Python pwned Hashed Password Cracker Tutorial

Contents

Time required: 60 minutes

NOTE: Do this tutorial in Kali Linux.

What is pwned?

- Watch this video: Passwords & Hash Functions (Simply Explained)

pwned, in a security context, means that your account has been the victim of a data breach.

The word itself takes its name from player-to-player messaging in online computer gaming. When one player is defeated, another might type out a message to say 'You've been owned'.

This was so frequently misspelt as 'pwned', the word itself took off.

Has Your Password been Pwned?

Check out your current email account or cell phone number. Has your information been involved in a data breach?

https://haveibeenpwned.com/

Password Lists

Password lists are text files of common passwords used to crack password hashes. These are from data breaches and are REAL passwords that people use.

The following sites have password lists. We will use some in this lab.

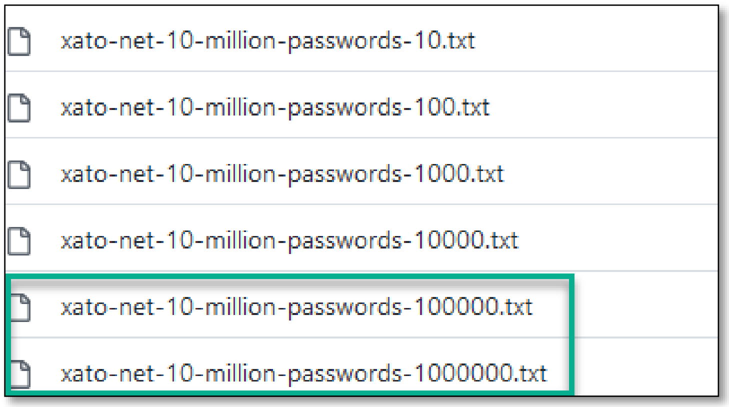
WARNING: Some of the passwords might be offensive. Remember, these are real passwords that people have used and are still used.

- https://github.com/danielmiessler/SecLists/tree/master/Passwords
- https://weakpass.com/

Tutorial 1: Get a Password List

We are going to use a password list that has popular passwords in it.

1. Go to https://github.com/danielmiessler/SecLists/tree/master/Passwords
2. I suggest using one of the two highlighted lists. I will be using the last one. 10 has the 10 most popular passwords, 100 has the 100 most popular passwords, etc. The bigger the list, the better chance of finding a password.



xato-net-10-million-passwords-10.txt
xato-net-10-million-passwords-100.txt
xato-net-10-million-passwords-1000.txt
xato-net-10-million-passwords-10000.txt
xato-net-10-million-passwords-100000.txt
xato-net-10-million-passwords-1000000.txt

3. Click the password file. Click **View Raw**.
4. It may take a bit to load the file.
5. Press **CTRL-A** to select all the passwords.
6. Press **CTRL-C** to copy all the passwords.
7. In the same folder that you will save your password cracker program: Use Visual Studio Code to create a text file named **passwords.txt**
   NOTE: if you use Notepad, it may freeze up.

8. Press **CTRL-V** to paste the passwords into the text file.

We are now ready to create our password cracking tool in Python.

Tutorial 2: Hash Password

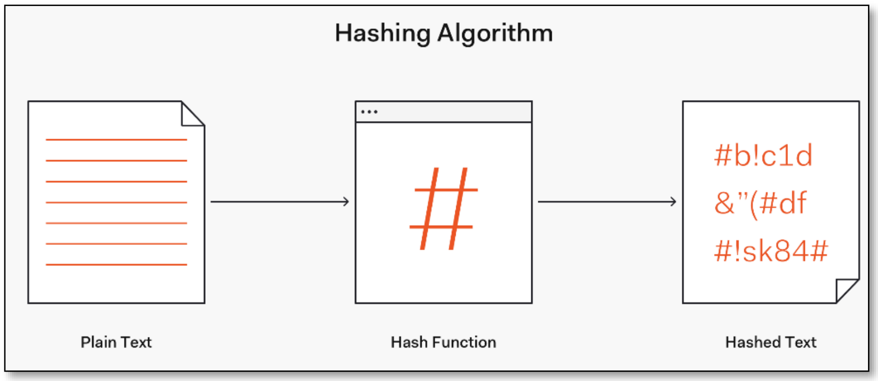
Hashing algorithms are mathematical functions that convert data into fixed-length hash values, hash codes, or hashes. The output hash value is literally a summary of the original value. The most important thing about these hash values is that it is impossible to retrieve the original input data just from hash values.

What's the benefit of using hashing algorithms? Why not just use encryption? Although encryption is important for protecting data (data confidentiality), sometimes it is important to be able to prove that no one has modified the data you're sending. Using hashing values, you'll be able to tell if a file hasn't been modified since creation (data integrity).

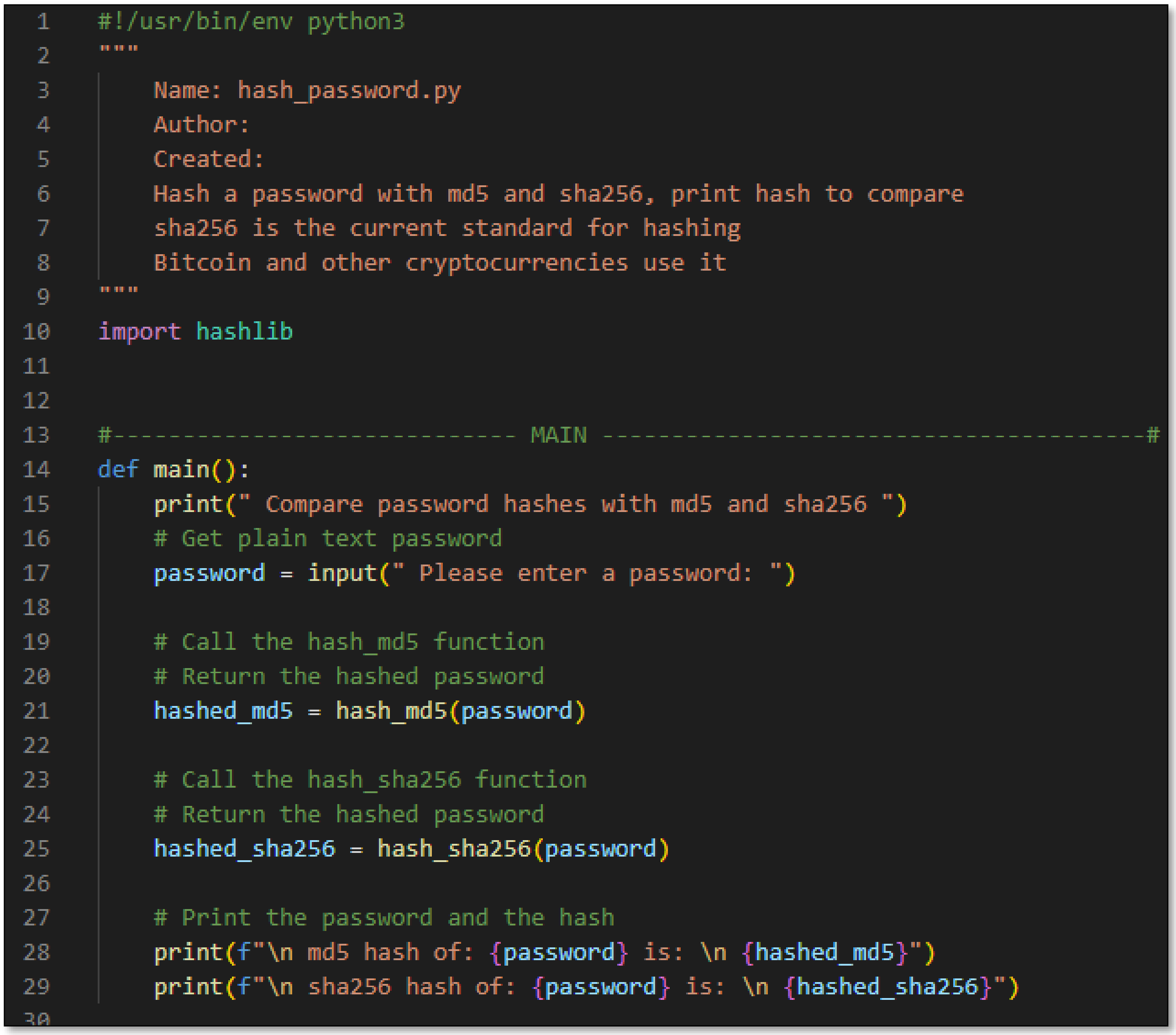
There are many examples of their use, some examples include digital signatures, public-key encryption, message authentication, password protection, and many other cryptographic protocols. Whether you're buying something online, checking your bank balance, storing your files on a cloud storage system, using a Git version control system, connecting to an HTTPS website, connecting to a remote machine using SSH, or even sending a message on your mobile phone, there's a hash function somewhere under the hood.

Storing passwords in cleartext is the equivalent of writing them down in a piece of digital paper. If an attacker was to break into the database and steal the passwords table, the attacker could then access each user account. This problem is compounded by the fact that many users re-use or use variations of a single password, potentially allowing the attacker to access other services different from the one being compromised. That all sounds like a security nightmare!

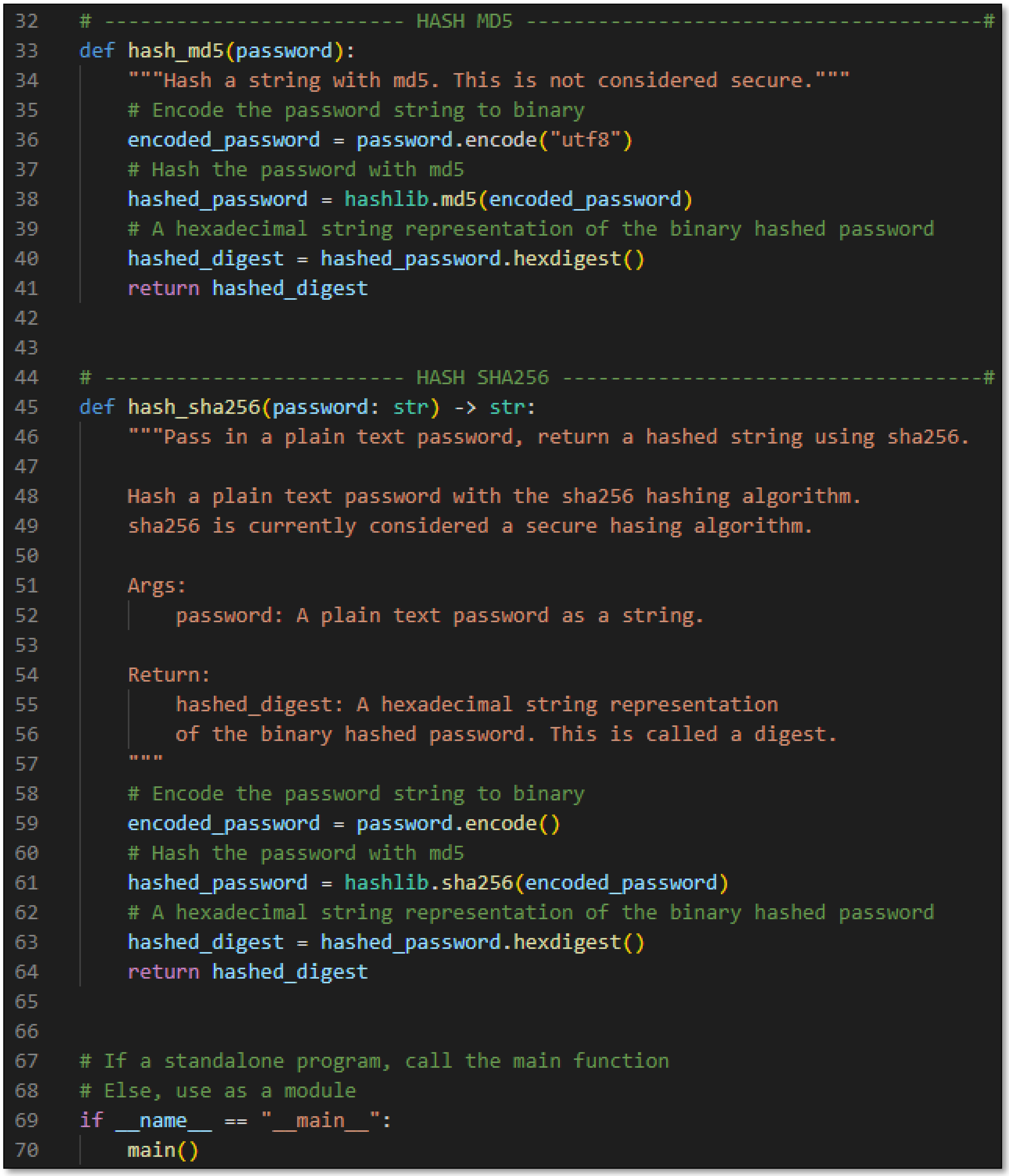
When you authenticate to a system, your password is not passed to the system, a hash of your password is passed. Your plain text password is never passed through a network. The hash of your password is compared to the hash stored on the system. If they match, you can login.



In Kali Linux, create a Python program named **hash_password.py**

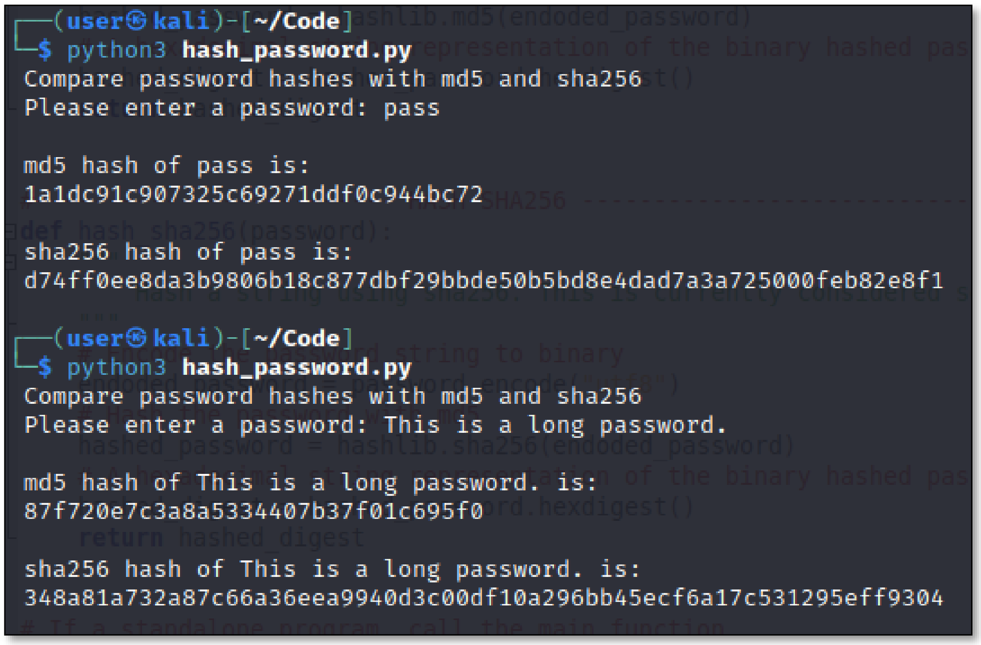


```python
#!/usr/bin/env python3
"""
    Name: hash_password.py
    Author:
    Created:
    Hash a password with md5 and sha256, print hash to compare
    sha256 is the current standard for hashing
    Bitcoin and other cryptocurrencies use it
"""
import hashlib


#---------------------------- MAIN --------------------------------------#
def main():
    print(" Compare password hashes with md5 and sha256 ")
    # Get plain text password
    password = input(" Please enter a password: ")

    # Call the hash_md5 function
    # Return the hashed password
    hashed_md5 = hash_md5(password)

    # Call the hash_sha256 function
    # Return the hashed password
    hashed_sha256 = hash_sha256(password)

    # Print the password and the hash
    print(f"\n md5 hash of: {password} is: \n {hashed_md5}")
    print(f"\n sha256 hash of: {password} is: \n {hashed_sha256}")
```



```python
# ------------------------ HASH MD5 -------------------------------------#
def hash_md5(password):
    """Hash a string with md5. This is not considered secure."""
    # Encode the password string to binary
    encoded_password = password.encode("utf8")
    # Hash the password with md5
    hashed_password = hashlib.md5(encoded_password)
    # A hexadecimal string representation of the binary hashed password
    hashed_digest = hashed_password.hexdigest()
    return hashed_digest


# ------------------------ HASH SHA256 ----------------------------------#
def hash_sha256(password: str) -> str:
    """Pass in a plain text password, return a hashed string using sha256.

    Hash a plain text password with the sha256 hashing algorithm.
    sha256 is currently considered a secure hasing algorithm.

    Args:
        password: A plain text password as a string.

    Return:
        hashed_digest: A hexadecimal string representation
        of the binary hashed password. This is called a digest.
    """
    # Encode the password string to binary
    encoded_password = password.encode()
    # Hash the password with md5
    hashed_password = hashlib.sha256(encoded_password)
    # A hexadecimal string representation of the binary hashed password
    hashed_digest = hashed_password.hexdigest()
    return hashed_digest


# If a standalone program, call the main function
# Else, use as a module
if __name__ == "__main__":
    main()
```

Example run.

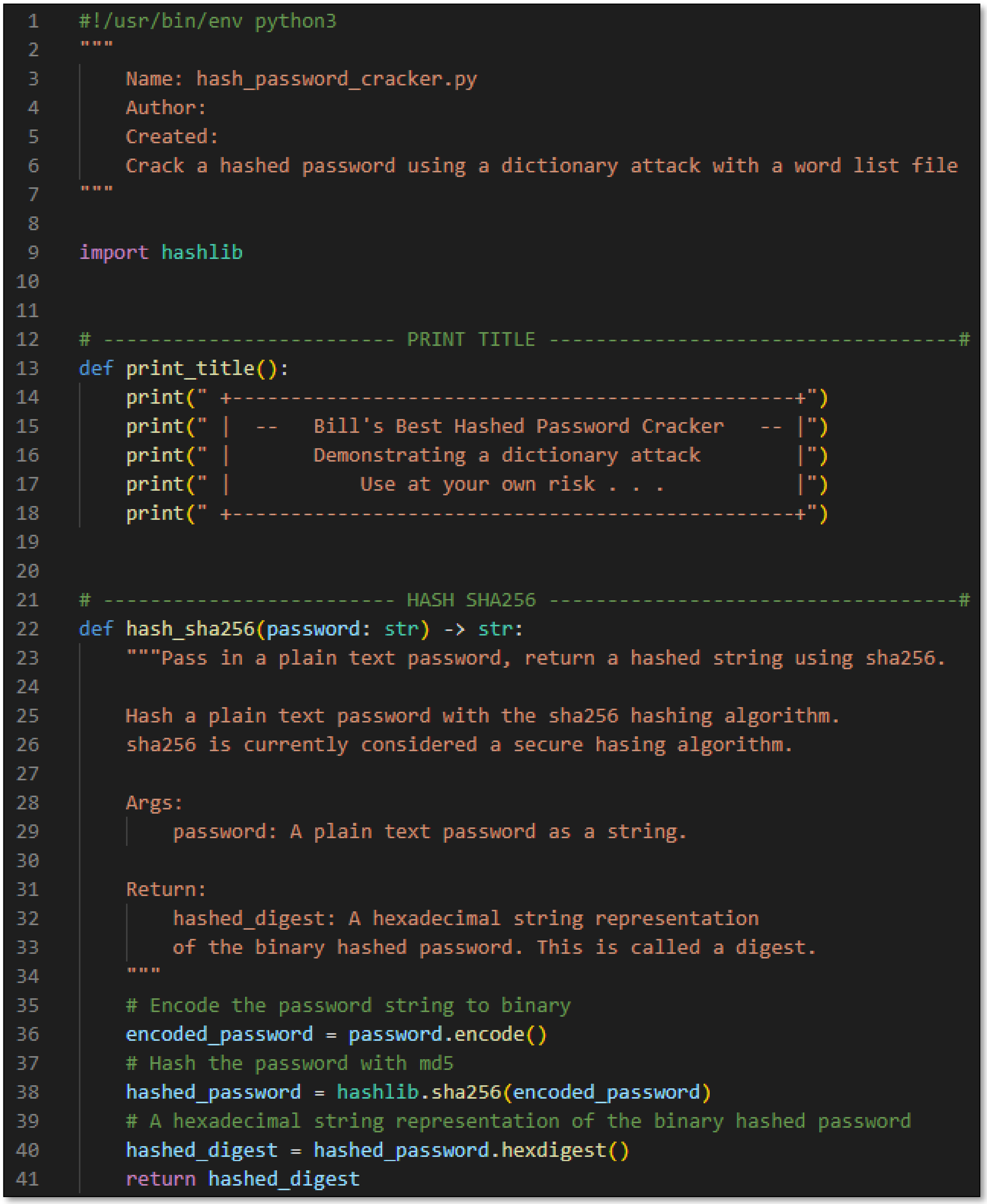

```
┌──(user㉿kali)-[~/Code]
└─$ python3 hash_password.py
Compare password hashes with md5 and sha256
Please enter a password: pass

md5 hash of pass is:
1a1dc91c907325c69271ddf0c944bc72

sha256 hash of pass is:
d74ff0ee8da3b9806b18c877dbf29bbde50b5bd8e4dad7a3a725000feb82e8f1

┌──(user㉿kali)-[~/Code]
└─$ python3 hash_password.py
Compare password hashes with md5 and sha256
Please enter a password: This is a long password.

md5 hash of This is a long password. is:
87f720e7c3a8a5334407b37f01c695f0

sha256 hash of This is a long password. is:
348a81a732a87c66a36eea9940d3c00df10a296bb45ecf6a17c531295eff9304
```

Notice that the has is the same length regardless of the password length.

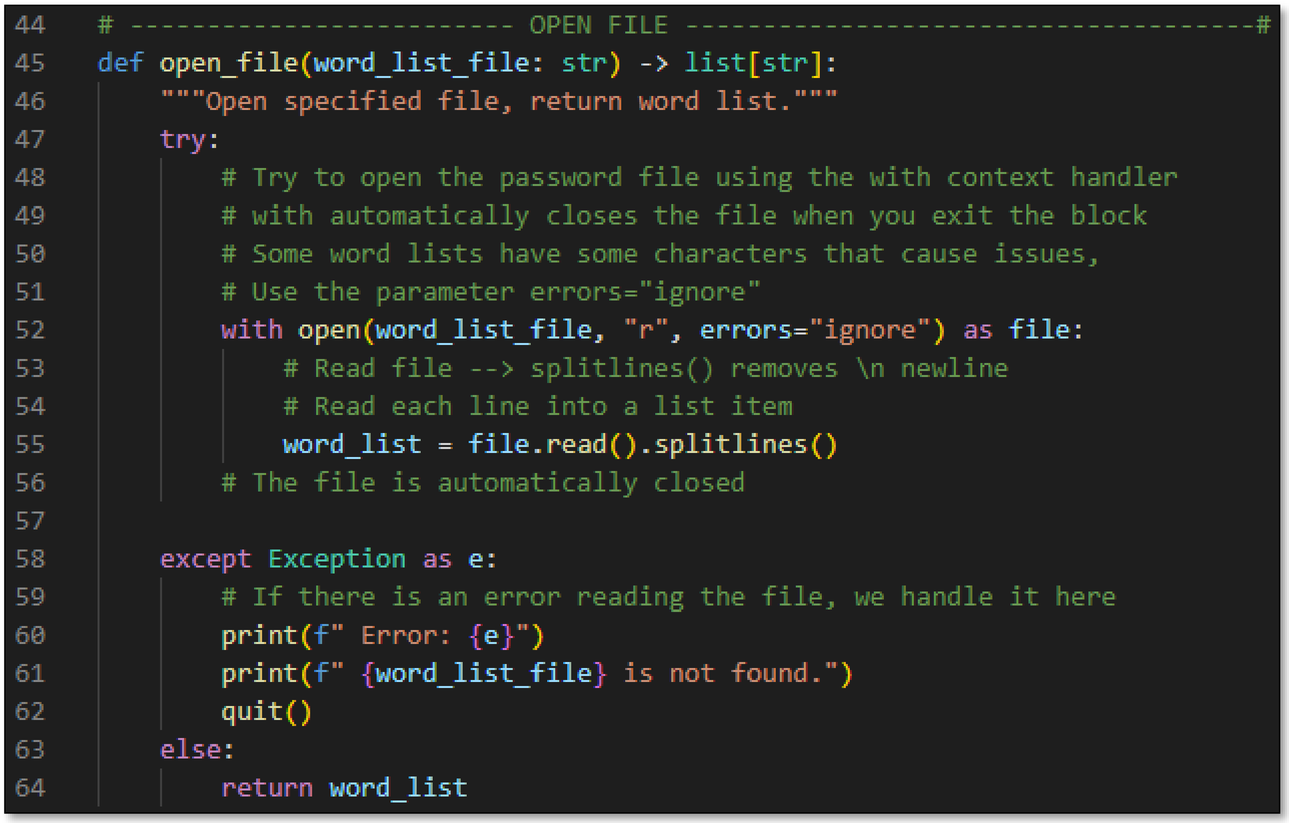
Tutorial 3: Crack Captured Hash

If we can capture an authentication hash, we can compare it against our word lists. This is called a dictionary attack. We are comparing hash digests of passwords. If the digests are the same, it is the same password.

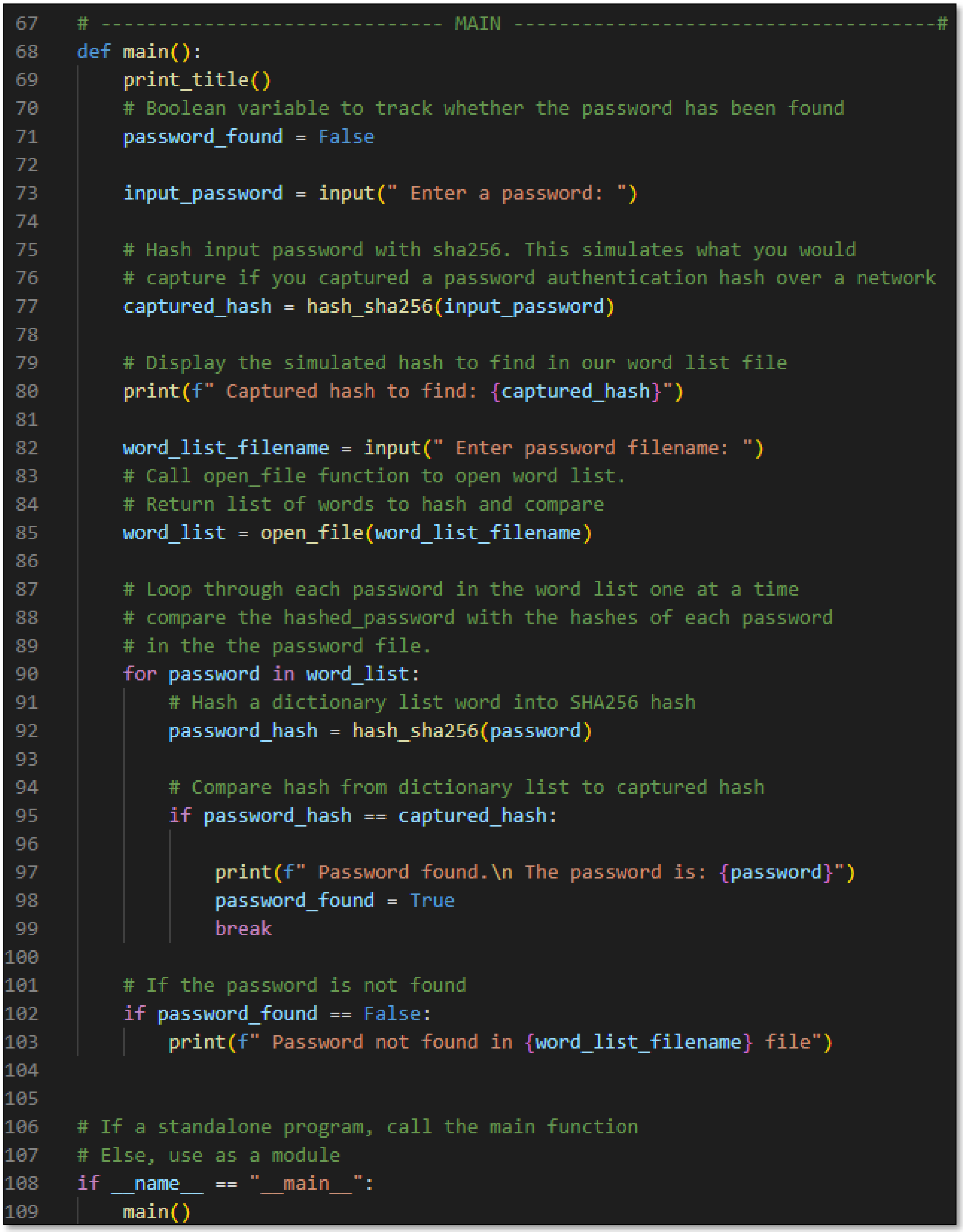
The hash_sha256 function is the same as the first program.



```python
#!/usr/bin/env python3
"""
    Name: hash_password_cracker.py
    Author:
    Created:
    Crack a hashed password using a dictionary attack with a word list file
"""

import hashlib


# ------------------------- PRINT TITLE -----------------------------------#
def print_title():
    print(" +-------------------------------------------------+")
    print(" |  --   Bill's Best Hashed Password Cracker   --  |")
    print(" |       Demonstrating a dictionary attack         |")
    print(" |           Use at your own risk . . .            |")
    print(" +-------------------------------------------------+")


# ------------------------- HASH SHA256 -----------------------------------#
def hash_sha256(password: str) -> str:
    """Pass in a plain text password, return a hashed string using sha256.

    Hash a plain text password with the sha256 hashing algorithm.
    sha256 is currently considered a secure hasing algorithm.

    Args:
        password: A plain text password as a string.

    Return:
        hashed_digest: A hexadecimal string representation
        of the binary hashed password. This is called a digest.
    """
    # Encode the password string to binary
    encoded_password = password.encode()
    # Hash the password with md5
    hashed_password = hashlib.sha256(encoded_password)
    # A hexadecimal string representation of the binary hashed password
    hashed_digest = hashed_password.hexdigest()
    return hashed_digest
```

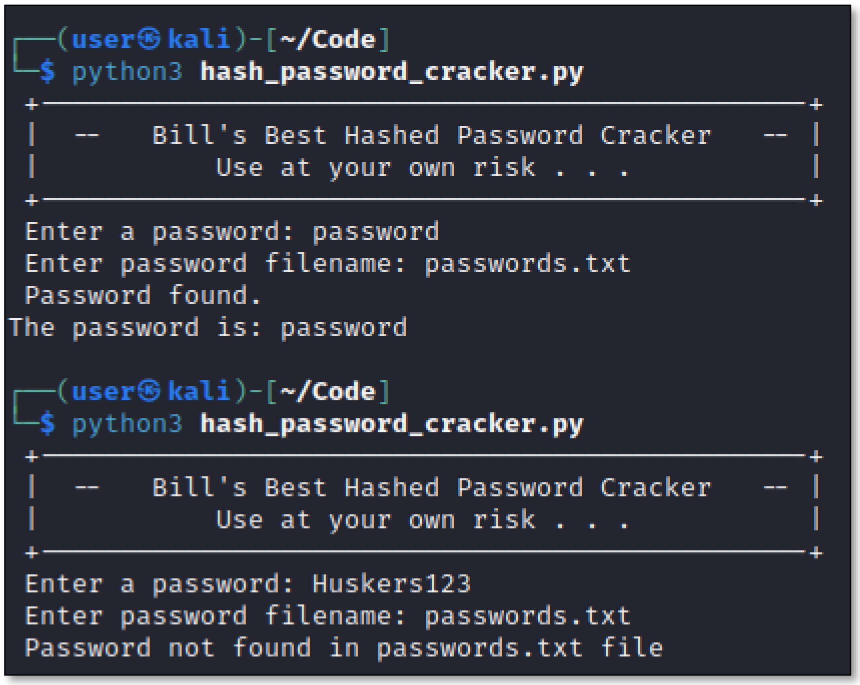


```python
# ------------------------- OPEN FILE -------------------------------------#
def open_file(word_list_file: str) -> list[str]:
    """Open specified file, return word list."""
    try:
        # Try to open the password file using the with context handler
        # with automatically closes the file when you exit the block
        # Some word lists have some characters that cause issues,
        # Use the parameter errors="ignore"
        with open(word_list_file, "r", errors="ignore") as file:
            # Read file --> splitlines() removes \n newline
            # Read each line into a list item
            word_list = file.read().splitlines()
        # The file is automatically closed

    except Exception as e:
        # If there is an error reading the file, we handle it here
        print(f" Error: {e}")
        print(f" {word_list_file} is not found.")
        quit()
    else:
        return word_list
```



```python
# ------------------------------ MAIN --------------------------------------#
def main():
    print_title()
    # Boolean variable to track whether the password has been found
    password_found = False

    input_password = input(" Enter a password: ")

    # Hash input password with sha256. This simulates what you would
    # capture if you captured a password authentication hash over a network
    captured_hash = hash_sha256(input_password)

    # Display the simulated hash to find in our word list file
    print(f" Captured hash to find: {captured_hash}")

    word_list_filename = input(" Enter password filename: ")
    # Call open_file function to open word list.
    # Return list of words to hash and compare
    word_list = open_file(word_list_filename)

    # Loop through each password in the word list one at a time
    # compare the hashed_password with the hashes of each password
    # in the the password file.
    for password in word_list:
        # Hash a dictionary list word into SHA256 hash
        password_hash = hash_sha256(password)

        # Compare hash from dictionary list to captured hash
        if password_hash == captured_hash:

            print(f" Password found.\n The password is: {password}")
            password_found = True
            break

    # If the password is not found
    if password_found == False:
        print(f" Password not found in {word_list_filename} file")


# If a standalone program, call the main function
# Else, use as a module
if __name__ == "__main__":
    main()
```

Example run:



```
┌──(user㉿kali)-[~/Code]
└─$ python3 hash_password_cracker.py
 +--------------------------------------------------+
 |  --    Bill's Best Hashed Password Cracker    -- |
 |            Use at your own risk . . .            |
 +--------------------------------------------------+
 Enter a password: password
 Enter password filename: passwords.txt
 Password found.
The password is: password

┌──(user㉿kali)-[~/Code]
└─$ python3 hash_password_cracker.py
 +--------------------------------------------------+
 |  --    Bill's Best Hashed Password Cracker    -- |
 |            Use at your own risk . . .            |
 +--------------------------------------------------+
 Enter a password: Huskers123
 Enter password filename: passwords.txt
 Password not found in passwords.txt file
```

Assignment Submission

1. Attach all program files.
2. Attach a screenshot of your functioning program.
3. Attach to the assignment in BlackBoard.