**University of Wolverhampton**

**School of Engineering, Computational and Mathematical Sciences**

**4CS001 Introduction Programming and Problem Solving**

**Coursework 1**

**This assignment is worth 40% of the overall module grade**

**Introduction**:

This coursework will assess your knowledge of data structures and iteration. It also builds on many of the topics we have covered over the course of the module.

The marking scheme for this task is on Canvas, make sure you check back on a regular basis as you work through the assessment.

**Task**

Your task is to develop a program that encrypts and decrypts text using the Caesar Cipher.

**Overview:**

The Caesar Cipher technique is one of the earliest and simplest methods of encryption technique. It is a type of simple substitution cipher, i.e., each letter of a given text is replaced by a letter with a fixed number of positions down the alphabet. For example, with a shift number of 1, A would be replaced by B, B would become C, and so on. The method is named after Julius Caesar, who apparently used it to communicate with his officials.

Thus, to encrypt a given text we need an integer value, known as a shift which indicates the number of positions each letter of the text has been moved down.

Thus, for a shift number of 4, the string:

**ABCDE**

becomes

**EFGHI**

Thus, the algorithm for Caesar Cipher:

Input:

* A text string of upper-case letters.
* A number between 0-25 denoting the required shift.

Procedure:

* Traverse the given text string one character at a time.
* For each character, transform the given character as per the rule, depending on whether we’re encrypting or decrypting the text.
* Return the new string generated.

For example, the message **HELLO WORLD**, a shift of **4** is: **LIPPS ASVPH**. Historically the Caesar Cipher ignored anything that is not a letter and it is case-insensitive, treating upper and lower case letters the same.

**Getting Started:**

Start by creating the file **caesar\_<<StudentID>>.py**. You **MUST** follow the instructions provided below in implementation part strictly. Add your name and student number to the top of the file.

**Program Requirements:**

You will develop a program capable of encrypt and decrypting text using Caesar cipher. In order to do this, you will be required to implement several functions, specified in the template provided.

Once complete, your programs ***main()***method should do the following:

1. Prompt users to select a mode (encrypt or decrypt).
2. Check if the mode the user entered is valid. If not, continue to prompt the user until a valid mode is selected.
3. Prompt the user for the message they would like to encrypt or decrypt.
4. Encrypt or decrypt the message as appropriate and print the output.
5. Prompt the user whether they would like to encrypt or decrypt another message.
6. Check if the user has entered a valid input (y/n) If not, continue to prompt the user until they enter a valid response. Depending upon the response you should either:
   1. End the program if the user selects no.
   2. Proceed directly to step 2 if the user says yes.

You should use a loop to keep the programming running if the user says that he/she would like to encrypt or decrypt more messages.

Your program should handle both uppercase and lowercase inputs. You can use

str.upper() and str.lower() to convert strings to a specific case. However, you should only output only uppercase messages.

**Example Simple Implementation (user input in red):**

Welcome to the Caesar Cipher

This program encrypts and decrypts text using Caesar Cipher. Would you like to encrypt (e) or decrypt (d)? : g

Invalid Mode

Would you like to encrypt (e) or decrypt (d) ?: e

What message would you like to encrypt: HELLO WORLD

What is the shift number: 4

LIPPS ASVPH

Would you like to encrypt or decrypt another message? (y/n): t

Would you like to encrypt or decrypt another message? (y/n): y

Would you like to encrypt (e) or decrypt (d): d

What message would you like to decrypt: LIPPS ASVPH

What is the shift number: 4

HELLO WORLD

Would you like to encrypt or decrypt another message? (y/n): n

Thanks for using the program, goodbye!

**Structure and Documentation**

The structure of your code and documentation will be analysed and assessed.

This will be done using a static analysis tool called **Pylint**. This software checks the code in your program, ensures that it follows Python conventions and that all functions, classes and modules have been documented. You can read more about it here: <https://www.pylint.org/>.

Python has an official style guide named PEP8 (<https://peps.python.org/pep-0008/>), which is where most Python conventions and coding standards originate from.

Example checks that Pylint carries out to ensure that the PEP8 coding standard is followed include things such as:

* + checking line-code's length
  + checking if variable names are well-formed
  + checking if imported modules/functions are used
  + checking if variables/function parameters are used

It is a good idea to run these checks on your code at regular intervals and before submitting.

**Note:** Marks will be deducted for warning and errors detected in your code.

**Implementation Details**

**Step 1 – Implementing the *welcome()* function (10%):**

To implement this function, you simply need to print an introduction to the user. You should use the following message:

Welcome to the Caesar Cipher

This program encrypts and decrypts text with the Caesar Cipher.

**Step 2 – Implementing the *enter\_message()* function(15%):**

To implement this function, you need to request input from the user to determine the mode of conversion and the message that they would like to encrypt or decrypt.

Your function should check if the mode the user entered is valid. If not, it should continue to prompt the user until a valid mode is selected.

The function should return two strings, the mode of conversation and the message. The message should be converted to upper case to avoid potential encrypting/decrypting issues.

**Example Implementation:**

>>> enter\_message()

Would you like to encrypt (e) or decrypt (d): g

Invalid Mode

Would you like to encrypt (e) or decrypt (d): e

What message would you like to encrypt: Hello WORld

What is the shift number: a

Invalid Shift

What is the shift number: 4

('e', 'HELLO WORLD',4)

**Step 3 – Implementing the *encrypt()* function (25%):**

To implement this function, you need to correctly encrypt a plain text message as encrypted text. It should take 2 parameters, the message to be encrypted, and the shift number.

Your function should return a single value, a string - the encrypted message.

**Example Implementation:**

>>> encrypt("Hello World", 4)

'LIPPS ASVPH'

**Step 4 – Implementing the *decrypt()* function(25%):**

To implement this function, you need to correctly decrypt a message in Morse Code. It should take 2 parameters, the message to be decrypted, and the shift number.

Your function should return a single value, a plain text string.

**Implementation:**

>>> decrypt('LIPPS ASVPH',4)

'HELLO WORLD'

**File Input/Output (25%):**

The Caesar Cipher that you’ve developed can only encrypt or decrypt a single message at a time. Therefore, your task for this task will be to attempt to resolve this issue by implementing file reading/writing capabilities so that multiple messages can be encrypted/decrypted in one go.

To do this, you will need to implement the **process\_file(),write\_messages(), is\_file()** and **message\_or\_file()** functions.

The first function **process\_file()** should take two parameters, the filename from which messages should be read and the mode of operation (encrypting/decrypting). It should process the lines in the file one by one, returning a list of encrypted/decrypted messages.

A file has to be added on messages.txt that includes a series of plain text strings, which you can use to test your function.

**Implementation:**

>>> process\_file('messages.txt', 'e')

The second function **is\_file()** should take a single parameter, a filename and return a Boolean value (True/False) depending upon whether the file exists within the current folder. There are several ways that you can achieve this, for example using exception handling logic or through the built-in OS module included with Python.

**Implementation:**

>>> is\_file('messages.txt')

True

The third function **write\_messages()** should take a single parameter, a list of strings. It should write these strings to a file called results.txt, with each item taking up a single line.

The function doesn’t need to return anything.

The fourth function **message\_or\_file()** should work similarly to the **enter\_message()** function you developed earlier. It shouldn’t take any parameters, but needs to return three values, the mode ("e" or "d" based on whether the user would like to encrypt or decrypt), a message if the user would like to process messages using the console (or None) and a filename if the user would like to process data in a file (or None). You should continue to validate the mode that the user enters as well as the input mechanism (file/console).

In addition, you should call your **is\_file()** function to verify filenames. Users should be repeatedly prompted to enter the filename if the file doesn’t exist.

**Example Implementation:**

>>> message\_or\_file()

Would you like to encrypt (e) or decrypt (d):g

Invalid Mode

Would you like to encrypt (e) or decrypt (d):e

Would you like to read from a file (f) or the console (c)? f

Enter a filename: something\_silly.txt

Invalid Filename

Enter a filename: messages.txt

What is the shift number: 4

('e', None, 'messages.txt')

>>> message\_or\_file()

Would you like to encrypt (e) or decrypt (d): g

Invalid Mode

Would you like to encrypt (e) or decrypt (d): e

Would you like to read from a file (f) or the console (c)? c

What message would you like to encrypt: Hello WORld

('e', 'HELLO WORLD', None)

You will also need to update your main() function to call the functions you’ve implemented.

**Full Implementation (user input in red)**

Welcome to the Caesar Cipher

This program encrypts and decrypts text using Caesar Cipher.

Would you like to encrypt (e) or decrypt (d): d

Would you like to read from a file (f) or the console (c)? c

What message would you like to decrypt: LIPPS ASVPH

What is the shift number: 4

HELLO WORLD

Would you like to encrypt or decrypt another message? (y/n): y

Would you like to encrypt (e) or decrypt (d): e

Would you like to read from a file (f) or the console (c)? c

What message would you like to encrypt: Hello WORld

What is the shift number: 4

LIPPS ASVPH

Would you like to encrypt or decrypt another message? (y/n): y

Would you like to encrypt (e) or decrypt (d): e

Would you like to read from a file (f) or the console (c)? f

Enter a filename: something\_silly.txt

Invalid Filename

Enter a filename: messages.txt

What is the shift number: 4

Output written to results.txt

Would you like to encrypt or decrypt another message? (y/n): n

Thanks for using the program, goodbye!

**Submission and Marking**

You should upload your submission to Canvas before the deadline on Canvas. If you fail to do so, you may receive a grade of 0 NS (Non-Submission).

Your work will be automatically tested using a program that tests each of the individual functions you have implemented. Therefore, it is very important that you do not alter any of the function signatures in the template and implement everything as instructed.