Classical ciphers in Python

Data Encryption and Decryption

- In cryptography, encryption is the process of encoding a message or information in such a way that only authorized parties can access it and those who are not authorized cannot.
- Encryption does not itself prevent interference, but denies the intelligible content to a would-be interceptor.
- A given encryption algorithm takes the original message, and a key, and alters the original message mathematically based on the key's bits to create a new encrypted message. Likewise, a decryption algorithm takes an encrypted message and restores it to its original form using one or more keys.

Encryption in Python

- Python doesn't have very much in its standard library that deals with encryption. Instead, you get 3rd party libraries.
- The most widely used packages are : PyCrypto and cryptography.
- PyCrypto is a collection of both secure hash functions (such as SHA256 and RIPEMD160), and various encryption algorithms (AES, DES, RSA, ElGamal, etc.).
- The **cryptography** package aims to be "cryptography for humans" much like the **requests** library is "HTTP for Humans". The idea is that you will be able to create simple cryptographic recipes that are safe and easy-to-use.

The cryptography package

- If you are using Python 3.5, you can install it with pip, like so:

  pip install cryptography

- One of the easy to use module of this package is Fernet. The Fernet module implements an easy-to-use authentication scheme that uses a symmetric encryption algorithm which guarantees that any message you encrypt with it cannot be manipulated or read without the key you define.
- The Fernet module also supports key rotation via **MultiFernet**.

Example

```
>>> from cryptography.fernet import Fernet
>>> cipher_key = Fernet.generate_key()
>>> cipher_key
b'APM1JDVgT8WDGOWBgQv6EIhvxl4vDYvUnVdg-Vjdt0o='
>>> cipher = Fernet(cipher_key)
>>> text = b'My super secret message'
>>> encrypted_text = cipher.encrypt(text)
>>> encrypted_text
(b'gAAAAABXOnV86aeUGADA6mTe9xEL92y_m0_TlC9vcqaF6NzHqRKkjEqh4d21PInEP3C9HuiUk
S9f'
 b'6bdHsSlRiCNWbSkPuRd_62zfEv3eaZjJvLAm3omnya8=')
>>> decrypted_text = cipher.decrypt(encrypted_text)
>>> decrypted_text
b'My super secret message'
```

- First off we need to import Fernet. Next we generate a key. We print out the key to see what it looks like. As you can see, it's a random byte string. If you want, you can try running the **generate key** method a few times. The result will always be different. Next we create our Fernet cipher instance using our key.
- Now we have a cipher we can use to encrypt and decrypt our message. The next step is to create a message worth encrypting and then encrypt it using the **encrypt** method. I went ahead and printed our the encrypted text so you can see that you can no longer read the text. To decrypt our super secret message, we just call **decrypt** on our cipher and pass it the encrypted text. The result is we get a plain text byte string of our message.

PyCrypto

- PyCrypto is another library, which provides secure hash functions and various encryption algorithms.
- Installation

  pip install pycrypto

Example

```
from Crypto.Cipher import AES
# Encryption
encryption_suite = AES.new('This is a key123', AES.MODE_CBC, 'This is an IV456')
cipher_text = encryption_suite.encrypt("A really secret message. Not for prying eyes.")

# Decryption
decryption_suite = AES.new('This is a key123', AES.MODE_CBC, 'This is an IV456')
plain_text = decryption_suite.decrypt(cipher_text)
```

Classical cipher

- In cryptography, a **classical cipher** is a type of cipher that was used historically but now has fallen, for the most part, into disuse. In contrast to modern cryptographic algorithms, most classical ciphers can be practically computed and solved by hand. However, they are also usually very simple to break with modern technology.
- Classical ciphers are often divided into *transposition ciphers* and *substitution ciphers*.

Substitution ciphers

- In a substitution cipher, letters (or groups of letters) are systematically replaced throughout the message for other letters (or groups of letters).
- A well-known example of a substitution cipher is the Caesar cipher. To encrypt a message with the Caesar cipher, each letter of message is replaced by the letter three positions later in the alphabet. Hence, A is replaced by D, B by E, C by F, etc. Finally, X, Y and Z are replaced by A, B and C respectively. So, for example, "WIKIPEDIA" encrypts as "ZLNLSHGLD". Caesar rotated the alphabet by three letters, but any number works.

Ceaser Cipher

```
print("Ceaser Cipher", end="\n")
message = input("Enter a message : ")
key = int(input("Enter a key : "))
encryptedMessage = '';
for i in message:
    if i.islower():
        enr = chr(ord(i) + key)
        if (ord(enr) > ord('z')):  # ord() convert charcter to its unicode example a = 97
            enr = chr(ord(enr) - ord('z') + ord('a') - 1)  # chr() convert unicode to character
        encryptedMessage += enr
    else:
        enr = chr(ord(i) + key)
        if (ord(enr) > ord('Z')):
            enr = chr(ord(enr) - ord('Z') + ord('A') - 1)
        encryptedMessage += enr
print("Encrypted Message : ",encryptedMessage)

decryptedMessage = '';
for i in encryptedMessage:
    if i.islower():
        dnr = chr(ord(i) - key)
        if (ord(dnr) < ord('a')):
            dnr = chr(ord(dnr) + ord('z') - ord('a') + 1)
    else:
        dnr = chr(ord(i) - key)
        if (ord(dnr) < ord('A')):
            dnr = chr(ord(dnr) + ord('Z') - ord('A') + 1)
    decryptedMessage += dnr
print("Decrypted Message : ",decryptedMessage)
```

Transposition ciphers

- In a transposition cipher, the letters themselves are kept unchanged, but their order within the message is scrambled according to some well-defined scheme. Many transposition ciphers are done according to a geometric design.
- Simple example is row transposition. In a columnar cipher, the original message is arranged in a rectangle, from left to right and top to bottom. Next, a key is chosen and used to assign a number to each column in the rectangle to determine the order of rearrangement. The number corresponding to the letters in the key is determined by their place in the alphabet, i.e. A is 1, B is 2, C is 3, etc. For example, if the key word is CAT and the message is THE SKY IS BLUE

C	A	T
3	1	20
T	H	E
S	K	Y
I	S	B
L	U	E

- Next, you take the letters in numerical order and that is how you would transpose the message. You take the column under A first, then the column under C, then the column under T, as a result your message "The sky is blue" has become: HKSUTSILEYBE

Columnar transposition

```python
def split_len(seq, length): # split the seq in pair equal to length of key
    return [seq[i:i + length] for i in range(0, len(seq), length)]

def encode(key, plaintext):

    order = {
        int(val): num for num, val in enumerate(key) #allows us to loop over something and have an automatic counter
    }

    ciphertext = ''
    for index in sorted(order.keys()):
        for part in split_len(plaintext, len(key)):
            try:
                ciphertext += part[order[index]]
            except IndexError:
                continue

    return ciphertext

print(encode('213', 'THESKYISBLUE'))
```

THANKYOU