Applying BRAD Framework in Generative AI

- Identify **B (Background)**, **R (Role)**, **A (Action)**, **D (Details)**.
- Design a **final prompt** using BRAD.
- Generate and analyze the AI output.

Question 1: Resume Generator

Scenario:
You are helping final-year students prepare resumes using a Generative AI tool.
Use the BRAD framework to create a prompt that generates a professional resume summary.

Question 2: Code Explanation

Scenario:
You are an instructor explaining a Python program that finds the factorial of a number.
Use BRAD to design a prompt that makes the AI explain the code in a simple, step-by-step manner for beginners.

Question 3: Poster Design

Scenario:
You are designing a poster for "Women in Technology."
Use BRAD to generate an AI image showing women working with futuristic AI tools in an office environment.

Question 4: Educational Content Summary

Scenario:
You are creating short revision notes for the topic "Cloud Computing."
Use BRAD to create a prompt that makes the AI generate concise and student-friendly notes under 80 words.

Question 5: Code Generation

Scenario:
You are learning Java programming.
Use BRAD to generate a Java program to check whether a number is prime, with short explanation comments.

Question 6: Logo Design

Scenario:
You are designing a logo for your college's AI Club.
Use BRAD to make an AI image generator create a modern, minimal logo with a neural network or brain symbol.

1. ### Text Generation using GPT-2 with BRAD Framework

Solution:

```
# Install and import

!pip install transformers torch -q

from transformers import GPT2LMHeadModel, GPT2Tokenizer

tokenizer = GPT2Tokenizer.from_pretrained("gpt2")

model = GPT2LMHeadModel.from_pretrained("gpt2")

background = "You are preparing content about Artificial Intelligence and its real-world impact."

role = "Act as a technology expert and educator."

action = "Explain how AI is transforming different industries in simple terms."

details = "Keep the tone informative and inspiring for college students."

# Combine BRAD parts into a structured prompt

input_text = f"Background: {background}\nRole: {role}\nAction: {action}\nDetails: {details}\n\nResponse:"

# Encode and generate

inputs = tokenizer.encode(input_text, return_tensors='pt')

outputs = model.generate(

    inputs,

    max_length=120,

    num_return_sequences=1,

    temperature=0.9,

    repetition_penalty=1.2,

    no_repeat_ngram_size=3

)


# Decode and print

print(tokenizer.decode(outputs[0], skip_special_tokens=True))
```