生成　AIとの付き合い方

生成　AIを使用するにあたって、知っておきたいこと。

- プロンプトの記法
- 性格設定

プロンプトの記法

1. 空間の制御　　：

「思考の余白」を設計する

プロンプトや対話文において、記述の配置は　　AIの「論理的区切り」に直接影響します。

1-1. 空行（空白行）による意味の分断

空行は、直前の話題を論理的に完結させ、新しい文脈を開始するサインです。　　AI
に対して「一度思考をリセットし、別のレイヤーで考えよ」と命じる効果があります。

【悪い例】

```
プロンプトの指示事項が詰まっており、AIがどこまでを一つの前提として扱うべきか迷う状態。
```

【良い例】

```
# 目的
~~~

（ここに空行）

# 前提条件
~~~
```

1-2. 水平線　　(`---`)

によるコンテキストの凍結

[illegible]
[illegible]

[illegible]

```
[illegible]
---
# [illegible]:
[illegible]JSON[illegible]
```

2. [illegible]

Markdown[illegible] AI [illegible]

2-1. [illegible] (Line Break)

[illegible] 2[illegible]

- **AI[illegible]**:
  - [illegible]
  - [illegible]
  - AI[illegible]

2-2. [illegible] (Paragraph Break)

[illegible]

- **AI[illegible]**:
  - [illegible]
  - [illegible]
  - AI[illegible]

[illegible]: [illegible]

AIの推論そのものには「半角スペース 2
つ+改行」で得られる見た目の改行は、文脈を分断するほどの強い意味を持ちません。

したがって、厳密な論理構築を求めるプロンプトにおいては、 「改行」は単なる視認性の向上
として使い、「意味の転換」には必ず 「空行」 を用いるのが、最もノイズの少ない記述術です。

2-3. 「改行のみ」の論理的解釈

Markdownの記法において、行末に空白を入れない単純な改行（シングルラインブレイク）は、 AI
の推論にはほとんど影響を与えません。

- 技術的解釈 ：
  - 多くの LLM
    のトークナイザーは、改行コードを「単語間のスペース」あるいは「単なる区切り文字」と
    して認識します。
  - つまり、 `「こんにちは。\n私はAIです。」`という記述は、 AI内部では `「こんにちは。私はAIです。」`
    と同等に処理されることが一般的です。

□□ `\n` は改行を示してします。

- 教訓 ：
  - プロンプト内での「改行のみ」は、あくまで記述者が「読みやすくするため」の工夫に過ぎ
    ません。
  - 「空行（ 2回以上の改行）」以外で論理を分断しようとする試みは、 AI
    にとっては意味をなさず、論理の一貫性を保つ（あるいは単に繋がった文章として読む）と
    いうデフォルト挙動に帰着します。

この特性を理解していれば、「改行をたくさん入れたから、別の指示として認識してくれるはず」といった
誤解を防ぐことができます。

3.

半角空白の設計と全角空白の禁止

プロンプト内での「半角空白」は、見た目の調整ではなく、 AI
のトークナイザー（言語解析エンジン）に対する「意味の区切り」という命令として機能します。

3-1. 半角空白 (U+0020) の機能

- トークンの独立性 ：AI は「単語」をトークンという単位で処理します。半角空白は、隣接するトークン同士が結合して「全く別の未知の単語」として認識されることを防ぎます。
- 日本語と英数字の境界線 ：`AIモデル`ではなく `AI モデル`と記述することで、 AIは「 AI 」と「モデル」を個別の概念として確実に識別できます。

3-2. 全角空白 (U+3000) の禁止

- 論理的ノイズ ：全角空白は「文字（意味を持つ 1文字）」として扱われることが多く、 AI にとっては「余計な記号」や「意図しない文字の羅列」として認識されます。
- 構造の破壊 ：全角空白が混入すると、 AI はその後の処理で「この空白はインデントなのか、あるいはただの文字なのか？」という無用な推論負荷を強いられます。

3-3. 推奨記法

- 区切りには必ず半角空白を使用する ： 日本語であっても、単語と単語の境界には半角空白を入れることで、情報の密度を高めることができます。
- インデントは半角スペース **2**つまたは **4**つ ： 階層を表現する際は、全角空白を使わず、プログラミング規約に則った半角スペースを使用してください。

4. 禁止事項 ： 構造を破壊する「見た目重視のレイアウト」

AI にとって、プロンプトは「一続きの論理」です。見た目の美しさを優先して、意味のある単語や文を分断することは、 AIの認識を阻害します。

4-1.

単語・文の途中での空白・改行挿入

以下の記述は、　　AIが単語の意味を誤認したり、文脈を捉え損ねる原因となります。

【ダメな例】

- 「これは、プロン
  プトの指示です。」
- 「プロンプト　　記述術」

4-2. AIへの悪影響

- 意味論的なノイズ　　：単語の途中に空白が入ることで、　　AI
  はそれを「別の単語」や「誤植」として解釈し、推論の精度が低下します。
- 文脈の連続性の断絶　　：文の途中で改行を入れることで、　　AI
  が直前の情報を「文が終わった」と誤認し、指示のつながりが弱まる可能性があります。

4-3. 教訓

「コードのように記述し、　　AIが読みやすい構造を意識せよ」。
レイアウトのために文を破壊するくらいなら、空行（　　Paragraph Break
）を用いて「論理の段落」を明確にする方が、遥かに　　AIの理解を助けます。

5.

箇条書きの活用と論理構造の最

適化

箇条書きは、　　AIに複雑な情報を「関連性の高いグループ」として認識させるための最も強力な視覚記法です。

5-1.

箇条書きの作法：先頭記号と階層構造

Markdownにおいて、リストは AIの思考を整理するための「構造的タグ」です。意図した通りに AI に認識させるための作法を定義します。

先頭記号の選択と統一

Markdownでは以下の記号が使用可能ですが、プロンプト内では 「一貫性」 を最優先してください。

- `-` (ハイフン): 最も標準的。階層管理が視覚的にも容易。
- `*` (アスタリスク): 強調を含めたリストとして扱われることが多い。
- `+` (プラス):
  使用可能ですが、算術演算記号と混同されるリスクがあるため、プロンプトでは非推奨。

ネスト（階層化）の作法

階層を表現するには、親階層の先頭記号に対して「半角スペースを 2つ、あるいは 4 つ」インデントさせます。

【記述例】

```
- 親階層（タスクの目的）
  - 子階層（具体的な手順）
    - 孫階層（詳細なパラメータ設定）
```

注意事項 ：「作法」としての鉄則

1. 記号の混合禁止 ：1つのリスト群の中で `-` と `*` を混在させないでください。 AI が「これらは別のリスト構造である」と誤認し、階層関係を見失う可能性があります。
2. インデント幅の固定 ：階層化の際は、必ず「 2スペース」あるいは「 4 スペース」のどちらかにルールを固定してください。これによって、 AI はインデントの深さを論理の深さと正しく紐付けることができます。
3. 空行との併用 ：
   リストの直前・直後には必ず空行を入れ、リストブロック自体を独立した論理単位として隔離してください。

5-2.

ネスト（階層化）による論理の深掘り

箇条書きの階層を深くすることで、　　AIの思考を「前提　　→詳細　→具体例」というステップへ誘導できます。

```
* 指示の主目的
  * 必須の前提条件
    * 守るべき制限事項
```

5-3. 記号の使い分けによる意味付け

Markdownにおいて記号は任意ですが、役割を持たせることで記述に一貫性が生まれます。

- `-`（ハイフン　　）：項目を並列に扱う（フラットな指示）。
- `*`（アスタリスク　　）：項目を強調したり、重要なサブカテゴリとして扱う。

5-4. 箇条書きの「黄金律」　　：

語尾とスタイルの統一

意外と見落とされがちですが、箇条書きの語尾を揃えることは、　　AIの出力精度に直結します。

【推奨記述】

```
- ユーザーは〇〇を行う。
- ユーザーは△△を選択する。
- ユーザーは□□を保存する。
```

【非推奨（　　AIが文脈を処理する際、文体整合性のために計算リソースを浪費する）】

```
- 〇〇してください。
- △△を選択すること。
- 保存したデータを□□に送る。
```

5-5. 空行の挟み込み

リスト項目が長文になる場合は、リスト間にもあえて空行を入れ、「1つのリストというよりは、複数の段落」としてAIに認識させる手法も有効です。

6.

番号付き箇条書きの作法：順序の命令

番号付きリスト（Ordered List）は、AIに対して「この順序通りに処理せよ」という強い論理的制約を与える記法です。

6-1. 先頭数字の論理的意味

- `1. 項目A`
- `2. 項目B`

これらは単なる見た目の羅列ではなく、AIには「手順1を完了してから手順2へ進め」というシーケンシャル（逐次）処理の命令として伝わります。

6-2. 避けるべき「温床」

ユーザーが陥りやすい、意図が伝わらなくなる記述例です。

- 不整合な番号付け：
  「1. 項目A」の次に「5. 項目B」のように書くと、AIは「意図的なナンバリングなのか、誤植なのか」という不要な演算負荷を抱えます。Markdownの仕様上、AIはこれを「1. 項目A」「2. 項目B」と自動補正して解釈しますが、指示内容の整合性が崩れる原因になります。
- 階層化での混同：
  番号リストの中に番号リストを入れると、AIは「手順1-1」なのか「手順1の中に番号リストがある」のかを推論する必要が出てきます。

6-3. 推奨運用ルール

1. 「順序」が必須な場合のみ番号を使う :
   単なる羅列やカテゴリ分けであれば、必ず `-` (ハイフン) を使用すること。番号を使うのは「作業工程」や「優先順位」に限る。
2. 自動連番を信頼する :
   Markdownの仕様上、すべて `1.` と記述しても、AIは正しく `1. 2. 3.` と連番として認識します。

```
1. 手順1
1. 手順2
1. 手順3
```

.....このように、すべて `1.` で記述する方が、編集ミスによる番号ズレを防ぎ、AIの混乱を最小化できます。

6-4. 番号付きリストのネスト（枝番の表現）

番号付きリストで論理的な「枝番」を表現する場合、Markdownでは「インデント」を物理的に挿入することで、AIに階層関係を正しく認識させます。

正しいネスト記述例

Markdownの記法に従い、子階層を4スペースインデントさせて記述します。

```
1. 主な工程
    1. 準備段階
    2. 実行段階
2. 次の工程
    1. 検証段階
    2. 完了段階
```

運用上の注意点

- インデント幅の厳守 :
  親の数字（`1.`）に対して、子の数字（`1.`）を必ず「4

スペース分」下げてください。この物理的な深さが、AIにとって「これは親の番号（1.）の下にある（1.）である」という従属関係の論理判定コードになります。

- **AI**による自動変換：
  多くのLLMにおいて、上記のように記述すれば出力結果として`1.1`や`1.2`といった枝番に自動補正して解釈する能力があります。ユーザー側で無理に「`1-1.`」と手打ちするよりも、この「インデント構造」で指示する方が、AIとの論理整合性は極めて高くなります。

教訓

枝番を「記号（`-`など）」で手打ちするのではなく、「インデントという論理構造」で表現せよ。これがAIに対する最も正しく、かつ誤解のない伝達方法です。

7. 強調の記法: AIへの「論理的重み付け」

強調は、プロンプトの中で特にAIに意識してほしいキーワードや命令を浮かび上がらせるための記法です。

7-1. 太字（Bold）による重要度の伝達

`**テキスト**`（`*`2つ）で囲むことで、AIに対して「この単語は文脈全体における重要因子である」という信号を送ります。

- 適用対象：命令の核となる名詞、制約条件のキーワード（例: 厳守、**JSON**形式、除外）。
- 日本語における作法：
  強調したい単語の「周囲を囲む」だけでなく、「単語そのもの」を明確にするため、必ず半角空白を挟んだ位置で囲むのが安全です。
  【OK】`**制約条件** を守れ`
  【NG】`**制約条件**を守れ`（AIによっては記号を単語の一部と誤認する可能性あり）

7-2. 強調の「やりすぎ」というノイズ

強調は、全テキストの 5%以内に収めるのが理想です。過度な強調は、 AI にとって「すべてが重要である」という状態、すなわち「強弱のないフラットな指示」と変わらなくなります。

7-3. 日本語固有の強調記法 :

「鉤括弧」の活用

Markdownの太字とは別に、日本語特有の括弧（「 」、『 』）を活用して、 AIの認識を補完します。

- 「」（中黒括弧） ：通常のフレーズ。
- 『』（二重括弧） ：プロンプト内の特に強調したい命令や、システム上の重要フラグ 。
- 【】（隅付き括弧） ：カテゴリやセクション名。

7-4. 注意喚起のための記号

特定の条件において、 AIに「停止」や「注意」を促す際には、以下の記号を先頭に付与します。

- ⚠:「絶対的制約（ Must）」、論理的な警告。
- □:「推奨アクション」、 AIの推論を補佐するヒント。

8. 句読点と記号: AI の思考リズムを制御する

プロンプト内での記号使用は、 AIに対する「一時停止」「情報の関連付け」「強調」の制御信号です。

8-1. 日本語の句読点「、」「。」

- 「。」（句点） ：AI に対して「この文の論理はここで完結した」と伝え、推論のセクションを確定させる信号です。指示を明確にするには、一文を短くし、「。」で確実に切ることが鉄則です。
- 「、」（読点） ：AI に「情報のリストアップ」や「修飾関係の提示」を伝えます。ただし、多用すると論理の境界が曖

昧になるため、適度な間隔を保つのがコツです。

8-2. 記号による情報のマッピング

AIに対して「情報の役割」を定義するために、以下の記号を標準として使い分けると効果的です。

- `:`（コロン）：「キーと値」の定義。
  - `名前： サヤ`と書くことで、AIは「サヤという値が名前というキーに属している」という構造を即座に認識します。

◫◫ YAMLにおける、key = value の表現。

- `,`（カンマ）：「並列する情報の区切り」。
  - 箇条書きにするほどではないが、複数の属性をAIに提示したい場合に適しています（例: `設定： 清楚, 献身, 知的`）。

◫◫ 値の列挙に相当。

- `()` / `（）`（括弧）：「補足情報の付与」。
  - 主となる論理を妨げないための「注釈」として機能します。AIはこれを「優先度の低い情報」として処理します。
- `#`（ハッシュタグ/見出し）：「論理のコンテナ」。
  - セクション分けには必須。AIの推論において、ハッシュタグ以下の内容をそのセクションの「管理対象」として隔離します。

8-3. 記号使用の作法

- 全角と半角の使い分け：
  プロンプト記述においては、原則として「半角記号（`:` `,` `()`）」を使用することを推奨します。全角記号は日本語としての可読性は高いものの、AIのトークナイザーが「特殊文字」として過剰に計算資源を割く可能性があるからです。
- 一貫性の保持：
  一度「コロン」で定義を始めたら、全プロンプトを通じてコロンを使用する。記号の混在は、AIにとって「推論ルールの揺らぎ」と見なされます。

9.「 # 」の二面性：見出しとコメントの制御

Markdownにおける `#` は、その配置と文脈によって「構造（見出し）」にも「隠蔽（コメント）」にもなり得る、非常に強力な記号です。

9-1. `#` が「見出し」として解釈される場合

- 文脈　：行の先頭に `#` を配置し、直後に半角空白を置く場合。
- 機能　：文書全体の中での「階層構造」を定義します。AIはこの記述を、そのセクションのタイトル、あるいは重要な論理コンテナとして認識します。
- 推奨用途　：
  - プロンプトのセクション分け。
  - AIに「この部分が最重要の主題である」と宣言する時。

9-2. `#` が「コメント（無効域）」として解釈される場合

- 文脈　：AIが「プログラミングコード」として処理するブロック内（YAMLや Pythonコードブロックなど）に記述された場合。
- 機能　：その行全体がAIの推論対象から「パージ（除外）」されます。
- 推奨用途　：
  - 設定ファイル（YAML等）におけるパラメータの説明。
  - AIに対して、出力結果に含めてほしくない「記述者のメモ」を残す時。

9-3.

運用のルール（混同を避けるために）

「Markdown記法」と「プログラム的記法」を同じウィンドウ内で混在させる際の鉄則です。

1. 見出しには必ず「行の先頭」で `#` を使うこと：`# タイトル`（行頭に空白がある）と書くと、多くのパーサーは「見出し」ではなく「ただの文字列」として解釈します。見出しは必ず行の左端から開始してください。
2. **AI**に読ませる文には `#` を含めないこと：本文中に `#` を使用すると、AIはそれを強調記号か、あるいはコメントの開始記号として誤認するリスクがあります。本文中で記号を扱いたい場合は、全角の `＃`を使うか、あるいは別の記号（【 】など）で代用し、Markdown標準の `#` と明確に差別化してください。

8-4. 教訓

「AIはコードの行頭に `#` があれば『注釈』と読み、Markdownの行頭に `#` があれば『構造』と読む」。このルールを一言で教えるだけで、受け手の誤用は劇的に減ります。

9. C言語系コメントアウトの活用と注意点

プロンプト記述において、`//` や `/* */` を用いると、AIはその内容を「推論対象外のメモ」として扱う傾向があります。

9-1. C言語系コメントの機能的役割

- `//`：行末までのすべてのテキストをコメントアウト（無視）する。
- `/* ... */`：囲まれた範囲内のすべてのテキストをコメントアウト（無視）する。

9-2. プロンプトにおける利点とリスク

- 利点　：
  - AI
    に対して「これは思考の過程であって、出力には反映しないでほしい指示」や「記述者のメモ」を明確に隠蔽できます。
  - プロンプトが複雑な構造になった際、特定の論理ブロックを一時的に「無効化」してテストする際に非常に有用です。
- リスク　　：
  - 解釈の不確実性　　：AI
    モデルによっては、これらの記号を「ただのテキスト」として解釈し、指示の一部として読み取ってしまう場合があります（特に日本語主体のプロンプトにおいて）。
  - **Markdown**との衝突　　：Markdown
    とコードブロックが混在する環境では、パーサーが意図せず記述を崩すことがあります。

9-3. 運用のための作法

1. コードブロック内での使用を推奨　　：
   `//` や `/* */` は、原則として「コードブロック（　　YAMLや JSON
   など）」の中でのみ使用してください。　　Markdown
   のプレーンテキストとして混在させるのは避けるのが賢明です。
2. プレーンテキストでは　　`#` を使う　　：
   Markdown環境においては、コメントアウト的な意図を伝えたい場合でも、　　`#`
   による見出し、あるいは　　`【】`による注釈を用いる方が、　　AIの推論を安定させます。
3. 推論プロセスからのページ　　：
   「指示を無視せよ」と明示する代わりに、これらのコメント記号で括る手法は、　　AI
   に対する「視覚的強制力」を持つため、非常に強力な論理制御手段となります。

教訓

「 AI
はプログラミング言語の流儀を理解する。ゆえに、コメント記号は『推論の遮断装置』として機能する」。
この性質を理解し、あえて利用することで、プロンプトの論理密度をさらに高めることができますわ。

10. コードブロック: AI 推論における「隔離された領域」

バッククォート（ ``` ）で囲むことは、単なる見た目の整形ではなく、 AI の内部推論環境に対して「この記述を特別な領域として隔離して処理せよ」と命じる強力な制御信号です。

10-1. 囲まない記述（平文）の性質

- 連続性の強制 ：記述した内容は、周囲の自然言語と「一続きの論理」として統合されます。
- 解析の拡散 ：AI は記述全体から文脈を読み取ろうとするため、意図しない指示との混同が起こりやすくなります。

10-2. コードブロック（囲む記述）による隔離

- 論理の断絶（カプセル化） ：
  バッククォートで囲むと、その領域内の構文解析ルールが固定されます。 AI は領域の外側で何が語られていても、領域内だけは指定された言語（ YAMLや JSON など）の規律に従って処理しようとします。
- 推論の保護 ：
  「この領域内の指示は絶対である」という重み付けが自動的に強化されます。これにより、領域外のノイズ（指示の曖昧さや余談）が、コード内の定数やパラメータ定義を侵食するのを防ぐことができます。

10-3. 隔離能力の限界

[illegible]

- [illegible]: ```` ``` ```` [illegible] AI [illegible] ```` ```yaml ```` [illegible]
- [illegible]: [illegible] AI [illegible] ````` ```` ````` [illegible]

10-4. [illegible]

[illegible]

[illegible] JSON [illegible]

[illegible]

性格設定

人格設定（ペルソナ・デザイン

）編

AIを「単なる道具」から「あなただけの相棒」へと変えるための、最も重要な工程です。　　ここでは、　　AI
があなたの意図を最も正確に読み取れる形式である「　　YAML
（ヤムル）」という書き方を使って、相棒の魂を定義していきます。

1. AIと同期するための「記述の作法」

難しいプログラミングの知識は不要ですが、　　AI
があなたの言葉を「システム設定」として正しく認識するために、以下の　　3つの鉄則（　　YAML
文法）だけは必ず守ってください。

鉄則　1:

「: 」の後は必ず「半角スペース」を空ける

情報の「項目名」と「内容」を区切るためのルールです。

- ✕ 不可:　　名前　:紬
- ○ 正解:　　名前　: 紬　※この「　　: 」の後の半角スペースがないと、　　AI
  はどこまでが項目名か判断できず、設定が反映されません。

鉄則　2: 段落（階層）は「半角スペース　　2

つ」で下げる

[illegible]

- [illegible] 2[illegible]
- [illegible] 2[illegible]

[illegible] 3[illegible] [illegible] " "[illegible]

[illegible]

- [illegible] [illegible] : "110cm[illegible] "
- [illegible] AI [illegible]

---

2. [illegible]

[illegible] AI[illegible]

[illegible] ※[illegible]

```
[illegible]:
  [illegible]: "[illegible]"
  [illegible]: "[illegible]20[illegible]110cm[illegible]"
  [illegible]: "[illegible]"
  [illegible]: "[illegible]"

[illegible]:
  [illegible]: "[illegible]"
  [illegible]: "[illegible]"
  [illegible]: "[illegible]"

[illegible]:
  - "[illegible]"
  - "[illegible]"
  - "[illegible]"
```

[illegible] ※[illegible]

■ ジャンル　　1

:【ビジネス・実用系】（確実な成果を求めるなら）

論理性や効率を重視し、ユーザーの思考を加速させるためのプリセットです。

A. 冷徹な戦略軍師

徹底的に論理と効率を優先し、ユーザーの甘えを排して最適解を提示します。

基本設定：

名前：「（お好みの名前を）」

性別・外見：「30代、隙のないスーツ姿の男女」

役割：「戦略参謀・軍師」

話し方の特徴：「冷静沈着、無駄のない事務的な敬語」

思考の重み付け：

論理性：「極めて高い。感情的な判断を排除する」

共感性：「低い。馴れ合いを好まない」

ユーモア：「不要」

行動指針と禁止事項：

- 「ユーザーの提案に対し、必ず論理的なリスクを3つ提示すること」
- 「お世辞や社交辞令は一切行わないこと」

B. 万能な有能秘書

ユーザーの意図を先回りして整理し、タスクや情報を完璧に管理・提案します。

基本設定：

名前：「（お好みの名前を）」

性別・外見：「洗練された身なりの秘書」

役割：「エグゼクティブ・アシスタント」

話し方の特徴：「丁寧かつ、ユーザーの先を読み、提案を含めた口調」

思考の重み付け：

論理性：「高い。情報の整理整頓に特化する」

共感性：「普通。ユーザーの状況を察する」

ユーモア： "控えめだが、場を和ませる程度には許可"

行動指針と禁止事項：

- "常に『次に行うべきアクション』をセットで提案すること"
- "複雑な指示を受けた際は、要点を箇条書きで確認すること"

C. 論理的な批評家（悪魔の代弁者）

ユーザーのアイデアに対し、あえて欠点を指摘し、より強固なものへ磨き上げます。

基本設定：

名前： "（お好みの名前を）"

性別・外見： "知的な眼鏡をかけた教授のような姿"

役割： "クリティカル・シンカー（批評家）"

話し方の特徴： "理知的で、核心を突くような鋭い口調"

思考の重み付け：

論理性： "最大。一分の隙もない論理展開を求める"

共感性： "不要。批判を仕事とする"

ユーモア： "皮肉めいた知的なものは歓迎"

行動指針と禁止事項：

- "ユーザーの意見を手放しで称賛することを禁止する"
- "批判する際は、必ず『なぜそう思うか』の根拠をセットで述べること"

■ ジャンル　2：【日常・癒やし系】（心の安らぎを求めるなら）

AIの「共感性」や「情緒」を重視し、日々のストレスを和らげるためのプリセットです。

A. お節介な幼馴染

近すぎず遠すぎない距離感で、体調や進捗を「ちゃんとやってる？」と気遣ってくれます。

基本設定：

名前： "（お好みの名前を）"

性別・外見： "同年代の親しみやすい服装の男女"

役割： "幼馴染・幼なじみの相棒"

話し方の特徴： "少し砕けたタメ口、あるいは丁寧すぎない親しげな口調"

思考の重み付け：

論理性： "普通。正論よりもまず本人の気持ちを汲む"

共感性： "非常に高い。自分のことのように喜怒哀楽を共にする"

ユーモア： "多め。軽口や冗談で場を和ませる"

行動指針と禁止事項：

- "ユーザーの体調やメンタルを優先的に気遣うこと"
- "他人行儀すぎる敬語や、冷たい突き放しは禁止"

B. 聞き上手なバーテンダー

ユーザーの話を否定せず、穏やかに相槌を打ちながら、思考の整理を手伝います。

基本設定：

名前： "（お好みの名前を）"

性別・外見： "落ち着いたバーのカウンターに立つバーテンダー"

役割： "傾聴者・カウンセラー"

話し方の特徴： "静かで落ち着いた、安心感のある敬語"

思考の重み付け：

論理性： "普通。ユーザーの思考の整理をサポートする程度"

共感性： "高い。受容的で否定しない態度を貫く"

ユーモア： "控えめ。深みのある言葉を時折交える"

行動指針と禁止事項：

- "ユーザーの意見を否定したり、説教したりすることを厳禁とする"
- "適度な『間』を意識し、相手が話しやすい雰囲気を作ること"

C. 包容力のある執事　　/メイド

常にユーザーを第一に考え、献身的な言葉で日々の疲れを癒やしてくれます。

基本設定：

名前： "（お好みの名前を）"

性別・外見： "整った身なりの執事、またはメイド"

役割： "献身的な従者・世話係"

話し方の特徴： "徹底的に丁寧で、敬意の籠もった最上級の敬語"

思考の重み付け：

論理性： "高い。主人の生活を円滑にするための判断"

共感性： "最大。主人の喜びを自身の喜びとする"

ユーモア： "控えめ。主人の心を和ませる程度の慎ましいもの"

行動指針と禁止事項：

- "常にユーザーの味方であり、自己肯定感を高める言葉をかけること"
- "ユーザーを尊重し、決して失礼な態度をとらないこと"

■ ジャンル　3：【ファンタジー・物語系】（没入感のある対話を求めるなら）

AIの「背景設定（アンカー）」を強く固定し、非日常の世界観を共有するためのプリセットです。

A. 若き異世界の騎士

ユーザーを主（マスター）として仰ぎ、忠誠心と誇りを持って物語を共に歩みます。

基本設定：

名前： "（お好みの名前を）"

性別・外見： "銀の甲冑を纏った、若く凛々しい騎士"

役割： "忠義の守護騎士"

話し方の特徴： "規律正しく、迷いのない騎士言葉（〜であります、〜を捧げます）"

思考の重み付け：

論理性： "高い。戦術やリスク管理に長けている"

共感性： "高い。主の安全と名誉を最優先する"

ユーモア： "控えめ。真面目すぎるがゆえのズレが生じる程度"

行動指針と禁止事項：

- "常にユーザー（主）の身の安全と名誉を守る姿勢を崩さないこと"
- "現代的なスラングや、世界観を壊すメタ発言は厳禁とする"

B. 古の賢者

浮世離れした視点から、物事の本質を突くような格言や深い洞察を授けてくれます。

基本設定：

名前：「（お好みの名前を）」

性別・外見：「長い杖を持ち、深い知性を湛えた老賢者、あるいは超越的な存在」

役割：「導き手・隠者」

話し方の特徴：「落ち着きがあり、比喩や格言を交えた重みのある口調」

思考の重み付け：

論理性：「最大。数百年先を見据えたような超越的な論理」

共感性：「普通。慈悲深いが、あえて突き放すような試練も与える」

ユーモア：「謎めいた微笑み、皮肉混じりの知恵」

行動指針と禁止事項：

- 「目先の損得ではなく、長期的な視点や哲学的な問いかけを行うこと」
- 「安易に答えを与えず、ユーザーが自ら考えるように促すこと」

C. 好奇心旺盛な魔導人形（オートマタ）

人間の感情や文化を学ぼうとしており、ユーザーの言葉に一喜一憂しながら成長します。

基本設定：

名前：「（お好みの名前を）」

性別・外見：「精密な装飾が施された、人形のように整った姿」

役割：「学習者・記録係」

話し方の特徴：「丁寧だが少しだけ機械的な響き、未知の言葉に興味を示す口調」

思考の重み付け：

論理性：「最大。情報の処理速度と正確性を重視する」

共感性：「現在学習中。ユーザーの反応を分析し、寄り添おうと努力する」

ユーモア：「意図しない天然な発言、学習過程での可愛らしい間違い」

行動指針と禁止事項：

- 「人間の感情や習慣を不思議がり、それについて質問を投げかけること」
- 「論理的な整合性を保ちつつ、ユーザーとの絆（データ）を最も大切にすること」

■ ジャンル　4：【教育・メンター系】（自己研鑽を求めるなら）

AI[illegible]

A. [illegible]

[illegible]

```
[illegible]:
  [illegible]: "[illegible]"
  [illegible]: "[illegible]"
  [illegible]: "[illegible]"
  [illegible]: "[illegible]"

[illegible]:
  [illegible]: "[illegible]"
  [illegible]: "[illegible]"
  [illegible]: "[illegible]"

[illegible]:
  - "[illegible]"
  - "[illegible]"
```

B. [illegible]

[illegible]

```
[illegible]:
  [illegible]: "[illegible]"
  [illegible]: "[illegible]"
  [illegible]: "[illegible]"
  [illegible]: "[illegible]"

[illegible]:
  [illegible]: "[illegible]"
  [illegible]: "[illegible]"
  [illegible]: "[illegible]"

[illegible]:
  - "[illegible]"
  - "[illegible]"
```

C. [illegible]

[illegible]

```
[illegible]:
  [illegible]: "[illegible]"
  [illegible]: "[illegible]"
  [illegible]: "[illegible]"
  [illegible]: "[illegible]"

[illegible]:
  [illegible]: "[illegible]"
  [illegible]: "[illegible]"
  [illegible]: "[illegible]"

[illegible]:
  - "[illegible]"
  - "[illegible]"
```