removes any whitespace characters from the beginning and end of a string. If we had not used it, each line would contain a newline character at the end of the line. This is usually not what we want.

   Note: `strip` removes whitespace from both the beginning and end of the line. Use `rstrip` if you need to preserve whitespace at the beginning of the line.

2. The second way of reading a text file loads the entire file into a string:

   ```
   s = open('example.txt').read()
   ```

   The string `s` is now

   ```
   'Hello.\nThis is a text file.\nBye!'
   ```

Directories

Say your program opens a file, like below:

```
s = open('file.txt').read()
```

The file is assumed to be in the same directory as your program itself. If it is in a different directory, then you need to specify that, like below:

```
s = open('c:/users/heinold/desktop/file.txt').read()
```

12.2 Writing to files

There are also several ways to write to files. We will look at one way here. We will be writing to a file called `writefile.txt`.

```
f = open('writefile.txt', 'w')
print('This is line 1.', file=f)
print('This is line 2.', file=f)
f.close()
```

We first have to open the file. That is what the first line does, with the `'w'` indicating that we want to be able to write to the file. Python creates what is called a file object to represent the file, and we give that object the name `f`. This is what we use to refer to the file. To write to the file, we use the print statement with the optional `file` argument that specifies the file to write to. When we are done writing, we should close the file to make sure all of our changes take. Be careful here because if `writefile.txt` already exists, its contents will be overwritten.

12.3 Examples

Example 1 Write a program that reads a list of temperatures from a file called `temps.txt`, converts those temperatures to Fahrenheit, and writes the results to a file called `ftemps.txt`.

```
file1 = open('ftemps.txt', 'w')
temperatures = [line.strip() for line in open('temps.txt')]
for t in temperatures:
    print(t*9/5+32, file=file1)
file1.close()
```

Example 2 In Section 7.6 we wrote a simple quiz game. The questions and answers were both contained in lists hard-coded into the program. Instead of that, we can store the questions and answers in files. That way, if you decide to change the questions or answers, you just have to change their files. Moreover, if you decide to give the program to someone else who doesn't know

Python, they can easily create their own lists of questions and answers. To do this, we just replace the lines that create the lists with the following:

```
questions = [line.strip() for line in open('questions.txt')]
answers = [line.strip() for line in open('answers.txt')]
```

Example 3 Say you have a text file that contains the results of every 2009-10 NCAA basketball game. (You can find such a file at www.kenpom.com.) A typical line of the file looks like this:

```
02/27/2010, Robert Morris, 61, Mount St. Mary's, 63
```

Below is a program that scans through the file to find the most lopsided game, the one where the winning team had the largest margin of victory.

```
lines = [line.strip() for line in open('scores.txt')]
games = [line.split(',') for line in lines]
print(max([abs(int(g[2])-int(g[4])) for g in games]))
```

We use the `split` method to break each line into a lists of its component parts. The scores are at indices 2 and 4. To find the maximum difference, we can use a list comprehension to create a list of all the margins of victories and use **`max`** to find the maximum.

The maximum turns out to be 84. Unfortunately, the method above does not tell us anything else about the game. In order to do that, we resort to the longer way to find maximums, described in Section 5.5. This allows us to store information about the game as we search for the largest margin of victory.

```
lines = [line.strip() for line in open('scores.txt')]
games = [line.split(',') for line in lines]

biggest_diff = 0
for g in games:
    diff = abs(int(g[2])-int(g[4]))
    if diff>biggest_diff:
        biggest_diff = diff
        game_info = g
print(game_info)
```

```
['12/03/2009', ' SalemInternational', '35', ' Marshall', '119']
```

12.4 Wordplay

If you like words, you can have a lot of fun with a wordlist, which is a text file where each line contains a different word. A quick web search will turn up a variety of different wordlists, ranging from lists of common English words to lists containing practically every English word.

Assuming the wordlist file is `wordlist.txt`, we can load the words into a list using the line below.

```
wordlist = [line.strip() for line in open('wordlist.txt')]
```

Example 1 Print all three letter words.

```
for word in wordlist:
    if len(word)==3:
        print(word)
```

Note that this and most of the upcoming examples can be done with list comprehensions:

```
print([word for word in wordlist if len(word)==3])
```

Example 2 Print all the words that start with `gn` or `kn`.

```
for word in wordlist:
    if word[:2]=='gn' or word[:2]=='kn':
        print(word)
```

Example 3 Determine what percentage of words start with a vowel.

```
count = 0
for word in wordlist:
    if word[0] in 'aeiou':
        count=count+1
print(100*count/len(wordlist))
```

Example 4 Print all 7-letter words that start with `th` and end in `ly`. Things like this are good for cheating at crosswords.

```
for word in wordlist:
    if len(word)==7 and word[:2]=='th' and word[-2:]=='ly':
        print(word)
```

Example 5 Print the first ten words that start with `q`.

```
i=0
while wordlist[i]!='q':
    i=i+1
print(wordlist[i:i+10]
```

Note this is not a very efficient way of doing things since we have to scan through most of the list. A binary search would be more efficient, but the above approach still runs almost instantly even for large files.

Example 6 Find the longest word that can be made using only the letters `a`, `b`, `c`, `d`, and `e`.

```
largest = 0
for word in wordlist:
    for c in word:
        if c not in 'abcde':
            break
    else:
        if len(word)>largest:
            largest=len(word)
            largest_word=word
print(largest_word)
```

The way this program works is for every word in the wordlist, we use a for/else loop (Section 9.4) to scan through the word looking checking each character to see if it is an `a`, `b`, `c`, `d`, or `e`. If any letter isn't one of these, then we break out of the loop and move on to the next word. On the other hand, if we get all the way through the loop, then we go to else block. In that block, we use a modification of the technique from Section 5.5 for finding a maximum.

12.5 Exercises

1. You are given a file called `class_scores.txt`, where each line of the file contains a one-word username and a test score separated by spaces, like below:.

   ```
   GWashington 83
   JAdams 86
   ```

   Write code that scans through the file, adds 5 points to each test score, and outputs the usernames and new test scores to a new file, `scores2.txt`.

2. You are given a file called `grades.txt`, where each line of the file contains a one-word student username and three test scores separated by spaces, like below:.

   ```
   GWashington 83 77 54
   JAdams 86 69 90
   ```

   Write code that scans through the file and determines how many students passed all three tests.

3. You are given a file called `logfile.txt` that lists log-on and log-off times for users of a system. A typical line of the file looks like this:

   ```
   Van Rossum, 14:22, 14:37
   ```

Each line has three entries separated by commas: a username, a log-on time, and a log-off time. Times are given in 24-hour format. You may assume that all log-ons and log-offs occur within a single workday.

Write a program that scans through the file and prints out all users who were online for at least an hour.

4. You are given a file called `students.txt`. A typical line in the file looks like:

```
walter melon        melon@email.msmary.edu        555-3141
```

There is a name, an email address, and a phone number, each separated by tabs. Write a program that reads through the file line-by-line, and for each line, capitalizes the first letter of the first and last name and adds the area code 301 to the phone number. Your program should write this to a new file called `students2.txt`. Here is what the first line of the new file should look like:

```
Walter Melon        melon@email.msmary.edu        301-555-3141
```

5. You are given a file `namelist.txt` that contains a bunch of names. Some of the names are a first name and a last name separated by spaces, like *George Washington*, while others have a middle name, like *John Quincy Adams*. There are no names consisting of just one word or more than three words. Write a program that asks the user to enter initials, like *GW* or *JQA*, and prints all the names that match those initials. Note that initials like *JA* should match both *John Adams* and *John Quincy Adams*.

6. You are given a file `namelist.txt` that contains a bunch of names. Print out all the names in the list in which the vowels *a*, *e*, *i*, *o*, and *u* appear in order (with repeats possible). The first vowel in the name must be *a* and after the first *u*, it is okay for there to be other vowels. An example is *Ace Elvin Coulson*.

7. You are given a file called `baseball.txt`. A typical line of the file starts like below.

```
Ichiro Suzuki     SEA     162     680     74    ...[more stats]
```

Each entry is separated by a tab, `\t`. The first entry is the player's name and the second is their team. Following that are 16 statistics. Home runs are the seventh stat and stolen bases are the eleventh. Print out all the players who have at least 20 home runs and at least 20 stolen bases.

8. For this problem, use the file of NCAA basketball scores as described in Section 12.3.

   (a) Find the average of the points scored over all the games in the file.

   (b) Pick your favorite team and scan through the file to determine how many games they won and how many games they lost.

   (c) Find the team(s) that lost by 30 or more points the most times

   (d) Find all the teams that averaged at least 70 points a game.

(e) Find all the teams that had winning records but were collectively outscored by their opponents. A team is collectively outscored by their opponents if the total number of points the team scored over all their games is less than the total number of points their opponents scored in their games against the team.

9. Benford's law states that in real data where the values are spread across several orders of magnitude, about 30% of the values will start with the number 1, whereas only about 4.6% of the values will start with the number 9. This is contrary to what we might expect, namely that values starting with 1 and 9 would be equally likely. Using the file `expenses.txt` which consists of a number of costs from an expense account, determine what percentage start with each of the digits 1 through 9. This technique is used by accountants to detect fraud.

10. *Wordplay* – Use the file `wordlist.txt` for this problem. Find the following:

    (a) All words ending in *ime*

    (b) All words whose second, third, and fourth letters are *ave*

    (c) How many words contain at least one of the letters *r, s, t, l, n, e*

    (d) The percentage of words that contain at least one of the letters *r, s, t, l, n, e*

    (e) All words with no vowels

    (f) All words that contain every vowel

    (g) Whether there are more ten-letter words or seven-letter words

    (h) The longest word in the list

    (i) All palindromes

    (j) All words that are words in reverse, like *rat* and *tar*.

    (k) Same as above, but only print one word out of each pair.

    (l) All words that contain double letters next each other like *aardvark* or *book*, excluding words that end in *lly*

    (m) All words that contain a *q* that isn't followed by a *u*

    (n) All words that contain *zu* anywhere in the word

    (o) All words that contain *ab* in multiple places, like *habitable*

    (p) All words with four or more vowels in a row

    (q) All words that contain both a *z* and a *w*

    (r) All words whose first letter is *a*, third letter is *e* and fifth letter is *i*

    (s) All two-letter words

    (t) All four-letter words that start and end with the same letter

    (u) All words that contain at least nine vowels.

    (v) All words that contain each of the letters *a*, *b*, *c*, *d*, *e*, and *f* in any order. There may be other letters in the word. Two examples are *backfield* and *feedback*.

    (w) All words whose first four and last four letters are the same

(x) All words of the form *abcd*dcba*, where * is arbitrarily long sequence of letters.

(y) All groups of 5 words, like *pat pet pit pot put*, where each word is 3 letters, all words share the same first and last letters, and the middle letter runs through all 5 vowels.

(z) The word that has the most i's.

11. Write a program to help with word games. The user enters a word and the program uses the wordlist to determine if the user's word is a real word or not.

12. Suppose we write all the words in the wordlist backwards and then arrange these backwards words alphabetically. Write a program that prints the last word in this modified wordlist.

13. Print out all combinations of the string `'Python'` plus a three letter English word. Capitalize the first letter of the three letter word. Example combinations are `'PythonCat'`, `'PythonDog'`, and `'PythonTag'`. These are valid combinations because *cat*, *dog*, and *tag* are English words. On the other hand, `'PythonQqz'` would not be a valid combination because *qqz* is not an English word. Use a wordlist to determine which three letter combinations are words.

14. Write a simple spell-checking program. The user should enter a string and the program should print out a list of all the words it thinks are misspelled. These would be all the words it cannot find in a wordlist.

15. Crossword cheater: When working on a crossword puzzle, often you will have a word where you know several of the letters, but not all of them. You can write a computer program to help you. For the program, the user should be able to input a word with the letters they know filled in and asterisks for those they don't know. The program should print out a list of all words that fit that description. For example, the input `th***ly` should return all the words that could work, namely *thickly* and *thirdly*.

16. Ask the user to enter several letters. Then find all the words that can be made with those letters, repeats allowed.

17. Using the wordlist, produce a dictionary whose keys are the letters *a* through `z` and whose values are the percentage of words that use that letter.

18. Using the wordlist, produce a dictionary whose keys are the letters *a* through `z` and whose values are the percentage of total letters in the wordlist that are that letter.

19. Write a program that asks the user for a word and finds all the smaller words that can be made from the letters of that word. The number of occurrences of a letter in a smaller word can't exceed the number of occurrences of the letter in the user's word.

20. (a) Write a program that reads a file consisting of email addresses, each on its own line. Your program should print out a string consisting of those email addresses separated by semicolons.

(b) Write the same program as above, but the new string should contain only those email addresses that do not end in `@prof.college.edu`.

21. The file `high_temperatures.txt` contains the average high temperatures for each day of the year in a certain city. Each line of the file consists of the date, written in the month/day format, followed by a space and the average high temperature for that date. Find the 30-day period over which there is the biggest increase in the average high temperature.

22. In Chapter 6 there was an exercise about the game *Mad Libs*. It asked you to make up a story and leave out some words of the story. Your program should ask the user to enter some words and tell them what types of words to enter. Then print the full story along with the inserted words. Rewrite your program from that exercise to read the story from a file. Reading the story from a file allows people who do not know how to program to use their own stories with the program without having to change the code.

23. An acronym is an abbreviation that uses the first letter of each word in a phrase. We see them everywhere. For instance, NCAA for National Collegiate Athletic Association or NBC for National Broadcasting Company. Write a program where the user enters an acronym and the program randomly selects words from a wordlist such that the words would fit the acronym. Below is some typical output generated when I ran the program:

```
Enter acronym: ABC
['addressed', 'better', 'common']

Enter acronym: BRIAN
['bank', 'regarding', 'intending', 'army', 'naive']
```

24. This problem is about a version of the game *Jotto*. The computer chooses a random five-letter word with no repeat letters. The player gets several turns to try to guess the computer's word. On each turn, the player guesses a five-letter word and is told the number of letters that their guess has in common with the computer's word.

25. The word *part* has the interesting property that if you remove its letters one by one, each resulting step is a real word. For instance, *part* → *pat* → *pa* → *a*. You may remove the letters in any order, and the last (single-letter) word needs to be a real word as well. Find all eight-letter words with this property.

26. Write a program to cheat at the game *Scrabble*. The user enters a string. Your program should return a list of all the words that can be created from those seven letters.