```
def text_to_speech():
    text_speech = pyttsx3.init()
    answer = input("What do you want to convert to speech? ")
    text_speech.say(answer)
    text_speech.runAndWait()
```

1) The text_to_speech() function is defined

2) An instance of the pyttsx3 library is created using pyttsx3.init() and assigned to the variable text_speech.

3) The user is prompted to enter the text they want to convert to speech using input().

4) The entered text is passed to the say() method of the text_speech object, which stores it for conversion.

5) The runAndWait() method is called to initiate the text-to-speech conversion and wait for it to complete.

```
def speech_to_text():
    r = sr.Recognizer()

    def record_text():
        while True:
            try:
                with sr.Microphone() as source2:
                    r.adjust_for_ambient_noise(source2, duration=0.2)
                    audio2 = r.listen(source2)
                    MyText = r.recognize_google(audio2)
                    return MyText
            except sr.RequestError as e:
                print("Could not request results; {0}".format(e))
            except sr.UnknownValueError:
                print("speak something")
        return

    def output_text(text):
        with open("output.txt", "a") as f:
            f.write(text)
            f.write("\n")

    while True:
        text = record_text()
        output_text(text)
        print("Text written to output.txt")
```

1) The speech_to_text() function is defined.

2) An instance of the sr.Recognizer() class from the speech_recognition library is created and assigned to the variable r.

3) The record_text() function is defined inside speech_to_text().

4) Inside record_text(), a while True loop is used to continuously record audio from the microphone using sr.Microphone() as source2.

5) The adjust_for_ambient_noise() method is called on r to reduce background noise.

6) The listen() method is used to record audio from the microphone and store it in the audio2 variable.

7) The recognize_google() method is called on r to convert the recorded audio to text using Google's speech recognition API. The recognized text is stored in the MyText variable.

8) The MyText variable is returned from the record_text() function.

9) The output_text() function is defined inside speech_to_text().

10) Inside output_text(), the recognized text is written to a file named "output.txt" using the with statement and the write() method.

11) Back in speech_to_text(), the record_text() function is called to record and recognize speech.

12) The recognized text is passed to the output_text() function to write it to the "output.txt" file.

13) A message is printed indicating that the text has been written to the file

```
def main():
    while True:
        choice = input("Enter 'text' for text-to-speech or 'speech' for speech-to-text (or 'exit' to quit): ")
        if choice.lower() == 'text':
            text_to_speech()
        elif choice.lower() == 'speech':
            speech_to_text()
        elif choice.lower() == 'exit':
            print("Exiting...")
            break
```

```
        else:
            print("Invalid choice. Please try again.")
```

1) The main() function is defined.

2) A while True loop is used to continuously prompt the user for input.

3) The user is asked to enter 'text' for text-to-speech, 'speech' for speech-to-text, or 'exit' to quit the program.

4) If the user enters 'text', the text_to_speech() function is called.

5) If the user enters 'speech', the speech_to_text() function is called.

6) If the user enters 'exit', a message is printed, and the loop is broken using break.

7) If the user enters an invalid choice, an error message is printed.

- Finally, the if **name** == "**main**": block ensures that the main() function is executed when the script is run directly.
- This script allows users to convert text to speech or speech to text based on their choice. The text-to-speech functionality uses the pyttsx3 library, while the speech-to-text functionality uses the speech_recognition library.

```
from gtts import gTTS, lang
import os
from tkinter import *
from tkinter import messagebox
import speech_recognition as sr
```

This Python script provides a graphical user interface (GUI) to convert text to speech and speech to text using the gTTS (Google Text-to-Speech) and speech_recognition libraries. Below is a detailed explanation of the code, broken down into sections for clarity.

Importing Required Libraries

1) gTTS: This library is used for converting text to speech using Google's Text-to-Speech API.

2) os: This module allows interaction with the operating system, such as playing audio files.

3) tkinter: This is a standard GUI toolkit in Python used to create windows and user interfaces.

4) speech_recognition: This library is used for recognizing speech and converting it to text.

```
def text_to_speech():
    text = text_entry.get("1.0", "end-1c")
    language = accent_entry.get()

    if (len(text) <= 1) | (len(language) <= 0):
        messagebox.showerror(message="Enter required details")
        return

    speech = gTTS(text=text, lang=language, slow=False)
    speech.save("text.mp3")
    os.system("mpg123 " + "text.mp3")
```

Text to Speech Conversion

1) Input Retrieval: The function retrieves text from a text box (text_entry) and the desired language from an entry field (accent_entry).

2) Input Validation: It checks if the text and language inputs are valid. If not, an error message is displayed.

3) Text-to-Speech Conversion: The gTTS function converts the text into speech, specifying the language and speed.

4) Saving and Playing Audio: The generated speech is saved as an MP3 file, which is then played using the os.system command.

```
def list_languages():
    messagebox.showinfo(message=list(lang.tts_langs().items()))
```

Listing Supported Languages

- This function displays a list of supported languages and their corresponding language codes in a message box.

```
def speech_to_text():
    recorder = sr.Recognizer()
    try:
        duration = int(duration_entry.get())
    except:
        messagebox.showerror(message="Enter the duration")
        return

    messagebox.showinfo(message="Speak into the microphone and wait after finishing the recording")
```

```
    with sr.Microphone() as mic:
        recorder.adjust_for_ambient_noise(mic)
        audio_input = recorder.listen(mic, duration=duration)
        try:
            text_output = recorder.recognize_google(audio_input)
            messagebox.showinfo(message="You said:\n " + text_output)
        except:
```

Speech to Text Conversion

1) Recognizer Initialization: A Recognizer instance is created to process audio input.

2) Input Duration: The function retrieves the recording duration from an entry field (duration_entry). If invalid, an error message is shown.

3) Recording Audio: The user is prompted to speak, and the audio is recorded using the microphone.

4) Speech Recognition: The recorded audio is processed to convert it to text using Google's speech recognition. The recognized text is displayed in a message box.

```
window = Tk()
window.geometry("500x300")
window.title("Convert Speech to text and text to Speech: PythonGeeks")

title_label = Label(window, text="Convert Speech to text and text to Speech: PythonGeeks").pack()

# Text-to-Speech input
text_label = Label(window, text="Text:").place(x=10, y=20)
text_entry = Text(window, width=30, height=5)
text_entry.place(x=80, y=20)

# Accent input
accent_label = Label(window, text="Accent:").place(x=10, y=110)
accent_entry = Entry(window, width=26)
accent_entry.place(x=80, y=110)

# Duration input
duration_label = Label(window, text="Duration:").place(x=10, y=140)
duration_entry = Entry(window, width=26)
duration_entry.place(x=80, y=140)

# Perform the functions
button1 = Button(window, text='List languages', bg='Turquoise', fg='Red', command=list_languages).place(x=10, y=190)
button2 = Button(window, text='Convert Text to Speech', bg='Turquoise', fg='Red', command=text_to_speech).place(x=130, y=190)
button3 = Button(window, text='Convert Speech to Text', bg='Turquoise', fg='Red', command=speech_to_text).place(x=305, y=190)

# Close the app
window.mainloop()
```

Creating the GUI

1) Window Initialization: A Tkinter window is created with specified dimensions and a title.

2) Labels and Entry Widgets: Labels and input fields are created for text input, accent selection, and duration for speech recording.

3) Buttons: Buttons are created to trigger the functions for listing languages, converting text to speech, and converting speech to text.

4) Main Loop: The mainloop() method keeps the window open and responsive.

- This code effectively combines speech recognition and text-to-speech functionalities within a user-friendly interface, allowing users to easily convert text to audio and vice versa.