CMSC 201 Spring 2016
Homework 7 – Strings and File I/O

Assignment: Homework 7 – Strings and File I/O
Due Date: Monday, April 4th, 2016 by 8:59:59 PM
Value: 40 points

Homework 7 is designed to help you practice using file I/O, including reading to files, writing to files, and making use of string functions like `split()` to help parse the input. More importantly, you will be solving problems using algorithms you create and code yourself.

Remember to enable Python 3 before you run your programs:

```
scl enable python33 bash
```

Instructions

Each one of these exercises should be in a **separate python file**. For this assignment, you may assume that all the input you get will be of the correct type (*e.g.,* if you ask the user for a whole number, they will give you an integer).

For this assignment, you'll need to follow the class coding standards, a set of rules designed to make your code clear and readable. The class coding standards are on Blackboard under "Course Documents" in a file titled "CMSC 201 - Python Coding Standards."

You should be commenting your code, and using constants in your code (not magic numbers or strings). You should also have a function header comment for every function that is not `main()`!

Re-read the coding standards!

You will **lose major points** if you do not following the 201 coding standards.

A very important piece of following the coding standards is writing a complete **file header comment block**. Make sure that each file has a comment block at the top (see the coding standards document for an example).

NOTE: **You must use `main()` in each of your files.**

Details

Homework 7 is broken up into three parts. **Make sure to complete all 3 parts.**

NOTE: Your filenames for this homework must match the given ones exactly.
And remember, filenames are case sensitive.

You must run and test your code before submitting!
Code that does not run will lose a significant number of points.

hw7_part1.py (Worth 8 points)

Write a program to take in a telephone number that uses letters instead of numbers, such as 1-800-CALL-ATT. The program should use a function called "**`convertLetter()`**" to convert each letter to the corresponding number.

For this exercise, assume the following:

```
ABC  = 2
DEF  = 3
GHI  = 4
JKL  = 5
MNO  = 6
PQRS = 7
TUV  = 8
WXYZ = 9
```

You may want to check if the input is a digit (0-9) or not. To do this, you can use the **`isdigit()`** function. It will return **`True`** if the string passed in is a digit, or **`False`** if it is not a digit. For example, this returns **`True`**:

```
var = "5"
myBool = var.isdigit()
```

Your code should work for both lower and upper case letters. We covered converting to upper or lower case in Lecture 7 (Strings and Lists).

NOTE:
You do not need use constants for ***this part of the homework only***. You may use "magic numbers" like `7, "A", 0, "Q",` etc. Again, this applies to Part 1 of Homework 7 ***only***.

Here is some sample output for **`hw7_part1.py`**, with the user input in blue.

```
bash-4.1$ python hw7_part1.py
Welcome to the Telephone Converter
Enter the phone number: 1-800-LUV-UMBC
1-800-588-8622

bash-4.1$ python hw7_part1.py
Welcome to the Telephone Converter
Enter the phone number: 1-410-cookies
1-410-2665437
```

hw7_part2.py **(Worth 12 points)**

For this exercise, you are going to build an application that works as a currency exchange. We are going to convert from US Dollars to Retriever Bux, and from Retriever Bux to US Dollars.

In this exercise, use the following constant exchange rates:
1 US Dollar ($) = 8 Retriever Bux
1 Retriever Bux = 0.125 US Dollars ($)

Some expectations for this assignment:

1. You must use the simple menu system shown in the output to allow the users to choose what they would like to do.
2. You must use a function (or functions) to do the conversion.
3. You must check to make sure that the user enters a 1, 2, or 3 for their choice. You should reprompt if they enter invalid choices.
4. You must use the values for conversion listed above.
5. You should use the format function shown below to limit the output to two decimal places
6.

```
dollars = 18.125
dollars = format(dollars, '.2f')
# dollars will now print out as "18.12"
print(dollars)
```

Make sure that you use are using constants, and not magic numbers, in your programs!

Sample output for this problem can be found on the next page.

Here is the **sample output** for hw7_part2.py, with the user input in blue.

```
bash-4.1$ emacs hw7_part2.py
bash-4.1$ python hw7_part2.py
Welcome to the Currency Converter
What would you like to do?
1. Convert US Dollars to Retriever Bux
2. Convert Retriever Bux to US Dollars
3. Exit
Enter your choice: 1
How much do you want to convert?: 22.75
22.75 US dollars equates to 182.00 Retriever Bux
Good bye, and thank you for using the Currency Converter

bash-4.1$ python hw7_part2.py
Welcome to the Currency Converter
What would you like to do?
1. Convert US Dollars to Retriever Bux
2. Convert Retriever Bux to US Dollars
3. Exit
Enter your choice: 2
How much do you want to convert?: 1014
1014.00 Retriever Bux equates to 126.75 US dollars
Good bye, and thank you for using the Currency Converter

bash-4.1$ python hw7_part2.py
Welcome to the Currency Converter
What would you like to do?
1. Convert US Dollars to Retriever Bux
2. Convert Retriever Bux to US Dollars
3. Exit
Enter your choice: 0
That is an invalid choice.
Enter your choice: 14
That is an invalid choice.
Enter your choice: 3
Good bye, and thank you for using the Currency Converter
```

hw7_part3.py **(Worth 16 points)**

*(*WARNING! *This part of the homework is the **most challenging**, so budget plenty of time and brain power. And read the instructions carefully!)*

Finally, you will write a program that takes in the name of a file from the user and calculates the total number of words in the file, the average word length, and the total number of sentences.

For this part of the homework, the filename you get from the user must be **checked for validity**: the filename must end in either "**`.dat`**" or "**`.txt`**" in order to be considered a valid filename for this program. If the user inputs an invalid filename (*e.g.*, "**`book.doc`**") you must continue to reprompt them until they give a valid filename.

You may **not** use built-in Python methods, such as **`endswith()`**, to check if the string ends with the correct "**`.dat`**" or "**`.txt`**".

(HINT: If an invalid filename is given, your program should also tell the user what a valid filename looks like – see the sample output for an example.)

Once you have a valid filename from the user, you should open the file and count the total number of words in the file, calculate the average word length, and count the total number of sentences in the file.

You may assume:

- A filename that ends in "**`.dat`**" or "**`.txt`**" will successfully open a file when used with the function **`open()`**
- A "word" in a file is any set of characters separated by **whitespace**
  - For example, "**`copy-right 1977 by author of book`**" would be six words ("**`copy-right`**" is a single word, and the number "**`1977`**" counts as a word). Tabs and newlines count as whitespace. You don't need to do any checking of word content.
- A sentence in a file is any set of characters separated by a **period**.

(HINT: Make sure to open the file for reading, and to close it when you're done.)

Sample output for this problem can be found on the next page.

PROTIP: **This would be a good time to use incremental programming!**
Incremental development is when you are only working on a small piece of the code at a time, and testing that the piece of code works before moving on to the next piece. This makes it a lot easier to fix any mistakes.
For example, for this problem, you might first write the code to get a valid filename from the user, and test that this works before moving on. Next, you might write the code to count the number of words, and test that this works before moving on. Then, you might write the code to calculate the average word length, and test that this works before moving on. etc...

Here is the **sample output** for hw7_part3.py, with the user input in blue. Your output does not need to be identical, but should be similar.

```
bash-4.1$ python hw7_part3.py
Please enter the name of the file to open: book.text
        The file must end in .txt or .dat to be valid.
        Please enter the name of the file to open: myFile.doc
        The file must end in .txt or .dat to be valid.
        Please enter the name of the file to open: zimmer.txt
The file zimmer.txt has 48925 words in it.
On average, each word is 4.679979560551865 characters long.
There are 3347 sentences in the file.
```

The **sample input file** that was used to create the sample output above, **`zimmer.txt`**, is much too long to include in this document. However, you can **directly download the file** using the "**`cp`**" command.

The command below will copy the file "**`zimmer.txt`**" from Professor Gibson's public directory to your current directory. The period at the end ("**`.`**") means that the file will have the same name after you copy it, so **`zimmer.txt`** will be the copied file's name. Make sure to run the command from the folder you want the file to be copied into!

```
cp /afs/umbc.edu/users/k/k/k38/pub/cs201/zimmer.txt .
```

Submitting

Once all three parts of your Homework 7 are complete, it is time to turn them in with the **`submit`** command.

Don't forget to complete the header block comment for each file! Make sure that you updated the header block's file name and description for each file.

You must be logged into your GL account, and you must be in the same directory as the Homework 7 files. To double check this, you can type **`ls`**.

```
linux1[3]% ls
hw7_part1.py  hw7_part2.py  hw7_part3.py
linux1[4]%
```

To submit your files, we use the **`submit`** command, where the class is **`cs201`**, and the assignment is **`HW7`**. Type in (all on one line)
`submit cs201 HW7 hw7_part1.py hw7_part2.py hw7_part3.py`
and press enter.

```
linux1[4]% submit cs201 HW7 hw7_part1.py hw7_part2.py
hw7_part3.py
Submitting hw7_part1.py...OK
Submitting hw7_part2.py...OK
Submitting hw7_part3.py...OK
linux1[5]%
```

If you don't get a confirmation like the one above, check that you have not made any typos or errors in the command.

You can **double-check that all three homework files were submitted** by using the **`submitls`** command. Type in **`submitls cs201 HW7`** and hit enter.

And you're done!