```
# Assignmet 8: Text generation: Develop applications for text generation tasks such as story generation using trained Generative AI mode

from transformers import GPT2LMHeadModel, GPT2Tokenizer
import torch

# Load model and tokenizer
model_name = "gpt2"
tokenizer = GPT2Tokenizer.from_pretrained(model_name)
model = GPT2LMHeadModel.from_pretrained(model_name)

# Set pad_token_id if not present
tokenizer.pad_token = tokenizer.eos_token

# Input prompt
prompt = "Once upon a time in a distant galaxy,"
inputs = tokenizer(prompt, return_tensors="pt", padding=True)

# Generate text with better decoding strategy
outputs = model.generate(
    inputs['input_ids'],
    attention_mask=inputs['attention_mask'],
    max_length=100,
    temperature=0.8,              # Controls randomness (lower = more focused)
    top_k=50,                     # Top-k sampling
    top_p=0.95,                   # Nucleus sampling
    repetition_penalty=1.2,      # Penalize repetition
    pad_token_id=tokenizer.eos_token_id,
    num_return_sequences=1,
    do_sample=True               # Sampling instead of greedy decoding
)

# Decode and print
generated_text = tokenizer.decode(outputs[0], skip_special_tokens=True)
print("Generated Story:\n", generated_text)
```

```
Generated Story:
 Once upon a time in a distant galaxy, an immense cosmic storm left its mark on Earth. It was the first of many "great" events to ta
It took only three weeks between June 24th 1947 when Apollo 14 landed at Kennedy Space Center with crew aboard NASA astronauts John
```

```
# Assignment 8: Text generation: Develop applications for text generation tasks such as dialogue generation using trained Generative AI m


from transformers import AutoModelForCausalLM, AutoTokenizer
import torch

# Load pre-trained English dialogue model
model_name = "microsoft/DialoGPT-medium"  # Use medium for better quality
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(model_name)

# Initialize chat history
chat_history_ids = None

print(" English Chatbot is ready! Type 'exit' to quit.")

# Run a loop for 5 dialogue exchanges
for step in range(5):
    user_input = input(" You: ")
    if user_input.lower() == "exit":
        break

    # Encode user input and add end-of-string token
    new_input_ids = tokenizer.encode(user_input + tokenizer.eos_token, return_tensors='pt')

    # Concatenate with history
    bot_input_ids = torch.cat([chat_history_ids, new_input_ids], dim=-1) if chat_history_ids is not None else new_input_ids

    # Generate a reply with tuned parameters
    chat_history_ids = model.generate(
        bot_input_ids,
        max_length=1000,
        pad_token_id=tokenizer.eos_token_id,
        do_sample=True,
        top_k=50,
        top_p=0.92,
        temperature=0.7,
        repetition_penalty=1.2,
    )
```

```
# Extract bot reply
response = tokenizer.decode(chat_history_ids[:, bot_input_ids.shape[-1]:][0], skip_special_tokens=True)
print(" Bot:", response)
```

```
tokenizer_config.json: 100%  614/614 [00:00<00:00, 69.3kB/s]
vocab.json:  1.04M/? [00:00<00:00, 47.2MB/s]
merges.txt:  456k/? [00:00<00:00, 21.0MB/s]
config.json: 100%  642/642 [00:00<00:00, 56.8kB/s]
pytorch_model.bin: 100%  863M/863M [00:43<00:00, 8.20MB/s]
model.safetensors: 100%  863M/863M [01:26<00:00, 11.2MB/s]
generation_config.json: 100%  124/124 [00:00<00:00, 1.67kB/s]
🤖 English Chatbot is ready! Type 'exit' to quit.
👤 You: Hi
🤖 Bot: Hey, what's your name?!
👤 You: amit patil
🤖 Bot: I've had a few beers
👤 You: yes, how many
🤖 Bot: A lot.
👤 You: plz give me 100
🤖 Bot: Hahahaha no problem
👤 You: what is the price of each
🤖 Bot: 10 for 5.
```