CS 1026 - Lab 6

Topics Covered:

- Files

SUBMISSION INSTRUCTIONS

- Submit you python file for the Lab Exercise **2 (Baseball Trivia)** through the Lab Submission tool on OWL.

Pre-Lab Questions

Question 1: Which of the following will correctly open the file text.txt for reading?

a) with open("text.txt", "r") as fh:
b) with open("text.txt") as fh:
c) with open("text.txt", "read") as fh:
d) with read("text.txt") as fh:
e) with read(fh) as “text.txt”:

Question 2: Which of the following will correctly open the file mytext.txt for writing?

a) with write("mytext.txt", "w") as fh:
b) with write("mytext.txt") as fh:
c) with open("mytext.txt", "w") as fh:
d) with open("mytext.txt") as fh:

Question 3: Consider the following code segment and text file. What will be the output of the code segment?

Code Segment	text.txt
with open("text.txt", "r") as text: for line in text: print(line)	cat dog monkey

a) catdogmonkey

b) cat
dog
monkey

c) cat

dog

monkey

Question 4: Consider the following code segment and text file. What will be the output of the code segment? Recall that print with end=’’ prints a line without a newline character at the end.

Code Segment	text.txt
with open("text.txt", "r") as text: for line in text: print(line, end=’’)	cat dog monkey

a) catdogmonkey

b) cat
dog
monkey

c) cat

dog

monkey

Question 5: Consider the following code segment and text file. What will be the output of the code segment?

Code Segment	text.txt
with open("text.txt", "r") as text: for line in text: print(line.strip())	cat dog monkey

a) catdogmonkey

b) cat
dog
monkey

c) cat

dog

monkey

Question 6: Suppose a text file contains the lines:

```
Train
Car
```

If you used the readline() function immediately after opening the file, what would the string that is returned by that call?

a) ‘Train’
b) ‘Train\n’
c) ‘Train\nCar’
d) ‘Train\nCar\n’

Question 7: Suppose a text file contains the lines:

```
Train
Car
```

If you used the readline**s**() function immediately after opening the file, what would the data that is returned by that call?

a) ‘Train\nCar’
b) ‘Train\nCar\n’
c) [‘Train’,’Car’]

d) [‘Train\n’,’Car\n’]

Question 8: What is the result of the following code?

```
with open('out.txt','w') as fh:
    for i in range(0,10,2):
        fh.write('{}'.format(i))
```

a) out.txt contains a single line, with the data 02468 on the line.
b) out.txt contains five lines, with the numbers 0, 2, 4, 6 and 8 on the first, second, … , fifth lines.
c) It depends on whether out.txt exists before the code is run.

Part 2 – Lab Exercises

1. Identify errors and fix the code.

Look at the following code segment. This reads from a file called text.txt which has only **one** line. The program should read this line and split it into individual words; these words should be written to the screen **and** to the file myfile.txt with each word on its own line. There are five errors in this code; find them and correct the code.

```
with open(file.txt,'r') as inp, open('myfile.txt',w) as out:
    line = inp.read
    words = line.split()
    for word in words:
        print(word+'\n')
        out.write(word)
```

2. Construct a program that reads baseball data and answers questions.

We are going to write a program that reads a data file containing baseball statistics from 1916-2015. The data set is adapted from this page: https://www.kaggle.com/open-source-sports/baseball-databank and this data: https://www.seanlahman.com/baseball-archive/statistics/.

The data is presented as a CSV file. In the file, there is a row for each team’s result in a season. For instance, this is the row for the 1992 Toronto Blue Jays season:

yearID	lgID	franchID	G	W	L	WSWin	R	AB	H	2B	3B	HR	attendance

1992	AL	TOR	162	96	66	Y	780	5536	1458	265	40	163	4028318

This represents, for instance:

1. The team ID is TOR, and they play in the AL (one of the two leagues in Major League Baseball)
2. The team played 162 games, winning 96 and losing 66
3. The team won the world series (the championship in Major League Baseball)
4. The team had 780 runs, 5536 At bats, 1458 hits, 265 doubles, 40 triples and 163 home runs.
5. The attendance for the entire year was 4,028,318 people.

To help read the file, we define some constants for the columns:

```
YEAR = 0
LEAGUE = 1
TEAM = 2
GAMES_PLAYED = 3
GAMES_WON = 4
GAMES_LOST = 5
WON_WS = 6
RUNS = 7
AT_BAT = 8
HITS = 9
DOUBLES = 10
TRIPLES = 11
HOME_RUNS = 12
ATTENDANCE = 13

NON_INT_COLS = [LEAGUE,TEAM,WON_WS]
```

These allow us to index our data using a name instead of a column number. This will make our code more readable. You are encouraged to copy and paste these constants into your code – a text file with this code is available on the website. The list NON_INT_COLS is a list of the columns that are not integers, which will help us with data processing.

Read the data

We will read the data into a 2D list. While there is a lot of data (over 2100 entries), it can all be stored in a list for easy and sufficiently fast access.

To do this, we open the file (Teams.csv) and read the file line by line, storing it in the list (called all_data)

```
all_data = []
with open('Teams.csv','r') as fh:

    for line in fh:
       all_data.append(line)
```

This reads all the data into all_data. If you print all_data (using print or the pprint library), you will see that there are several things we need to fix:

1. The first line of the CSV is the header line, and is not data.
2. The data is read as a string, not a list.
3. The data is not stored as integers.

Let's solve these problems.

Remove the header line

To remove the header line, we can use the readline function to read one line before the loop. Since we aren't using this data, we can store it in a variable, but we won't use that variable:

```
headers = fh.readline()
```

Store the data in a list

We can use strip and split to convert the string data into a list. Instead of appending the line, we can append a list of data. Replace the for loop with this code:

```
for line in fh:
    data = line.strip().split(',')
    all_data.append(line)
```

Store data as integers

After storing the data as a list, we can see that the data is still stored as strings, not integers. We can use the NON_INT_COLS variable to identify those that do not need to be converted. Use a for loop and an if statement, after defining the data list, to convert elements in the data list to integers:

```
for i in range(len(data)):  # go through all data elements.
```

```
            if i not in NON_INT_COLS:  #column filled with an integer?
                data[i] = int(data[i])
```

We can also add an elif to convert the Y/N data for the world series:

```
             elif i == WON_WS:
                data[i] = (data[i]=='Y')
```

You should be able to convince yourself that this statement replaces ‘Y’ with True and ‘N’ with False.

Put together, the code for the loop looks like this:

```
all_data = []
with open('Teams.csv','r') as fh:
    headers = fh.readline()

    for line in fh:
        data = line.strip().split(',')

        for i in range(len(data)):
            if i not in NON_INT_COLS:
                data[i] = int(data[i])
            elif i == WON_WS:
                data[i] = (data[i]=='Y')

        all_data.append(data)
```

Answer some baseball trivia

Now that we have the data set up, we can ask some questions about the data. Write code to answer some of the following questions:

1. What team had the highest number of home runs in a season?
2. What was the total attendance at all games in 1999?
3. What team had the lowest percentage of games won (i.e., games won / total games played) and also won the world series in a season?
4. What team had the highest percentage of games won but lost the world series in a season?
5. In what four seasons did all teams play the same number of games in the season? (For this question, you can use the fact that the data is ordered by year in the file to make

the computation more efficient, or loop over the entire dataset for each year to find the data in a simpler but less efficient way.)

All of these questions require a loop over all the season results in all_data. Use the constants like HOME_RUN to index data from each season result.

Submit your solution with at least two of the questions answered, making sure to print out a readable statement that answers the question.