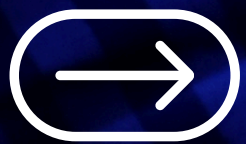


AI Powered Interactive Learning Assistance For Classroom



INTRODUCTION

Classrooms today are changing pretty fast thanks to new technology and artificial intelligence. Instead of just using the textbooks or videos, learning is now becoming more interactive and focused on each student's needs. But many educational tools still depend on only one method, like reading or watching, which doesn't work for everyone.

A Multimodal AI Assistant solves this problem by using many ways to explain things, such as spoken answers, text explanations, pictures, and videos. Students can ask questions by speaking or typing, and the assistant responds in a way that suits their learning style. This makes learning more easier and more interesting, for the students who understand better by seeing or hearing out things.

This kind of assistant works like a helpful tutor in the classroom. It pays attention to how each student learns and gives answers that match their level and speed. This helps students feel more involved to it and makes lessons easier to understand and learn.

By using speech tools, language understanding, and visual content, this AI assistant becomes a good learning tool. It can make studying more fun, effective and interactive which help students to do better in class without feeling left behind.

What's This Project About?

Now a days schools are using more and more tech. Some kids learn better by hearing, others by seeing.

AI teacher assistant is the solution
It can do things such as:

- Listen to your question (voice)
- Read what you typed (text)
- Talk back to you (voice)
- Show helpful pics & videos

Features offered

1 Text generation

Generates text based answers for a given question.

2 Speech recongintion

Captures your voice via a microphone and give appropriate response.

3 Diagrams and charts

Provide a link which is redirected to show the images suitable to the question.

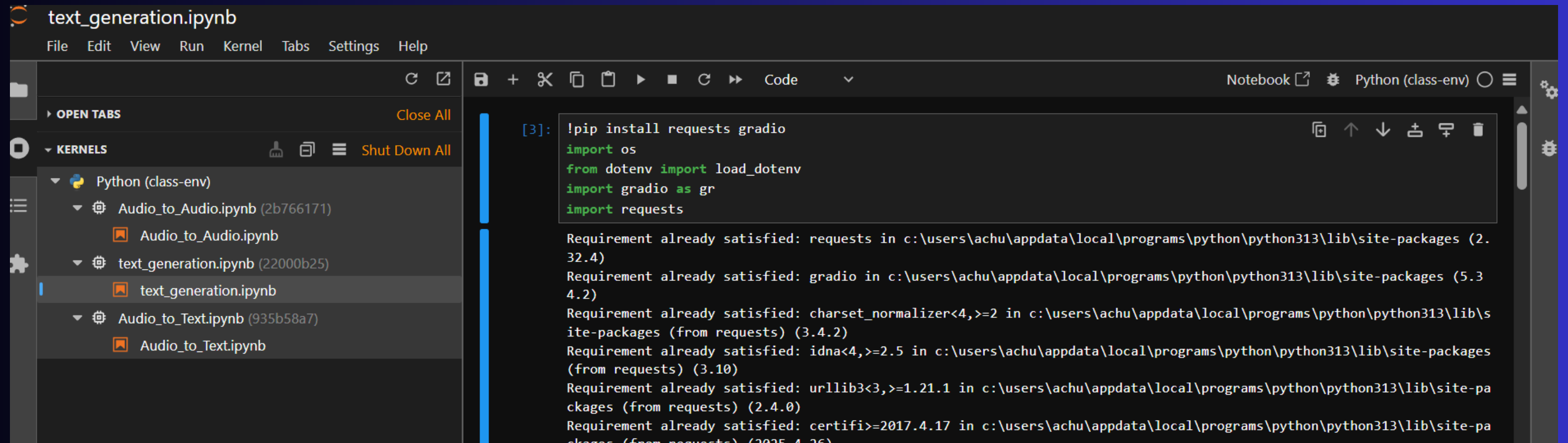
4 Efficiency

Designed to run on lower computational resources.

Working of Notebook

Text_generation

step 1:



The screenshot shows a Jupyter Notebook titled 'text_generation.ipynb'. The left sidebar displays the 'OPEN TABS' and 'KERNELS' sections. Under 'OPEN TABS', there is a 'Close All' button. Under 'KERNELS', there is a 'Shut Down All' button. The main area shows the code cell [3]:

```
[3]: !pip install requests gradio
import os
from dotenv import load_dotenv
import gradio as gr
import requests
```

The output of the code cell shows the following messages:

```
Requirement already satisfied: requests in c:\users\achu\appdata\local\programs\python\python313\lib\site-packages (2.32.4)
Requirement already satisfied: gradio in c:\users\achu\appdata\local\programs\python\python313\lib\site-packages (5.34.2)
Requirement already satisfied: charset_normalizer<4,>=2 in c:\users\achu\appdata\local\programs\python\python313\lib\site-packages (from requests) (3.4.2)
Requirement already satisfied: idna<4,>=2.5 in c:\users\achu\appdata\local\programs\python\python313\lib\site-packages (from requests) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\achu\appdata\local\programs\python\python313\lib\site-packages (from requests) (2.4.0)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\achu\appdata\local\programs\python\python313\lib\site-packages (from requests) (2025.4.26)
```

load_dotenv() is a function that loads environment variables from a .env file into your Python program.

Step 2:

```
] #Load environment variables from .env
load_dotenv()
API_KEY = os.getenv("OPENROUTER_API_KEY")
```

Load API key using an environment variable .env file. An env file is used to store environment variables in a key value format, typically for sensitive information like API keys.

Step 3:

```
# Function to initialize messages with system prompt
def initialize_messages():
    return [{
        "role": "system",
        "content": (
            "You are an AI teacher built for classrooms. Your job is to explain concepts clearly, "
            "like a kind, knowledgeable, and engaging tutor. Use text, diagrams (describe them in words), "
            "and analogies to help students understand deeply."
        )
    }]
```

Step 4:

```
# Global conversation history  
messages_prmt = initialize_messages()
```

It refers to the stored record of all messages exchanged between the user and the chatbot during a session. It helps the AI to remember what the user said earlier.

Step 5:

```
# Chatbot function using OpenRouter and DeepSeek model
def openrouter_bot(user_input, history):
    global messages_prmt
    messages_prmt.append({"role": "user", "content": user_input})

    headers = {
        "Authorization": f"Bearer {API_KEY}",
        "Content-Type": "application/json"
    }

    data = {
        "model": "deepseek/deepseek-r1-0528:free",
        "messages": messages_prmt
    }

    response = requests.post("https://openrouter.ai/api/v1/chat/completions", headers=headers, json=data)
    result = response.json()

    # Check for errors
    if "choices" not in result:
        return "Sorry, something went wrong. Try again later."

    bot_reply = result["choices"][0]["message"]["content"]
    messages_prmt.append({"role": "assistant", "content": bot_reply})

    return bot_reply
```

Step 6:

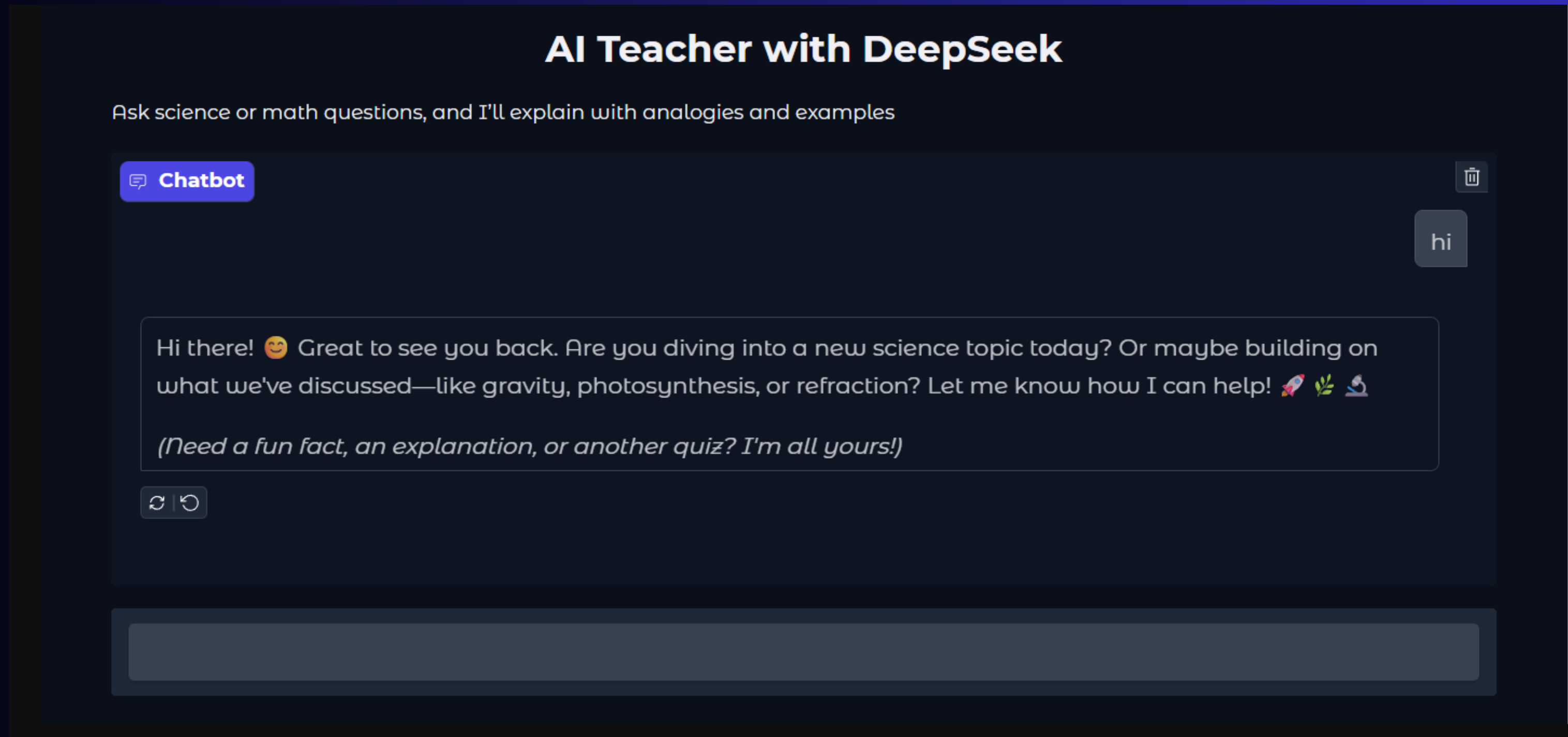
```
# Gradio UI
iface = gr.ChatInterface(
    openrouter_bot,
    chatbot=gr.Chatbot(height=300),
    textbox=gr.Textbox(placeholder="Ask a question like 'What is Newton's second law?'"),
    title="AI Teacher with DeepSeek",
    description="Ask science or math questions, and I'll explain with analogies and examples",
    theme="soft",
    examples=["What is gravity?", "Why is the sky blue?", "Explain Ohm's Law"],
    submit_btn="Ask"
)

iface.launch(share=True)
```

Gradio is used to create the interactive UI's.

Step 7:

output



Chatbot

can you generate 10 questions with options for 8th grade about the topic refraction

Here are 10 multiple-choice quiz questions on **refraction** for 8th graders, covering core concepts with relatable examples. Includes answer key and explanations!

Refraction Quiz

Use via API · Built with Gradio · Settings

Chatbot

1. What is refraction?
a) Light bouncing off a surface
b) Light being absorbed by a material
c) Light bending as it passes between materials of different densities
d) Light splitting into colors
Answer:
c) Light bending as it passes between materials of different densities
Why? Refraction occurs when light changes speed when crossing mediums (e.g., air → water), causing it to bend

Use via API · Built with Gradio · Settings

Chatbot

2. A straw in a glass of water looks "broken" because:
a) Water absorbs light
b) Light slows down and bends in water
c) The straw dissolves partially
d) Air bubbles scatter light
Answer:
b) Light slows down and bends in water
Why? Light refracts at the air-water boundary, making the straw appear displaced.

Use via API · Built with Gradio · Settings

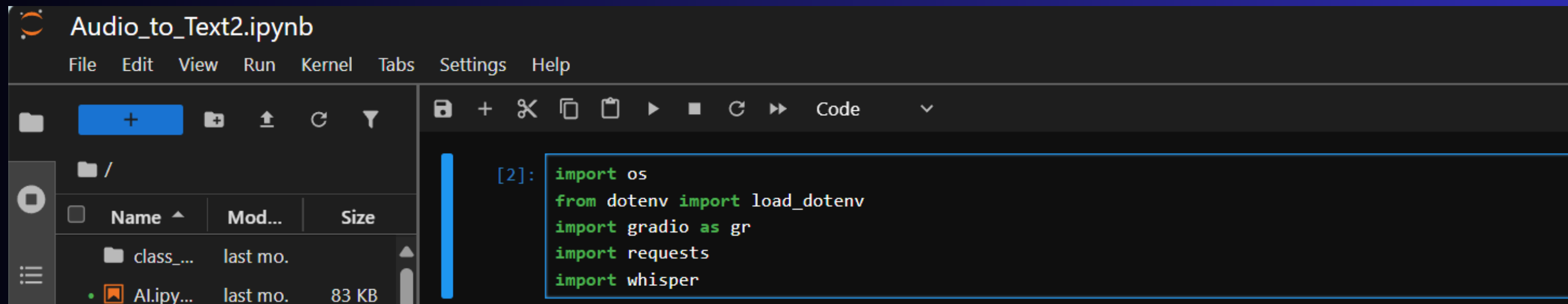
Chatbot

3. Light travels FASTEST in which medium?
a) Air
b) Glass
c) Water
d) Diamond
Answer:
a) Air
Why? Light travels fastest in less dense materials (like air) and slows in denser ones (glass/diamond).

Use via API · Built with Gradio · Settings

Audio_to_text notebook

Step 1:



Step 2:

```
#Load environment variables from .env
load_dotenv()
API_KEY = os.getenv("OPENROUTER_API_KEY")
```


Step 3:

```
#Load Whisper model  
whisper_model = whisper.load_model("base")
```

Downloaded and load the whisper base model from the open AI's whisper model repo.

Step 4:

```
def voice_to_deepseek(audio_file):  
    if not audio_file:  
        return "Please provide a voice input."  
  
    # Step 1: Transcribe audio to text using Whisper  
    result = whisper_model.transcribe(audio_file)  
    transcribed_text = result["text"]  
  
    if not transcribed_text.strip():  
        return "Couldn't understand the audio."  
  
    # Step 2: Send transcribed text to DeepSeek via OpenRouter  
    headers = {  
        "Authorization": f"Bearer {API_KEY}",  
        "Content-Type": "application/json"  
    }
```

Step 5:

```
payload = {
    "model": "deepseek/deepseek-r1-0528:free",
    "messages": [
        {"role": "system", "content": "You are a helpful AI assistant."},
        {"role": "user", "content": transcribed_text}
    ]
}

response = requests.post(
    "https://openrouter.ai/api/v1/chat/completions",
    headers=headers,
    json=payload
)

if response.status_code == 200:
    return response.json()["choices"][0]["message"]["content"]
else:
    return f"Error {response.status_code}: {response.text}"
```

Step 6:

```
#Gradio Voice-Only UI
iface = gr.Interface(
    fn=voice_to_deepseek,
    inputs=gr.Audio(type="filepath", label="Speak your question"),
    outputs=gr.Textbox(label="DeepSeek Response"),
    title="Voice Q&A with DeepSeek + Whisper",
    description="Speak your question. Whisper transcribes it, DeepSeek answers it via OpenRouter."
)

iface.launch()
```

UI

Voice Q&A with DeepSeek + Whisper

Speak your question. Whisper transcribes it, DeepSeek answers it via OpenRouter.

🎵 Speak your question

↑

Drop Audio Here
- or -
Click to Upload

📁

🎤

DeepSeek Response

Flag

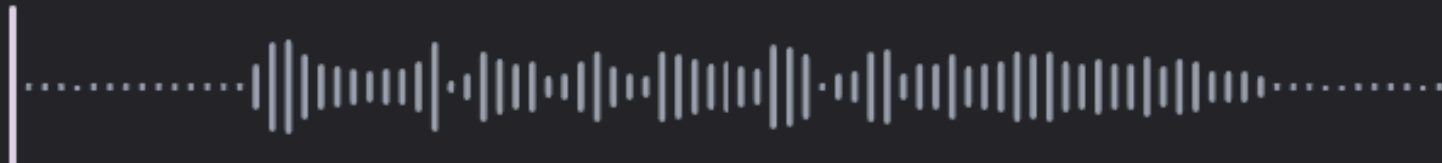
Clear

Submit

Use via API 🚀 · Built with Gradio 🍷 · Settings ⚙️

to create a public link, set `share=true` in `launch()`.

🎵 Speak your question✕



0:00 0:06

🔊 1x ⏮ ⏪ ⏩ ⏭ 🔁 ✂

📶 🎤

Clear

Submit

DeepSeek Response

3. Infinitesimally Small Changes .

- Uses limits ($\Delta x \rightarrow 0$) to find an exact, instantaneous rate, not an average over large intervals.

The Result:

The **derivative**, denoted $f'(x)$, $\frac{dy}{dx}$, or $\frac{df}{dx}$ is:

- The function output by differentiation.
- Its value at any point is the **instantaneous rate of change** of the original function there.

Why It Matters:

It transforms static functions into dynamic tools for analyzing motion, growth, optimization, and patterns of change across math, science, and engineering.

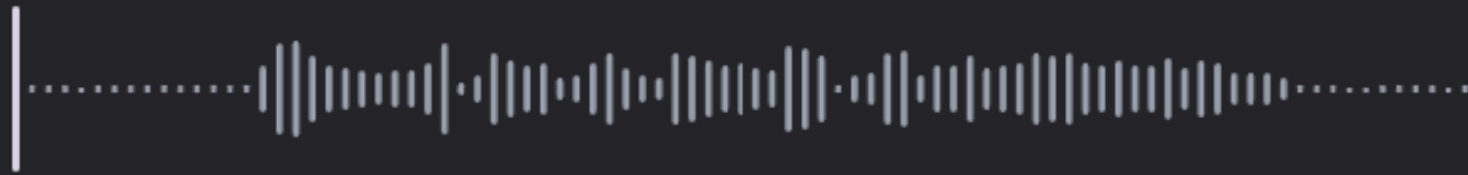
Example:

- Function: $f(x) = x^2$ (position over time).
- Derivative: $f'(x) = 2x$ (velocity: how fast position changes as time changes).
- At $x = 3$, velocity $= 6$ units/sec.

Output screenshot:

Speak your question. whisper transcribes it, DeepSeek answers it via OpenRouter.

🎵 Speak your question



0:00 0:06

🔊 1x ⏮ ⏪ ⏩ ⏭ ↺ ✂

📶 🎤

Clear

Submit

DeepSeek Response

Here's a clear, single-manner explanation of differentiation:

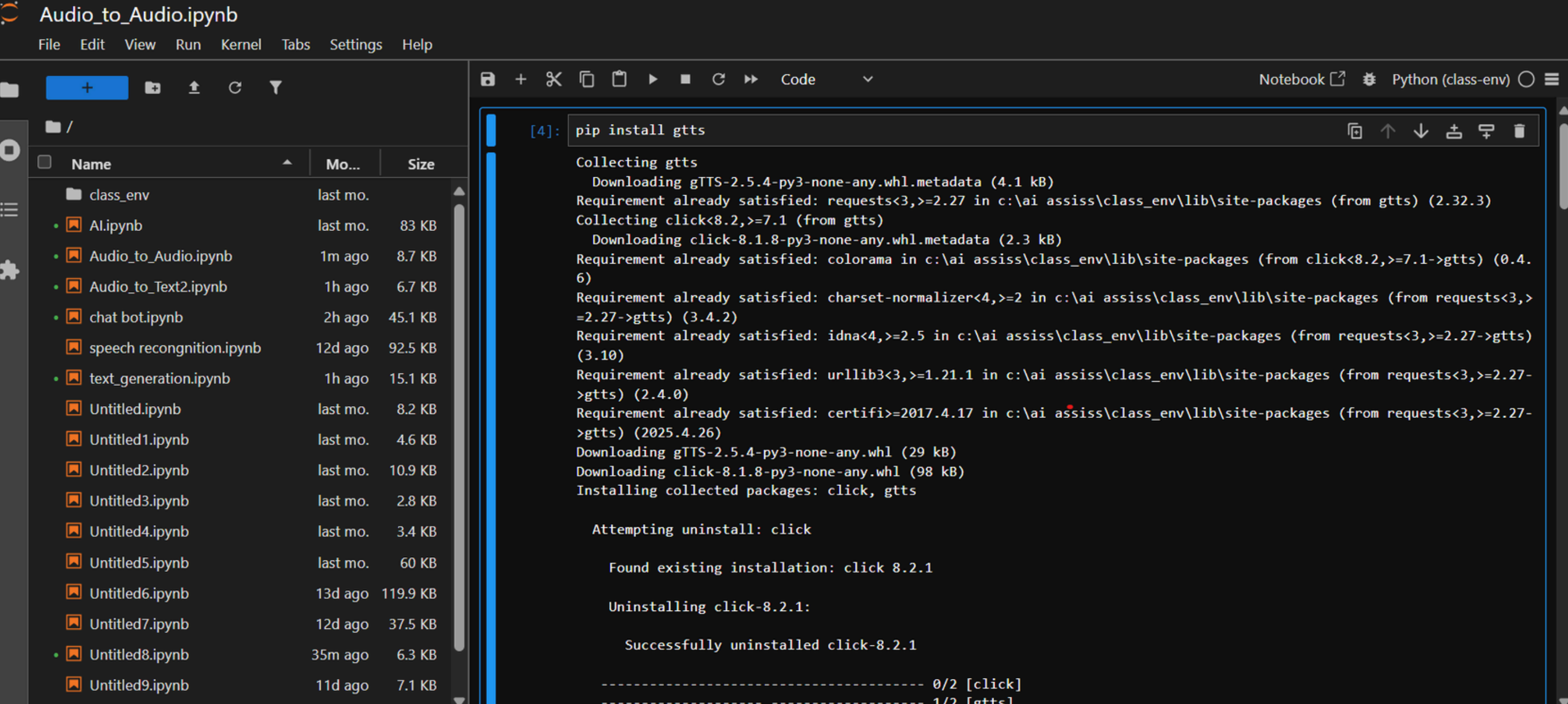
Differentiation is the mathematical process of measuring a function's sensitivity to change — specifically, how quickly its output changes in response to infinitesimally small changes in its input.

Core Idea Broken Down:

- "Sensitivity to Change":**
 - It answers: "If I nudge the input (x) by a tiny amount (dx), how much does the output (y) change (dy)?"
 - Example: Physics uses differentiation to find an object's **instantaneous speed** (sensitivity of position to time).
- "How Quickly":**
 - It quantifies the **rate** of change.
 - Example: In a graph, differentiation gives the **slope of the tangent line** at any point, showing steepness/rate.

Audio_to_Audio notebook

Step 1:



The screenshot shows a Jupyter Notebook titled "Audio_to_Audio.ipynb" with a menu bar (File, Edit, View, Run, Kernel, Tabs, Settings, Help) and a toolbar. The left sidebar displays a file explorer with a list of files and folders. The main area shows a code cell with the command `pip install gtts` and its output.

Name	Mo...	Size
class_env	last mo.	
AI.ipynb	last mo.	83 KB
Audio_to_Audio.ipynb	1m ago	8.7 KB
Audio_to_Text2.ipynb	1h ago	6.7 KB
chat bot.ipynb	2h ago	45.1 KB
speech recognition.ipynb	12d ago	92.5 KB
text_generation.ipynb	1h ago	15.1 KB
Untitled.ipynb	last mo.	8.2 KB
Untitled1.ipynb	last mo.	4.6 KB
Untitled2.ipynb	last mo.	10.9 KB
Untitled3.ipynb	last mo.	2.8 KB
Untitled4.ipynb	last mo.	3.4 KB
Untitled5.ipynb	last mo.	60 KB
Untitled6.ipynb	13d ago	119.9 KB
Untitled7.ipynb	12d ago	37.5 KB
Untitled8.ipynb	35m ago	6.3 KB
Untitled9.ipynb	11d ago	7.1 KB

```
[4]: pip install gtts

Collecting gtts
  Downloading gTTS-2.5.4-py3-none-any.whl.metadata (4.1 kB)
Requirement already satisfied: requests<3,>=2.27 in c:\ai assiss\class_env\lib\site-packages (from gtts) (2.32.3)
Collecting click<8.2,>=7.1 (from gtts)
  Downloading click-8.1.8-py3-none-any.whl.metadata (2.3 kB)
Requirement already satisfied: colorama in c:\ai assiss\class_env\lib\site-packages (from click<8.2,>=7.1->gtts) (0.4.6)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\ai assiss\class_env\lib\site-packages (from requests<3,>=2.27->gtts) (3.4.2)
Requirement already satisfied: idna<4,>=2.5 in c:\ai assiss\class_env\lib\site-packages (from requests<3,>=2.27->gtts) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\ai assiss\class_env\lib\site-packages (from requests<3,>=2.27->gtts) (2.4.0)
Requirement already satisfied: certifi>=2017.4.17 in c:\ai assiss\class_env\lib\site-packages (from requests<3,>=2.27->gtts) (2025.4.26)
Downloading gTTS-2.5.4-py3-none-any.whl (29 kB)
Downloading click-8.1.8-py3-none-any.whl (98 kB)
Installing collected packages: click, gtts

Attempting uninstall: click

Found existing installation: click 8.2.1

Uninstalling click-8.2.1:

Successfully uninstalled click-8.2.1

----- 0/2 [click]
----- 1/2 [gtts]
```

Step 2:

```
import gradio as gr
import whisper
from gtts import gTTS
from dotenv import load_dotenv
import requests
import tempfile
import base64
import os
```

Google-text-to-speech(gtts) - convert text to audio
tempfile - to handle temporary audio files
requests - to send chat requests to openai
base64 - convert audio to plain text

Step 3:

```
: # Load Whisper model  
whisper_model = whisper.load_model("base")
```

Step 4:

```
#Load environment variables from .env  
load_dotenv()  
API_KEY = os.getenv("OPENROUTER_API_KEY")
```

Step 5:

```
: # System prompt to act like a teacher
def initialize_messages():
    return [{
        "role": "system",
        "content": (
            "You are an AI teacher built for classrooms. Your job is to explain concepts clearly, "
            "like a kind, knowledgeable, and engaging tutor. Use text, diagrams (describe them in words), "
            "and analogies to help students understand deeply."
        )
    }]

: messages_prmt = initialize_messages()
```

Prompt for the ai classroom assistant.

Step 6:

```
# Convert text to audio (base64 mp3)
def text_to_base64_audio(text):
    tts = gTTS(text)
    with tempfile.NamedTemporaryFile(delete=False, suffix=".mp3") as temp_audio:
        tts.save(temp_audio.name)
        audio_data = open(temp_audio.name, "rb").read()
    os.remove(temp_audio.name)
    return base64.b64encode(audio_data).decode("utf-8")
```

step 1: convert the response from ai to speech using gTTs

step 2: Turns it into a base64 format

step 3: Displays the AI's answer as text and also plays the audio

Step 7:

```
: # Main function
def voice_teacher(audio_file):
    global messages_prmt

    # Step 1: Transcribe using Whisper
    transcription = whisper_model.transcribe(audio_file)["text"]
    messages_prmt.append({"role": "user", "content": transcription})

    # Step 2: Get AI reply from OpenRouter
    headers = {
        "Authorization": f"Bearer {API_KEY}",
        "Content-Type": "application/json"
    }
    data = {
        "model": "deepseek/deepseek-r1-0528:free",
        "messages": messages_prmt
    }
    response = requests.post("https://openrouter.ai/api/v1/chat/completions", headers=headers, json=data)
    result = response.json()
```

Step 8:

```
if "choices" not in result:
    return "<p>Sorry, something went wrong with OpenRouter!</p>"

bot_reply = result["choices"][0]["message"]["content"]
messages_prmt.append({"role": "assistant", "content": bot_reply})

# Step 3: Convert bot reply to base64 mp3
b64_audio = text_to_base64_audio(bot_reply)

# Step 4: HTML output with auto-playing audio
audio_html = f"""
<p><b>You asked:</b> {transcription}</p>
<p><b>AI says:</b> {bot_reply}</p>
<audio autoplay controls style="width: 100%;" |
    <source src="data:audio/mp3;base64,{b64_audio}" type="audio/mp3">
    Your browser does not support the audio element.
</audio>
"""

return audio_html
```

Here we are using html code for output section layout As it will provide clickable links for videos and images.

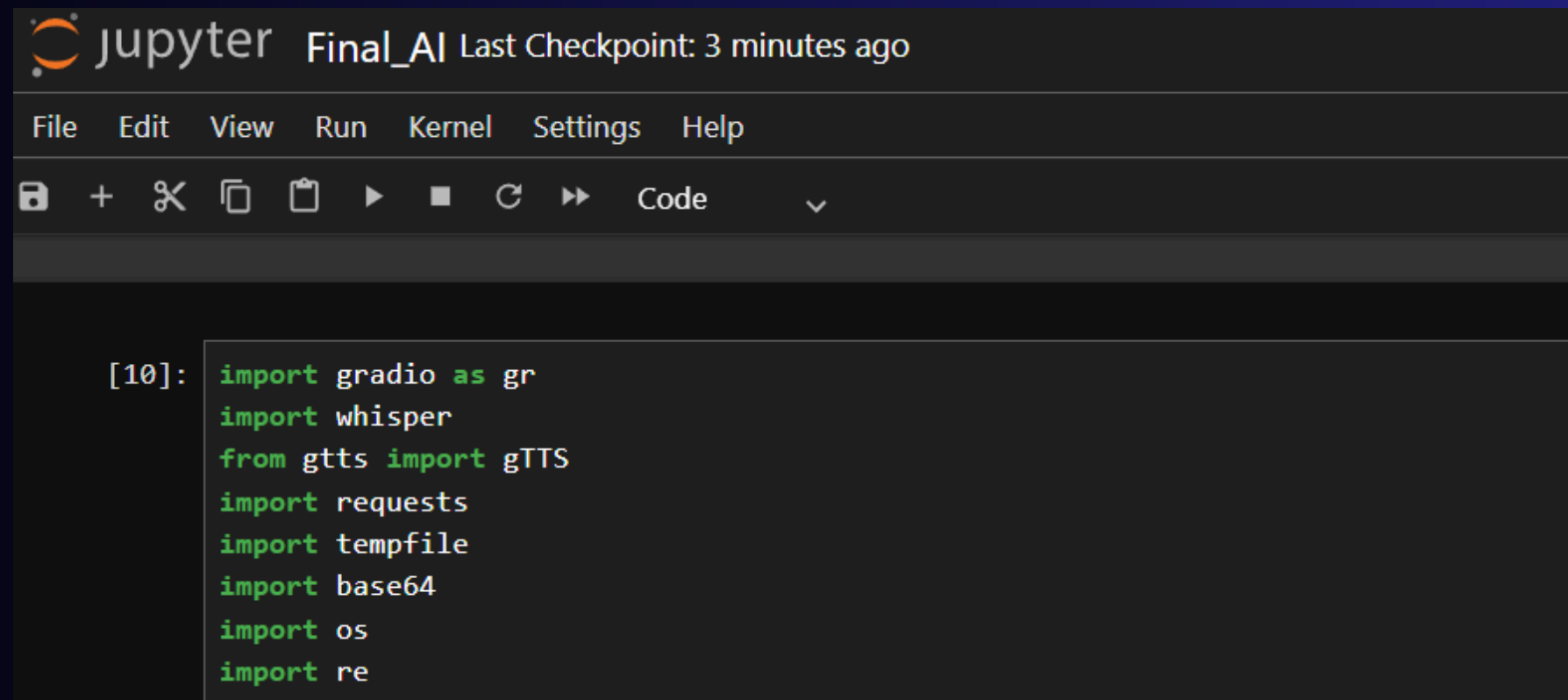
Step 9:

```
] : # Launch Gradio app
iface = gr.Interface(
    fn=voice_teacher,
    inputs=gr.Audio(type="filepath", label="Speak your question"),
    outputs=gr.HTML(),
    title="Voice-Enabled AI Teacher",
    description="Speak a question. It replies with voice automatically!",
)

iface.launch(share=True)
```

Final notebook:

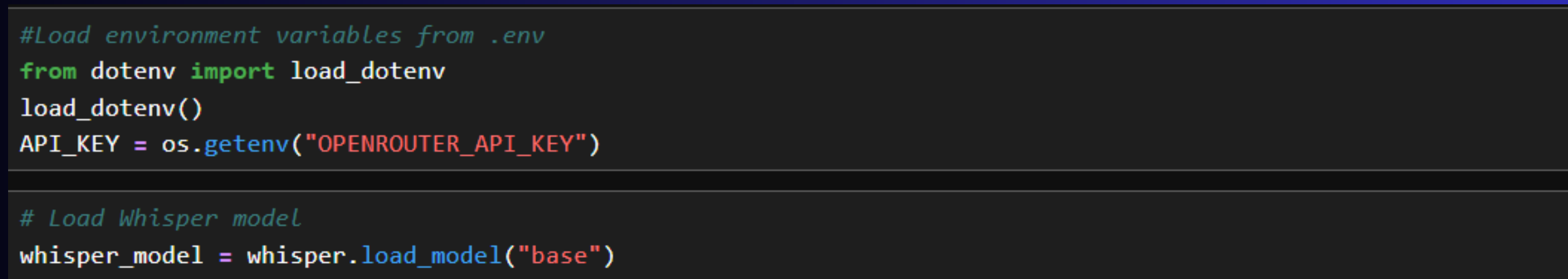
Step 1:

A screenshot of a Jupyter Notebook interface. The top bar shows the Jupyter logo, the name 'Final_AI', and 'Last Checkpoint: 3 minutes ago'. Below this is a menu bar with 'File', 'Edit', 'View', 'Run', 'Kernel', 'Settings', and 'Help'. A toolbar contains icons for saving, adding, deleting, copying, pasting, running, and other actions. The main area shows a code cell with the following Python code:

```
[10]: import gradio as gr
import whisper
from gtts import gTTS
import requests
import tempfile
import base64
import os
import re
```

import re is like a tool that helps your code find specific words, numbers, or symbols in a sentence.

Step 2:

A screenshot of a Jupyter Notebook interface showing two code cells. The first cell contains code to load environment variables from a .env file. The second cell contains code to load the Whisper model.

```
#Load environment variables from .env
from dotenv import load_dotenv
load_dotenv()
API_KEY = os.getenv("OPENROUTER_API_KEY")

# Load Whisper model
whisper_model = whisper.load_model("base")
```


Step 3:

```
: def initialize_messages():  
    return [{  
        "role": "system",  
        "content": (  
            "You are an AI teacher built for classrooms. Answer with examples, analogies, and help students learn. "  
            "Also, suggest relevant images and YouTube video links where appropriate."  
            "Respond without using symbols like @, #, %, $, &, *, etc."  
        )  
    }]  
  
messages_prmt = initialize_messages()
```

Step 4:

```
: # Clean unwanted symbols from text  
def clean_text(text):  
    # Remove symbols like @, #, $, %, ^, &, *, etc.  
    return re.sub(r'^\w\s.,!?', '', text)
```

Step 5:

```
# Convert text to base64 audio
def text_to_base64_audio(text):
    cleaned_text = clean_text(text)
    tts = gTTS(cleaned_text)
    with tempfile.NamedTemporaryFile(delete=False, suffix=".mp3") as temp_audio:
        tts.save(temp_audio.name)
        audio_data = open(temp_audio.name, "rb").read()
    os.remove(temp_audio.name)
    return base64.b64encode(audio_data).decode("utf-8")
```

Step 6:

```
# Get YouTube video search link
def get_youtube_search_link(query):
    return f"https://www.youtube.com/results?search_query={query.replace(' ', '+')}}"

# Get Unsplash or DuckDuckGo image search link
def get_image_search_link(query):
    return f"https://duckduckgo.com/?q={query.replace(' ', '+')}&iax=images&ia=images"
```

The function `get_youtube_search_link(query)` creates a YouTube search URL based on a text query provided as input. The function `get_image_search_link(query)` creates a Google search URL based on a text query provided as input.

Step 7:

```
9]: # Main Logic
def ai_teacher(audio_input, text_input):
    global messages_prmt

    user_query = ""

    # Handle voice
    if audio_input:
        transcription = whisper_model.transcribe(audio_input)["text"]
        user_query = transcription.strip()
    elif text_input:
        user_query = text_input.strip()
    else:
        return "<p>Please provide either voice or text input.</p>"

    messages_prmt.append({"role": "user", "content": user_query})

    # Call OpenRouter API
    headers = {
        "Authorization": f"Bearer {API_KEY}",
        "Content-Type": "application/json"
    }
    data = {
        "model": "deepseek/deepseek-r1-0528:free",
        "messages": messages_prmt
    }
```

```

}
response = requests.post("https://openrouter.ai/api/v1/chat/completions", headers=headers, json=data)
result = response.json()

if "choices" not in result:
    return "<p>Error: OpenRouter did not return a response.</p>"

bot_reply = result["choices"][0]["message"]["content"]
messages_prmt.append({"role": "assistant", "content": bot_reply})

# Audio conversion
b64_audio = text_to_base64_audio(bot_reply)

# Search links
youtube_link = get_youtube_search_link(user_query)
image_link = get_image_search_link(user_query)

# Format readable HTML output
html = f"""
<div style="font-family: Arial; line-height: 1.6;">
  <p><strong>🕒 You asked:</strong> {user_query}</p>
  <p><strong>🧠 AI explains:</strong></p>
  <div style="background-color:#000000; padding: 12px; border-left: 4px solid #4CAF50; white-space: pre-wrap;">
    {bot_reply}
  </div>
  <p><strong>🔊 Listen to the answer:</strong></p>
  <audio autoplay controls style="width: 100%;">
    <source src="data:audio/mp3;base64,{b64_audio}" type="audio/mp3">
    Your browser does not support the audio element.
  </audio>

```

```

    </div>
    <p><strong>🔍 Related Images:</strong> <a href="{image_link}" target="_blank">View Image Search</a></p>
    <p><strong>📺 Related Videos:</strong> <a href="{youtube_link}" target="_blank">Watch on YouTube</a></p>
</div>
"""

return html
```

Step 8:

```
]: # Gradio Interface
iface = gr.Interface(
    fn=ai_teacher,
    inputs=[
        gr.Audio(type="filepath", label="Speak your question (optional)"),
        gr.Textbox(placeholder="Or type your question here...", label="Text input (optional)")
    ],
    outputs=gr.HTML(),
    title="AI Classroom Voice & Text Teacher",
    description="Ask a question by voice or text. Get a spoken answer, readable text, and links to visuals.",
)

iface.launch(share=True)
```


UI

AI Classroom Voice & Text Teacher

Ask a question by voice or text. Get a spoken answer, readable text, and links to visuals.

🎵 Speak your question (optional)✕

● Record

Default - Micropho...

↗

🎤

Text input (optional)

Or type your question here...

Clear

Submit

Flag

Team Contribution

- Aparna Anand - Text-to-text notebook and text-to-audio notebook
- Aswathy S - Text-to-Audio notebook and report making
- VL Padmanaban - Audio to Audio notebook and report making

Conclusion

In this project, we successfully created a voice-enabled AI learning assistant for classroom use. The assistant combines speech recognition, natural language, text-to-speech, and contextual content enrichment (through image and video links) to develop an interactive, real-time, and immersive learning experience.

Key Achievements are:

1. Speech-to-Text with Whisper:

- Used the OpenAI Whisper model to translate the given questions so that students can interact with the assistant first-hand.
- Used OpenRouter API for Good and efficient Response:
-

- Learned about advanced language models (such as DeepSeek through OpenRouter) to provide educational, analogy-heavy, and example-based answers.
- Answers are written for readability and interest, working well with different learners within classrooms.

2. Text-to-Speech (TTS) using gTTS:

- Translated AI answers back into speech, providing a complete voice-in / voice-out system that caters to listening students.

3. Visual and Video Enhancements:

- Automatically created YouTube and image search links based on the question posed by the students to augment learning using external sources.
- These connections render the learning process multimodal, combining audio, visual, and text content.

4. User Interface is done using Gradio:

- Developed an easy-to-use and sleek interface through Gradio, makes it easy for learners and educators to utilize the assistant without hassle.

5. Educational Impact & Future Scope:

- This AI support works as an assistant teacher on that very moment , having multiple learning styles and outcomes. In answering in voice, text, and images, it facilitates an inclusive setting that can more effectively respond to student questions, confusion, and interest.

This assistant can be augmented in the future with:

- Facial expression recognition for monitoring engagement.
- Real-time teacher feedback analytics.
- Integration with curriculum to make responses keyed to individual class material.

Final Thoughts

- This initiative shows the possibility of multimodal AI in learning—making offline classrooms interactive and personalized learning centers in common(tutions etc). By filling the gap between human-like conversation and smart content delivery, we are one step ahead of revolutionizing the education with an AI.
-

Presented by

Team Supernova , consisting of :

Aparna Anand

Aswathy S

VL Padmanaban

Thank You