Computer Code for Beginners

Week 5

Letters in a String

Write a program to count the amount of each letter present in a string and print these values – for example a = 7, b = 4, etc.

- Open up the `lettersInString.py` file
- Start by using the 'hard-coded' `string` and then try asking the user for a string
- For each character in the string, we have two basic cases:
  - if the character is in the `letters` dictionary, then we need to get the number that it maps to (the current number of times that letter appears in the string) and add 1 to it
  - If the character is not in the `letters` dictionary, we can simply add it, mapping to 1

Challenge:

- Capital and lower case letters should be treated as the same letter.
- We don't want to include characters that aren't alphabetic. Using `s.isalpha()` returns true if the string `s` contains only alphabetic characters
- Once you've added these more challenging features, alter the string you use, so that it includes some mixed case letters and non-alphabetic characters to test the new features

Morse Code

Write a program to convert a string to Morse Code. The `morseCode.py` module contains a dictionary that maps characters to their Morse Code equivalent.

As well as the characters in the dictionary:

- Letters must be separated by three spaces
- Words must be separated by seven spaces

Implement the `textToMorse(string)` function so that it accepts a string of plain text as a parameter and returns a corresponding string of Morse Code.

When writing this program, you may find it useful to start by writing code that translates one letter, then one word, then a string of several words. Make sure you run your program and test it at each stage.

- Translating a letter is as simple as finding what that letter maps to in the `Morse` dictionary
- Translating a word requires you to loop through the word,translate each letter, and add three spaces between each letter
- Translating a string of several words requires you to loop through each word in the string, translate each word, and add seven space between each word

Remember that a string may be made up of several words:

- Using `s.split("")` returns a list of the words in the string `s`, using a space (`" "` to decide where one word ends and another begins
  - For example if `s = "Octopus Pie"` then using `s.split("")` returns the list `["Octopus ", "Pie"]`
- This will be useful to you in finding all the words in the plain text string

- Be careful to:
  - Deal with a plaint text string of mixed case (upper and lower case)
  - Only add these separators between words or letters, and not to the end of the translation
  - Check for characters that aren't in the dictionary's keys

Testing

Come up with some translations to test your function, `morseCode.py` contains one already

Challenge:

- Once you've finished the function that translates text to Morse Code, implement the `morseToText(string)` function so that it reverses the translation (returns the string translation of a Morse code parameter)
  - You'll need to change what you use split the input parameter and split each word to get each letter
  - You need to implement the small `reverseDict(dictionary)` function to reverse the dictionary (swap the keys and values) to help with this.
    - Have a go at it, but the function is available on the website if you need it
    - If you ise the version on the website, you can simply import it using `from dictionaryReverse import reverseDict`
  - The `morseCode.py` file contains two large strings of Morse to use to test your `morseToText()` function.