UMBC

AN HONORS UNIVERSITY IN MARYLAND

CMSC 201 Spring 2019
Homework 5 – Functions and Strings

Assignment: Homework 5 – Functions and Strings
Due Date: Friday, March 15th, 2019 by 11:59:59 PM
Value: 40 points

Collaboration: For Homework 5, collaboration is allowed. Make sure to consult the syllabus about the details of what is and is not allowed when collaborating. You may not work with any students who are not taking CMSC 201 this semester.

If you work with someone, remember to note their name, email address, and what you collaborated on by filling out the Collaboration Log.

You can find the Collaboration Log at http://tinyurl.com/spring19-201-collab

Remember that all collaborators need to fill out the log each time; even if the help was only "one way" help.

Make sure that you have a complete file header comment at the top of each file, and that all of the information is correctly filled out.

```
# File:    FILENAME.py
# Author:  YOUR NAME
# Date:    THE DATE
# Section: YOUR DISCUSSION SECTION NUMBER
# E-mail:  YOUR_EMAIL@umbc.edu
# Description:
#    DESCRIPTION OF WHAT THE PROGRAM DOES
```

Instructions

For each of the questions below, you are given a problem that you must solve or a task you must complete.

This assignment will focus on using functions to make code that is more modular. Some of these functions may be useful in later assignments!

At the end, your Homework 5 files must run without any errors.

NOTE: Your filenames for this homework must match the given ones exactly.

And remember, filenames are case sensitive!

Additional Instructions – Creating the hw5 Directory

During the semester, you'll want to keep your different Python programs organized, organizing them in appropriately named folders (also known as directories).

Just as you did for previous homeworks, you should create a directory to store your Homework 5 files. We recommend calling it **`hw5`**, and creating it inside the **`Homeworks`** directory inside the **`201`** directory.

If you need help on how to do this, refer back to the detailed instructions in Homework 1. (You don't need to make a separate folder for each file. You should store all of the Homework 5 files in the same **`hw5`** folder.)

Coding Standards

Prior to this assignment, you should re-read the Coding Standards, available on Blackboard under "Assignments" and linked on the course website at the top of the "Assignments" page.

For now, you should pay special attention to the sections about:

- Comments
  - ***Function header comments***
  - We've given a few of the function headers as examples, but you will need to create the rest on your own
- Constants
  - For Homework 5, you must use constants instead of magic numbers!!! Magic strings are also forbidden!!!!!!
- Make sure to **read the last page of the Coding Standards document**, which prohibits the use of certain tools and Python keywords
  - Also, this section states that **you may not use built-in functions** or concepts we haven't covered in class!
    If you use a built-in function to solve a problem, you will earn **zero points** for that entire problem.

Additional Specifications

For this assignment, **you must use `main()`** as seen in your **`lab2.py`** file, and as discussed in class.

For this assignment, you should pay attention to each problem's instructions on using "input validation." For example, the user may enter a negative value, but your program may require a positive value. **Make sure to follow each part's instructions about input validation.**

If the user enters a different type of data than what you asked for, your program may crash. This is acceptable.

You must create the functions specified in each problem, and they must be named ***exactly*** as shown.

Questions

Each question is worth the indicated number of points. Following the coding standards is worth 4 points. If you do not have complete file headers and correctly named files, you will lose points.

hw5_part1.py **(Worth 9 points)**

For this part of the homework you will create a program that asks the user for their name, and then prints their initials. You must create a function called **`getInitials()`** that takes the name string and prints out the initials. It must be case *in*sensitive and the initials must be printed in upper case.

The program must contain a **`main()`** and a function called **`getInitials()`**, implemented as described in the function header comment given below. (You should include this function header comment in your own code.)

```
##############################################################
# getInitials() counts the instances of a letter in a string
# Parameters:   name; a string containing the name
# Return:       None
```

Here is some sample output, with the user input in **blue**.
(Yours does not have to match this exactly, but it should be similar.)

```
linux[0]$ python hw5_part1.py
Enter your name: Dr. Freeman A. Hrabowski, III
Your initials are D.F.A.H.I.

linux[0]$ python hw5_part1.py
Enter your name: supreme cat king Ben Johnson
Your initials are S.C.K.B.J.

linux[0]$ python hw5_part1.py
Enter your name: Viceroy Charlie Dog
Your initials are V.C.D.
```

hw5_part2.py (Worth 6 points)

You will create a program to count how many words in a sentence look like they are in past tense, i.e. end in "-ed".

The program must contain a **`main()`** and a function called **`countPastTense()`**, implemented as described in the function header comment given below. (You should include this function header comment in your own code.)

```
############################################################
# countPastTense() counts past tense words
# Parameters:      phrase; a string to count past tense words
# Return:          none
```

Here is some sample output, with the user input in **blue**.
(Yours does not have to match this word for word, but it should be similar.)

```
linux[0]$ python hw5_part2.py
Tell me something: This show has jumped the baby shark
There appear to be 1 past tense words in that.


linux[0]$ python hw5_part2.py
Tell me something: All those red hermit crabs attacked me
when I fed them
There appear to be 3 past tense words in that.


linux[0]$ python hw5_part2.py
Tell me something: Periods make the word be ignored.
There appear to be 0 past tense words in that.
```

hw5_part3.py **(Worth 6 points)**

For this part of the homework, you will write code to search through a phrase, looking for occurrences of a word. Each time an occurrence is found, the program must report the index at which the word starts as well. After searching, it must print out the total number of times word was found.

The program must be case *in*sensitive, and must use a function called **`inPhrase()`** that takes in the phrase and word as its two formal parameters, and returns nothing. All of the searching and printing of results must happen inside the **`inPhrase()`** function.

Inside **`main()`**, the program does need to ensure that the word is shorter than the phrase, re-prompting the user as necessary. You may assume that both inputs will be at least 1 character.

Here is some sample output, with the user input in **blue**.
(Yours does not have to match this exactly, but it should be similar.)

```
linux[0]$ python hw5_part3.py
Please enter a phrase: Dogs are good, dogs dogs dogs
Please enter a word to search for: DOGS
Found DOGS at index 0
Found DOGS at index 15
Found DOGS at index 20
Found DOGS at index 25
Found DOGS a total of 4 times

linux[0]$ python hw5_part3.py
Please enter a phrase: Hello!
Please enter a word to search for: Good morning
The word cannot be longer than the phrase.
Please enter a shorter word to search for: Goodbye
The word cannot be longer than the phrase.
Please enter a shorter word to search for: ello
Found ello at index 1
Found ello a total of 1 times
```

hw5_part4.py **(Worth 9 points)**

Create a program that translates English into Pig Latin.

The function will take the first letter of each word in the sentence only if it's a not a vowel, and place it at the end of the word followed by "ay".

Your program must be case *in*sensitive. You must write the function **`translate()`** that takes in a single English word and returns its Pig Latin translation. Remember translate must take *only one word as input*.

```
##############################################################
# translate() takes a single english word translates it to
#               pig latin
# Parameters: english_word; an English word
# Return:     the pig latin translation
```

Here is some sample output, with the user input in **blue**.
(Yours does not have to match this word for word, but it should be similar.)

```
linux[0]$ python hw5_part4.py
Enter an English phrase: You look nice today
I think you meant to say: Youay ooklay icenay odaytay

Enter an English phrase: Again we don't have to worry
about punctuation.
I think you meant to say: Againay eway on'tday avehay otay
orryway aboutay unctuation.pay

Enter an English phrase: I realized that this problem
isn't as easy as I had thought.
I think you meant to say: Iay ealizedray hattay histay
roblempay isn'tay asay easyay asay Iay adhay hought.tay

Enter an English phrase: Owls have 14 vertebrae in their
necks.  So fancy!
I think you meant to say: Owlsay avehay 41ay ertebraevay
inay heirtay ecks.nay oSay ancy!fay
```

hw5_part5.py **(Worth 6 points)**

Write a program that, within **`main()`**,

1. Asks for the number of words the user will enter into the program
2. Then prompts for each of the words (and stores them in a list)

Once all of the words have been entered, the program must print them out in reverse order from how they were entered, and must also show what the word would be backwards. (See the sample output for an example.)

The program must use a function called **`backwards()`**, that works as specified in the function header provided below.

Note that this function does not print out any information, but returns the reversed string to the **`main()`** function that called it. The printing out of the strings will need to occur in **`main()`**.

```
######################################################
# backwards() reverses a string and returns the result
# Parameters: forString;  a string to reverse
# Return:     backString; the reversed string
```

(Again, do not use any built-in function or "trick" that circumvents the point of this assignment, or you will earn zero points. If you're not using a loop to create the backwards string, you're doing it wrong.)

(See the next page for sample output.)

Here is some sample output for **hw5_part5.py**, with the user input in **blue**. (Yours does not have to match this word for word, but it should be similar.)

```
linux[0]$ python hw5_part5.py
How many words would you like to turn backwards: 5
Please enter string #1: dog
Please enter string #2: bird
Please enter string #3: horse
Please enter string #4: fish
Please enter string #5: llama
The string 'llama' reversed is 'amall'.
The string 'fish' reversed is 'hsif'.
The string 'horse' reversed is 'esroh'.
The string 'bird' reversed is 'drib'.
The string 'dog' reversed is 'god'.

linux[0]$ python hw5_part5.py
How many words would you like to turn backwards: 3
Please enter string #1: Kayak
Please enter string #2: Racecar
Please enter string #3: Stats
The string 'Stats' reversed is 'statS'.
The string 'Racecar' reversed is 'racecaR'.
The string 'Kayak' reversed is 'kayaK'.

linux[0]$ python hw5_part5.py
How many words would you like to turn backwards: 0

linux[0]$ python hw5_part5.py
How many words would you like to turn backwards: 1
Please enter string #1: step on NO pets
The string 'step on NO pets' reversed is 'step ON no
pets'.
```

(Some of these examples are palindromes, which means they are the same backwards and forwards. If you pay attention to the capitalization, the reversal becomes a bit clearer.)

Submitting

Once your **`hw5_part1.py`**, **`hw5_part2.py`**, **`hw5_part3.py`**, **`hw5_part4.py`**, and **`hw5_part5.py`** files are complete, it is time to turn them in with the **`submit`** command. (You may also turn in individual files as you complete them. To do so, only **`submit`** those files that are complete.)

You must be logged into your account on GL, and you must be in the same directory as your Homework 5 Python files. To double-check you are in the directory with the correct files, you can type **`ls`**.

```
linux1[3]% ls
hw5_part1.py  hw5_part3.py  hw5_part5.py
hw5_part2.py  hw5_part4.py
linux1[4]% 
```

To submit your Homework 5 Python files, we use the **`submit`** command, where the class is **`cs201`**, and the assignment is **`HW5`**. Type in (all on one line) **`submit cs201 HW5 hw5_part1.py hw5_part2.py hw5_part3.py hw5_part4.py hw5_part5.py`** and press enter.

```
linux1[4]% submit cs201 HW5 hw5_part1.py hw5_part2.py
hw5_part3.py hw5_part4.py hw5_part5.py
Submitting hw5_part1.py...OK
Submitting hw5_part2.py...OK
Submitting hw5_part3.py...OK
Submitting hw5_part4.py...OK
Submitting hw5_part5.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 check that your homework was submitted by following the directions in Homework 0. Double-check that you submitted your homework correctly, since **an empty file will result in a grade of zero for this assignment.**