Applying a Caesar Shift to a Single Letter

Caesar Shift 1 'a' -> 'b' Caesar Shift 2 'f' -> 'h' Caesar Shift 4 'y' -> 'c'

Abstraction

We want to develop an algorithm that will solve the problem for a string of characters, and output to a file the encryption for each letter of the alphabet.

Some of the key components of the problem are :

- A loop to iterate through a validated string of characters, and build the encrypted string.
- For each single letter is extracted, convert the letter character into a UTF-8 (or ASCII) number and add a valid cipher key.
- If the new UTF8 code is greater than the code for 'z', subtract 26. This is the wraparound situation.
- Iterate through the entire alphabet, and encrypt each letter with the user's chosen cipher key.

Writing your Thinking

Take 5 minutes to think about how you tackle this problem.

- **Did you use pen and paper to try out specific examples of applying the Caesar Shift?**
- How is the wraparound different to converting letters that do not need a wraparound?
- **Did you figure out how to solve the problem for just lowercase first, before tackling uppercase?**
- What are good examples to enter as the user to test your program? Would it be better if your partner tested your program and you tested your partner's?

Using Think-Pair-Share-Square (TPSS), go through how you and your partner were thinking about how to solve the problem.

Pseudo-Code

```
Ask the user for an inputString and a key;
#always assume the user might enter unsuitable data
If user_input of wrong type {
        set some default values for the output;
}
If the inputString is a text and the key is between 1 and 25 {
        For each letter in inputString {
                utf8Number = utf_of(letter);  #ord(letter) in Python
                cipherNumber = utf8Number + Key;
                If letter is lowercase {
                        If cipherNumber > utf8_of('z') {
                                cipherNumber = cipherNumber – 26;
                        }
                }
                If letter is uppercase {
                        If cipherNumber > utf8_of('Z') {
                                cipherNumber = cipherNumber – 26;
                        }
                }
                codedLetter = character_of(cipherNumber);
                encryptedString = encryptedString + codedLetter;
        }
}
Output the user's string, key and encrypted string;
```

```
#output the catalogue to a txt file
If the user’s cipher key is not valid{
        Set the cipher key to 0
}
Create a list of the alphabet and a dictionary for the cipherAlphabet;
For each letter in the alphabetList {
        Set the cipherAlphabet key to the letter;
        Set the cipherAlphabet value to the encrypted letter;
}
Open a new file;
Output the dictionary items to the file;
```