Caesar Cipher - part 1: encryption

The purpose of this project is to familiarize you with the use of strings and looping. For this assignment, you will create an encryption program. You will complete this assignment with your assigned partner from partners1. Don't forget good coding style and comments, as well as the quality/efficiency of your solution!

Deliverables: Your encryption program in a file named *user1_user2.py*. Make sure both authors names are also included in a comment at the top of your file!

Background

The Caesar cipher is named after Julius Caesar who used this type of encryption to keep his military communications secret. A Caesar cipher replaces each plain-text letter with one that is a fixed number of places down the alphabet. The plain-text is your original message; the cipher-text is the encrypted message. For example in a shift of 3, “B” in the plain-text becomes “E” in the cipher-text, a “C” becomes “E”, and so on. The mapping wraps around so that “X” maps to “A”, “Z” maps to “C” and so on.

Here is the complete mapping for a shift of three:

Plain:ABCDEFGHIJKLMNOPQRSTUVWXYZ
Cipher:DEFGHIJKLMNOPQRSTUVWXYZABC

To encrypt a message simply substitute the plain-text letters with the corresponding cipher-text letter. For example, here is an encryption of “the quick brown fox jumps over the lazy dog” using our shift-three cipher (case is ignored):

Plaintext: the quick brown fox jumps over the lazy dog
cipher-text: WKH TXLFN EURZQ IRA MXPSV RYHU WKH ODCB GRJ

To decrypt the message you simply reverse the process.

The encryption can be represented using modular arithmetic after first transforming the letters into numbers according to the scheme: A = 0, B = 1, etc. (which is the index of the alphabet if it is in a string or a list).

Your Task

Create a program in encrypt.py that reads some text input and a shift to use from the user, encrypts the text using the given shift for the Caesar Cypher, and prints the cipher-text to the screen, along with the shift used. Your program should do this continually until the user enters "q" to quit. You can only shift letters, but input may have other characters, such as spaces and punctuation. You should just place these in the cipher-text as they are. If a letter in the plaintext is uppercase, you should use the uppercase version of the encrypted letter in the cipher-text.

Below is some sample output for your program, user input is bold:

```
enter plaintext to encrypt: Hello my name is Sherri!
enter the shift to use: 3
cipher-text: Khoor pb qdph lv Vkhuul!
a shift of 3 was used

enter plaintext to encrypt: I love math symbols, like + and
* ...
enter the shift to use: 3
cipher-text: L oryh pdwk vbperov, olnh + dqg * ...
a shift of 3 was used

enter plaintext to encrypt: Don't YOU like CS?!
enter the shift to use: 23
cipher-text: Alk'q VLR ifhb ZP?!
a shift of 23 was used

enter plaintext to encrypt: q
```

Some possibly helpful notes:

1. Python has a built-in function "ord(character)" that returns an integer value (ASCII) representing the character. The letters in the alphabet are represented by the values 97 to 122 for lowercase 'a' to 'z', and 65 to 90 for uppercase 'A' to 'Z'. For example:

```
ord('a') # returns 97
ord('b') # returns 98
ord('B') # returns 66
ord('Z') # returns 90
```

2. Python has another built-in function "chr(int)" that returns the character represented by the given integer value in ASCII. For example:

```
chr(97) # returns 'a'
ord(98) # returns 'b'
ord(66) # returns 'B'
ord(90) # returns 'Z'
```

   Why do the capital letters have lower ASCII integer values than the lowercase letters? I have no idea, just the way it is...

3. You can compare strings using the same operators as comparing numbers, such as <, >, ==, etc. String comparison uses alphabetical order, just like in a dictionary, so 'a'<'b' is True, and 'camel' < 'cat' is True, and 'dog' >= 'horse' is False. Think about how you can use this to determine if a character is lowercase, capital, or something else entirely.

Bonus:

Add functionality to accept the input "random" for the shift value from the user. Your program should still work exactly as before if the user enters a number for the shift value, however if they enter "random" instead, you should generate a random value between 1 and 25 to use as the shift. You should still output the cipher-text and the shift used. Check out the random module in Python for helpful functions.

Here is example output with the bonus, user input is bold:

```
enter plaintext to encrypt: Test 1
enter the shift to use: 2
cipher-text: Vguv 1
a shift of 2 was used

enter plaintext to encrypt: Test 2
enter the shift to use: random
cipher-text: Bmab 2
a shift of 8 was used

enter plaintext to encrypt: q
```