Python Project Substitution Cipher

Assignment Overview

In this assignment, we are going to implement a simple encoding and decoding of text. As a result of working on this project, you will gain more experience with the use of:

1. string
2. if statement
3. for loop and while loop

Background

The substitution cipher is a cipher that has been in use for many hundreds of years. It basically consists of substituting every plaintext character for a different ciphertext character. The cipher alphabet may be shifted or reversed or scrambled. In this project, we will **combine** two types of substitution cyphers: a simple substitution cipher followed by an affine cipher.

Simple substitution cipher: the cipher alphabet is created by first writing out a keyword, removing repeated letters in it, then writing all the remaining letters in the alphabet in the usual order. Consider the following example. Consider the alphabet being a single string consisting of the lower-case English letters as below (shown with each letter's associated index, e.g. m's index is 12):

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Then, using this system, the keyword "michigan" gives us the following alphabet mapping:

m	**i**	**c**	**h**	**g**	**a**	**n**	b	d	e	f	j	k	l	o	p	q	r	s	t	u	v	w	x	y	z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Note how "michigan" starts the mapping, how the second "i" is skipped, and how the remaining letters continue in alphabetical order without the characters of "michigan".

Using the example above, the word "green" is translated as follows:

- the 'g' is found at index 6 in the alphabet. The letter in the rotated alphabet is 'n'.
- the 'r' is found at index 17, the rotated letter is 'r'
- the 'e' is found at index 4, the rotated letter is 'g'
- the 'e' is found at index 4, the rotated letter is 'g'
- the 'n' is found at index 13, the rotated letter is 'l'

Thus, the string 'green' becomes the string 'nrggl'. You will use that cyphertext 'nrggl' as the starting point ("plaintext") for the next stage, the affine cipher.

Affine cipher: The key for the Affine cipher consists of two numbers, a and b. It takes in a plain-text string, and translates it into a new string based on an encryption/decryption functions that rotate the alphabet of size m: (1) the encryption function is y = (a*x + b) mod m, (2) the decryption function is

the inverse operation, i.e. do the encryption backwards (mathematically decryption is $x=a^{-1}*(y - b) \bmod m$ where a^{-1} satisfies the equation $1= a*a^{-1} \bmod m$, but in this case thinking mathematically will make programming much more difficult).

To encrypt consider our plaintext 'green' and its cyphertext 'nrggl' now using our affine key (a=5, b=8). After scrambling the alphabet using the keyword "michigan" and encrypting the plaintext "green" to "nrggl", we will encrypt "nrggl" using the table mentioned above for the numeric values of each letter, from our key a is 5 and b is 8, and finally m is 26 since there are 26 characters in the alphabet being used. The first step is to write the numeric values of each letter using the scrambled alphabet. Then, take each value of x, and solve the first part of the equation, (5x + 8). After finding the value of (5x + 8) for each character, take the remainder when dividing the result of (5x + 8) by 26. The final step in encrypting the message is to look up each numeric value in the table for the corresponding letters. In this example, the encrypted text would be "kpccv". The table below shows the completed table for encrypting a message in the Affine cipher.

plaintext	n	r	g	g	l
x	6	17	4	4	13
(5x+8)	38	93	28	28	73
(5x+8) mod 26	12	15	2	2	21
ciphertext	k	p	c	c	v

Important Hint: build the entire affine mapping and then use it to encrypt the plaintext. (Why take this approach? If you build an affine mapping, it makes decryption as easy as encryption.)

To decrypt you want to reverse that process. That is, start with the ciphertext "k" and work backwards (using the affine mapping) to get "n" and repeat for the remaining characters.

Project Description / Specification

1) The program should prompt the user for one of three commands:
   a) `"e"` to encode a string
   b) `"d"` to decode a string
   c) `"q"` to quit

   Any other command should raise an error and reprompt. Accept either upper or lower case letters (Hint: use the string method `upper()` or `lower()`). Use 'q' to control your `while` loop.
2) If the command is encode, then the program prompts for a string to encode and a keyword (which consist only of letters). The program then returns the encoded string (cipher text).
   a) To build the symmetric key, start with an empty string and as you loop through the keyword add characters to the key string (remember to check for duplicates before insertion. Hint: the string `in` operator is useful and remember that you can also use `not` as part of a Boolean expression, i.e. `not in`). Repeat to add the remaining characters in the alphabet.
   b) Hint: To encode you will find the string `index()` or `find()` methods useful for finding the index of a character in the alphabet.

c) Important, the program should **not encode** any letter that is not in the lower-case alphabet. If upper case, then convert it to lower case. Punctuations, special characters and numbers should simply be passed through to the encoded string.
d) The *keyword* should only contain lower case letters and its length should be between at least 1 and less or equal to 26. If not the case, then the program should display an error message and prompt the user to give another keyword
e) For the affine cipher always use *a=5, b=8*, and *m=26.* (Hint: *mod* is Python's % operator. Also, start with an empty cyphertext and add each encrypted character one at a time to the cyphertext as you encrypt each one.)
f) Hint: the string `isalpha()` method may be helpful. Also, `import string` and `string.ascii_lowercase` might also be useful.

3) If the command is decode, then the program should prompt for a string to decode and a keyword. The output should be the decoded string (plain text).
4) If the command is quit, then the program ends and prints a nice exit message.

Deliverables

The deliverable for this assignment is the following file:

projCipher.py – the source code for your Python program

Notes and Hints:

You should start with this program, as with all programs, by breaking the program down into parts. Here is some of that breakdown to help you

1. Your program should continuously prompt for a command. Can you write a loop that prompts for one of the three commands `"d", "e"` or `"q"` and reports an error if it is not one of those commands?
2. Your program should continuously prompt for a keyword until it is valid. The Python method `isalpha()` is helpful here. For example:
   i. `"abcd".isalpha()`→`True`
   ii. `"abc3".isalpha()`→`False`
   iii. `"1234".isalpha()`→`False`
3. A scrambling of an alphabet. Start with a string that consists of the letters a-z in a single string. How can you create a new string of the same length that starts with the keyword and does not have repeated letters?
4. A rotation of an alphabet. Start with the scrambled alphabet in step 3. How can you create a new string of the same length that has a particular rotation function (a*x+b) mod m where x is the character index in a string, m is the length of the alphabet and (a,b) is the affine cipher keys? Think of the string slicing operators. What do you need to do to the original string to create a new string of the indicated rotation? What pieces can you concatenate together to make the rotated alphabet?
5. Given two strings, one the lower-case alphabet and one a scrambled and rotated alphabet, how can you encode a given string? You need to go through each letter of the string to encode, find it in the regular alphabet, remember its location/index in that alphabet, then find the letter in the scrambled and rotated alphabet at the same index. The Python method `index` is helpful here. It indicates the location of a letter. Thus `"abcdef".index("c")` should return the value 2; `"xyz".index("c")` returns -1, as `"c"` does not occur in the string.

6. When encoding a string, you should check to see if the letter from the original string is in the alphabet. You may use the `in` operator: `"a" in "abcde"` returns `True`. You can also include the Boolean `not` operator: `"x" not in "abcde"` returns `True`.
7. When decoding, you are given a string to decode and a keyword. For example, the string 'go green, go white!', when encoded with a keyword 'michigan', becomes 'km kpccv, km orwzc!'. When decoding, you provide the encoded string and the keyword 'michigan'.
    i. can you undo the encoding process if you **know** the keyword? Here, we find the letter in the rotated alphabet and decode it to the letter in the normal alphabet (the opposite for what we did for encoding).

Test Cases

Test 1

```
would you like to Decrypt or Encrypt data? (e=Encode, d=Decode, q=Quit)q
See you again soon!
```

Test 2

```
would you like to Decrypt or Encrypt data? (e=Encode, d=Decode, q=Quit)e
Please enter a keyword: michigan
Enter your message: go green go white!
your encoded message:  km kpccv km orwzc!
would you like to Decrypt or Encrypt data? (e=Encode, d=Decode, q=Quit)d
Please enter a keyword: michigan
Enter your message: km kpccv
your decoded message:  go green
would you like to Decrypt or Encrypt data? (e=Encode, d=Decode, q=Quit)q
See you again soon!
```

Test 3

```
would you like to Decrypt or Encrypt data? (e=Encode, d=Decode, q=Quit)e
Please enter a keyword: MSU2
There is an error in the keyword. It must be all letters and a maximum length of
26 Please enter a keyword: abcdefghijklmnopqrstuvwxyzz
There is an error in the keyword. It must be all letters and a maximum length of
26 Please enter a keyword: MSU
Enter your message: Go Green Go White
your encoded message:  jm jnuuv jm lpwzu
would you like to Decrypt or Encrypt data? (e=Encode, d=Decode, q=Quit)q
See you again soon!
```