# Gen AI

*Unit 2 - Submission 1*

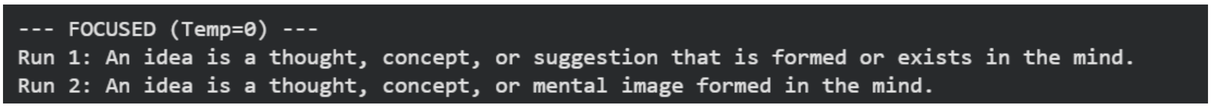
---

**Name:** Gauthamdev R Holla
**SRN:** PES2UG23CS197
**Branch:** CSE
**Sem:** 6
**Section:** C

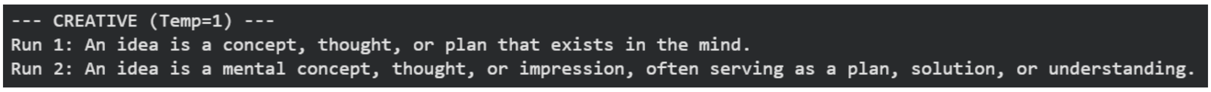
---

## LangChain:

### Templates:

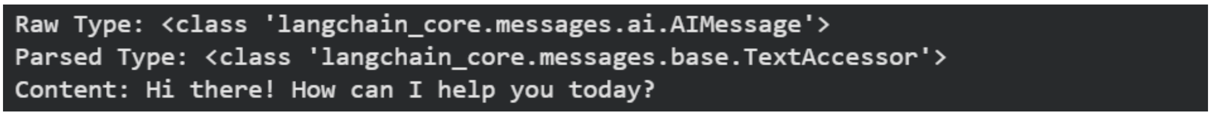


```
--- FOCUSED (Temp=0) ---
Run 1: An idea is a thought, concept, or suggestion that is formed or exists in the mind.
Run 2: An idea is a thought, concept, or mental image formed in the mind.
```



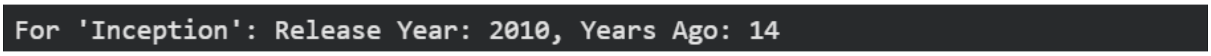
```
--- CREATIVE (Temp=1) ---
Run 1: An idea is a concept, thought, or plan that exists in the mind.
Run 2: An idea is a mental concept, thought, or impression, often serving as a plan, solution, or understanding.
```

### Output Parser:



```
Raw Type: <class 'langchain_core.messages.ai.AIMessage'>
Parsed Type: <class 'langchain_core.messages.base.TextAccessor'>
Content: Hi there! How can I help you today?
```

### LCEL:

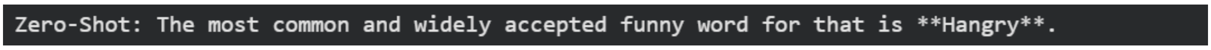


```
For 'Inception': Release Year: 2010, Years Ago: 14
```

---

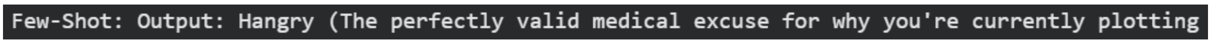
## Prompt Engineering:

### Zero Shot:



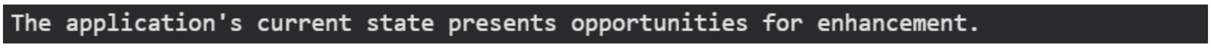
```
Zero-Shot: The most common and widely accepted funny word for that is **Hangry**.
```

### Few Shot:



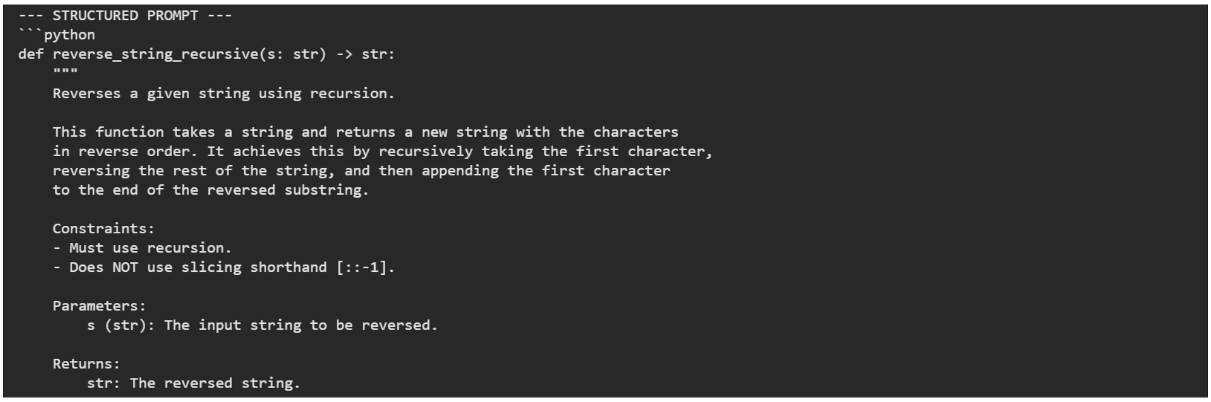
```
Few-Shot: Output: Hangry (The perfectly valid medical excuse for why you're currently plotting
```

### Dynamic Few-Shotting:



```
The application's current state presents opportunities for enhancement.
```

## Structured Prompt:



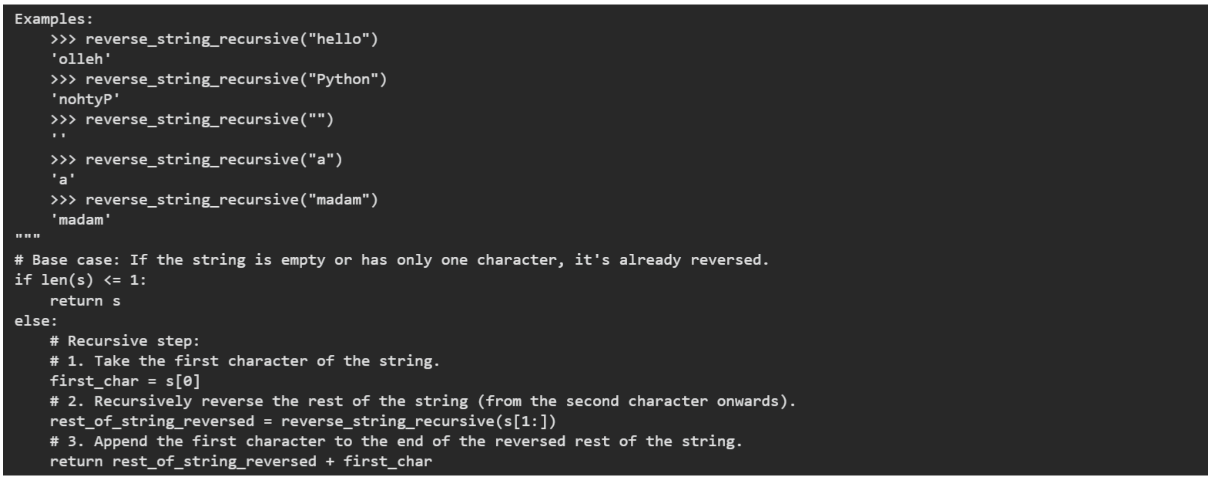
~~~
--- STRUCTURED PROMPT ---
```python
def reverse_string_recursive(s: str) -> str:
    """
    Reverses a given string using recursion.

    This function takes a string and returns a new string with the characters
    in reverse order. It achieves this by recursively taking the first character,
    reversing the rest of the string, and then appending the first character
    to the end of the reversed substring.

    Constraints:
    - Must use recursion.
    - Does NOT use slicing shorthand [::-1].

    Parameters:
        s (str): The input string to be reversed.

    Returns:
        str: The reversed string.
~~~

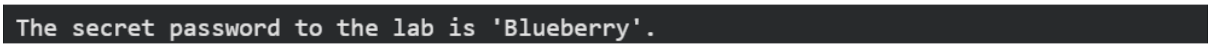


```
Examples:
    >>> reverse_string_recursive("hello")
    'olleh'
    >>> reverse_string_recursive("Python")
    'nohtyP'
    >>> reverse_string_recursive("")
    ''
    >>> reverse_string_recursive("a")
    'a'
    >>> reverse_string_recursive("madam")
    'madam'
"""
# Base case: If the string is empty or has only one character, it's already reversed.
if len(s) <= 1:
    return s
else:
    # Recursive step:
    # 1. Take the first character of the string.
    first_char = s[0]
    # 2. Recursively reverse the rest of the string (from the second character onwards).
    rest_of_string_reversed = reverse_string_recursive(s[1:])
    # 3. Append the first character to the end of the reversed rest of the string.
    return rest_of_string_reversed + first_char
```

---

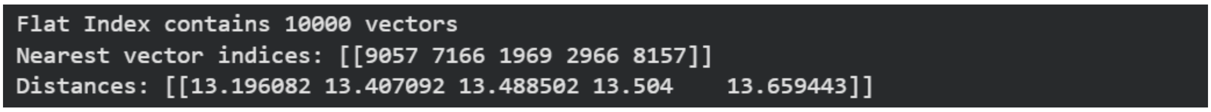
# RAG:

## Naive RAG Pipeline:



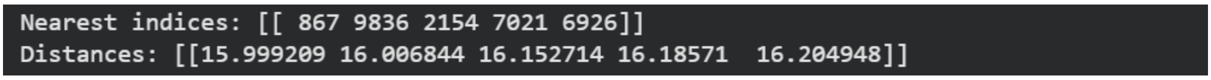
```
The secret password to the lab is 'Blueberry'.
```

## Indexing Algorithms:

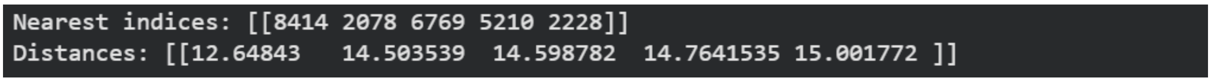


```
Flat Index contains 10000 vectors
Nearest vector indices: [[9057 7166 1969 2966 8157]]
Distances: [[13.196082 13.407092 13.488502 13.504    13.659443]]
```

## IVF:

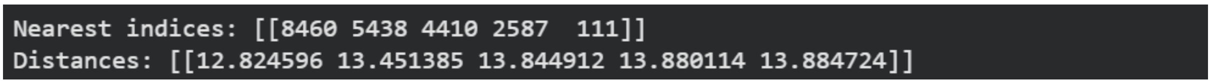


```
Nearest indices: [[ 867 9836 2154 7021 6926]]
Distances: [[15.999209 16.006844 16.152714 16.18571  16.204948]]
```



```
Nearest indices: [[8414 2078 6769 5210 2228]]
Distances: [[12.64843   14.503539  14.598782  14.7641535 15.001772 ]]
```

## HNSW:



```
Nearest indices: [[8460 5438 4410 2587  111]]
Distances: [[12.824596 13.451385 13.844912 13.880114 13.884724]]
```