**Report**

1. **User Requirements**
2. Input message to be encrypted
3. Input Date of birth (DOB) of user for key generation
4. Encryption key generation
5. Encryption method – Caesar Cipher
6. Output time taken to encrypt and decrypt message
7. Output encrypted message
8. Output decrypted message
9. **Analysis**

In this assignment, we must design a program that will take a message as input from the user and encrypt it. The user should also provide their DOB to generate an encryption key using which the encrypted message can be decrypted later. The program will display the encrypted and decrypted messages, along with the time in seconds both operations took to complete. The problem domain in this assignment is the logic used to encrypt and decrypt as per the algorithm. I have chosen to implement the Caesar Cipher algorithm for this assignment. Caesar cipher is a type of substitution cipher in which each letter in the plaintext is replaced by a letter some fixed number of positions known as offset down the alphabet [1]. The main logic of this algorithm is to shift all the letters of the message by a value of offset modulo 26. For example, if the offset given is 3, the letter ‘a’ will change to ‘d’, ‘b’ to ‘e’, ‘y’ to ‘b’ and so on [4]. As the DOB of user must be used as the encryption key, the program will sum all the digits of user’s DOB and set it as the offset value. This value will be passed to both the encryption and decryption functions. For the decryption part of this algorithm, each letter in the encrypted message, it replaces each letter with an inverse alphabet shift by a value of offset modulo 26 i.e. a previous letter in the alphabet. Hence, this program will output both encrypted and decrypted messages in the end.

1. **Design**

**Pseudocode**

alphabet = 'abcdefghijklmnopqrstuvwxyz'

INPUT user message

INPUT user DOB

for each digit in DOB:

sum += digit

offset <= sum

FUNCTION encryption(message, offset):

result <= “ ”

message.lower()

for each letter in message:

if letter in alphabet:

shift <= (alphabet.index(letter) + offset) % 26

result + = alphabet[shift]

else:

result += letter

return result

encrypted <= encryption(message, offset)

FUNCTION decryption(encrypted, offset):

result <= “ ”

for each letter in encrypted:

if letter in alphabet:

shift <= (alphabet.index(letter) - offset) % 26

result + = alphabet[shift]

else:

result += letter

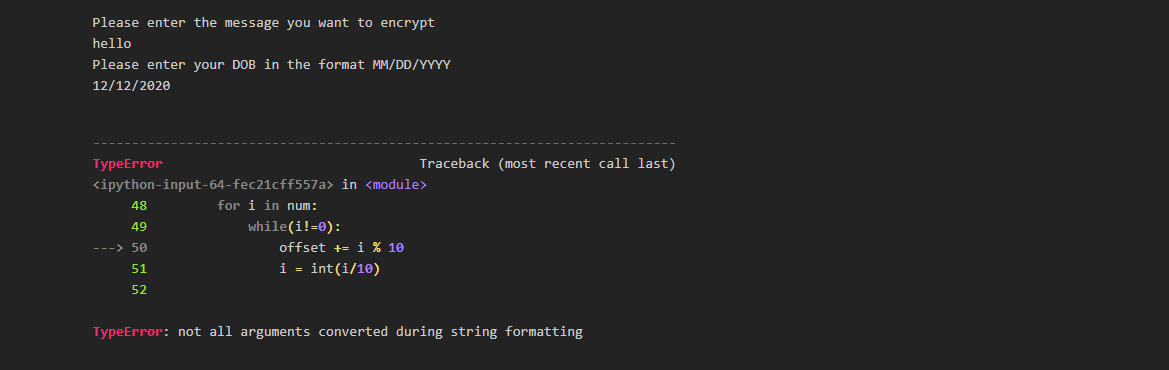
return result

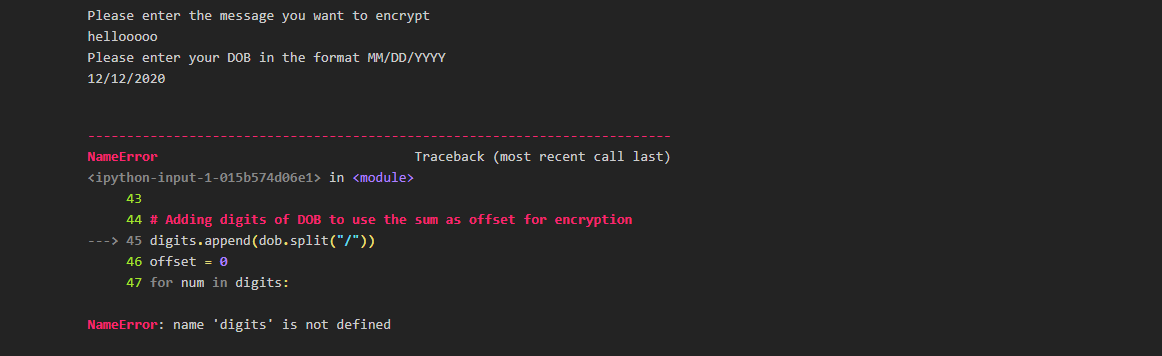
decrypted <= decryption(encrypted, offset)

PRINT encrypted

PRINT decrypted

**Few Initial Syntax Errors**

 b.



**Troubleshooting the Errors**

1. To troubleshoot the first error, I had to convert i to an integer using i = int(i).
2. I had to initialize a list ‘digits’ to solve this error.
3. **Implementation**

import time

# encrypt and decrypt a message using Caesar Cipher

key = 'abcdefghijklmnopqrstuvwxyz'

# Encryption Function

def encrypt(n, plaintext):

"""Encrypt the string and return the ciphertext"""

#Start timer

start = time.time()

result = ''

# Convert all letters to lowercase

#Substitute every letter

for l in plaintext.lower():

try:

i = (key.index(l) + n) % 26

result += key[i]

except ValueError:

result += l

#Stop timer

end = time.time()

#Calculate run time

run = end - start

print("Encryption took {:.5f} seconds".format(run))

return result.lower()

# Decryption function

def decrypt(n, ciphertext):

"""Decrypt the string and return the plaintext"""

start2 = time.time()

result = ''

for l in ciphertext:

try:

i = (key.index(l) - n) % 26

result += key[i]

except ValueError:

result += l

end2 = time.time()

run2 = end2 - start2

print("Decryption took {:.5f} seconds".format(run2))

return result

# Taking user input

digits=[]

text = input("Please enter the message you want to encrypt \n")

dob = input("Please enter your DOB in the format MM/DD/YYYY \n")

# Adding digits of DOB to use the sum as offset for encryption

digits.append(dob.split("/"))

offset = 0

for num in digits:

for i in num:

i = int(i)

while( i!=0 ):

offset += i % 10

i = int(i/10)

# Printing both encrypted and decrypted messages

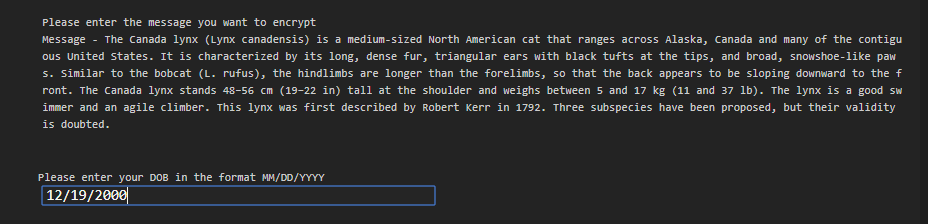
encrypted = encrypt(offset, text)

print('Encrypted:', encrypted)

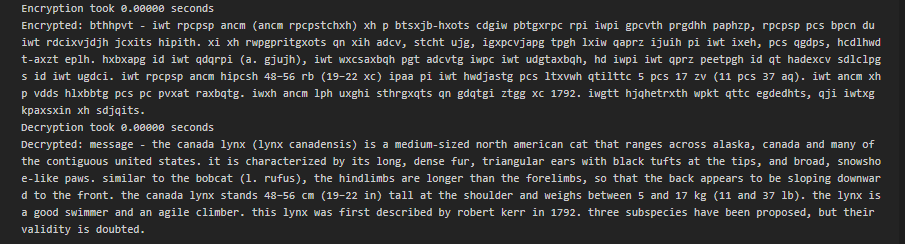
decrypted = decrypt(offset, encrypted)

print('Decrypted:', decrypted)

1. **Sample input**

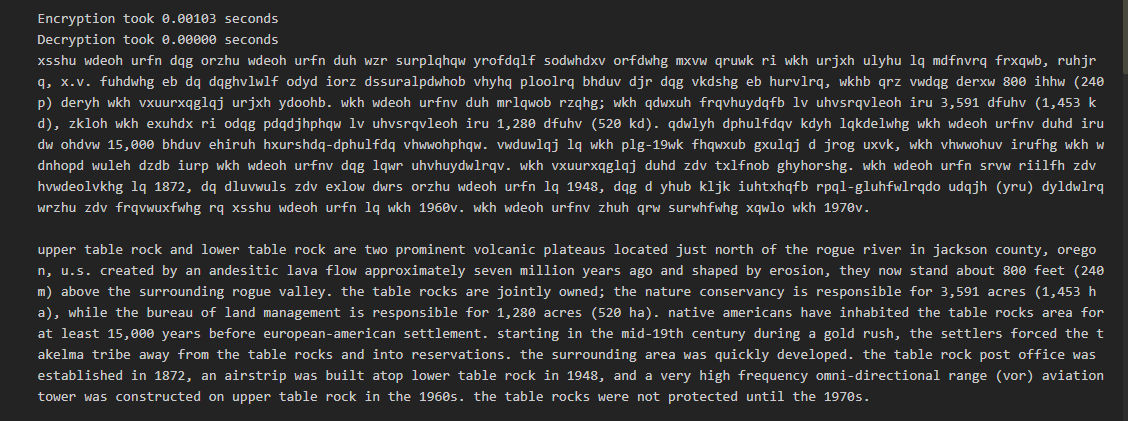


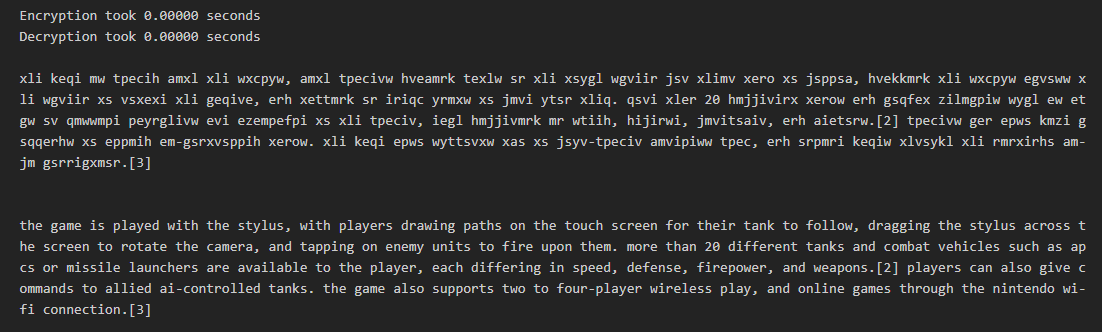
1. **Sample output**

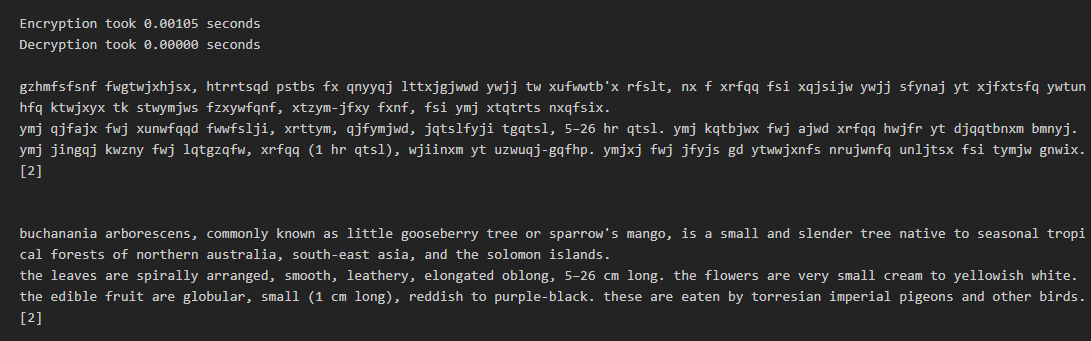


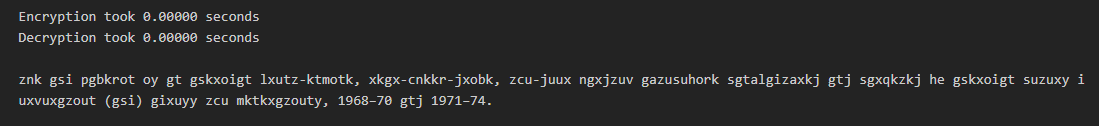
1. **Testing**

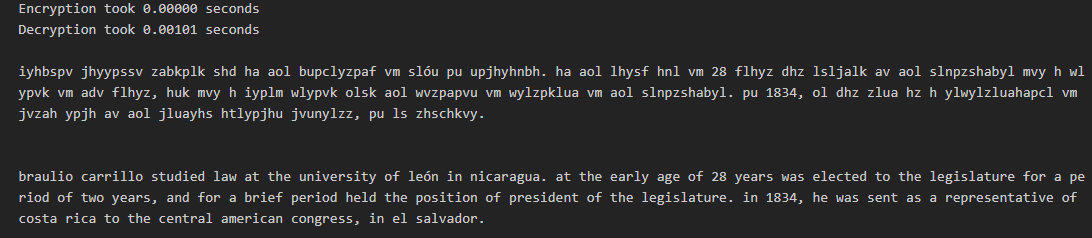
Test suite is a container that has a set of tests which helps testers in executing and reporting the test execution status. It can take any of the three states namely Active, in progress and completed. A Test case can be added to multiple test suites and test plans. After creating a test plan, test suites are created which in turn can have any number of tests [3]. To test my program, I made a text file of ten messages with varying characters and sizes, and compared the time taken by both the functions of every message of the test suite. I conclude that the program takes very less time for encryption and decryption, even for long messages. The accuracy of the output is 100%. This is because Caesar Cipher is a fairly simple encryption algorithm with a low time complexity of O(n) [2]. Following are the screenshots for some of the outputs of the test suite,

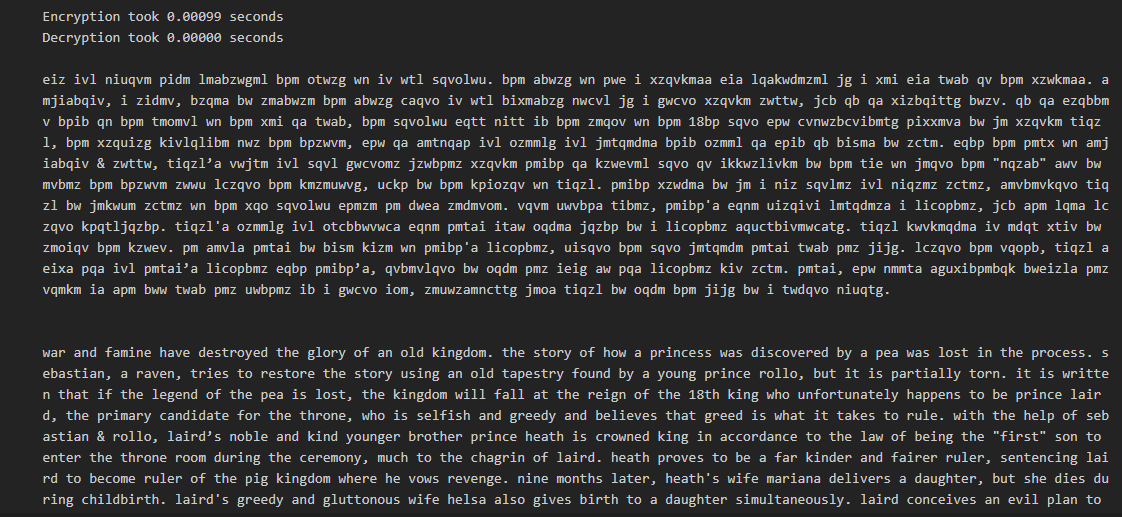












**References**

[1]B. Science Buddies, "Crack the Code! Make a Caesar Cipher", *Scientific American*, 2020. [Online]. Available: https://www.scientificamerican.com/article/crack-the-code-make-a-caesar-cipher/. [Accessed: 24- May- 2020].

[2] "Is the complexity of Caesar (shift) ciphers “n \* n!”?", *Cryptography Stack Exchange*, 2020. [Online]. Available: https://crypto.stackexchange.com/questions/22893/is-the-complexity-of-caesar-shift-ciphers-n-n. [Accessed: 24- May- 2020].

[3]"Test Suite - Tutorialspoint", *Tutorialspoint.com*, 2020. [Online]. Available: https://www.tutorialspoint.com/software\_testing\_dictionary/test\_suite.htm. [Accessed: 24- May- 2020].   
[4]"Practical Cryptography", *Practicalcryptography.com*, 2020. [Online]. Available: http://practicalcryptography.com/ciphers/caesar-cipher/. [Accessed: 24- May- 2020].