Assignment 1

Joshua Hwang (44302650)

11 March

1 Caesar cipher

1.1 test_caesar.py

```
# -*- coding: utf-8 -*-
"""
Caesar cypher script

Encode and decode messages by scrambling the letters in your message

Created on Fri Feb  1 23:06:50 2019

@author: shakes
"""
import string

letters = string.ascii_letters #contains 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'

message = "The quick brown fox jumped over the lazy dog" #type your message here
print("Message:", message)

#create the Caesar cypher
offset = 1 #choose your shift
totalLetters = 26
keys = {} #use dictionary for letter mapping
invkeys = {} #use dictionary for inverse letter mapping, you could use inverse search from original dict
for index, letter in enumerate(letters):
    # cypher setup
    if index < totalLetters: #lowercase
        newIndex = (index + offset) % totalLetters
        keys[letters[index]] = letters[newIndex]
        invkeys[letters[newIndex]] = letters[index]
    else: #uppercase
        uppIndex = index - totalLetters
```

```
        newIndex = ((uppIndex + offset) % totalLetters) + totalLetters
        keys[letters[index]] = letters[newIndex]
        invkeys[letters[newIndex]] = letters[index]
print("Cypher Dict:", keys)

#encrypt
encryptedMessage = []
for letter in message:
    if letter == ' ': #spaces
        encryptedMessage.append(letter)
    else:
        encryptedMessage.append(keys[letter])
print("Encrypted Message:", ''.join(encryptedMessage)) #join is used to put list into string

#decrypt
decryptedMessage = []
for letter in encryptedMessage:
    if letter == ' ': #spaces
        decryptedMessage.append(letter)
    else:
        decryptedMessage.append(invkeys[letter])
print("Decrypted Message:", ''.join(decryptedMessage)) #join is used to put list into string
```

The output is,

```
Message: The quick brown fox jumped over the lazy dog
Cypher Dict: {'a': 'b', 'b': 'c', 'c': 'd', 'd': 'e' ...
Encrypted Message: Uif rvjdl cspxo gpy kvnqfe pwfs uif mbaz eph
Decrypted Message: The quick brown fox jumped over the lazy dog
```

1.2 test_caesar_break.py part b and c

```
# -*- coding: utf-8 -*-
"""
Determine the shift of the Caesar Cypher

Created on Sat Feb  2 23:03:02 2019

@author: shakes
"""
from collections import Counter
import string

message = "Zyp cpxpxmpc ez wzzv fa le esp delcd lyo yze ozhy le jzfc qppe Ehz ypgpc rtgp fa hzcv Hzcv rtgpd jz
```

```
#frequency of each letter
letter_counts = Counter(message)
#print(letter_counts)

#find max letter
maxFreq = -1
maxLetter = None
for letter, freq in letter_counts.items():
    print(letter, ":", freq)
    if letter != ' ' and freq > maxFreq:
        maxLetter = letter
        maxFreq = freq
print("Max Ocurring Letter:", maxLetter)

#predict shift
#assume max letter is 'e'
letters = string.ascii_letters #contains 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'

shift = ord(maxLetter) - ord('e')
print("Predicted Shift:", shift)

#attempt decipher
upperLetters = string.ascii_uppercase
lowerLetters = string.ascii_lowercase
decrypt = []
for letter in message:
    letters = [];
    if letter in upperLetters:
        letters = upperLetters
    elif letter in lowerLetters:
        letters = lowerLetters
    else:
        decrypt.append(letter)
        continue

    index = letters.index(letter)
    decrypt.append(letters[(index - shift + len(letters))%len(letters)])

print("Message:", "".join(decrypt));
```

The output is,

```
Z : 1
y : 15
p : 27
  : 50
```

```
c : 14
x : 6
m : 2
e : 18
z : 18
w : 4
v : 5
f : 9
a : 6
l : 12
s : 7
d : 6
o : 7
h : 7
j : 6
q : 4
E : 2
g : 4
r : 5
t : 13
H : 1
n : 1
D : 1
S : 1
Max Ocurring Letter: p
Predicted Shift: 11
Message: One remember to look up at the stars and not down at your feet
Two never give up work
Work gives you meaning and purpose and life is empty without it
Three if you are lucky enough to find love remember it is there and dont throw it away
Stephen Hawking
```

2 Enigma cipher

2.1 test_enigma_simple.py part a and b

```
# -*- coding: utf-8 -*-
"""
Create and test an Enigma machine encryption and decoding machine

This code is based on the implementation of the Enigma machine in Python
called pyEnigma by Christophe Goessen (initial author) and Cdric Bonhomme
https://github.com/cedricbonhomme/pyEnigma

Created on Tue Feb  5 12:17:02 2019
```

```
@author: uqscha22
"""
import enigma
import rotor

engine = enigma.Enigma(rotor.ROTOR_Reflector_A, rotor.ROTOR_I,
                               rotor.ROTOR_II, rotor.ROTOR_III, key="ABC",
                               plugs="AA BB CC DD EE")

#print(engine)

# Part a)
message = "Hello World"
print("Message:", message)
secret = engine.encipher(message)
print("Encoded Message:", secret)

#Write code to decrypt message below
#HINT: Reuse the code above to do it. You do not need to write a decrypt function.
engine = enigma.Enigma(rotor.ROTOR_Reflector_A, rotor.ROTOR_I,
                               rotor.ROTOR_II, rotor.ROTOR_III, key="ABC",
                               plugs="AA BB CC DD EE")
print("Decoded Message:", engine.encipher(secret))

#Part b)
ShakesHorribleMessage = "Vxye ajgh D yf? Ptn uluo yjgco L ws nznde czidn. Bsj ccj qdbk qjph wpw ypxvu!"

#Write code to decrypt message above
engine = enigma.Enigma(rotor.ROTOR_Reflector_A, rotor.ROTOR_I,
                               rotor.ROTOR_II, rotor.ROTOR_III, key="SSC",
                               plugs="AA BB CC DD EE")
print("Decoded Message:", engine.encipher(ShakesHorribleMessage))
```

The output is,

```
Message: Hello World
Encoded Message: Ncsmm Ywdpy
Decoded Message: Hello World
Decoded Message: What will I do? The same thing I do every night. Try and take over the world!
```

2.2 test_enigma_break.py part c, d and e

```
# -*- coding: utf-8 -*-
"""
```

```
Create and test an Enigma machine encryption and decoding machine

This code is based on the implementation of the Enigma machine in Python
called pyEnigma by Christophe Goessen (initial author) and Cdric Bonhomme
https://github.com/cedricbonhomme/pyEnigma

Created on Tue Feb  5 12:17:02 2019

@author: uqscha22
"""
import string
import enigma
import rotor

letters = string.ascii_letters #contains 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'
capitalLetters = letters[-26:]
#print(capitalLetters)

ShakesHorribleMessage = "Xm xti ca idjmq Ecokta Rkhoxuu! Kdiu gm xex oft uz yjwenv qik parwc hs emrvm sfzu qnw
crib = "Hail Shakes!"
crib_substring = "Gdwm Qopjmw!"
print(crib_substring)

##Break the code via brute force search
from itertools import product
all_keys = (''.join(x) for x in
        product(capitalLetters, capitalLetters, capitalLetters))

attempt_key = ''
attempt = crib_substring
N = 0 # number of attempts
for key in all_keys:
    engine = enigma.Enigma(rotor.ROTOR_Reflector_A, rotor.ROTOR_I,
                                    rotor.ROTOR_II, rotor.ROTOR_III, key=key,
                                    plugs="AA BB CC DD EE")
    attempt = engine.encipher(ShakesHorribleMessage)
    if crib == attempt[-len(crib_substring):]:
        attempt_key = key
        break;
    N += 1


#Print the Decoded message
engine = enigma.Enigma(rotor.ROTOR_Reflector_A, rotor.ROTOR_I,
                                rotor.ROTOR_II, rotor.ROTOR_III, key=attempt_key,
                                plugs="AA BB CC DD EE")
```

```
print(engine.encipher(ShakesHorribleMessage))
print("Tries:", N)
```

The output is,

```
Gdwm Qopjmw!
To all my loyal Spider Monkeys!
Half of you are to attack the enemy on their left flank.
Half of you are to attack the enemy on their right flank
and the rest of you shall come with me down the middle! Hail Shakes!

Tries: 11771
```

My computer took roughly 5 seconds to crack the cipher. We can use Moore's law as an approximation. Since 1940 was 80 years ago there would've been 40 doublings since then, the time would have halved every two years.

$$\frac{5 \times 2^40}{60 \times 60 \times 24 \times 365} \approx 174326$$

A 1940s computer will take 174326 years to solve.

By adding two new rotors we now have ${}^5P_3 = 60$ possible rotor combos. Not only that but we a plug board where each plug can be placed in any remaining letter giving us ${}^{26}C_10 = 5.3 \times 10^6$ (Note: you could choose not using a plug and certain ordering are not equivalent but we'll ignore those). Multiplying these possibilities gives us 3.19×10^8 times longer than our previous method.

Thus it would take 50 years to complete.

3 Code Breaking

The first part appears to be the work "ATTACK" where A is 19, T is 17 etc. The next words are "PEARL" and "HARBOR" since they fit with the other A values and also because of what happened in 1941. The remaining parts are "DECEMBER" and "SEVEN" from context and the placement of other Es.

The full message read: "ATTACK PEARL HARBOR DECEMBER SEVEN".