**1. Imports**

import os

import json

import hashlib

from cryptography.fernet import Fernet

from datetime import datetime

* os → lets us handle files/folders (create, delete, check existence).
* json → we save/load memories in a JSON file (like a digital notebook).
* hashlib → used to **hash the PIN** (turns it into an irreversible fingerprint so no one sees your actual PIN).
* cryptography.fernet → encrypts/decrypts your memories so they’re safe even if someone opens the file.
* datetime → helps us check the unlock date of each memory.

**2. Configuration**

BASE\_DIR = os.path.expanduser("~") # User's home folder

DATA\_FOLDER = os.path.join(BASE\_DIR, "MemoirData")

KEY\_FILE = os.path.join(DATA\_FOLDER, "secret.key")

DATA\_FILE = os.path.join(DATA\_FOLDER, "memories.json")

PIN\_FILE = os.path.join(DATA\_FOLDER, "pin.json")

os.makedirs(DATA\_FOLDER, exist\_ok=True)

* We decide where everything is stored:
  + **MemoirData folder** inside your user folder.
  + Inside it we keep:
    - secret.key → secret encryption key.
    - memories.json → where encrypted memories are stored.
    - pin.json → where the hashed PIN is stored.
* os.makedirs(..., exist\_ok=True) → creates the folder if it doesn’t exist (won’t crash if it already exists).

**3. Encryption Functions**

def load\_key():

if not os.path.exists(KEY\_FILE):

key = Fernet.generate\_key()

with open(KEY\_FILE, "wb") as f:

f.write(key)

else:

with open(KEY\_FILE, "rb") as f:

key = f.read()

return Fernet(key)

fernet = load\_key()

* If a **key doesn’t exist**, we generate one and save it in secret.key.
* If it exists, we just read it.
* fernet is the object we’ll use for **encrypting and decrypting** memories.

**4. PIN Handling**

def hash\_pin(pin: str) -> str:

return hashlib.sha256(pin.encode()).hexdigest()

* This turns the PIN into a **SHA256 hash** (like a fingerprint).
* No one can “unhash” it, so even if the file is leaked, your PIN is safe.

def setup\_pin():

if not os.path.exists(PIN\_FILE):

# First-time setup

else:

# Ask for PIN every time after

* If you’re running the program for the **first time**, it asks you to set a PIN and saves it (hashed).
* If PIN exists:
  + You get **3 attempts** to enter it correctly.
  + If wrong 3 times → program exits.
  + If right → you’re in!

**5. Memory Handling**

def load\_memories():

if not os.path.exists(DATA\_FILE):

return []

with open(DATA\_FILE, "r") as f:

return json.load(f)

* Reads all saved memories from memories.json.
* If file doesn’t exist yet → returns an empty list.

def save\_memories(memories):

with open(DATA\_FILE, "w") as f:

json.dump(memories, f, indent=4)

* Saves all memories back into the file in nice readable format.

def add\_memory():

text = input("Enter your memory: ")

unlock\_date = input("Enter unlock date (YYYY-MM-DD): ")

encrypted\_text = fernet.encrypt(text.encode()).decode()

memories = load\_memories()

memories.append({

"memory": encrypted\_text,

"unlock\_date": unlock\_date

})

save\_memories(memories)

print("✅ Memory saved!")

* Asks you for the **memory text** and the **unlock date**.
* Encrypts the memory text using our Fernet key.
* Saves it in the memories.json file.

def view\_specific\_memory():

# Loads all memories

# Shows list of memories with status (Unlocked or Locked until date)

# Lets you pick which memory to open

# Decrypts if unlock date has passed

* Lists all memories with their unlock date.
* Lets you pick one.
* If today’s date >= unlock date → decrypt and show it.
* If not yet → tells you it’s still locked.

def delete\_memory():

# Similar to view

# Shows list of memories

# Lets you pick one to delete

# Confirms before deleting

* Lets you choose which memory to delete.
* Removes it from the list and updates the file.

**6. Main Menu**

def main\_menu():

while True:

print("\n===== MEMOIR - Your Digital Diary =====")

print("1. Add a memory")

print("2. View a memory")

print("3. Delete a memory")

print("4. Exit")

choice = input("Enter choice: ")

if choice == "1":

add\_memory()

elif choice == "2":

view\_specific\_memory()

elif choice == "3":

delete\_memory()

elif choice == "4":

print("👋 Goodbye! Stay safe with your memories.")

break

else:

print("❌ Invalid choice, try again.")

* Simple loop that keeps showing options until you exit.
* Based on your choice → calls the right function.

**7. Program Start**

if \_\_name\_\_ == "\_\_main\_\_":

setup\_pin()

main\_menu()

* setup\_pin() → checks PIN (or sets it up first time).
* Then takes you to the main\_menu().

✅ **In short:**

* We built a **secure digital diary**.
* Memories are **encrypted**.
* Access is protected with a **PIN**.
* Memories unlock **only after the date you set**.
* You can **add, view, delete** memories safely.