CS 100 Programming I
Project 1

Revision Date: October 2, 2015

Preamble

You may develop your code anywhere, but you must ensure it runs correctly under a Linux distribution before submission.

Rolling Rock Magazine!

> *Without an appreciation for grace and beauty, there's no pleasure in creating things and no pleasure in having them.* – Calvin and Hobbes

It's that time of year again. Your editor wants you to compile a list of the 10 best bands of all time. He knows you are the worldly sort and have excellent musical taste. What he doesn't know, is that for the last 6 months you have done nothing but indulge your guilty 80's pop pleasure. While you do enjoy Soft Cell's "Tainted Love", you don't think it will be making the list.

To solve your dilemma, you have decided to write a program that will help you in your task. You have two data files, one which holds a list of bands, the year the band was formed, and the number of albums sold (in millions). The other data file has a list of bands and a rating of their critical aclaim, from zero to five stars.

Here is an example from the first data file:

```
"Bob Seger And The Silver Bullet Band"   1975     32.3
"The Mudmen"                             2003     0.26
```

Here is an example from the second

```
"The Mudmen"  4
"The Who"     3.5
"Pink Floyd"  4.8
```

Your task is to merge these two lists. The output should be the band name, followed by the number of albums sold, followed by the critics' rating.

Getting the Data

Your main program should be placed in a file named *bands.py*. The program should accept two command-line arguments that represent the file name holding the albums sold list and the file name holding the ratings list.

Here is an example call:

```
python3 bands.py albums_sold.txt ratings.txt
```

The file names are just examples; you will need to invent your own data and place into files named by your choosing.

Command-line arguments

Here is a short python program that checks if there are two command-line arguments and, if so, prints out the two arguments:

```
import sys

def main():
     # check to see if there are two arguments
     # look at the length of sys.argv
     # sys.argv holds the name of the program plus the arguments

     if len(sys.argv) != 3:
        print("incorrect number of arguments, should be 2")
        sys.exit(1)

    print("the name of the program is",sys.argv[0])
    print("argument 1 is", sys.argv[1])
    print("argument 2 is", sys.argv[2])

main()
```

Strategy

Here is one strategy you could use to solve this problem:

```
created the scanner for the albums sold file
for each record r in the albums sold file
    create the scanner for the ratings file
    for each record s in the ratings file
        if there is a match between the band names of r and s,
            print out a line of data
    close the scanner for the ratings file
close the scanner for the albums sold file
```

This is not very efficient, since the ratings file is read over and over. Your task is to come up with a much more efficient strategy.

Scanner

For your program you will need to do some file I/O (Input/Output). For this you will use a common computer science tool called a *Scanner*. This will make your IO process much simpler. To get the Scanner, enter the following command.

```
wget troll.cs.ua.edu/cs100/python/projects/scanner.py
```

Typically, a scanner is used with a loop. Suppose we wish to read each token (a token is a series of characters surrounded by empty space) in a file. Here is a loop that does that:

```
s = Scanner(fileName)              #create the scanner
token = s.readtoken()              #read the first token
while token != "":                 #check if the read was good
    token = s.readtoken()          #read the next token
s.close()                          #always close the scanner when done
```

Using a scanner always means performing the five steps as given in the comments.

Of course, the code above doesn't have any output; Let's modify the code to print each token, one per line:

```
s = Scanner(fileName)
token = s.readtoken()
while token != "":
    print(token)
    token = s.readtoken()
s.close()
```

Often, adjacent tokens in a file are often related. Here is a loop that presumes each three tokens in the file are related:

```
s = Scanner(fileName)
token1 = s.readtoken()
token2 = s.readtoken()
token3 = s.readtoken()
while token1 != "":
    print(token1,token2,token3)
    token1 = s.readtoken()
    token2 = s.readtoken()
    token3 = s.readtoken()
s.close()
```

Typically, we define a function to read one collection of tokens (a collection of such tokens is known as a *record*) at a time:

```
def readRecord(s):                    # we pass the scanner in
    token1 = s.readtoken()
    if token1 == "":
        return "","",""
    token2 = s.readtoken()
    token3 = s.readtoken()
    return token1,token2,token3
```

Now our reading loop becomes:

```
s = Scanner(fileName)
token1,token2,token3 = readRecord(s)
while token1 != "":
```

```
        print(token1,token2,token3)
        token1,token2,token3 = readRecord(s)
    s.close()
```

Note that it is the job of the caller of *readRecord* to create the scanner, repeatedly call *readRecord*, and close the scanner when done.

Note also, that the scanner can read things other than tokens; see the documentation at the top of *scanner.py*.

To use a scanner, you will need to import it into your program:

```
from Scanner import *
```

You should never need to modify the scanner code. However, the scanner file contains comments that explain its use and would be worth reading.

Special Cases

If a band appears in the albums sold file but not in the ratings file, the character '*' should be printed for the band rating.

If a band appears in the ratings file but not in the albums sold file, it is ignored.

Challenges

Try to format the output so that the band name starts in column 0, the albums sold in column 50, and the rating in column 65.

Submission Instructions

Change to the directory containing your assignment. Do an *ls* command to make sure you are in the correct place. You should see, at least, your *bands.py* file. Extra files are OK. Submit your program like this:

```
submit cs100 xxxxx project1
```

Remember to replace `xxxxx` with your instructor name.