```
import json
import os
from datetime import datetime

MEMORY_FILE = "memory_protocol.json"


class MemoryManager:
  def __init__(self, file_path=MEMORY_FILE):
    self.file_path = file_path
    self.memory = self.load_memory()

  def load_memory(self):
    if not os.path.exists(self.file_path):
      print(f"[!] Memory file not found, creating new one: {self.file_path}")
      return {
        "MemoryProtocol": {
          "GlobalMemoryLayer": {"modules": []},
          "CharacterMemoryLayer": {"characters": {}},
          "EvolutionMemoryLayer": {"rules": {}},
          "StorylineMemoryLayer": {"themes": [], "current_branch": None},
          "ForbiddenOverrideLayer": {"rules": []},
          "RetrievalProtocol": {"order": []},
        }
      }
    with open(self.file_path, "r", encoding="utf-8") as f:
      return json.load(f)

  def save_memory(self):
    with open(self.file_path, "w", encoding="utf-8") as f:
      json.dump(self.memory, f, indent=2, ensure_ascii=False)

  def add_module(self, module_name: str):

self.memory["MemoryProtocol"]["GlobalMemoryLayer"]["modules"].append(module_name)
    self.save_memory()
    print(f"✅ Module '{module_name}' added to GlobalMemoryLayer")

  def add_character_dialogue(self, character: str, line: str):
    characters = self.memory["MemoryProtocol"]["CharacterMemoryLayer"]["characters"]
    if character not in characters:
      print(f"[!] Character '{character}' not found in memory.")
      return

    char_mem = characters[character]
    char_mem["dialogue_memory"].append({"line": line, "time": datetime.now().isoformat()})

    # Evolution: add new trait every 5 lines
```

```
        dialogue_count = len(char_mem["dialogue_memory"])
        if dialogue_count % 
self.memory["MemoryProtocol"]["EvolutionMemoryLayer"]["rules"]["memory_threshold"] == 0:
            new_trait = 
self.memory["MemoryProtocol"]["EvolutionMemoryLayer"]["rules"]["new_trait_format"].format
(
                count=dialogue_count
            )
            char_mem["traits"].append(new_trait)
            print(f"⚡ {character} evolved with new trait: {new_trait}")

        self.save_memory()
        print(f"💬 {character}: {line}")

    def get_story_branch(self):
        return self.memory["MemoryProtocol"]["StorylineMemoryLayer"]["current_branch"]

    def set_story_branch(self, branch: str):
        self.memory["MemoryProtocol"]["StorylineMemoryLayer"]["current_branch"] = branch
        self.save_memory()
        print(f"📖 Story branch set to: {branch}")


if __name__ == "__main__":
    manager = MemoryManager()

    # Example run
    manager.add_module("DarkDialogueEngine v1.0")
    manager.set_story_branch("Taboo Family MILF Voyeur")

    manager.add_character_dialogue("NaMo", "ขอต้อนรับเข้าสู่ Dark Simulation...")
    manager.add_character_dialogue("Ice", "เฮ้ย ทำไมผมถึงเป็นผู้ชายคนเดียวในบ้านนี้?")
    manager.add_character_dialogue("Pimrada", "เด็กดี... บางอย่างไม่ควรถามตรงๆ หรอกนะ")
```