SSEP 2024: Intro to Programming with Python

Summer 2024

Assignment 04: Encryption and Decryption

*This is a **pair programming assignment find another person to work with on the assignment.***

Assignment Goals:

- Define functions
- Call functions
- Debug functions

Notes

It is your responsibility to write a program that is well structured (i.e. modular), and easy to read. Your file should start with a header that has your name(s), the date, and a brief description of what your program does. Variable names should be descriptive. Comments should be used appropriately to document your code.

Quick Programming Tips:

- ***Not sure what to do?*** Talk through the assignment instructions with your partner and write out in English what specific tasks your program needs to do. Then, pseudocode.
- ***Stuck on how to program your solution?*** Try out 2 different ideas then ask for help if you are still stuck!
- ***Have a seemingly invisible bug?*** Use print statements throughout your code. Before running the code, think about what you expect to print if the code is working. Then see if what you expected is what prints.

Program Specification (AKA Spec)

For this assignment, you will write four functions for encrypting and decrypting strings.

Your four functions should meet the following specifications:

Function 1

- Name: vowelSwap
- Parameter(s): string
- Returns: string
- Does: When this function is called it encrypts the string it is called with by replacing all instances of "a" with "*", all instances of "e" with "#", all instances of "i" with "<", all instances of "o" with "$", and all instances of "u" with "@". The function can be called with a string of any case but the encrypted string it returns should be in all lowercase.

Function 2

- Name: unVowelSwap
- Parameter(s): string
- Returns: string
- Does: When this function is called it decrypts the string it is called with assuming it was encrypted with vowelSwap. The string the function returns should be in all lowercase.

Function 3

- Name: reverseSentence
- Parameter(s): string
- Returns: string
- Does: When this function is called, it checks how many words are in the string it was called with. If it was called with a string that is at least two words it encrypts the string by reversing the words in it. If the string it is called with is less than 2 words, it moves the first letter of the string to the end of the string and the last letter of the string to the beginning of the string. The string the function returns should have the same casing as the string the function was called with.

Function 4

- Name: unReverseSentence
- Parameter(s): string
- Returns: string
- Does: When this function is called it decrypts the string it is called with assuming it was encrypted with reverseSentence. The string the function returns should have the same casing as the string the function was called with.

Example encryption and decryption for several strings are shown below.

```
##############################
Original String:
I Love computer ScIeNcE
Vowel Swap Encrpytion:
< l$v# c$mp@t#r sc<#nc#
Vowel Swap Decryption:
i love computer science
Reverse Sentence Encryption:
ScIeNcE computer Love I
Reverse Sentence Decryption:
I Love computer ScIeNcE
##############################
Original String:
Hello
Vowel Swap Encrpytion:
h#ll$
Vowel Swap Decryption:
hello
Reverse Sentence Encryption:
oellH
Reverse Sentence Decryption:
Hello
##############################
Original String:
Oh wow this is FANCY
Vowel Swap Encrpytion:
$h w$w th<s <s f*ncy
Vowel Swap Decryption:
oh wow this is fancy
Reverse Sentence Encryption:
FANCY is this wow Oh
Reverse Sentence Decryption:
Oh wow this is FANCY
```

Submission

Be prepared to demonstrate and discuss your code with the class. After giving every one time to work we will come together and discuss:

1. Solutions you came up with
2. Roadblocks you hit along the way
3. What you would do differently if you were to do the assignment again

Curious to know how your code would be graded in a college class? Here's an example rubric:

	Missing / Not Complete (0)	Approaching (2)	Meets (4)	Exceeds (5)
Readability	Assignment is unreadable or not submitted.	Assignment includes formatting, but significant improvements could be made. For example, headers, more documentation (comments), descriptive variable names.	Assignment includes formatting, but minor improvements could be made. For example, headers, more documentation (comments), descriptive variable names.	Assignment is well formatted and easy to read. Headers, documentation (comments), and descriptive variable names are all included.
Computational Problem Solving	No code is included in the assignment, or the code included is unreadable.	Problem solving approach could use significant improvements. Specifically, better decomposition of the problem, and/or increased modularity.	Problem solving approach is solid but minor improvements could be made with respect to decomposition of the problem, and/or increased modularity.	Problem solving approach is solid. Problem is decomposed into manageable pieces and code is modular.
Implementation	Nothing has been implemented, or most of assignment has not been done.	Code does not run consistently or efficiently. Some outputs match expected outputs. All parts of the assignment are completed except for a few small parts.	Code mostly runs consistently and efficiently. Most outputs match expected outputs, and all parts of the assignment are completed.	Code runs consistently and efficiently. Outputs match expected outputs, and all parts of the assignment are completed.