```
import gradio as gr
import torch
from transformers import AutoTokenizer, AutoModelForCausalLM

# Load model and tokenizer
model_name = "ibm-granite/granite-3.2-2b-instruct"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(
    model_name,
    torch_dtype=torch.float16 if torch.cuda.is_available() else torch.float32,
    device_map="auto" if torch.cuda.is_available() else None
)

if tokenizer.pad_token is None:
    tokenizer.pad_token = tokenizer.eos_token

def generate_response(prompt, max_length=512):
    inputs = tokenizer(prompt, return_tensors="pt", truncation=True, max_length=5

    if torch.cuda.is_available():
        inputs = {k: v.to(model.device) for k, v in inputs.items()}

    with torch.no_grad():
        outputs = model.generate(
            **inputs,
            max_length=max_length,
            temperature=0.7,
            do_sample=True,
            pad_token_id=tokenizer.eos_token_id
        )

    response = tokenizer.decode(outputs[0], skip_special_tokens=True)
    response = response.replace(prompt, "").strip()
    return response

def concept_explanation(concept):
    prompt = f"Explain the concept of {concept} in detail with examples:"
    return generate_response(prompt, max_length=800)

def quiz_generator(concept):
    prompt = f"Generate 5 quiz questions about {concept} with different question
    return generate_response(prompt, max_length=1000)

# Create Gradio interface
with gr.Blocks() as app:
    gr.Markdown("# Educational AI Assistant")

    with gr.Tabs():
        with gr.TabItem("Concept Explanation"):
            concept_input = gr.Textbox(label="Enter a concept", placeholder="e.g.
            explain_btn = gr.Button("Explain")
            explanation_output = gr.Textbox(label="Explanation", lines=10)

            explain_btn.click(concept_explanation, inputs=concept_input, outputs=

        with gr.TabItem("Quiz Generator"):
            quiz_input = gr.Textbox(label="Enter a topic", placeholder="e.g., phy
            quiz_btn = gr.Button("Generate Quiz")
            quiz_output = gr.Textbox(label="Quiz Questions", lines=15)
```

```
                quiz_btn.click(quiz_generator, inputs=quiz_input, outputs=quiz_output

    app.launch(share=True)
```

```
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to acces
  warnings.warn(
```

tokenizer_config.json: 8.88k/? [00:00<00:00, 145kB/s]

vocab.json: 777k/? [00:00<00:00, 3.61MB/s]

merges.txt: 442k/? [00:00<00:00, 6.38MB/s]

tokenizer.json: 3.48M/? [00:00<00:00, 39.5MB/s]

added_tokens.json: 100% 87.0/87.0 [00:00<00:00, 1.62kB/s]

special_tokens_map.json: 100% 701/701 [00:00<00:00, 15.0kB/s]

config.json: 100% 786/786 [00:00<00:00, 29.6kB/s]

```
`torch_dtype` is deprecated! Use `dtype` instead!
```

model.safetensors.index.json: 29.8k/? [00:00<00:00, 1.90MB/s]

Fetching 2 files: 100% 2/2 [01:23<00:00, 83.96s/it]

model-00001-of-00002.safetensors: 100% 5.00G/5.00G [01:23<00:00, 73.3MB/s]

model-00002-of-00002.safetensors: 100% 67.1M/67.1M [00:10<00:00, 5.41MB/s]

Loading checkpoint shards: 0% 0/2 [00:00<?, ?it/s]

```
generation_config.json:   0%|          | 0.00/137 [00:00<?, ?B/s]
Colab notebook detected. To show errors in colab notebook, set debug=True
* Running on public URL: https://55d29b03744d6d03a6.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgr
```

Educational AI Assistant

Concept Explanation ...

Quiz Generator

Enter a concept

e.g., machine learning