Daniel Beck

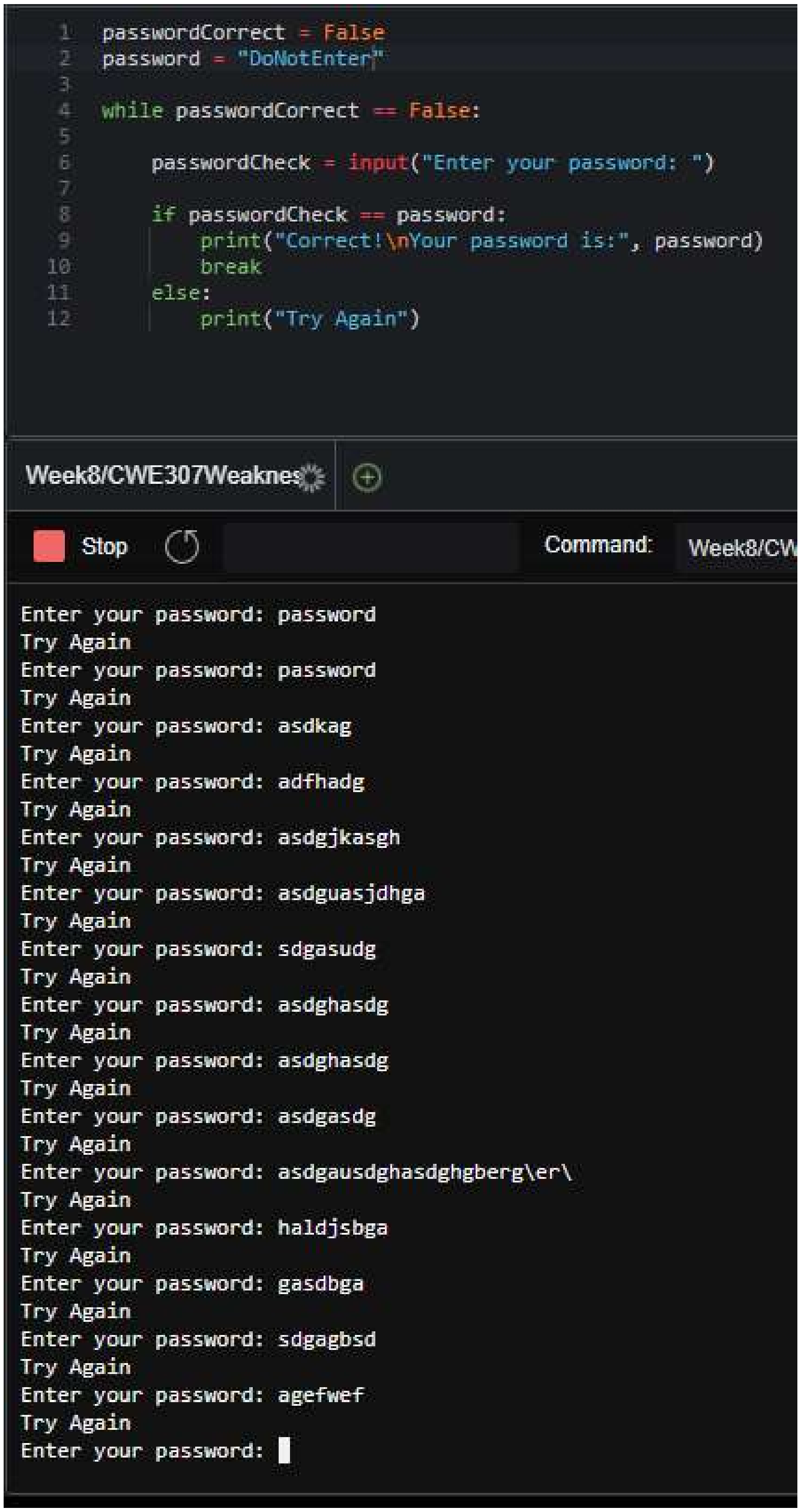
SDEV-325

October 13, 2020

1. The first software vulnerability that was addressed was CWE-307: Improper Restriction of Excessive Authentication Attempts. A CWE-307 vulnerability occurs when "The software does not implement sufficient measures to prevent multiple failed authentication attempts within in a short time frame, making it more susceptible to brute force attacks (*Common Weakness Enumeration,* 2020)." If there are no restrictions on how many attempts can be made, then hackers can continuously attack an account until access is granted. 1a shows a program that continuously will allow someone trying to gain access to the account, multiple attempts, without restriction. 1b shows the result of the weakness being mitigated by adding a counter to the password attempts and exits the program after four failed attempts.

2. The second software vulnerability that was addressed was CWE-327: Use of a Broken or Risky Cryptographic Algorithm. A CWE-327 vulnerability occurs when "The use of a broken or risky cryptographic algorithm is an unnecessary risk that may result in the exposure of sensitive information (*Common Weakness Enumeration,* 2020)." If users' passwords are not encrypted correctly, there is a chance that their passwords may be exposed, or the password may become lost, locking the user out of their account. 2a shows what it may look like when using a python extension, cryptography.fernet that has become broken and is not properly encrypting/decrypting passwords. The output of the program shows that the password that is stored does not get decrypted correctly. 2b shows the mitigation of the vulnerability by using a corrected method that runs the password through a cryptography algorithm.

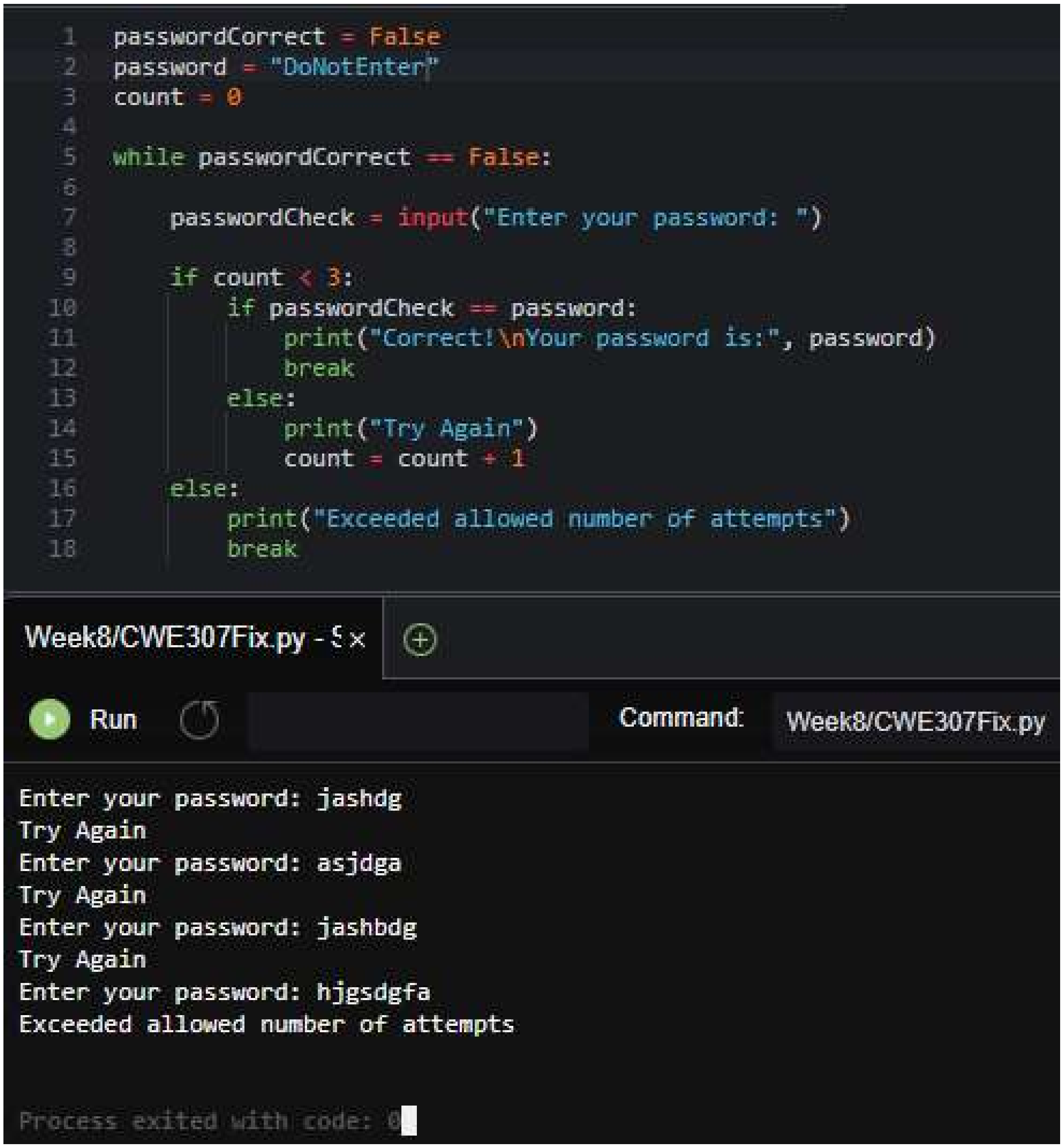
1a.



```python
passwordCorrect = False
password = "DoNotEnter"

while passwordCorrect == False:

    passwordCheck = input("Enter your password: ")

    if passwordCheck == password:
        print("Correct!\nYour password is:", password)
        break
    else:
        print("Try Again")
```

```
Enter your password: password
Try Again
Enter your password: password
Try Again
Enter your password: asdkag
Try Again
Enter your password: adfhadg
Try Again
Enter your password: asdgjkasgh
Try Again
Enter your password: asdguasjdhga
Try Again
Enter your password: sdgasudg
Try Again
Enter your password: asdghasdg
Try Again
Enter your password: asdghasdg
Try Again
Enter your password: asdgasdg
Try Again
Enter your password: asdgausdghasdghgberg\er\
Try Again
Enter your password: haldjsbga
Try Again
Enter your password: gasdbga
Try Again
Enter your password: sdgagbsd
Try Again
Enter your password: agefwef
Try Again
Enter your password: 
```

1b.



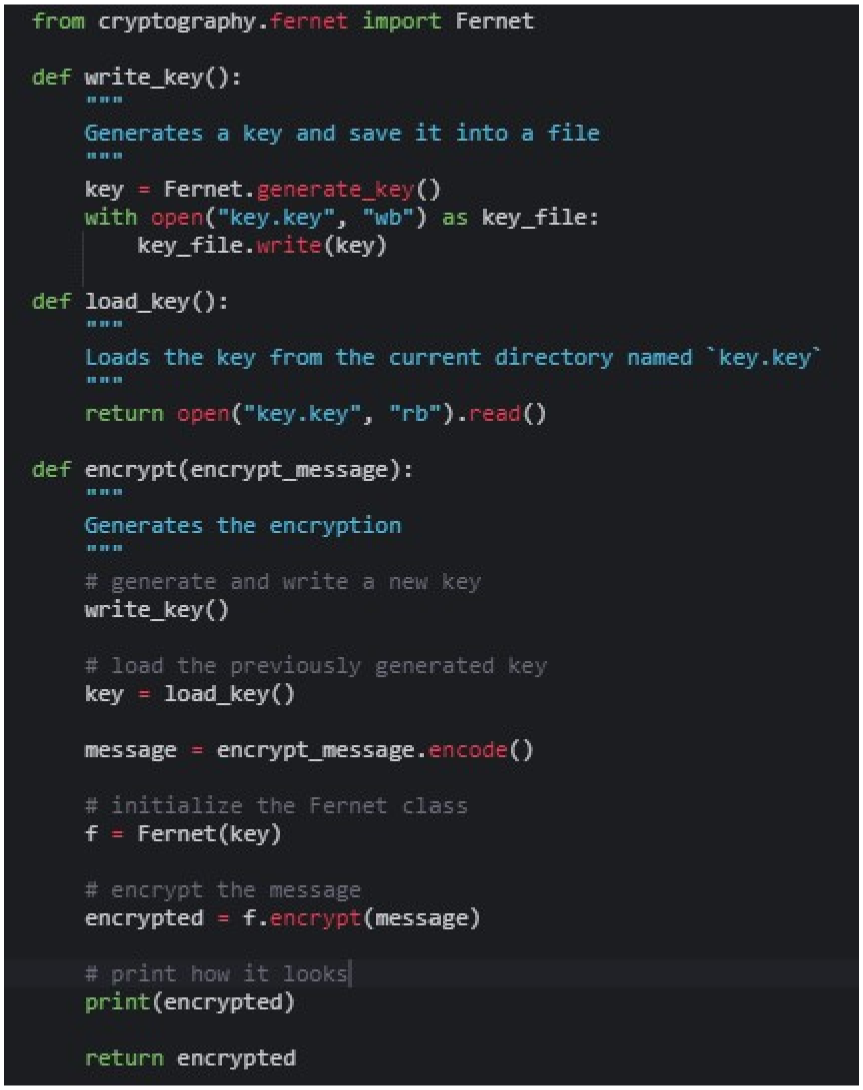
```python
passwordCorrect = False
password = "DoNotEnter"
count = 0

while passwordCorrect == False:

    passwordCheck = input("Enter your password: ")

    if count < 3:
        if passwordCheck == password:
            print("Correct!\nYour password is:", password)
            break
        else:
            print("Try Again")
            count = count + 1
    else:
        print("Exceeded allowed number of attempts")
        break
```

Week8/CWE307Fix.py - S ×

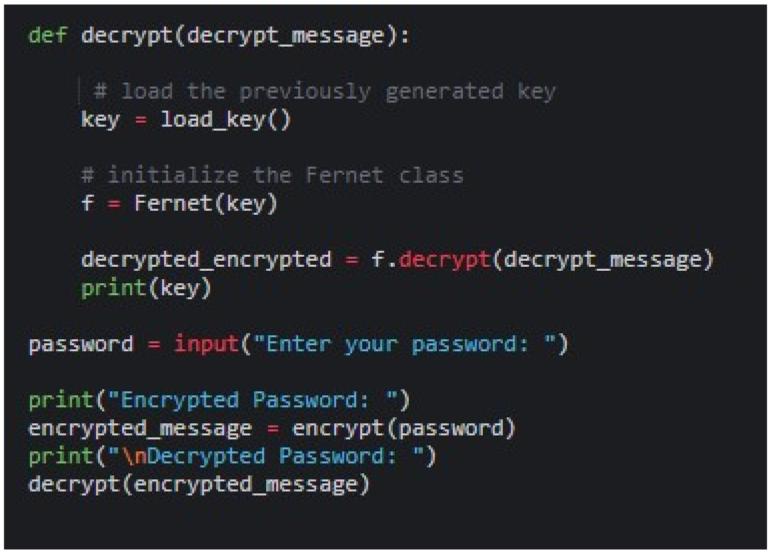
Run   Command: Week8/CWE307Fix.py

```
Enter your password: jashdg
Try Again
Enter your password: asjdga
Try Again
Enter your password: jashbdg
Try Again
Enter your password: hjgsdgfa
Exceeded allowed number of attempts


Process exited with code: 0
```

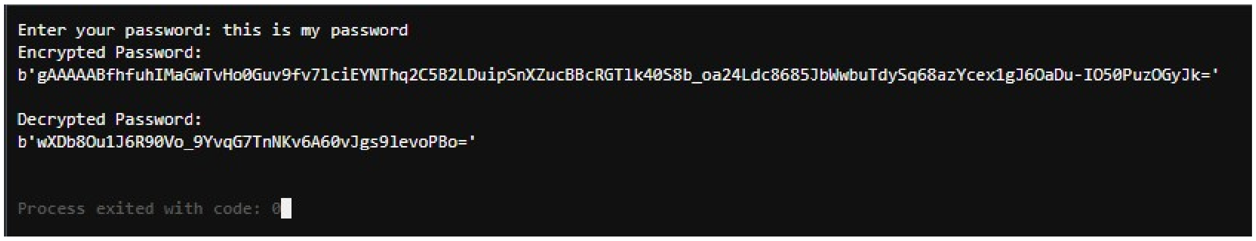
2a.



```python
from cryptography.fernet import Fernet

def write_key():
    """
    Generates a key and save it into a file
    """
    key = Fernet.generate_key()
    with open("key.key", "wb") as key_file:
        key_file.write(key)

def load_key():
    """
    Loads the key from the current directory named `key.key`
    """
    return open("key.key", "rb").read()

def encrypt(encrypt_message):
    """
    Generates the encryption
    """
    # generate and write a new key
    write_key()

    # load the previously generated key
    key = load_key()

    message = encrypt_message.encode()

    # initialize the Fernet class
    f = Fernet(key)

    # encrypt the message
    encrypted = f.encrypt(message)

    # print how it looks
    print(encrypted)

    return encrypted
```

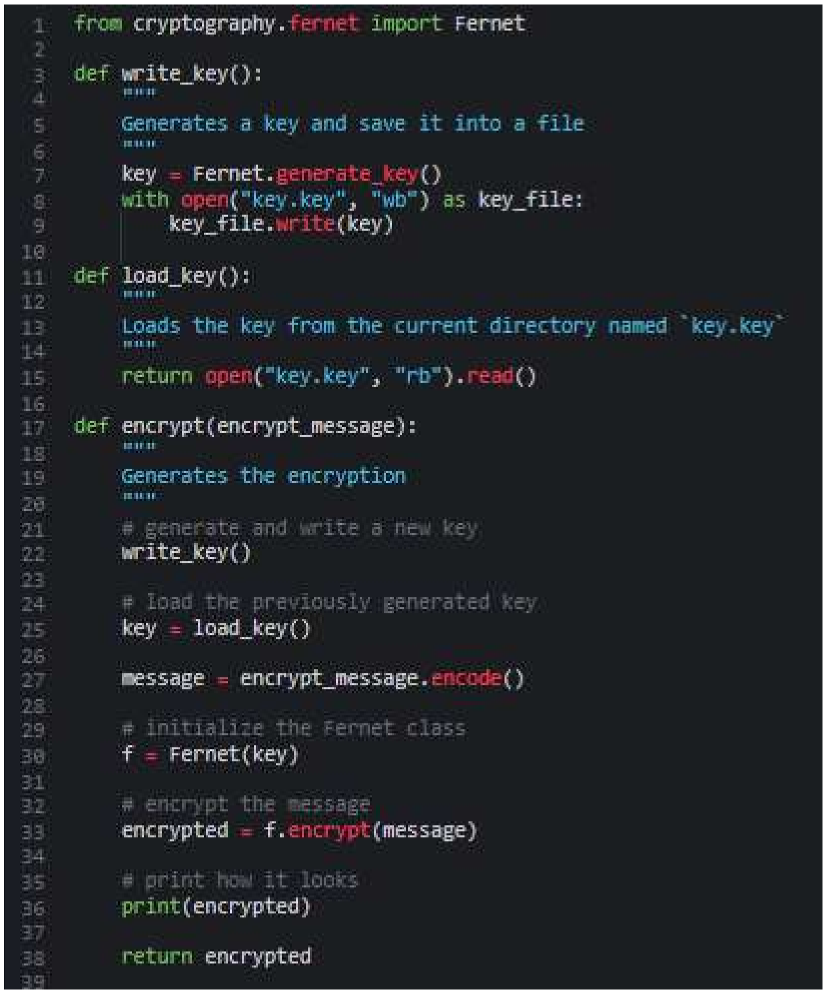


```python
def decrypt(decrypt_message):

    # load the previously generated key
    key = load_key()

    # initialize the Fernet class
    f = Fernet(key)

    decrypted_encrypted = f.decrypt(decrypt_message)
    print(key)

password = input("Enter your password: ")

print("Encrypted Password: ")
encrypted_message = encrypt(password)
print("\nDecrypted Password: ")
decrypt(encrypted_message)
```

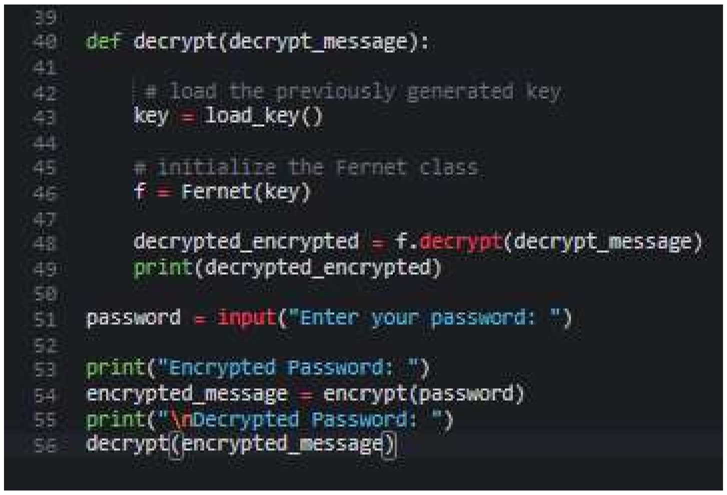


```
Enter your password: this is my password
Encrypted Password:
b'gAAAAABfhfuhIMaGwTvHo0Guv9fv71ciEYNThq2C5B2LDuipSnXZucBBcRGT1k40S8b_oa24Ldc8685JbWwbuTdySq68azYcex1gJ6OaDu-IO50PuzOGyJk='

Decrypted Password:
b'wXDb8Ou1J6R90Vo_9YvqG7TnNKv6A60vJgs91evoPBo='


Process exited with code: 0
```

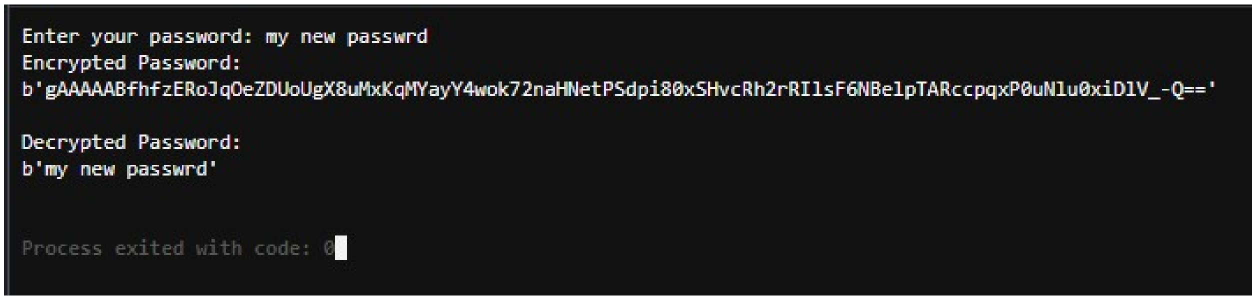
2b.



```
from cryptography.fernet import Fernet

def write_key():
    """
    Generates a key and save it into a file
    """
    key = Fernet.generate_key()
    with open("key.key", "wb") as key_file:
        key_file.write(key)

def load_key():
    """
    Loads the key from the current directory named `key.key`
    """
    return open("key.key", "rb").read()

def encrypt(encrypt_message):
    """
    Generates the encryption
    """
    # generate and write a new key
    write_key()

    # load the previously generated key
    key = load_key()

    message = encrypt_message.encode()

    # initialize the Fernet class
    f = Fernet(key)

    # encrypt the message
    encrypted = f.encrypt(message)

    # print how it looks
    print(encrypted)

    return encrypted
```



```

def decrypt(decrypt_message):

    # load the previously generated key
    key = load_key()

    # initialize the Fernet class
    f = Fernet(key)

    decrypted_encrypted = f.decrypt(decrypt_message)
    print(decrypted_encrypted)

password = input("Enter your password: ")

print("Encrypted Password: ")
encrypted_message = encrypt(password)
print("\nDecrypted Password: ")
decrypt(encrypted_message)
```



```
Enter your password: my new passwrd
Encrypted Password:
b'gAAAAABfhfzERoJqOeZDUoUgX8uMxKqMYayY4wok72naHNetPSdpi80xSHvcRh2rRIlsF6NBelpTARccpqxP0uNlu0xiDlV_-Q=='

Decrypted Password:
b'my new passwrd'


Process exited with code: 0
```

References:

Common Weakness Enumeration. (2020, August 20). Retrieved October 10, 2020, from https://cwe.mitre.org/data/definitions/