COSC2429 Introduction to Programming

Files

Outline

- Working with data files
- Reading text file
- Writing text file
- Example: Processing quiz scores

Working with data files

- So far, the data we have used have all been either coded right into the program, or have been entered by the user.
- However, in real life data normally reside in files. For example: images , web pages, word processing documents, music, etc.
- For our purposes, we assume that our data files are **text files**, which could be created use a **text editor** (e.g. **Notepad++**) or downloaded from a website.
- Regardless of how the file is created, Python allows us to manipulate the contents of the file with ease.

Working with data files

- In Python, we must **open** files before we can use them and **close** them when we finish working with them.
- Once a file is opened it becomes a Python **object** just like all other data.
- Here are the functions that can be used to open and close files.

Function	**Use**	**Explanation**
open	open(filename, 'r')	Open a file called filename and use it for reading. This will return a reference to a file object.
open	open(filename, 'w')	Open a file called filename and use it for writing. This will also return a reference to a file object.
close	filevariable.close()	File use is complete.

Reading text file

- Suppose we have a text file called **qbdata.txt** that contains the following data representing statistics about NFL quarterbacks. The format of the data file is as follows:

```
FName LName Pos Team Completions Attempts Yards TDs Ints Comp% Rating
```

- Here is an example of the file:

```
Colt McCoy QB CLE  135 222 1576    6   9   60.8%   74.5
Josh Freeman QB TB 291 474 3451    25  6   61.4%   95.9
Michael Vick QB PHI    233 372 3018    21  6   62.6%   100.2
Matt Schaub QB HOU 365 574 4370    24  12  63.6%   92.0
Philip Rivers QB SD    357 541 4710    30  13  66.0%   101.8
Matt Hasselbeck QB SEA 266 444 3001    12  17  59.9%   73.2
Jimmy Clausen QB CAR   157 299 1558    3   9   52.5%   58.4
Joe Flacco QB BAL  306 489 3622    25  10  62.6%   93.6
Kyle Orton QB DEN  293 498 3653    20  9   58.8%   87.5
…
```

Reading text file: using for loop

- Because a text file is a sequence of lines (i.e. a list of lines), we can read the lines using a for loop as follow. Here we assume that the given .txt file is in the same directory with the .py file.

```
infile = open("qbdata.txt", "r")

for line in infile:                  # traverse file by line using for loop
    values = line.split()            # split line by whitespace to a list of strings
    print('QB ', values[0], values[1], 'had a rating of ', values[10])

infile.close()
```

- Note that a line is just a string with the **newline character "\n"** at the end.
- The values in a line can be extracted easily into a list using the **split** function.
- If the file resides in a different directory with the program, you have to specify its full path name, e.g. qbfile = open("**C:\\Users\\v81025\\Desktop\\qbdata.txt**", "r")
- Note that we used **"\\"** to describe the **"\"** because it is a special character similar to **"\t"** or **"\n"**

Reading text file: using while loop

```
infile = open("qbdata.txt","r")

line = infile.readline()                    # read the 1st line
while line:                                 # traverse file by line using while loop
    values = line.split()                   # split line by whitespace into a list of strings
    print('QB ', values[0], values[1], 'had a rating of ', values[10])
    line = infile.readline()                # read the next line

infile.close()
```

- **“while line:”** means while line is not the empty string (empty string has an ASCII code of 0)

Writing text file

- One of the most commonly performed data processing tasks is to **read** data from a file, **manipulate** it in some way, and then **write** the resulting data out to a new data file to be used for other purposes later.
- As an example, consider the qbdata.txt file once again. Assume that we have been asked to provide a file consisting of **only the names** of the quarterbacks. In addition, the names should be in the order **last name followed by first name with the names separated by a comma**.
- Here is the first attempt to solve the problem:

```
infile = open("qbdata.txt","r")
line = infile.readline()
while line:
    items = line.split()
    data_line = items[1] + ',' + items[0]          # add ', ' between the names
    print(data_line)
    line = infile.readline()

infile.close()
```

Writing text file

- The program is quite good as it prints out the names as required. However, it doesn’t write the names into a file.
- Here is the second attempt:

```
infile = open("qbdata.txt","r")
outfile = open("qbnames.txt","w")

line = infile.readline()
while line:
    items = line.split()
    data_line = items[1] + ',' + items[0] + '\n'     # add '\n' to make a line
    outfile.write(data_line)                         # write a string to outfile
    line = infile.readline()

infile.close()
outfile.close()
```

Exercise: Processing quiz scores

- The following sample file called **quiz_scores.txt** contains one line for each student in an imaginary class. The students name is the first thing on each line, followed by some quiz scores. The number of scores might be different for each student.

```
Joe, 10, 15, 20, 30, 40
Bill, 23, 16, 19, 22
Sue, 8, 22, 17, 14, 32, 17, 24, 21, 29, 11, 17
Grace, 12, 28, 21, 45, 26, 10
John, 14, 32, 25, 16, 89
…
```

- Using the text file **quiz_scores.txt** write a program that prints out the names of students who have at least six quiz scores.
- Modify the above program to calculate the **average score** for each student, and write out the student's names along with their average scores to another text file called **avg_scores.txt**. Each student name and average score is separated by a comma.