APPLIED CRYPTOGRAPHY LAB-2

NAME: VISHWAS M

SRN: PES2UG20CS390

SEC: F

DATE: 08/09/2022

Question 1

Create and display a file SRN.txt with the following contents:

srn.txt
~/Desktop/lab2

```
excited about his newly-earned powers, midas started touching every
surrounding objects like stones, grass, trees and cart, turning each item into pure
gold. but soon, midas became hungry. as he picked up a piece of food, he
found he could not eat it. it had turned to gold in his hand.
```

seed@VM: ~/.../lab2

```
[09/08/22]seed@VM:~/.../lab2$ python3
Python 3.8.5 (default, Jul 28 2020, 12:59:40)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> text=open("srn.txt","r").read()
>>> text
'EXCITED ABOUT HIS NEWLY-EARNED POWERS, MIDAS STARTED TOUCHING every\nsur
rounding objects like stones, grass, trees and cart, TURNING EACH ITEM IN
TO PURE\nGOLD. BUT SOON, MIDAS BECAME HUNGRY. AS HE PICKED UP A PIECE OF
FOOD, HE\nFOUND HE COULD NOT EAT IT. IT HAD TURNED TO GOLD IN HIS HAND.\n
'
>>> open("srn.txt","w").write(text.lower())
287
>>> exit()
[09/08/22]seed@VM:~/.../lab2$ cat srn.txt
excited about his newly-earned powers, midas started touching every
surrounding objects like stones, grass, trees and cart, turning each item
 into pure
gold. but soon, midas became hungry. as he picked up a piece of food, he
found he could not eat it. it had turned to gold in his hand.
[09/08/22]seed@VM:~/.../lab2$
```

Question 2

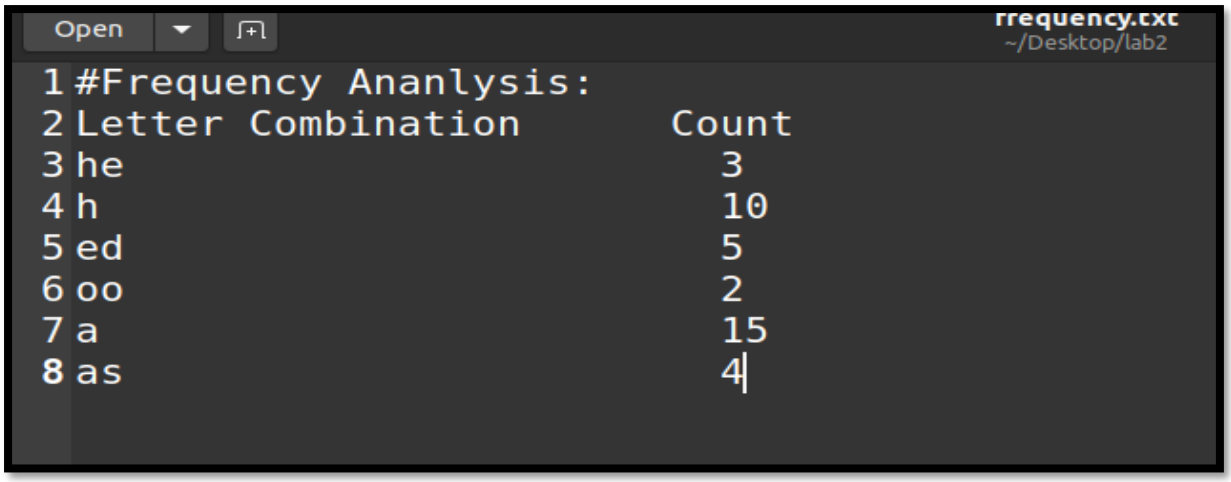
In SRN.txt, convert uppercase letters to lowercase and find the frequencies of the following words:

a. he
b. h
c. ed
d. oo
e. a
f. as

Finding Frequencies

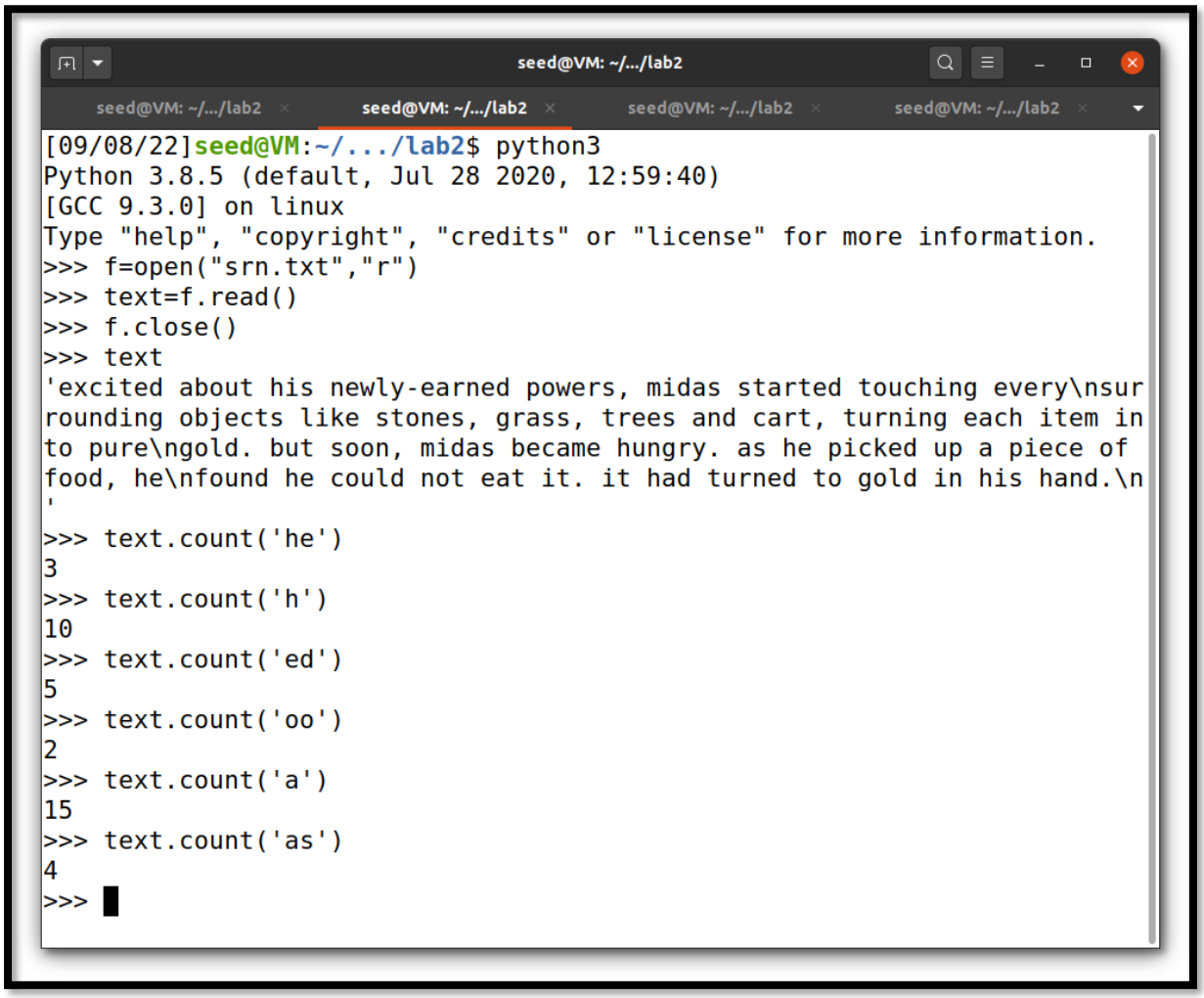
This is also achieved using python as follows:

```
seed@VM: ~/.../lab2
[09/08/22]seed@VM:~/.../lab2$ python3
Python 3.8.5 (default, Jul 28 2020, 12:59:40)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> f=open("srn.txt","r")
>>> text=f.read()
>>> f.close()
>>> text
'excited about his newly-earned powers, midas started touching every\nsur
rounding objects like stones, grass, trees and cart, turning each item in
to pure\ngold. but soon, midas became hungry. as he picked up a piece of
food, he\nfound he could not eat it. it had turned to gold in his hand.\n
'
>>> text.count('he')
3
>>> text.count('h')
10
>>> text.count('ed')
5
>>> text.count('oo')
2
>>> text.count('a')
15
>>> text.count('as')
4
>>>
```



Converting to lowercase

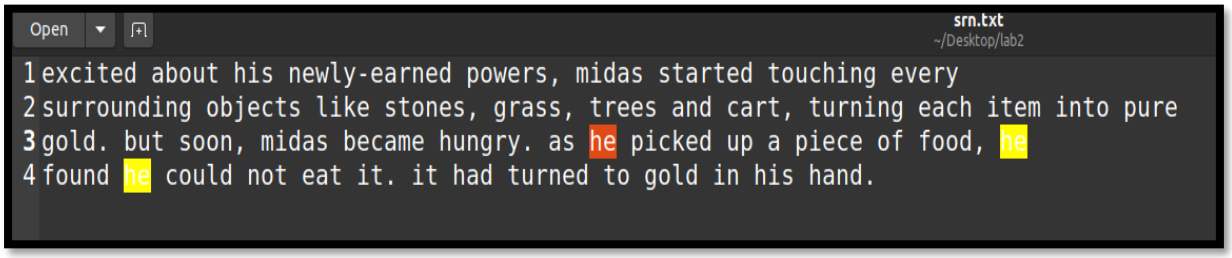
A simple python script helps achieve this:



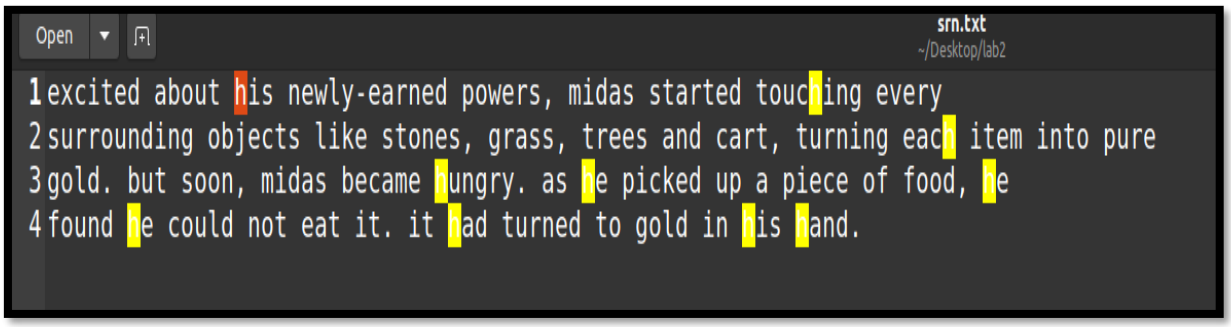
Question 3

Highlighting the words given in question 2

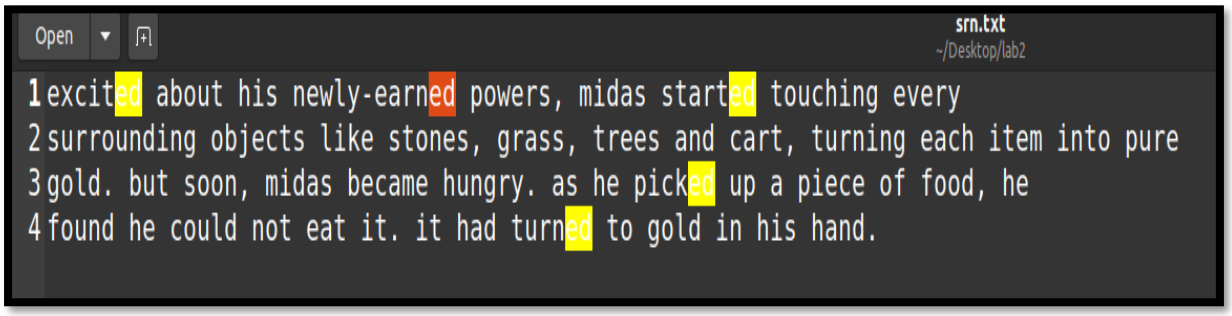
a) Occurrences of 'he' highlighted:



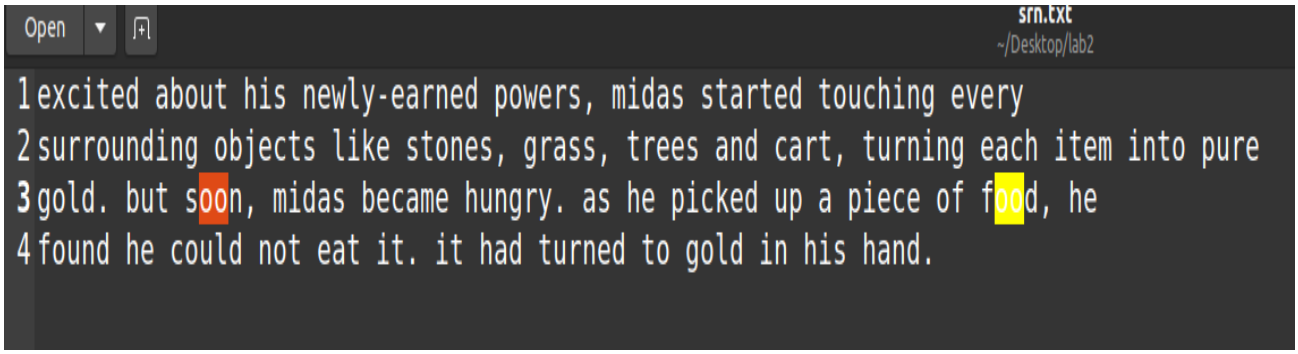
b) Occurrences of 'h' highlighted:



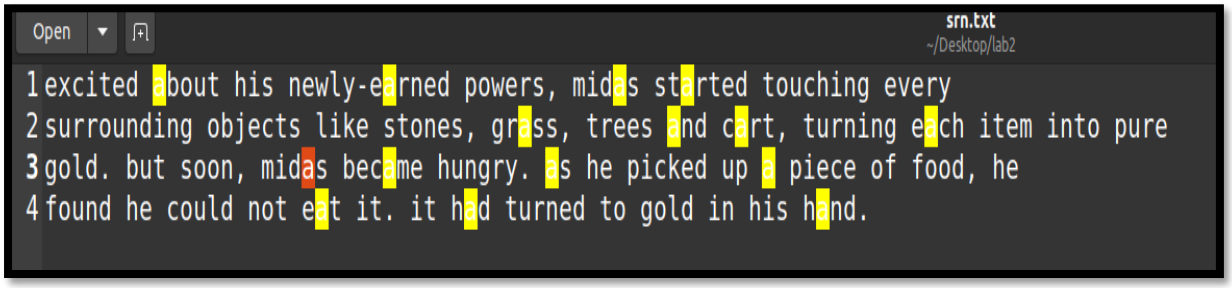
c) Occurrences of 'ed' highlighted:



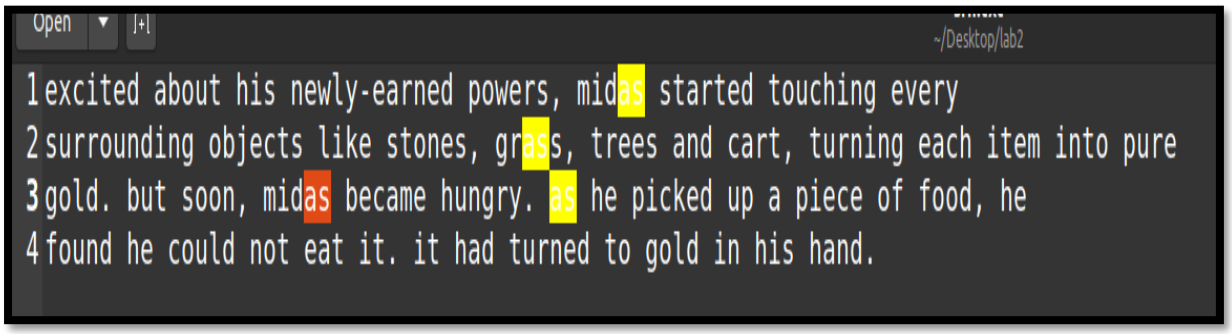
d) Occurrences of 'oo' highlighted:



e)Occurrences of 'a' highlighted:



f) Occurrences of 'as' highlighted:



Question 4

Generate the substitution cipher key
Python's random module has a 'shuffle' functionality that lets us generate random permutations of a list. This has been used to generate the
substitution cipher key from the alphabet. However, a key can be generated from online sources like random.org as well.

Function used to generate the key:

```python
def generate_key(alphabet_string):
    """
        Generates a substitution key,
        given a string of the alphabet
    """
    import random as r
    l=list(alphabet_string)
    r.shuffle(l)
    return ''.join(l)
```

Key generation:

```
[09/08/22]seed@VM:~/.../lab2$ python3
Python 3.8.5 (default, Jul 28 2020, 12:59:40)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from util import *
>>> generate_key('abcdefghijklmnopqrstuvwxyz')
'jidxgukfzvtemhqycnpwosrlab'
>>> generate_key('abcdefghijklmnopqrstuvwxyz')
'gunhmezrvydlkjxfwoctasbiqp'
>>> generate_key('abcdefghijklmnopqrstuvwxyz')
'wserkcmqfltgbznxyaivuphojd'
>>> generate_key('abcdefghijklmnopqrstuvwxyz')
'xywanqdbrhplfjtgkoevsumczi'
>>> generate_key('abcdefghijklmnopqrstuvwxyz')
'pusekjdngbqxaowfzlmyhivtrc'
>>> generate_key('abcdefghijklmnopqrstuvwxyz')
'hjdyqbeksgopnmfizcvuwlaxtr'
>>> generate_key('abcdefghijklmnopqrstuvwxyz')
'pounekqdcmljabgxizstrhywvf'
>>>
```

Question 5

Generate the ciphertext using the key generated in question 4

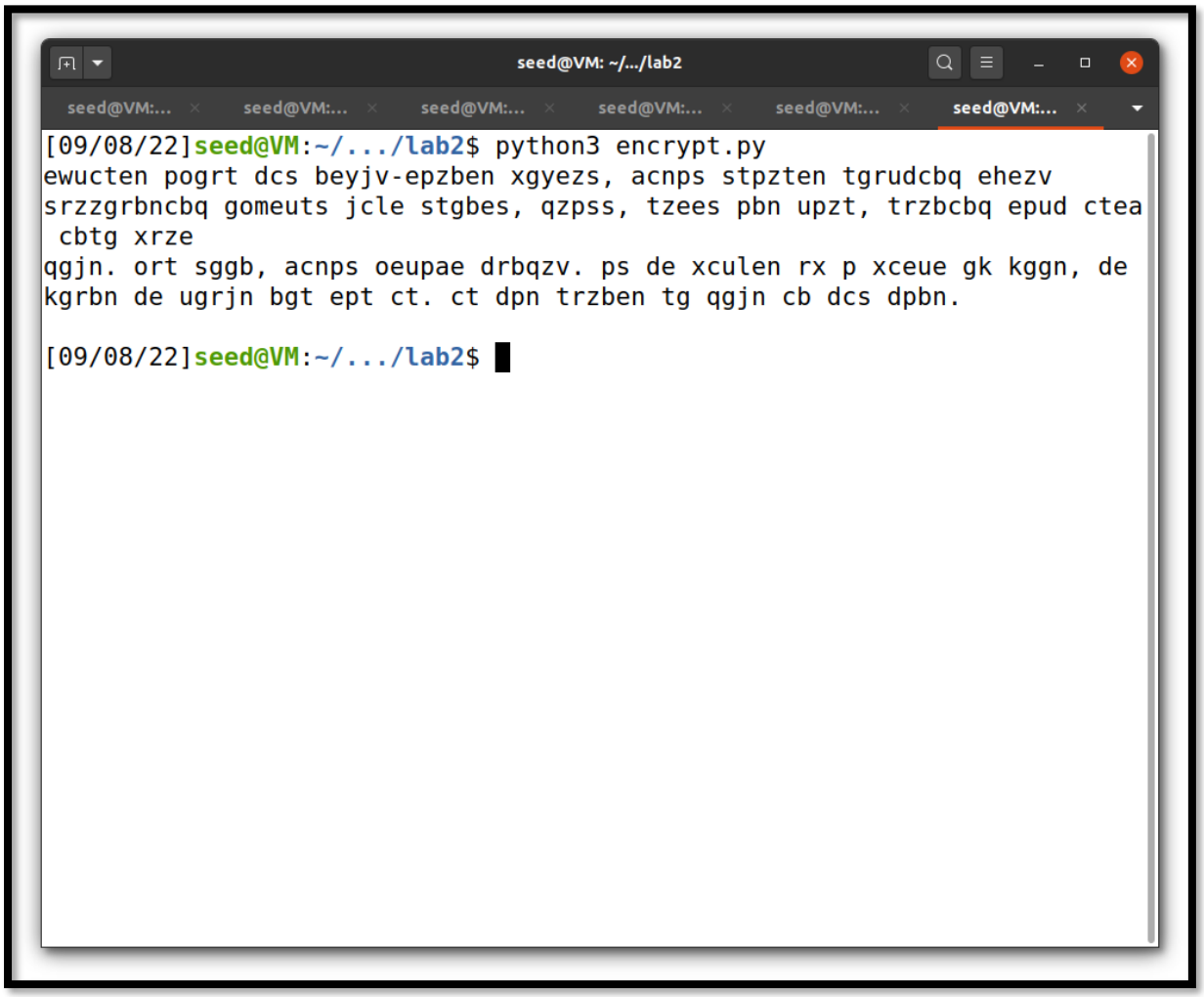
Writing a python script to achieve it:

encrypt.py
~/Desktop/lab2

```python
#!.usr/bin/env.python3

#Read the plaintext
f=open('srn.txt','r')
plaintext=f.read()
f.close()

#ciphertext variable
ciphertext=''

alphabet='abcdefghijklmnopqrstuvwxyz'
#TODO:Change This

key='pounekqdcmljabgxizstrhywvf'

#iterate over every character in plaintext

for i in range(len(plaintext)):
   #If current plaintext character is not in the alphabet
   if plaintext[i] not in alphabet:
       ciphertext +=plaintext[i]
   # If the current plaintext character is present in the alphabet
   else:
      index_in_alphabet = alphabet.index(plaintext[i])
      ciphertext +=key[index_in_alphabet]

print(ciphertext)
```

Python Tab Width: 8 Ln 25, Col 42 INS

Ciphertext generated:



Question 6

Decrypt the ciphertext back to plaintext

The script used to achieve it:

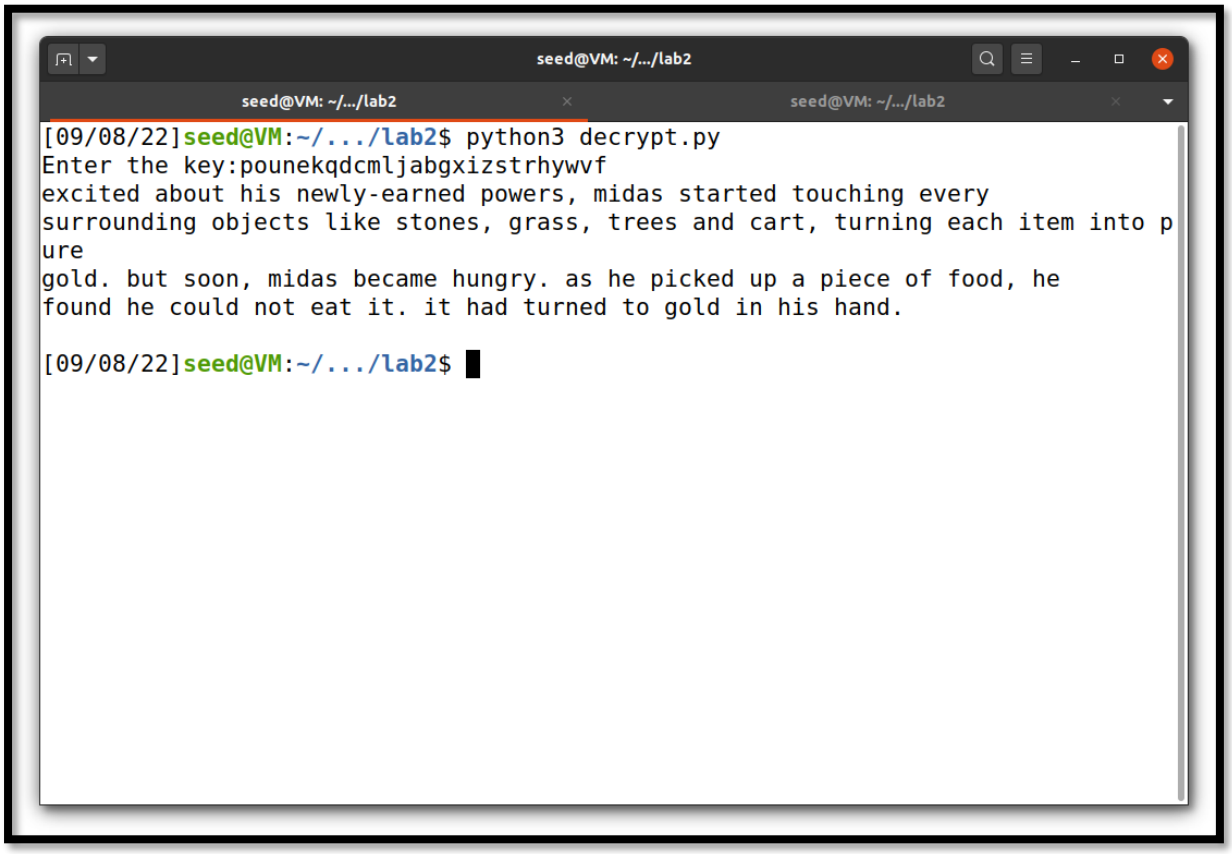
encrypt.py
~/Desktop/lab2

encrypt.py | decrypt.py | util.py

```python
#!.usr/bin/env.python3

#Read the plaintext
f=open('srn.txt','r')
plaintext=f.read()
f.close()

#ciphertext variable
ciphertext=''

alphabet='abcdefghijklmnopqrstuvwxyz'
#TODO:Change This

key='pounekqdcmljabgxizstrhywvf'

#iterate over every character in plaintext

for i in range(len(plaintext)):
  #If current plaintext character is not in the alphabet
  if plaintext[i] not in alphabet:
      ciphertext +=plaintext[i]
  # If the current plaintext character is present in the alphabet
  else:
     index_in_alphabet = alphabet.index(plaintext[i])
     ciphertext +=key[index_in_alphabet]

print(ciphertext)
```

Python | Tab Width: 8 | Ln 14, Col 6 | INS

Decrypted plaintext:



Question 7

Suppose the input file is:

Hungry, Midas groaned, "I'll starve! Perhaps this was not such an excellent wish after all!"…

Ciphertext generated:

```
[09/08/22]seed@VM:~/.../lab2$ python3 encryt.py
python3: can't open file 'encryt.py': [Errno 2] No such file or directory
[09/08/22]seed@VM:~/.../lab2$ python3 encrypt.py
Hrbqzv, Mcnps qzgpben, "I'jj stpzhe! Pezdpxs tdcs yps bgt srud pb ewuejjebt ycsd
 pktez pjj!"…

[09/08/22]seed@VM:~/.../lab2$
```

Decrypted plaintext:

```
[09/08/22]seed@VM:~/.../lab2$ python3 decrypt.py
Enter the key:pounekqdcmljabgxizstrhywvf
Hungry, Midas groaned, "I'll starve! Perhaps this was not such an excellent wish
 after all!"…


[09/08/22]seed@VM:~/.../lab2$
```

Comment on the ciphertext generated:

This ciphertext is shorter than the one before. The previous ciphertext, due to its long length, is more permeable to frequency analysis attacks. This ciphertext, though still insecure, is less vulnerable to frequency analysis attacks.
However, care must be taken to remove all punctuations, lest they give attackers hints as to the contents of the message (for example, it's pretty obvious that “ z'yy ” stands for “ i'll ”, hence two letters of the key are revealed).