Published

Week 10: Strings 6 of 6

10 Week 10 Assignment

Computer SCIENCE

IMPORTANT: Please review the statement on Academic Integrity. Submitting code that is not your own may result in an automatic F for the class.
Statement on Academic Integrity

Objectives

1. Classes
2. Objects
3. Operator Overloading

Task 1 - Wordinator

Starter: **unit5_task3_starter.py** (https://usu.instructure.com/courses/681553/files/81729325/download?download_frd=1)

You will create a program that will generate specific output based on two words. You will design, implement, and use a Wordinator object that will perform the specific operations. Make sure to watch the video and read the description below carefully. Many parts of this are simple. There is one part that will take some thinking to get implemented properly. Note the sections where you must do things a specific way. In other words, you will not be able to use specific operators or functions, and must program the solution in another way (manually).

Use the starter code file unit5_task3_starter.py. Modify the name so it is called task1.py. Your plan should be created in plan1.txt. It must include the UML and design for the Wordinator class. You do not need to design the code that will be in task1.py. Create another file called wordinator.py to create your Wordinator class. Meet the following requirements.

As always, please see rubric for grading details.

Video Demo

Requirements

1. task1.py
    1. The code in the file should not change from what is given in unit5_task3_starter.py.
    2. You should become familiar with how the menu system works. It will help you on a future assignment.
2. wordinator.py
    1. Use appropriate private data and method members. Items not created as private that should be will lose points.
    2. Create a constructor that takes a single word as a parameter
    3. Write appropriate methods to meet all requirements as indicated in the user menu. Details of each are below
3. Menu Item Output(use as a guide to determining method requirements)
    1. First Word
        1. Prints the word that comes first **alphabetically**.
        2. Note that alphabetically is bolded.
    2. Big Word
        1. Prints the two words into one big concatenated word.
        2. The first letter should be capitalized with all others lowercase.
        3. NOTE: There is an inconsistency in the video. Match the requirements here for your submission.
    3. Words Factor
        1. Ask the user for an integer factor value

2. Print each words repeated the integer factor number of times.
3. All letters should be uppercase
4. NOTE: There is an inconsistency in the video. Match the requirements here for your submission.

4. Mix Words
    1. Print a new word where each sequential letter is from the other word.
    2. Example:
        1. Word1: Apple
        2. Word2: Banana
        3. MixedWord: Abpapnlaena
    3. If one word is longer, then it's letters do not alternate after the other word's letters are used up
    4. The first letter should be capitalized with all others lowercase.
5. Middle Words
    1. This one will take some thought. Please read this carefully, especially #6.
    2. Print the middle portion of each of the two words
    3. The middle portion is approximately half of the length of the entire word
    4. The number of characters that come before and after the middle portion should be equal
    5. The length of the middle word must meet the following requirements:
        1. Difference between the number of middle characters and outside characters cannot be greater than two
        2. The number of characters in the middle cannot be more than one greater than the number of outside characters
    6. Hint: If you can meet the previous requirements for words of length 7, 8, 9, and 10, you will be set.
    7. You may not have duplicate code to calculate this for each word. This should make you think of code reuse.
6. Switch Case
    1. You may not use the switchcase() method, or any other method that swaps case automatically.
    2. Print each of the words with the case of each letter swapped from the original
    3. You may not have duplicate code to calculate this for each word.
7. Back Words Slice
    1. Print each word backwards
    2. You must use the slice operator to reverse the ordering.
        1. In class we didn't discuss the entire capabilities of the slice operator.
        2. The full description could be notated as [start:end:step]
        3. The step is very similar to the step in the range function. (hint hint)
    3. Each letter should maintain its original capitalization.

8. Back Words Manual
    1. Print each word backwards
    2. You MAY NOT use the slice operator or any other method that reverses a string automatically. A loop might help.
    3. Each letter should maintain its original capitalization.

Common Errors

Pay attention to the starter code, and how it uses the Wordinator objects for each menu item. This gives you insight in to how to design each associated method. One common error is to not realize that this task is an exercise in operator overloading for many of the menu items.

Points	100
Submitting	a text entry box or a file upload
File Types	zip

Due	For	Available from	Until
Nov 12, 2021	Everyone	-	-

Unit 5 (2)

Criteria	Ratings		Pts
Task 3: Software Design Plan Only needed for Wordinator class. Include UML as well	**5 pts** **Full Marks**	**0 pts** **No Marks**	5 pts
Task 3: starter file not modified Only the name of the file should be changed.	**5 pts** **Full Marks**	**0 pts** **No Marks**	5 pts
Task 3: Appropriate private data and methods All that should be private are private.	**12 pts** **Full Marks**	**0 pts** **No Marks**	12 pts
Task 3: Constructor takes 1 word as parameter initializer method is constructed properly	**6 pts** **Full Marks**	**0 pts** **No Marks**	6 pts
Task 3: First Word Menu Item Appropriate method implemented and performs properly	**8 pts** **Full Marks**	**0 pts** **No Marks**	8 pts
Task 3: Big Word Menu item Appropriate method implemented and performs properly	**8 pts** **Full Marks**	**0 pts** **No Marks**	8 pts
Task 3: Words Factor Menu item Appropriate method implemented and performs properly	**8 pts** **Full Marks**	**0 pts** **No Marks**	8 pts
Task 3: Mix Words Menu item Appropriate method implemented and performs properly	**12 pts** **Full Marks**	**0 pts** **No Marks**	12 pts
Task 3: Middle Words Menu item Appropriate method implemented and performs properly	**15 pts** **Full Marks**	**0 pts** **No Marks**	15 pts
Task 3: Switch Case Menu item May NOT use a built-in function or any method that does this automatically. Must code it manually. May NOT have duplicate code to calculate for each word.	**7 pts** **Full Marks**	**0 pts** **No Marks**	7 pts

Criteria	Ratings		Pts
Task 3: Back Words Slice Menu item Method implemented and called properly. Should use the slice operator.	**7 pts** **Full Marks**	**0 pts** **No Marks**	7 pts
Task 3: Back Words Manual Menu item Method implemented and called properly. Cannot use the slice operator or any other method/function that reverses a string automatically. You must do it manually. May not have duplicate code to calculate for each word.	**7 pts** **Full Marks**	**0 pts** **No Marks**	7 pts
			Total Points: 100