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 initially for just one letter (either lower or upper case) from the user.

Some of the key components of the problem are :

- Converting the letter character into a UTF-8 (or ASCII) number.
- Adding on the Cipher Key, that we know is from 1-25.
- If the new UTF8 code is greater than the code for ‘z’, what do we SUBTRACT in order to get back to a code for a letter? This is the wraparound situation.
- Each character will need to be examined to determine if it is uppercase or lowercase

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?
- **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 a character 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 character is a letter and the key is between 1 and 25 {
        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);
Output the user’s letter, key and codedLetter;
```