Part 2 - Encoding / Decoding Strings

This relates to Part A only.

1. Python constants
The specification shows the variable ALPHABET in capital letters. Why is that?

ALPHABET = 'abcdefghijklmnopqrstuvwxyz'

- Most languages allow for constants – like variables except that once they are defined they **cannot** be changed. To make them easy to identify they are written in capital letters. Although Python does not allow a constant variables a programmer can capitalise any variable name to indicate that it **should** not be changed.

---

2. Specification - Part 2 - Encoding / Decoding Strings
This assignment will involve the use of strings and functions. Create a Python program that will apply a simple encoding and decoding of a message.

The program should prompt the user to enter a message and then enter a key (the shift value). The key should be in the range 1 to 25. The program will then produce an encoded string by applying the key (shift value) to the alphabetic characters in the message. The shift amount instructs the program on how many places in one direction to move to find the encoding of any letter of the alphabet.

Examples: A key of 3 will shift each character 3 characters to the right.

- the character '**a**' as the character '**d**',
- the character '**b**' as '**e**', etc.,
- the string **'this'** becomes the string **'wklv'** using a rotation of 3.

Wrap the encoding around the alphabetic characters, so that 'w' is encoded as 'z', 'x' is encoded as 'a', 'y' is encoded as 'b', and 'z' is encoded as 'c' etc.

Important note:

- There are various solutions to this type of challenge on the internet. These solutions convert a letter to ASCII code using ord() and convert an ASCII code to a letter using chr(). Instead your solution should be built around rotating a character in string. E.g.,

  ALPHABET = 'abcdefghijklmnopqrstuvwxyz'

- **Any code solutions submitted using the ord() and chr() approach will receive a mark of zero.**

Part A: Encoding
The program prompts for a string to encode and a rotation integer in the range of 1-25. The program then returns the encoded string. Important, the program should not encode any letter that is not in the lower case alphabet. Those letters should simply be passed through to the encoded string. E.g.,

- the string **'*this!'** becomes the string **'*wklv!'** using a rotation of 3.
- the string **'This'** becomes the string **'Tklv'** using a rotation of 3.

---

3. User/Program requirements
Step 1: understand what the user requires:

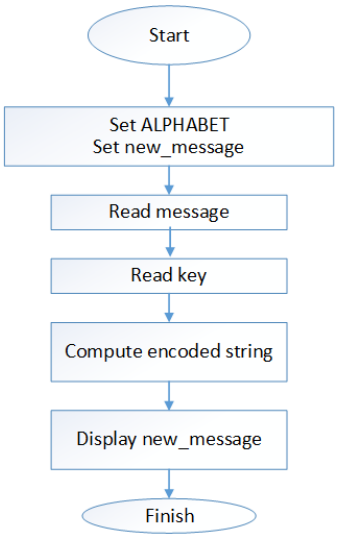
- The user wants to enter a message from the keyboard.
- The user wants to enter a key from the keyboard.
- The user wants the program to 'encode the string'.
- The user wants to see the encoded string displayed on the screen.

Step 2: understand what the program requires:

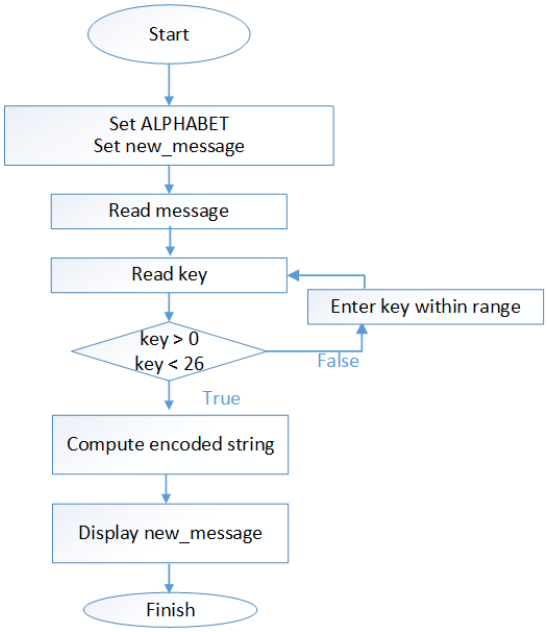
- Understand the inputs to the program, the outputs (results) and the process between both.
- The inputs are the message and key entered by the user from the keyboard. The key should be in the range 1 to 25.
- The output is the encoded string.
- The process encodes the entered message by applying the key (shift value).

4. Flowchart - Part A

Using one process box to represent the encoding of the string, we could create a first draft of a flowchart with the initiation phase, inputs and outputs.



- Where could we check if the key is in the range 1 to 25?



5. Compute Encoded String

The process encodes the entered message by applying the key (shift value). A key of 3 will encode the character 'a' as the character 'd', the character 'b' as 'e', etc., (each character's encoding is shifted 3 characters to the right).

- Apply the key to the lowercase alphabetic characters in the message to produce the encoded new message.
- Characters that are not lowercase characters are NOT encoded and the original character is used in the encoded message.

- For each character in the message, check if that character is found in the string ALPHABET. If found:
    - get the position in ALPHABET
    - add the key to its current position to get new position
    - find new character using the new position
    - add new character to new message.

- Cyclic, wrap-around. How do we encode the characters w, x, y?
    - Wrap the encoding around the alphabetic characters, so that with a key of 3, 'w' is encoded as 'z', 'x' is encoded as 'a', 'y' is encoded as 'b', and 'z' is encoded as 'c'. Think of this as encoding where the alphabet is arranged in a circle, and the shift amount tells you how many places in one direction to move to find the encoding of any letter of the alphabet.
    - You can use the operator % here. See example of a similar problem. The days of the week are cyclical. Suppose we assign the following numeric code to the days of the week:

        0 Sunday
        1 Monday
        2 Tuesday
        3 Wednesday
        4 Thursday
        5 Friday
        6 Saturday

    - To determine what day of the week is 2 days from Tuesday, we could find the code for Tuesday (2), add 2 (for 2 days). This would give us 4 meaning 2 days from Tuesday is a Thursday.
    - To determine what day of the week is 10 days from Tuesday, we can look at the code for Tuesday (2), add 10 and use % 7 (for number of days). Python tells us that ((2+10) % 7) = 5 meaning 10 days from Tuesday is a Friday**.**
    - This example works well because our numeric code started at 0 for Sunday incremented by one each time until Saturday.

    - Each letter of *ALPHABET* has a position (index) starting at position 0 to 25 inclusive:
    - So the letter ‘a’ is at position 0 of the alphabet, and ‘z’ is at position 25:

        ALPHABET = 'abcdefghijklmnopqrstuvwxyz'