day33_cipher
11/19/19

Today's task is to write a program that encodes text using a Caesar Cipher, a simple method of encryption historically credited with Julius Caesar. See the following Wikipedia link for more information:

http://en.wikipedia.org/wiki/Caesar_cipher

To make a Caesar cipher, you first choose an offset (a number between 1 and 25 inclusive), and then map all letters of the alphabet to new letters of the alphabet shifted over.

For example, if you use an offset of 3, the following happens:

```
English:  abcdefghijklmnopqrstuvwxyz
Cipher:   defghijklmnopqrstuvwxyzabc
```

Look at the top line for the "c". Notice that below it in the Cipher line the letter right there is "f". This means in the encrypted text every c will be replaced with an f. The English word "cat" becomes "fdw". The phrase "I wish I knew more palindromic cyclops primes" becomes "l zlvk l nqhz pruh sdolqgurplf fbforsv sulphv ". Not at all obvious, eh?!

This kind of cipher is susceptible to hacking using a frequency analysis, or just brute force. If you had enough text to work with you could run it through a frequency analysis program and then figure out which letter was "e", the usual most common letter in English text, figure out the offset to e, and you've most likely broken the code. Obviously this type of cipher isn't used where security matters anymore.

However, it's a great puzzle. So, your task is to create a program that does the following:

- your program is named "day33_cipher"
- it asks the user for an offset to use
- uses a try/except structure to make sure the offset is an integer
- the program then verifies that the integer is one that will work with how you have set up the offset procedure (Make sure the user enters a number between 1 and 25 inclusive. 0 and 26 should not be allowed (they would mean that you weren't encrypting anything!))
- asks the user for text to encode, make the text lowercase
- encrypts the text and prints it

Then on the same run:

- asks the user for text to decode, make what they enter lowercase
- prints the decoded text (using the same offset as the encoding, don't ask for an offset here)
- if you enter nothing on any encode/decode line, the program quits

I give you the code you can use to encrypt and decrypt on the next page. You just have to do everything around it, making the program look good.

Tips and sample output on next page.

Tips:

- You only use a try/except structure for making sure the user is entering an integer. Inside the try/except structure use an if statement to make sure that the offset is between 1 and 25 inclusive.
- Make the user-entered text all lowercase before you try to encrypt or decrypt it.
- Then print the encrypted text, letter by letter, using the code shown below (the code shown is an image, do not try to copy and paste it into Python; get the actual code from the file day33_starter_code.txt in the class worksheets folder):

```
#Use this code to encrypt
for i in range(len(text)):
    ascii_code=ord(text[i])
    if text[i] in "abcdefghijklmnopqrstuvwxyz":
        ascii_code = ascii_code + offset
        if ascii_code > ord('z'):
            ascii_code = ascii_code - 26
    print(chr(ascii_code),end="")
print()
```

- To decrypt, use the following code (again, see above about getting this code into your program):

```
#Use this code to decrypt
for i in range(len(text)):
    ascii_code=ord(text[i])
    # Offset any lowercase letter, wrapping
    if text[i] in "abcdefghijklmnopqrstuvwxyz":
        ascii_code = ascii_code - offset
        if ascii_code < ord('a'):
            ascii_code = ascii_code + 26
    print(chr(ascii_code),end="")
print()
```

In both code snippets the text to be encrypted or decrypted is in a variable named **text** and the offset you are using is in a variable called **offset**.

See sample output here:

```
Enter cipher offset: 5
Enter a message to encode: palindromic cyclops primes
Cipher text: ufqnsiwtrnh hdhqtux uwnrjx

Enter a message to decode: hdhqtux
English text: cyclops
```

Optional extra credit:

Ask the user to enter encrypted text and decrypt it using all 25 possible decryptions (print them all). One will clearly be the English phrase.

Or, more challenging: make a decoding program that uses your frequency program functionality to analyze encrypted text to find the most commonly used letter, peg that as the letter "e", figure out the encoding offset, and decode the user-entered text. (Obviously this would only work if the user pasted in a large amount of text, so perhaps you should tell the user this program requires 500 characters or more of text.)

If you do either of these, name it day33_cipherEC.py and be sure to tell me to look for it.