SOURCE CODE-ENCRYPTION-DECRYPTION

import os

import secrets

import shutil

from pathlib import Path

from string import ascii\_letters

from time import sleep

from Crypto import Random

from Crypto.Cipher import AES

def generate\_random\_password():

flag = True

print('\nPlease enter the number of bytes in the password')

print('Allowed Values:16, 24, 32')

while flag:

nbytes = int(input('Byte number:'))

if nbytes in [16, 24, 32]:

flag = False

else:

print('ERROR: Please enter one of the options given above')

random\_password = ''

ascii\_extended = ascii\_letters + '0123456789' + \

r'!"#$%&()\*+,-./;<>?@[\]^\_`{|}~'

for i in range(nbytes):

random\_password += secrets.choice(ascii\_extended)

return random\_password

def generate\_password():

flag = True

while flag:

password = input('\nEnter a password:')

passwordLen = len(password)

if passwordLen in [16, 24, 32]:

flag = False

else:

print(

'ERROR: Your password have {} characters. The length of the password must be 16, 24 or 32'.format(passwordLen))

return password

def pad(s):

padding\_size = AES.block\_size - len(s) % AES.block\_size

return s + b"\0" \* padding\_size, padding\_size

def encrypt(message, key):

message, padding\_size = pad(message)

iv = Random.new().read(AES.block\_size)

cipher = AES.new(key, AES.MODE\_CFB, iv)

enc\_bytes = iv + cipher.encrypt(message) + bytes([padding\_size])

return enc\_bytes

def decrypt(ciphertext, key):

iv = ciphertext[:AES.block\_size]

cipher = AES.new(key, AES.MODE\_CFB, iv)

plaintext = cipher.decrypt(ciphertext[AES.block\_size:-1])

padding\_size = ciphertext[-1] \* (-1)

return plaintext[:padding\_size]

def encrypt\_file(filePATH, key):

with open(filePATH, 'rb+') as in\_file:

plaintext = in\_file.read()

in\_file.seek(0)

enc = encrypt(plaintext, key)

in\_file.write(enc)

in\_file.truncate()

encfilePATH = filePATH + '.enc'

os.rename(filePATH, encfilePATH)

in\_file.close()

def decrypt\_file(encfolderPATH, encfilePATH, key):

with open(encfilePATH, 'rb+') as in\_file:

plaintext = in\_file.read()

dec = decrypt(plaintext, key)

with open(encfolderPATH[:-3] + encfilePATH[len(encfolderPATH):-4], 'wb') as out\_file:

out\_file.write(dec)

out\_file.close()

def get\_all\_filePaths(folderPATH):

result = []

for dirpath, dirnames, filenames in os.walk(folderPATH):

result.extend([os.path.join(dirpath, filename)

for filename in filenames])

return result

def get\_all\_folderPATHS(folderPATH):

folderPATHS = []

for dirpath, dirnames, filenames in os.walk(folderPATH):

folderPATHS.append(dirpath)

return folderPATHS

def mkdir\_folder(folderPATHS):

mainfolderPATH = folderPATHS[0][:-3]

p = Path(mainfolderPATH)

p.mkdir(parents=True, exist\_ok=True)

for folderPATH in folderPATHS[1:]:

folderPATH = mainfolderPATH + folderPATH[len(mainfolderPATH)+3:]

p = Path(folderPATH)

p.mkdir(parents=True, exist\_ok=True)

# ----- Encrypting/Decrypting every file inside a folder ----- #

def encrypt\_folder(folderPATH, key):

for filePATH in get\_all\_filePaths(folderPATH):

encrypt\_file(filePATH, key)

encfolderPATH = folderPATH + 'ENC'

os.rename(folderPATH, encfolderPATH)

def decrypt\_folder(encfolderPATH, key):

for encfilePATH in get\_all\_filePaths(encfolderPATH):

decrypt\_file(encfolderPATH, encfilePATH, key)

def run\_folderlocker\_encryption():

print('\nPlease type the path of the folder')

flagPATH = True

while flagPATH:

folderPATH = input('PATH:')

if folderPATH[-3:] == 'ENC':

print(

'ERROR: Encryption cannot be applied to the folders with ENC extension!')

print('\nPlease re-enter the path')

elif folderPATH[-3:] != 'ENC':

flagPATH = False

print('\nPlease choose the type of the encryption')

print('------------')

print('Type "rp" to generate random password')

print('Type "up" to enter a password')

print('------------')

flag\_encryption\_type = True

while flag\_encryption\_type:

encryption\_type = input('Encryption type:')

if encryption\_type != 'rp' and encryption\_type != 'up':

print('ERROR: Please type one of the commands given above!')

else:

flag\_encryption\_type = False

if encryption\_type == 'rp':

password = generate\_random\_password()

print('Generating random password...')

sleep(1.5)

elif encryption\_type == 'up':

password = generate\_password()

key = str.encode(password)

print('\nIMPORTANT: Save this password to decrypt your folder!')

print('PASSWORD:{}\n'.format(password))

print('Encrypting the folder...')

sleep(1.5)

encrypt\_folder(folderPATH, key)

print('Encryption is successful!')

def run\_folderlocker\_decryption():

print('\nPlease type the path of the folder')

flagPATH = True

while flagPATH:

encfolderPATH = input('PATH:')

if encfolderPATH[-3:] != 'ENC':

print(

'ERROR: Decryption can only be applied to the folders with ENC extension!')

print('\nPlease re-enter the path')

else:

flagPATH = False

print('\nPlease enter the password')

password = input('PASSWORD:')

key = str.encode(password)

print('\nDecrypting the folder...')

sleep(1.5)

mkdir\_folder(get\_all\_folderPATHS(encfolderPATH))

try:

decrypt\_folder(encfolderPATH, key)

print('Decryption is successful!')

print('\nDo you want to remove the encrypted folder?')

print('[Y]: Yes\t[N]: No')

flag\_answer = True

while flag\_answer:

answer = input('')

if answer != 'Y' and answer != 'N':

print('ERROR: Please type one of the commands given above!')

else:

flag\_answer = False

if answer == 'Y':

shutil.rmtree(encfolderPATH, ignore\_errors=True)

print('\nRemoving the encrypted folder...')

sleep(1.5)

elif answer == 'N':

pass

except:

print('\nDecryption is not successful!')

print('Please enter the correct password')

shutil.rmtree(encfolderPATH[:-3], ignore\_errors=True)

def run\_folderlocker():

print('\t--Welcome to the Folder Locker--\n')

print('Do you want to encrypt or decrypt the folder?')

print('[E]: Encrypt\t[D]: Decrypt')

flag\_method = True

while flag\_method:

answer = input('')

if answer != 'E' and answer != 'D':

print('ERROR: Please type one of the commands given above!')

else:

flag\_method = False

if answer == 'E':

run\_folderlocker\_encryption()

os.system('pause')

print('\nThis page will close in 90 seconds. Please save the password to decrypt the folder!')

sleep(90)

elif answer == 'D':

run\_folderlocker\_decryption()

os.system('pause')

print('\nThis page will close in 10 seconds')

sleep(10)

run\_folderlocker()