CSC401- Assignment 2

Jay Tang

Due Wednesday, April 20, 11:59pm

Reading

Read **Chapter 4 and 7.3** in Introduction to Computing using Python: An Application Development Focus, Second Edition by Ljubomir Perković.

Logistics

You need to do this assignment on a computer which has Python 3 installed on it. Python 3.10 download page can be found here.

You are encouraged to work with your classmates on the assignments. If you do work with someone on the assignments, please include the name of your collaborators at the top of the file you submit. If you worked alone, please indicate that at the top of your submission. **A submission without collaboration information will not receive credit.**

A submission that includes code which does not run will not get any points for the part unless specifically documented reason of the error.

Submission

Submit the assignment using Assignment 2 folder. Submit only a **single python file** using your name as file name (e.g., Jay_Tang_Assign_2.py).

This assignment is due Wednesday, April 20, 11:59pm. Submissions after the deadline will be automatically rejected by the system.

Assignment

1. Decisions, loops and stings (10pt)

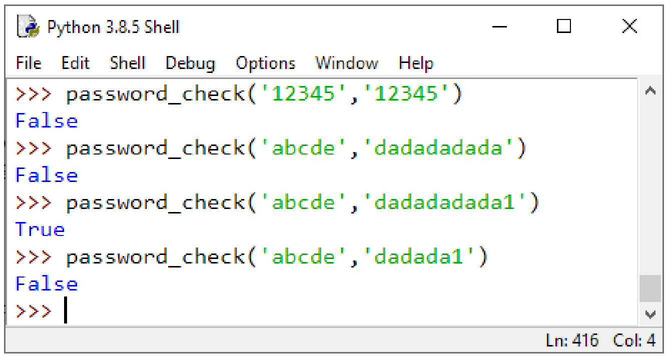
Write a function that takes a list of strings as a parameter, say `['Antheil', 'Saint-Saens', 'Price', 'Easdale', 'Nielsen']`, and prints the strings where last letter is the same as the first letter. (so Saint-Saens, Easdale and Nielsen should get listed, but not the others.) Note that the case of the letters should not matter. You will do a case insensitive comparison this time.

2. Strings (30pt)

a. (15pt) Write a function `password_check(oldpwd, newpwd)` that has two parameters, `newpwd` and `oldpwd`, which represent a new password and the old password. The function accepts the new password (returns `True`) if the `newpwd` fulfills all of the following conditions:

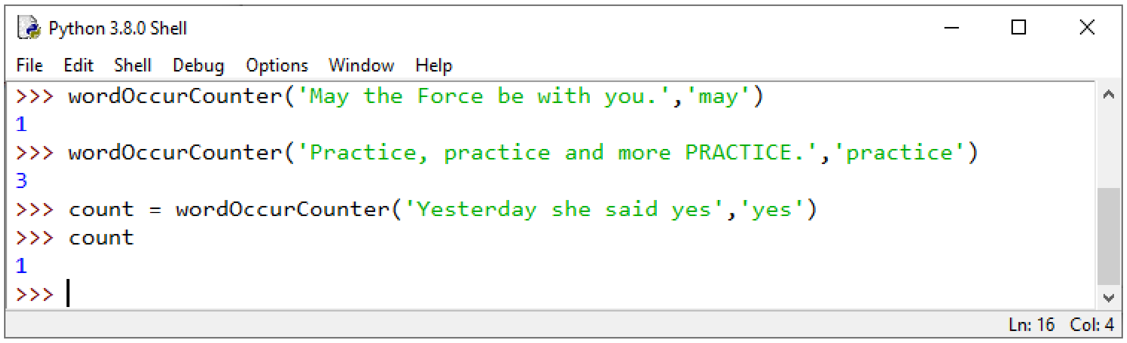
   i. `newpwd` is different from `oldpwd`
   ii. `newpwd` is at least 8 letters long
   iii. `newpwd` contains some character which is not a letter (i.e. a number, or some special character). *Hint*: research the string type using `help(str)` to look for appropriate tools to do this.

   If the new password fails the check, your function should return `False`.



```
>>> password_check('12345','12345')
False
>>> password_check('abcde','dadadadada')
False
>>> password_check('abcde','dadadadada1')
True
>>> password_check('abcde','dadada1')
False
>>> |
```

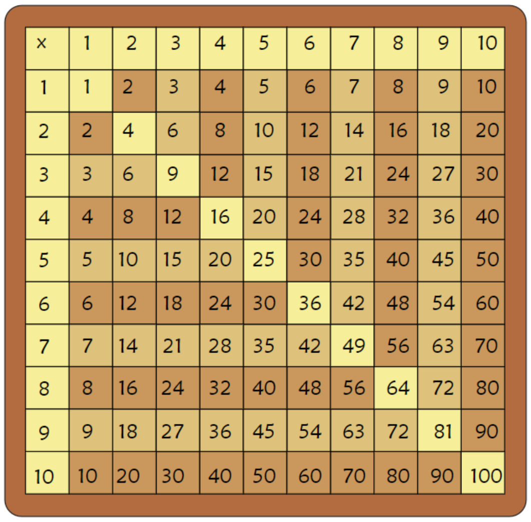
b. (15pt) Implement a function `wordOccurCounter(sentence, word)` that has two parameters: `sentence` representing a sentence and `word` representing a word, and returns the number of times the word appears in the sentence. Please make sure to remove any punctuation from the "sentence" before counting words. For the purposes of this question, punctuation is restricted to a comma and a period. If either parameter string is empty the function should return 0. It should not be case sensitive when counting words, i.e., capitalization of the word of the sentence should not make any difference. The function should not consider substrings when checking for the word. The following shows several examples of how the function could be used:



```
>>> wordOccurCounter('May the Force be with you.','may')
1
>>> wordOccurCounter('Practice, practice and more PRACTICE.','practice')
3
>>> count = wordOccurCounter('Yesterday she said yes','yes')
>>> count
1
>>> |
```

3. Print (20pt)

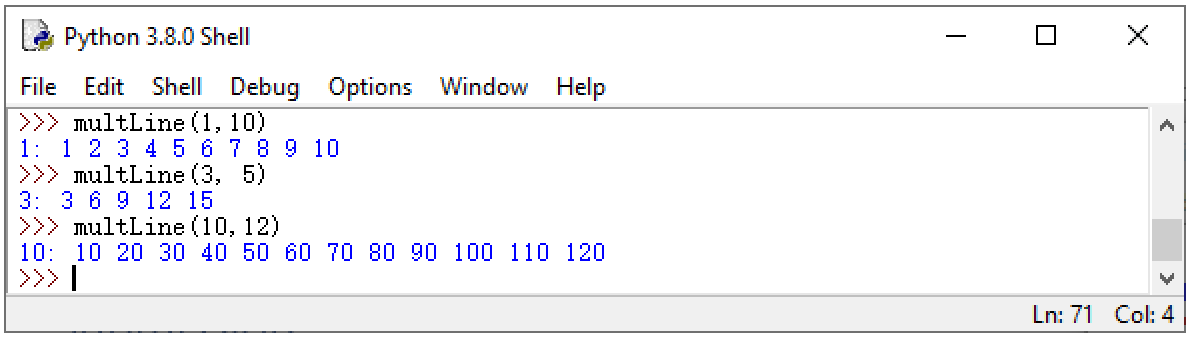
We have seen a multiplication chart like this



x	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	2	4	6	8	10	12	14	16	18	20
3	3	6	9	12	15	18	21	24	27	30
4	4	8	12	16	20	24	28	32	36	40
5	5	10	15	20	25	30	35	40	45	50
6	6	12	18	24	30	36	42	48	54	60
7	7	14	21	28	35	42	49	56	63	70
8	8	16	24	32	40	48	56	64	72	80
9	9	18	27	36	45	54	63	72	81	90
10	10	20	30	40	50	60	70	80	90	100

It could be 1x to 10x, but sometimes we can stretch it to 12x or more.

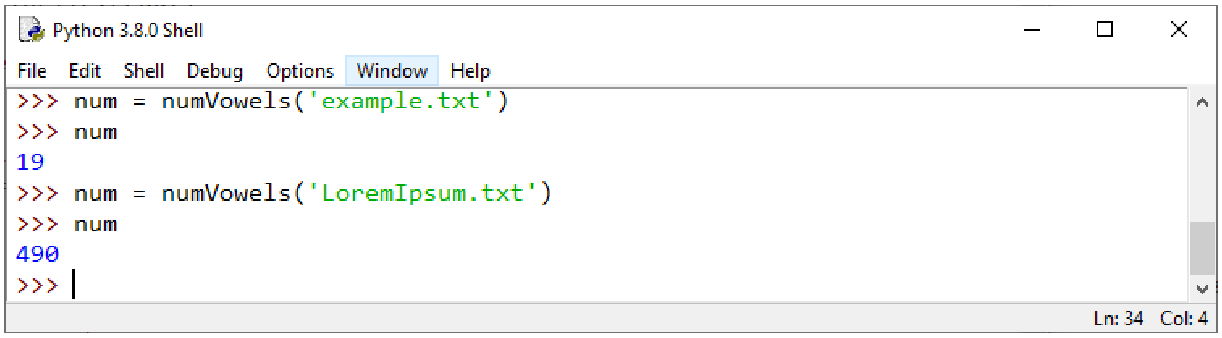
Implement a function `multLine(line, numCount)` that takes two parameters: first one indicating which line of the multiplication chart to print and second one deciding how many numbers to multiply. First print the line number then followed by multiples.



```
>>> multLine(1, 10)
1:  1 2 3 4 5 6 7 8 9 10
>>> multLine(3,  5)
3:  3 6 9 12 15
>>> multLine(10, 12)
10:  10 20 30 40 50 60 70 80 90 100 110 120
>>> 
```

4. File I/O (40pt)

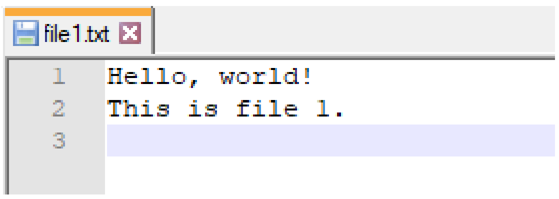
a. (20pt) Write a function `numVowels(filename)` that has one parameter `filename` representing a file name and returns the number of vowels (case insensitive) that appear in the file. For the purposes of this question a vowel is one of a, e, i, o, or u.



```
>>> num = numVowels('example.txt')
>>> num
19
>>> num = numVowels('LoremIpsum.txt')
>>> num
490
>>> 
```

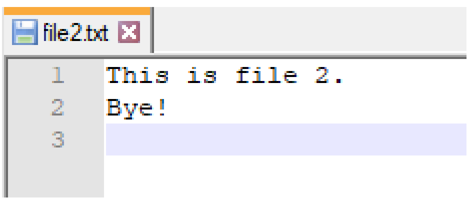
b. (20pt) Implement the function `merge(file1, file2)` that takes two strings as parameters: both represent filenames. The function will merge the two files. Content in the second file(`file2`) will be appended to the end of the first file(`file1`). Below is the simple example of how it works.

Before merge:



```
Hello, world!
This is file 1.

```



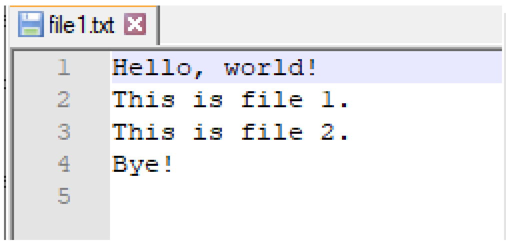
```
This is file 2.
Bye!

```

Call `merge()` function

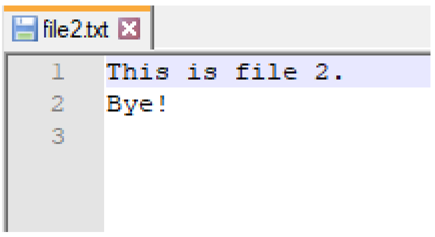
```
>>> merge('file1.txt', 'file2.txt')
```

After merge:



```
Hello, world!
This is file 1.
This is file 2.
Bye!

```



```
This is file 2.
Bye!

```