Great question! 👂

In your program, **this line is the key** to detecting when you stop speaking:

audio = r.listen(source)

This uses the speech\_recognition library’s Recognizer.listen() method, which **automatically detects silence** based on a built-in **energy threshold and pause threshold**.

**🔍 How It Knows You Stopped Talking:**

**✅ 1. pause\_threshold**

r.pause\_threshold = 1

This means:

* After you stop speaking for **1 second**, it **assumes you're done**.
* The recognizer then stops listening and processes what you said.

🧠 **Example:**  
You say: “Hello Violet” → then wait 1 second silently → recognition happens.

**✅ 2. energy\_threshold (optional)**

This is a **volume threshold** to detect voice vs. background noise.

* It adjusts automatically, but you can set it manually like this:

r.energy\_threshold = 300

Higher values ignore softer sounds (good for noisy places).

**🧪 Want to Fine-Tune It?**

If it’s **waiting too long** after you stop:

r.pause\_threshold = 0.5 # responds faster

If it’s **cutting off too early**:

r.pause\_threshold = 1.5 # waits longer silence

**💡 Bonus Tip:**

You can add:

r.adjust\_for\_ambient\_noise(source, duration=1)

before r.listen() to calibrate for background noise. That helps reduce false silence triggers.

Would you like to see a version of your code with these improvements added?

Excellent question! Let’s **break it down step-by-step** so you clearly understand how this function works.

**🔍 Function Signature:**

def listen\_for\_wake\_word(wake\_word="violet"):

This defines a function named listen\_for\_wake\_word, with a **default argument**:

* wake\_word="violet"  
  👉 This means **if you don’t pass anything**, it will look for the word **"violet"** by default.

✅ You can call it like:

listen\_for\_wake\_word() # uses "violet"

listen\_for\_wake\_word("nova") # changes it to "nova"

**🧠 What the Function Does:**

with sr.Microphone() as source:

print("Listening for wake word...")

audio = recognizer.listen(source)

* It turns on the **microphone** and listens until you speak and pause.
* Your speech is stored in the audio variable.

**🎤 Recognizing What You Said:**

text = recognizer.recognize\_google(audio).lower()

* Converts your audio into **text** using Google's API.
* .lower() makes it lowercase so matching is easy.

**🧪 Wake Word Detection:**

return wake\_word in text

* This is the key condition:  
  👉 It checks if the wake word (like "violet") is **inside the spoken text**.

**✅ Example:**

If you say:

“Hello violet, what’s the weather?”

Then:

* text = "hello violet, what's the weather?"
* "violet" in text → ✅ True  
  → The function returns **True**, which triggers your assistant to respond.

**❌ Exception Handling:**

except:

return False

If there’s an error (like no internet, or unrecognized audio), it safely returns False instead of crashing.

**✅ Summary:**

The function:

1. Listens to your voice 🎙️
2. Converts it to text 📄
3. Checks if that text contains the wake word 🔍
4. Returns **True** or **False** accordingly ✅❌

Let me know if you’d like to:

* Make it detect multiple wake words (e.g., "violet" or "nova")
* Add a sound or animation when it hears the wake word
* Use offline recognition instead of Google API

Why does the command executes infinitly?

Great observation. Yes, you're absolutely right to question this. Let me explain clearly:

**❓ Problem**

You're seeing that when command() runs and you say **nothing**, it **seems to get stuck** or keep showing:

Listening...

**🎯 Root Cause**

Here’s your command() function:

def command():

with sr.Microphone() as source:

print("Listening...")

r.pause\_threshold = 1

audio = r.listen(source)

The issue is this line:

audio = r.listen(source)

By **default**, r.listen(source) **waits indefinitely** for **you to say something**.  
This is **why your assistant looks stuck at "Listening..."** — because it's patiently waiting for you to speak.

**✅ Solution: Add a timeout and phrase\_time\_limit**

You can limit:

* timeout: how long to wait before it gives up **if you say nothing at all**.
* phrase\_time\_limit: how long to keep recording **once you start talking**.

**✅ Fix:**

audio = r.listen(source, timeout=5, phrase\_time\_limit=7)

This means:

* If the user says **nothing in 5 seconds**, it'll raise WaitTimeoutError.
* Once the user starts speaking, it'll record up to **7 seconds** max.

**✅ Updated command() Function:**

def command():

with sr.Microphone() as source:

print("Listening...")

r.pause\_threshold = 1

try:

audio = r.listen(source, timeout=5, phrase\_time\_limit=7)

except sr.WaitTimeoutError:

print("No speech detected. Timing out.")

return "None"

try:

print("Recognizing...")

query = r.recognize\_google(audio).lower()

print(f"\nYou said: {query}\n")

speak(query)

return query

except sr.UnknownValueError:

print("Sorry, I didn't catch that. Please repeat.")

return "None"

except sr.RequestError:

print("Sorry, I cannot reach Google's servers right now.")

return "None"

**🔄 Result:**

Now, if you stay silent for 5 seconds, it won’t wait forever — it will move on and return "None".

Let me know if you want to **add a retry**, or **let Violet say something** when you don’t speak!

# FINAL CODE OF JARVIS DAY 3

import speech\_recognition as sr  
import pyttsx3  
  
r = sr.Recognizer()  
engine = pyttsx3.init()  
  
def speak(audio):  
  
 voices = engine.getProperty('voices')  
 female\_voice\_found = False  
 for voice in voices:  
 if "zira" in voice.name.lower():  
 engine.setProperty('voice', voice.id)  
  
 female\_voice\_found = True  
 break  
   
 engine.say(audio)  
 engine.runAndWait()  
  
def command():  
  
 with sr.Microphone() as source:  
 print("Listening...")  
 r.pause\_threshold = 1  
 audio = r.listen(source)  
  
 try:  
 print("Recognizing...")  
 query = r.recognize\_google(audio).lower()  
 print(f"\nYou said: {query}\n")  
 speak(query)  
 return query  
 except sr.UnknownValueError:  
 print("Sorry, I didn't catch that. Please repeat.")  
 return "None"  
 except sr.RequestError:  
 print("Sorry, I cannot reach Google's servers right now.")  
 return "None"  
  
def call\_assistant(call\_word="violet"):  
 with sr.Microphone() as source:  
 print("listening to your command...")  
 r.pause\_threshold=1  
 try:  
 audio2 = r.listen(source,timeout=5)  
 except sr.WaitTimeoutError:  
 return None  
  
 try:  
 query2=r.recognize\_google(audio2,language='en-in').lower()  
 print(f"\nYou said: {query2}\n")  
 return query2  
  
 except :  
  
 return None  
  
  
  
# Call the function to test  
while True:  
 result = call\_assistant()  
 if result is not None and "violet" in result:  
 speak("yes iam here")  
 commander=command()  
 if commander=="stop":  
 speak("ok bye")  
 break  
  
  
 elif result is not None: #quer2 can be anything  
 speak("say my name please")