# MUST RUN CODE AS ADMINISTRATOR IN ORDER TO CREATE REGISTRY CHANGES!  
import winreg  
from tkinter import \*  
import glob  
import hashlib  
import os  
import sys  
import base64  
from cryptography.fernet import Fernet  
from cryptography.hazmat.backends import default\_backend  
from cryptography.hazmat.primitives import hashes  
from cryptography.hazmat.primitives.kdf.pbkdf2 import PBKDF2HMAC  
  
  
def main():  
 #check if registry changes been made already:  
 try:  
 reg=winreg.ConnectRegistry(None,winreg.HKEY\_CLASSES\_ROOT)  
 k=winreg.OpenKey(reg,"Directory\\Shell\\encrypt/decrypt\\command",0,winreg.KEY\_READ)  
 except:  
 # adding registry value that will appear as option on right-click on a folder:  
 key = winreg.CreateKeyEx(winreg.HKEY\_CLASSES\_ROOT, "Directory\\Shell\\encrypt/decrypt", 0, winreg.KEY\_WRITE)  
 subkey = winreg.CreateKeyEx(winreg.HKEY\_CLASSES\_ROOT, "Directory\\Shell\\encrypt/decrypt\\command", 0,winreg.KEY\_WRITE)  
 # the python file that will run when choose 'encrypt/decrypt option in a folder:  
 winreg.SetValueEx(subkey, "", 0, winreg.REG\_SZ, r"C:\Users\Johnny\AppData\Local\Programs\Python\Python37\python.exe C:/Users/Johnny/PycharmProjects/folderEncryption/venv/main.py %1 '-c'")  
  
  
 #after run the python file , a window (GUI) will open and let us to manipulate the folder - using Tkinter on python:  
 root = Tk() #just the window that opens..  
 title = Label(root,text="encrypt/decrypt")  
 lineSpace=Label(root,text=" ")  
 subtitle = Label(root, text="insert user & password for encrypt/decrypt:")  
 title.grid(row=1,column=1)  
 lineSpace.grid(row=2,column=1)  
 subtitle.grid(row=3,column=1)  
 userLabel=Label(root,text="user: ")  
 passLabel=Label(root,text="password: ")  
 userLabel.grid(row=4,column=1)  
 passLabel.grid(row=5,column=1)  
 userInputUser=Entry(root,fg="red")  
 userInputUser.grid(row=4,column=2)  
 userInputPassword=Entry(root,fg="red")  
 userInputPassword.grid(row=5,column=2)  
 #TOTALLY ME!!!! GOT TO GET DIRECTORY FROM REGISTRY OF THE RIGHT CLICK OUTSIDE MY CODE!  
 try:  
 path=sys.argv[1]  
 except:  
 path=sys.argv[0]  
 mylabel = Label(root,text=path)  
 mylabel.grid(row=0,column=0)  
  
 def encryptPressed():  
 #encrypt all files in folder:  
 salt=userInputUser.get()  
 salt=hashlib.sha256().digest()  
 passwordProvided = userInputPassword.get()  
 password=passwordProvided.encode()  
 kdf=PBKDF2HMAC(  
 algorithm=hashes.SHA256,  
 length=32,  
 salt=salt,  
 iterations=1000000,  
 backend=default\_backend()  
 )  
 key = base64.urlsafe\_b64encode(kdf.derive(password)) #FERNET KEY FOR AES !!!!  
 Cipher = Fernet(key)  
 files = [f for f in glob.glob(path + "\*\*/\*.\*", recursive=True)]  
 for f in files:  
 input\_file = f  
 output\_file = f  
 # read the bytes from current file data and store it in data variable.  
 with open(input\_file, 'rb') as currentFile:  
 data = currentFile.read()  
 # encrypt data with fernet(AES) and the known key that we use..  
 encrypted = Cipher.encrypt(data)  
 # write the encrypted data instead current data:  
 with open(output\_file, 'wb') as currentFile:  
 currentFile.write(encrypted)  
  
 def decryptPressed():  
 # decrypt all files in folder:  
 salt = userInputUser.get()  
 salt = hashlib.sha256().digest()  
 passwordProvided = userInputPassword.get()  
 password = passwordProvided.encode()  
 kdf = PBKDF2HMAC(  
 algorithm=hashes.SHA256,  
 length=32,  
 salt=salt,  
 iterations=1000000,  
 backend=default\_backend()  
 )  
 key = base64.urlsafe\_b64encode(kdf.derive(password)) # FERNET KEY FOR AES !!!!  
 Cipher = Fernet(key)  
 files = [f for f in glob.glob(path + "\*\*/\*.\*", recursive=True)]  
 for f in files:  
 input\_file = f  
 output\_file = f  
 # read the bytes from current file data and store it in data variable.  
 with open(input\_file, 'rb') as currentFile:  
 data = currentFile.read()  
 # decrypt data with fernet(AES) and the known key that we use..  
 decrypted = Cipher.decrypt(data)  
 # write the decrypted data instead current data:  
 with open(output\_file, 'wb') as currentFile:  
 currentFile.write(decrypted)  
  
  
 encryptBtn = Button(root, text="encrypt!", fg="blue",command=encryptPressed)  
 decryptBtn = Button(root, text="decrypt!",fg="blue",command=decryptPressed)  
 encryptBtn.grid(row=6, column=0, padx=50)  
 decryptBtn.grid(row=6, column=3, padx=50)  
  
 root.mainloop()  
  
if \_\_name\_\_=="\_\_main\_\_":  
 main()