# Secret Messages Extensions – no scaffolding!

Any Code, Screenshot, or Typed answer should be placed into the table below….
If there were options on which question to do… the question text should also be entered.

|  |  |
| --- | --- |
| Q | Answer |
| 1 | **Add a menu Item**Almost always there as part of a bigger question Add option “X” which when selected prints the first line of “Never going to give you up” by Rick Astley.Add a new option “x” to DisplayMenu() |
|  |  |
| 2 | **Validate GetTextFromUser** Does this need validating at input? Or at the point of using the text for encryption/decryption?Much better to validate at input so can keep re-requesting until valid data is input, than to validate when it is used, when it may be difficult to go back to the input code.Add a presence check to *GetTextFromUser*. This uses a while loop and the *valid* boolean variable to enable exit of the while loop once the user has entered a valid input: |
| 2(b) |  |
|  | We could go further and test for erroneous data by checking that each character in the input text is either alphanumeric or punctuation. Note that we need to use the *string* library:  |
|  |  |
| 3 | **Validate GetKeyforCaesarCipher**TypeCheck (integer), Range Check (not more than 25? Not less than -25) NOT ZeroAdd a while loop which only exits if the user inputs valid data.The type check can be implemented by using a try/except block to catch the run-time error that occurs when an alpha character is converted to an integer.If the type check succeeds then check that the numeric value is valid via a range check. |
|  |  |
| 4 | **Brute force find Caesar Key**Create a sub routine that outputs the plaintext for each Caesar Key. The new subroutine can leverage the existing Use subroutine which applies a Caesar shift. The new subroutine loops through 1 to 25 to apply all possible shifts:Need to display the new menu option and add the code to handle the new choice. |
| 5 | **Frequency Analysis**Create a sub routine that output the frequency of each of the Characters in the cypher text.First add code to handle the new option:And then create the new subroutine to output the results of the frequency analysis. |
| 6 | **Steganography Encryption**Really hard to automate BUT… using diary.txt using a start position of 1, and end position of 208 and a value for n of 9 then you should be able to see the hidden message “MeeT ME at the co1D room”.The encrypted message should be simply the numbers that each jump between the letter represents.This finds and reports the jumps needed to hide the text in the *Plaintext* variable within the *diary.txt* file.Note that I have opened *diary.txt* as a binary file and therefore needed to use *.decode(‘ASCII’)* to convert each character to text. It would have been easier to just open the file as a text file using *.open(‘diary.txt’, ‘r’)* but I chose to copy the method already used in the skeleton code for opening this file.It’s not really necessary to store the jumps in an array – they could be concatenated to the return string as they are found, and return the string without the final comma. |
| 7 | **Steganography brute force decrypt** Try all the values of n below 20 for all the starting positions below 20 and output all the results. |
|  | Note – only see text for N=1, which is a poor way to hide a message in the text. Better to start at N=2:The output for N=5 is an anomoly – we can see it’s not plain text even though three common words were found in the output. |
| 8 | **Steganography Stepped N** Use in another code N+1 N+pi Function(N)? a non-linear way of jumping through the letters in the diary. There is a non-linear message hidden.To implement a non-linear increment N, we could change the *EveryNthCharacterStenography* function to take an additional parameter, *NIncrease*. You can see that this value is added to N in the while loop. The ‘= 0’ part in the parameter list means that if it is not added as an argument then the value of NIncrease will be zero (i.e. it will behave as though there is no non-linear element to the increment). def EveryNthCharacterSteganography(StartPosition, EndPosition, N, NIncrease = 0): CurrentPosition = StartPosition HiddenMessage = '' while CurrentPosition <= EndPosition: HiddenMessage += GetTextFromFile(CurrentPosition, CurrentPosition) CurrentPosition = CurrentPosition + N N += NIncrease return HiddenMessageOriginally, if *N* were 2, then *CurrentPosition* would iterate through the following values:1, 3, 5, 7, 9, 11, … i.e. a linear progression, each value incrementing by 2 each time. |
| 9 | Bonus – implement the vernam cipher… |