Cryptology

“Cryptology is the science or study of cryptanalysis and cryptography.” [dictionary.com] Cryptography deals with procedures and methods that generate codes or ciphers to create secret messages. Cryptography is used to authenticate, proving one’s identity. It is also used to ensure privacy or confidentiality of communication, integrity (original item not altered prior to being received). Mechanisms, such as non-repudiation, which ensures that the sender really sent the communication, are highly dependent on strong cryptographic techniques.
Cryptography has been around, probably, as long as writing has been around. There are documented cases dating back 4000 years. In class, one of the earlier ciphers used in the Roman Empire and attributed to Caesar, the Caesar Cipher, was studied and implemented. The cipher is a member of a class of ciphers that uses substitution as its base.

The Caesar Cipher

During the Roman Empire, Julius Caesar utilized a simple cipher to encrypt messages back and forth to his battlefield commanders. The cipher shifts the alphabet by a certain number of letters and wraps the letters that extend beyond the last letter back onto the remaining unmapped starting letters. Consider the following:

```
Original  : ABCDEFGHIJKLMNOPQRSTUVWXYZ
Shift by 3: DEFGHIJKLMNOPQRSTUVWXYZABC
```

This is a Caesar cipher or shift cipher with a shift of 3. Each letter in a message is replaced by a letter that is three letters down the alphabet. The last three letters are wrapped onto the starting three letters. Using this cipher, a plain text message transforms into the following cipher text.

```
Plaintext : DATA STRUCTURES IS DIFFICULT
Ciphertext: GDWD VWUXFWXUHV LV GLIILFXOW
```

Computers can easily perform such transformations using modular arithmetic. Any letter can be expressed numerically by a one-to-one mapping as shown below.

```
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
```

To encrypt a letter in a plain text message, use the transformation,

$E(M) = (M + K) \pmod{26}$

where $E(M)$ is the numeric value of the encrypted letter, M is the numeric equivalent of the plain text letter [Luciano et al.] and K is the key, the value to shift. **Note:** *if the alphabet used includes other characters, then take modulo the length of the alphabet instead of 26*.

To decrypt the message, use the transformation,

$D(C) = (C + (26 - K)) \pmod{26} = (C - K) \pmod{26}$

where $D(C)$ is the numeric value of the plain text letter, C is the numeric value of the encrypted letter, and K is the key. Using the mapping above, the plain text message, WATER, becomes GKDOB as shown below.

```
W   A   T   E   R
22  0   19  4   17
      |
      |    Encrypt(Shift by +10)
      |
      V
6   10   3   14   1
G   K    D   O    B
```

Let's Password Protect It!

In this recitation exercise, the encryption and decryption functions that were implemented in class will be extended to use a password. The idea is to utilize the same scheme of shifting the alphabet, except the shifting is relative to the ordinal value of each character in the password. That is each character in the password or more precisely the ordinal value of each character in the password will form the shift key.

The following Python functions are provided as skeletons in the cypher.py file. Instruction on how to obtain this file is provided later in this document.

```
def encrypt(message, password):
    pass # replace this line with your code

def decrypt(message, password)
    pass # replace this line with your code
```

Once implemented, one should be able to encrypt a message using the password provided. For example,

```
>>> encrypt('This is my message',' My#1Secret Password')
'N$Q}\t+G6NMng$CUCMC'
```

Calling the decrypt function on the encrypted message with the same password should result in the original message.

```
>>> decrypt(encrypt('This is my message',' My#1Secret Password'),'My#1Secret
Password')
'This is my message'
```

Implementing the Password

To use the password to encrypt or decrypt a message,
the ordinal value of each character in the password can serve as the shift value.
So, consider defining a simple alphabet as,

```
alphabet='abcd'
ordinal_value = {'a':0, 'b':1, 'c':2, 'd':3}
```

then a password like **'bdac'** would shift the first letter in a message by 1 (i.e. ordinal_value['b']==1), and shift the second letter in a message by 3 (i.e. ordinal_value['d']==3), and shift the third letter in a message by 0 (i.e. ordinal_value['a']==0), and so forth. What must be addressed is how to handle the case where the message is larger than the password. In that case,

once all the characters in the password have been exhausted, then the next character to use in the password should be reset to the first character. So, using the example alphabet and password above, the encryption would proceed as follows:

Message:	b	a	d	c	a	d	c	d
Password:	b	d	a	c	b	d	a	c
Encrypted Message:	c	d	d	a	b	c	c	b

Notice that the password was repeated (i.e. in red in the table above). The way to implement this is to keep track of the index of the character in the message being encrypted and take the modulo with respect to the length of the password. This will ensure that the index into the password consistently rotates back to 0. For example, for the message **'badcadcd'**, the second **a** in red has the index of 4. The length of the password **'bdac'** is 4. Taking 4 % 4 results in 0. This means that the password character with an index of 0 will be used.

Implementation

You will be required to accept the assignment on Github via the link on Blackboard. Once you have a repository, you will need to use PyCharm to clone the repository. You will be guided by your recitation instructor on how to carry out this activity.

IMPORTANT: You are only to modify the **cypher.py** file and nothing else. Modifying other files may result in a broken project folder.

The first function, called **`encrypt`**, takes two arguments, a message and a password, returning the encrypted message using the password.
The second function, called **`decrypt`**, takes two arguments, a message and a password, returning the decrypted message using the password. The alphabet that should be used is the one representing all the printable characters, provided in the string module. These functions should work as follows:

```
>>> encrypt('Shhh.... This is a secret','IamBond!')
'\x0brDS\x0c\x0b_BC$DTQhv{Ckg$CzE/,'

>>> decrypt(encrypt('Shhh.... This is a secret','IamBond!'),'IamBond!')
'Shhh.... This is a secret'
```