The objective of this program is to **create an integrative visual and sound dictionary system** using Python. Students will learn to:

1. Combine multiple libraries (e.g., `gTTS`, `IPython.display`, `requests`) to build a functional application.
2. Display images and play sounds based on user input.
3. Understand how to integrate external resources (e.g., image URLs, sound files) into a program.
4. Develop problem-solving and debugging skills while creating an interactive system.

This lab emphasizes **integrative programming** by merging multimedia, user interaction, and external data handling into a single application.

A **visual dictionary system** is a tool that allows users to search for words and retrieve corresponding images or visual representations. Below is a simple implementation of a visual dictionary system using Python in Google Colab. This example uses a pre-defined set of words and their corresponding image URLs.

Simple Visual Dictionary System in Python

```
import requests
from IPython.display import Image, display

# Predefined dictionary of words and their corresponding image URLs
visual_dictionary = {
  "apple": "https://upload.wikimedia.org/wikipedia/commons/thumb/1/15/Red_Apple.jpg/1200px-Red_Apple.jpg",
  "banana": "https://upload.wikimedia.org/wikipedia/commons/8/8a/Banana-Single.jpg",
  "cat": "https://upload.wikimedia.org/wikipedia/commons/thumb/4/4d/Cat_November_2010-1a.jpg/1200px-Cat_November_2010-1a.jpg",
  "dog": "https://upload.wikimedia.org/wikipedia/commons/thumb/5/5f/Dog_Breeds.jpg/1200px-Dog_Breeds.jpg",
  "car": "https://upload.wikimedia.org/wikipedia/commons/thumb/5/5a/2011_Toyota_Camry_--_NHTSA.jpg/1200px-2011_Toyota_Camry_--_NHTSA.jpg",
  "tree": "https://upload.wikimedia.org/wikipedia/commons/thumb/4/4f/Tree_in_the_Desert.jpg/1200px-Tree_in_the_Desert.jpg"
}

# Function to display image for a given word
def display_image(word):
```

```
    if word.lower() in visual_dictionary:
        image_url = visual_dictionary[word.lower()]
        display(Image(url=image_url, width=300))
    else:
        print(f"No image found for the word: {word}")

# User input
word = input("Enter a word to see its visual representation: ")
display_image(word)
```

How to Run in Google Colab:

1. Open Google Colab.
2. Create a new notebook.
3. Copy and paste the code above into a code cell.
4. Run the cell.
5. When prompted, enter a word (e.g., "apple", "cat", "car") to see its visual representation.

want to create a **sound dictionary system** where users can input a word and hear its corresponding sound (e.g., pronunciation, animal sound, or object sound), you can use Python in Google Colab. Below is a simple implementation that uses pre-defined sound files or text-to-speech (TTS) for pronunciation.

Simple Sound Dictionary System in Python

Option 1: Using Predefined Sound Files

This example uses pre-defined sound files (e.g., MP3 or WAV) stored online.

```
from IPython.display import Audio, display
import requests

# Predefined dictionary of words and their corresponding sound URLs
sound_dictionary = {
    "cat": "https://www.soundhelix.com/examples/mp3/SoundHelix-Song-1.mp3",  # Example sound
    "dog": "https://www.soundhelix.com/examples/mp3/SoundHelix-Song-2.mp3",  # Example sound
    "bird": "https://www.soundhelix.com/examples/mp3/SoundHelix-Song-3.mp3",  # Example sound
    "cow": "https://www.soundhelix.com/examples/mp3/SoundHelix-Song-4.mp3",  # Example sound
}

# Function to play sound for a given word
def play_sound(word):
    if word.lower() in sound_dictionary:
```

```
    sound_url = sound_dictionary[word.lower()]
    display(Audio(url=sound_url, autoplay=True))
  else:
    print(f"No sound found for the word: {word}")

# User input
word = input("Enter a word to hear its sound: ")
play_sound(word)
```

Option 2: Using Text-to-Speech (TTS)

This example uses a TTS library like `gTTS` (Google Text-to-Speech) to generate pronunciation for any word.

```
from gtts import gTTS
from IPython.display import Audio, display
import os

# Function to generate and play pronunciation for a given word
def play_pronunciation(word):
  tts = gTTS(text=word, lang='en')  # Generate TTS audio
  tts.save("output.mp3")  # Save as MP3 file
  display(Audio("output.mp3", autoplay=True))  # Play the audio
  os.remove("output.mp3")  # Clean up the file

# User input
word = input("Enter a word to hear its pronunciation: ")
play_pronunciation(word)
```

How to Run in Google Colab:

1. Open Google Colab.
2. Create a new notebook.
3. Copy and paste the code for either Option 1 or Option 2 into a code cell.
4. Run the cell.
5. When prompted, enter a word (e.g., "cat", "dog") to hear its sound or pronunciation.

Explanation:

Option 1: Predefined Sound Files

1. **Sound Dictionary**: A dictionary (`sound_dictionary`) is created where each key is a word (e.g., "cat") and the value is a URL pointing to a sound file (e.g., MP3).

2. **User Input**: The user is prompted to enter a word.
3. **Sound Playback**: If the word exists in the dictionary, the corresponding sound is played using `IPython.display.Audio`. If the word is not found, a message is shown.

Option 2: Text-to-Speech (TTS)

1. **TTS Library**: The `gTTS` library is used to convert the input word into speech.
2. **Audio Playback**: The generated audio is saved as an MP3 file, played using `IPython.display.Audio`, and then deleted to clean up.

Example Output:

- If the user enters `"cat"` in **Option 1**, the corresponding sound file will play.
- If the user enters `"hello"` in **Option 2**, the TTS system will pronounce the word "hello".

Enhancements:

1. **Add More Sounds**: Expand the `sound_dictionary` with more words and sound URLs.
2. **Dynamic Sound Fetching**: Use an API to fetch sounds dynamically (e.g., animal sounds from an online database).
3. **Language Support**: Modify the TTS system to support multiple languages.
4. **UI**: Use `ipywidgets` to create a more interactive interface.

Explanation:

1. **Visual Dictionary**: A dictionary (`visual_dictionary`) is created where each key is a word (e.g., "apple") and the value is a URL pointing to an image of that word.
2. **User Input**: The user is prompted to enter a word.
3. **Image Display**: If the word exists in the dictionary, the corresponding image is displayed using `IPython.display.Image`. If the word is not found, a message is shown.

Example Output:

If the user enters "apple", the output will display an image of an apple.

Enhancements:

1. **Add More Words**: Expand the visual_dictionary with more words and image URLs.
2. **Dynamic Image Search**: Use an API like Google Custom Search or Unsplash to dynamically fetch images for any word.
3. **Error Handling**: Improve error handling for invalid inputs or broken image URLs.
4. **UI**: Use a library like ipywidgets to create a more interactive interface.

Let me know if you'd like help with any of these enhancements!