

**You can view the demo here:** <https://youtu.be/JGFGXgqw73o>

GitHub link: [https://github.com/nafisahumyra/pw\\_mgr](https://github.com/nafisahumyra/pw_mgr)

## Setup

In command prompt, type `pip install cryptography`

Cryptography is an external python library that we'll use to encrypt and decrypt password files

## Introduction

A password manager is a tool designed to securely store and manage your collection of passwords in an encrypted file or database. The primary purpose of a password manager is to keep your passwords safe and accessible.

When using a password manager, each password is associated with an identifier, such as an account name or URL. For example, if you have multiple Facebook accounts, they might be stored as follows:

- facebook Account1 pw: mypassword123
- facebook Account2 pw: mypassword123
- facebook Account3 pw: mypassword123

This way, you have a collection of key-value pairs, with each key representing an account and each value representing the corresponding password. The password manager allows you to store different passwords for various accounts across different platforms.

To access your stored passwords, you log in to the password manager using a master password or key. Once authenticated, the database is decrypted, and you can retrieve the password for any specific account you need.

## Implementation Details

```
main.py > ...
1  from cryptography.fernet import Fernet
2
3  class PasswordManager:
4      def __init__(self):
5          self.key = None
6          self.password_file = None
7          self.password_dict = {}
8
9      def create_key(self, path):
10         self.key = Fernet.generate_key()
11         print(self.key)
12
13 pm = PasswordManager()
14 pm.create_key(None)
```

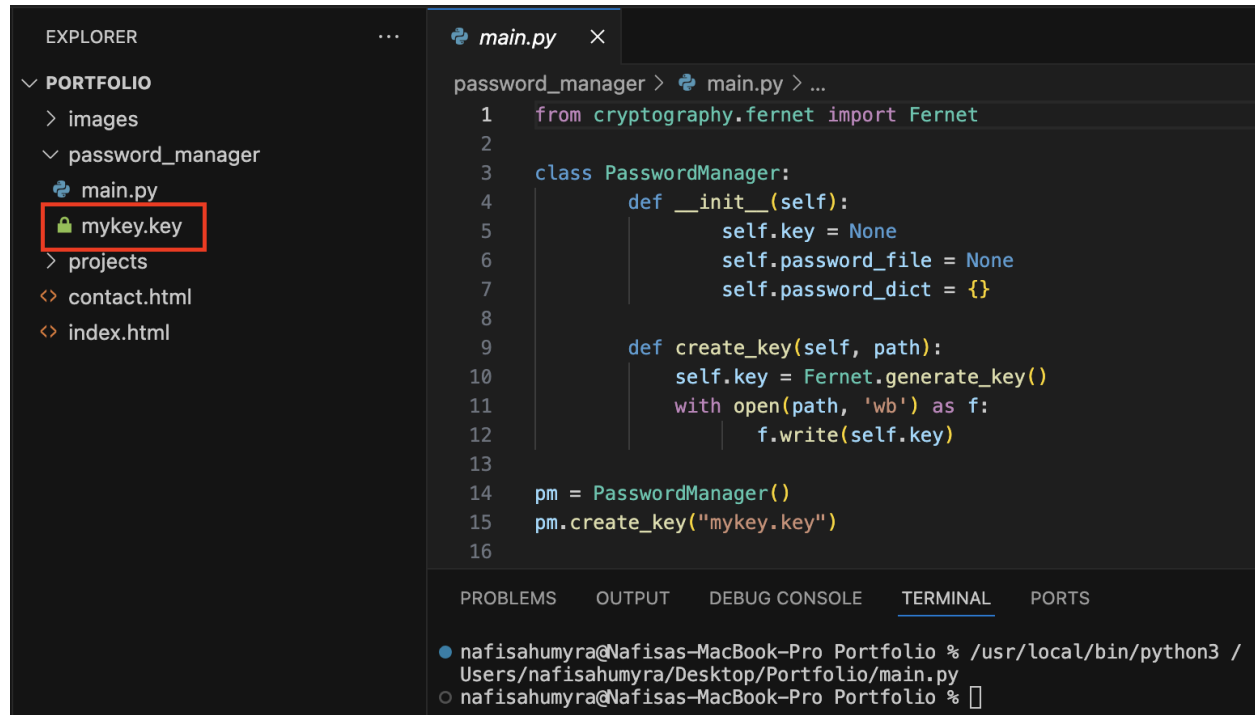
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

```
● nafisahumyra@Nafisas-MacBook-Pro Portfolio % python3 main.py
b'hqE0kDVeQ5Vzd1TGHdBbA00RK10_Sm1kylyDnrds0qc='
○ nafisahumyra@Nafisas-MacBook-Pro Portfolio %
```

We need a key to encrypt and decrypt whatever we want to open. So we're creating a method that generates a key, which is the `create_key` function. This is going to set the `key` attribute of the class to `Fernet.generate_key()`. Calling this will return a key. We want it to print a key so we create a `PasswordManager` instance

When we run this script, you can see in the terminal it returns a key. It can be used for encryption and decryption.

We provide a path because we want to store the key into a file so that we can load it later on. Running this script generates a file mykey.key

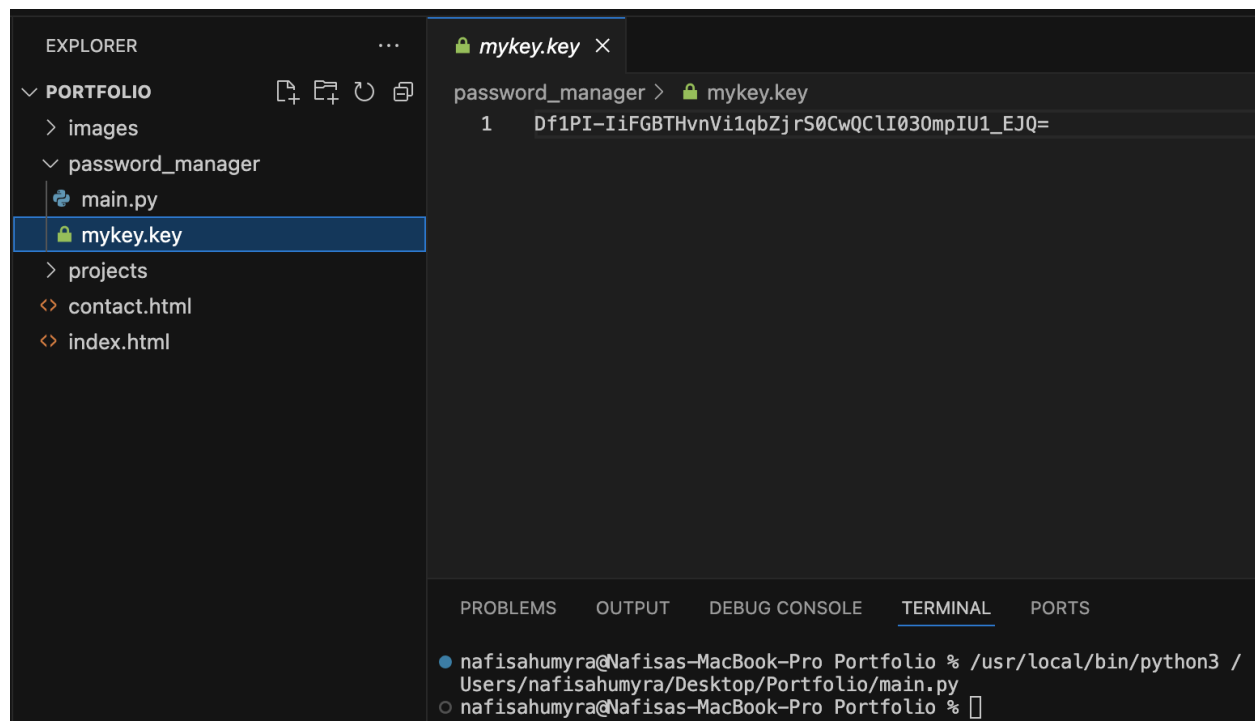


This screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar displays a project named 'PORTFOLIO' with a subdirectory 'password\_manager' containing 'main.py' and 'mykey.key'. The 'mykey.key' file is highlighted with a red rectangle. The main editor window shows the code for 'main.py', which imports 'Fernet' from 'cryptography.fernet' and defines a 'PasswordManager' class. The class has an '\_\_init\_\_' method that initializes 'self.key' as None, 'self.password\_file' as None, and 'self.password\_dict' as an empty dictionary. It also has a 'create\_key' method that generates a key using 'Fernet.generate\_key()' and writes it to a file specified by 'path'. The code at the bottom of the file creates a 'PasswordManager' instance and calls 'create\_key' with the path 'mykey.key'. The bottom panel shows the 'TERMINAL' tab with the command '/usr/local/bin/python3 /Users/nafisahumyra/Desktop/Portfolio/main.py' executed successfully.

```
password_manager > main.py > ...
1  from cryptography.fernet import Fernet
2
3  class PasswordManager:
4      def __init__(self):
5          self.key = None
6          self.password_file = None
7          self.password_dict = {}
8
9      def create_key(self, path):
10         self.key = Fernet.generate_key()
11         with open(path, 'wb') as f:
12             f.write(self.key)
13
14     pm = PasswordManager()
15     pm.create_key("mykey.key")
16
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS

● nafisahumyra@Nafisas-MacBook-Pro Portfolio % /usr/local/bin/python3 /Users/nafisahumyra/Desktop/Portfolio/main.py  
○ nafisahumyra@Nafisas-MacBook-Pro Portfolio %



This screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left shows the 'PORTFOLIO' project with 'password\_manager' containing 'main.py' and 'mykey.key'. The 'mykey.key' file is selected and highlighted. The main editor window shows the content of 'mykey.key', which is a single line of text: 'Df1PI-IiFGBTHvnVi1qbZjrS0CwQClI030mpIU1\_EJQ='. The bottom panel shows the 'TERMINAL' tab with the same command as the previous screenshot, indicating the file was successfully created and its content is now visible.

```
password_manager > mykey.key
1  Df1PI-IiFGBTHvnVi1qbZjrS0CwQClI030mpIU1_EJQ=
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS

● nafisahumyra@Nafisas-MacBook-Pro Portfolio % /usr/local/bin/python3 /Users/nafisahumyra/Desktop/Portfolio/main.py  
○ nafisahumyra@Nafisas-MacBook-Pro Portfolio %

Now we have to have a function for loading because once you create a key and a password file and you save your passwords, you want to be able to decrypt it again with the same key so the existing key has to be loaded, you cannot create a new key every time.

```
3 class PasswordManager:
4     def __init__(self):
5         self.key = None
6         self.password_file = None
7         self.password_dict = {}
8
9     def create_key(self, path):
10        self.key = Fernet.generate_key()
11        with open(path, 'wb') as f:
12            f.write(self.key)
13
14        def load_key(self, path):
15            with open(path, 'rb') as f:
16                self.key = f.read()
17
```

At this point we can create keys, load keys and now we do the same thing for the password files. We want to have a password file which has the information of that password dictionary. So either we already have a password file and we want to load the content into the password dictionary, or we already have the password dictionary with the values and we want to write those into the password file.

So we make a function for creating the password file `create_password_file`

## Encrypting and Decrypting Passwords

The next step is to handle the actual encryption and decryption of passwords. This is where the `add_password` and `load_password_file` functions come into play.

The `add_password` function takes a site and a password, encrypts the password, and writes it to the password file. This ensures that even if someone gets access to your password file, they won't be able to read the passwords without the key.

The `load_password_file` function reads the encrypted passwords from the file, decrypts them using the key, and loads them into the password dictionary. This allows you to retrieve your passwords whenever you need them.

## Retrieving Passwords

Finally, the `get_password` function allows you to retrieve a password for a specific site. It simply looks up the site in the password dictionary and returns the corresponding password.

## User Interaction

The `main` function provides a simple command-line interface for interacting with the password manager. It allows you to:

- Create a new key
- Load an existing key
- Create a new password file
- Load an existing password file
- Add a new password
- Get a password
- Quit the program

The `main` function prompts the user to choose an action, and then calls the appropriate function based on the user's choice.

## Conclusion

This password manager provides a simple and secure way to store and manage your passwords. By using encryption, it ensures that your passwords are protected, even if someone gains access to your password file. The command-line interface makes it easy to use, and the ability to load and save keys and password files ensures that you can always access your passwords when you need them.

Here's the complete code for the password manager:

```

1  from cryptography.fernet import Fernet
2
3  class PasswordManager:
4      def __init__(self):
5          self.key = None
6          self.password_file = None
7          self.password_dict = {}
8
9      def create_key(self, path):
10         self.key = Fernet.generate_key()
11         with open(path, 'wb') as f:
12             f.write(self.key)
13
14         def load_key(self, path):
15             with open(path, 'rb') as f:
16                 self.key = f.read()
17
18         def create_password_file(self, path, initial_values=None):
19             self.password_file = path
20             if initial_values is not None:
21                 for key, value in initial_values.items():
22                     self.add_password(key, value)
23
24         def load_password_file(self, path):
25             self.password_file = path
26
27             with open(path, 'r') as f:
28                 for line in f:
29                     site, encrypted = line.split(":")
30                     self.password_dict[site] = Fernet(self.key).decrypt(encrypted.encode()).decode()
31
32         def add_password(self, site, password):
33             self.password_dict[site] = password
34
35             if self.password_file is not None:
36                 with open(self.password_file, 'a+') as f:
37                     encrypted = Fernet(self.key).encrypt(password.encode())
38                     f.write(site + ":" + encrypted.decode() + "\n")
39
40         def get_password(self, site):
41             return self.password_dict[site]
42
43
44  def main():
45      password = {
46          "Email": "1234567",
47          "FaceBook": "myfbpassword",
48          "YouTube": "helloworld123",
49          "Something": "mypassword123"
50      }
51      pm = PasswordManager()
52      print("""What do you want to do?
53      (1) Create a new key
54      (2) Load an existing key
55      (3) Create new password file
56      (4) Load existing password file
57      (5) Add a new password
58      (6) Get a password
59      (q) Quit
60      """)
61
62      done = False
63
64      while not done:
65
66          choice = input("Enter your choice: ")
67          if choice == "1":
68              path = input("Enter path: ")
69              pm.create_key(path)
70          elif choice == "2":
71              path = input("Enter path: ")
72              pm.load_key(path)
73          elif choice == "3":
74              path = input("Enter path: ")
75              pm.create_password_file(path, password)
76          elif choice == "4":
77              path = input("Enter path: ")
78              pm.load_password_file(path)
79          elif choice == "5":
80              site = input("Enter the site: ")
81              password = input("Enter the password: ")
82              pm.add_password(site, password)
83
84          elif choice == "6":
85              site = input("What site do you want: ")
86              print(f"Password for {site} is {pm.get_password(site)}")
87          elif choice == "q":
88              done = True
89              print("Bye")
90          else:
91              print("Invalid choice!")
92
93  if __name__ == "__main__":
94      main()

```