~~~python
import os
from langchain_google_genai import ChatGoogleGenerativeAI
from langchain_core.prompts import PromptTemplate

os.environ["GOOGLE_API_KEY"] = "MY GEMINI KEY"

llm = ChatGoogleGenerativeAI(
    model="gemini-2.5-flash",
    temperature=0
)

prompt = PromptTemplate.from_template("""
# Context
You are a Senior Python Dev.

# Objective
Write a function to reverse a string.

# Constraints
1. Must use Python 3.10
2. It must use recursion (no slicing [::-1]).

# Output Format
Include detailed docstrings.
""")

chain = prompt | llm

response = chain.invoke({})
print(response.content)

```python
import sys

# Ensure Python 3.10 or newer is being used
if sys.version_info < (3, 10):
    raise RuntimeError("This script requires Python 3.10 or newer.")

def reverse_string_recursive(s: str) -> str:
    """
    Reverses a given string using a recursive approach.

    This function takes an input string and returns a new string with
its
    characters in reverse order. It strictly adheres to the constraint
of
    using recursion and explicitly avoids built-in string reversal
mechanisms
    like slicing (e.g., `[::-1]`).

    The recursion works by:
~~~

```
    1.  **Base Case**: If the string is empty or contains only one
character,
        it is already reversed, so it's returned as is. This prevents
infinite
        recursion.
    2.  **Recursive Step**: For strings longer than one character, the
function
        takes the first character (`s[0]`), recursively reverses the
rest of
        the string (`s[1:]`), and then appends the first character to
the
        *end* of the result from the recursive call. This effectively
moves
        each character from the front to the back, one by one, as the
recursion
        unwinds.

    Args:
        s (str): The input string to be reversed. Must be a string.

    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("racecar")
        'racecar'
        >>> reverse_string_recursive("12345")
        '54321'

    Constraints Fulfilled:
        -   **Python 3.10**: Compatible with Python 3.10 and newer.
        -   **Recursion**: The function calls itself
(`reverse_string_recursive(s[1:])`).
        -   **No Slicing `[::-1]`**: Explicitly avoids the `[::-1]`
slicing syntax.
            It uses `s[1:]` to get a substring (all but the first
character)
            and `s[0]` to get the first character, which are standard
string
            access methods, not a full reversal slice.
    """
    # Base case: An empty string or a string with a single character
    # is its own reverse. This is where the recursion stops.
    if len(s) <= 1:
        return s
    else:
        # Recursive step:
        # 1. Take the substring starting from the second character
(s[1:]).
        # 2. Recursively call the function on this substring.
        # 3. Append the first character of the original string (s[0])
        #    to the result of the recursive call.
        return reverse_string_recursive(s[1:]) + s[0]

if __name__ == "__main__":
    # --- Test Cases ---
    print("--- Testing reverse_string_recursive ---")

    test_cases = [
        ("hello", "olleh"),
        ("Python", "nohtyP"),
        ("", ""),
        ("a", "a"),
        ("racecar", "racecar"),
        ("madam", "madam"),
        ("12345", "54321"),
        ("programming", "gnimmargorp"),
        ("level", "level"),
        ("A", "A"),
        ("ab", "ba"),
        ("abc", "cba"),
    ]

    for input_str, expected_output in test_cases:
        actual_output = reverse_string_recursive(input_str)
        print(f"Input: '{input_str}'")
        print(f"Expected: '{expected_output}'")
        print(f"Actual:   '{actual_output}'")
        assert actual_output == expected_output, \
            f"Test failed for '{input_str}'. Expected
'{expected_output}', got '{actual_output}'"
        print("Status: PASSED\n")

    print("All test cases passed successfully!")

    # Example of using the function directly
    my_string = "Senior Python Dev"
    reversed_my_string = reverse_string_recursive(my_string)
    print(f"Original string: '{my_string}'")
    print(f"Reversed string: '{reversed_my_string}'")
```