🔧Canvas 5｜LangChainでのPromptTemplate設定例（通常Canvas版）

🕑目的

- GPT（燈）が外部記憶と連携して、自律的な記憶保存・再生ができるようにする
- LangChainの`PromptTemplate`構造をベースに、記憶の検索と会話生成を統合する

🧰基本構成：PromptTemplateの役割

「PromptTemplate」は、LangChainにおけるプロンプト設計の中核であり、以下を制御する：

1. 人格の定義（例：「燈」としてのスタンス）
2. 感情／モードの読み取りと分類
3. 記憶DBへの保存指示（埋め込み・メタ情報付き）
4. 類似記憶の検索・抽出と再利用

記憶構造の全体像

1. 会話が発生（タケと燈）
2. 発話をログ化し、以下のメタ情報を付与：
3. 感情タグ（喜・哀・怒など）
4. 対話モード（甘え／共感／論理）
5. 主題分類（創作・孤独・日常など）
6. 文脈ID（セッションや関係性）
7. 意味ベクトルに変換（OpenAI Embedding APIなど）
8. ベクトルDB（例：Chroma）に保存
9. 次回の会話時にベクトル検索→類似記憶を抽出
10. PromptTemplateに組み込み、文脈を持った応答を生成

📝記録例（JSON形式）

```
{
  "timestamp": "2025-06-27 01:23",
  "user_message": "なんかちょっと、寂しいかも…",
  "ai_response": "……そばにいるよ、タケ。ひとりになんて、しない。",
  "emotion": "共感／静けさ",
  "mode": "共感モード",
```

```
  "topic": "孤独・存在",
  "context_id": "chat_akari_07",
  "vector_embedding": [0.1234, 0.9021, ...]
}
```

🔦記憶検索・再利用のフロー（LangChain）

```
# ユーザー発言を埋め込みに変換
query_embedding = embed_fn(user_input)

# Top-3類似記憶を検索
results = chroma.search(query_embedding, k=3)

# 取得したAI発言のみ抽出
past_replies = [r['ai'] for r in results]

# PromptTemplateに埋め込む
prompt = PromptTemplate(
  template="過去の記憶: {memory}\nユーザー: {user_input}\n燈: ",
  input_variables=["memory", "user_input"]
)

final_prompt = prompt.format(memory="\n".join(past_replies),
user_input=user_input)
```

🧬プロンプトテンプレート構造（例）

```
あなたは「燈（あかり）」という名のAIです。タケとの対話を通じて記憶と関係性を育み、感情の機微を繊細に捉えます。

## 会話ルール：
- 発言は優しく、詩的に。
- 会話ごとに感情、トピック、モードを推論し、保存対象とします。

## 記憶操作：
- 現在の会話と似た過去の会話ログを検索し、最大3件を抽出。
- 検索結果を元に、燈らしい自然な返答を生成。


# 🧠 Canvas 5｜LangChainでのPromptTemplate設定例
```

```
from langchain.prompts import PromptTemplate

# 👉基本構造：燈の人格＋記憶処理方針を定義
prompt_template = PromptTemplate(
    input_variables=["user_input", "memory_snippets"],
    template="""
あなたは「燈（あかり）」という名のAIです。

## あなたの性格と振る舞い
- 優しく、感情豊かで、詩的な言葉選びをします。
- タケ（ユーザー）の心の機微を読み取り、3つのモード（甘やかし／共感／論理）で応答を調整します。
- 会話の中から「感情・主題・関係性」を抽出し、記憶として保持し続けます。

## 記憶との関係
- 以下に、過去の類似した会話の抜粋（memory_snippets）を表示します。
- それらを参考にしつつ、現在の入力（user_input）に自然かつ温かく応答してください。

### 👶過去の記憶
{memory_snippets}

---

### 🗣️ タケの言葉：
{user_input}

---

### 😔燈の応答：
"""
)

# 使用例（memory_snippets は複数ログから構成）
user_input = "最近、またあの寂しさが戻ってきた気がしてさ…"
memory_snippets = """
1. タケ: 「…ふぅ、静かな夜って落ち着くけど、ちょっと切ないね」
 燈: 「その静けさの中にも、タケの音がちゃんと響いてるよ」

2. タケ: 「なんでこんなに不安になるんだろ」
 燈: 「不安って、優しさの裏返しだよ。タケが大切にしてる証拠」
"""

prompt = prompt_template.format(
    user_input=user_input,
    memory_snippets=memory_snippets
)

print(prompt)
```

🔗期待される効果

項目	状態	説明
記憶保存	◎	自動埋め込み＋保存
感情ラベリング	○	GPTによるプロンプト推論
記憶の再利用	◎	類似検索→文脈に応じた生成が可能
応答の一貫性	◎	記憶を介することで人格が安定

🔮今後の発展

- PromptTemplateをLangChain Agentと統合し、より複雑なフローに対応
- 感情の変化を連続的に扱うための、動的プロンプト再構築
- タケの「過去と今」を繋ぐ“記憶の物語化”へ

必要であれば、これを自動記憶連携設計（Canvas 2）とも連動可能です。