Explain **how call 1002 → 1001 works**, how **Asterisk**, **AI listening**, and **AI speaking** all connect together.

🧠 BIG PICTURE (FIRST)

Think of your system like this:

📞 **Phone Call (1002 → 1001)**
⬇
☎ **Asterisk PBX** (controls the call)
⬇
🤖 **AGI Python Script** (this file)
⬇
🎤 **Listen to caller voice**
⬇
✍ **Convert voice → text (Deepgram STT)**
⬇
🧠 **AI thinks (OpenAI GPT)**
⬇
🔊 **Convert text → voice (Deepgram TTS)**
⬇
📞 **Play voice back to caller**

PART 1️⃣ – WHAT IS ASTERISK & AGI?

🔹 What is Asterisk?

Asterisk is a **phone system software**.
It:

- Connects extensions like **1001**, **1002**
- Handles calls
- Runs **AGI scripts** (like your Python file)

🔹 What is AGI (Asterisk Gateway Interface)?

AGI allows **Asterisk to talk to a program** (Python, PHP, etc.)

Your Python script:

- Answers calls
- Records audio
- Plays audio
- Talks to AI services

👉 **This file is an AGI program**

PART2️⃣– FILE HEADER (BASIC SETUP)

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
```

✔ Tells Linux:

- Use **Python 3**
- Use **UTF-8 text**

```
import sys, os, time, json, requests, subprocess, traceback
```

These are **Python tools**:

Module	Why used
sys	Talk to Asterisk
os	File handling
time	Delays
json	API responses
requests	Call APIs
subprocess	Run `sox`
traceback	Show errors

PART3️⃣ – API CONFIGURATION

```
DEEPGRAM_API_KEY = "my key"
OPENAI_API_KEY = "my key"
OPENAI_MODEL = "gpt-4o-mini"
```

🔹 What is Deepgram?

Deepgram does:

- 🎤 **Speech → Text** (STT)
- 🔊 **Text → Speech** (TTS)

🔹 What is OpenAI?

OpenAI provides:

- 🧠 AI brain (GPT model)

PART4️⃣ – AGI HELPER FUNCTION

```
def agi(cmd):
    sys.stdout.write(cmd + "\n")
    sys.stdout.flush()
    return sys.stdin.readline()
```

🔹 What does this do?

This is how **Python talks to Asterisk**.

Example:

```
agi("ANSWER")
```

➡ tells Asterisk:

> “Answer the call now”

PART5️⃣ – READ AGI ENVIRONMENT

```
while True:
    line = sys.stdin.readline().strip()
    if not line:
        break
```

✔ Asterisk sends **call info** (caller ID, extension, etc.)
✔ You read it first (mandatory for AGI)

PART6️⃣– CALL STARTS

```
agi('VERBOSE "VOICE AI AGENT STARTED" 1')
agi("ANSWER")
```

📞 **Call is answered**
🖥️ Message appears in Asterisk CLI

PART7️⃣– MEMORY (CONVERSATION)

```
conversation = [
 {"role":"system","content":"You are a helpful voice assistant..."}
]
```

🧠 This stores:

- What user says
- What AI replies

So AI remembers conversation.

PART8️⃣– MAIN CALL LOOP (VERY IMPORTANT)

```
while True:
```

This loop:

- Listens
- Thinks
- Talks
- Repeats

Until user presses **#**

🎤 STEP 1 – RECORD USER VOICE

agi('RECORD FILE /tmp/user_1 wav "#" 20000 0 BEEP s=2')

Meaning:

Part	Meaning
RECORD FILE	Record audio
wav	Save as WAV
"#"	Stop if # pressed
20000	Max 20 seconds
BEEP	Play beep first

📂 File saved as:

/tmp/user_1.wav

📝 STEP 2 – SPEECH → TEXT (STT)

requests.post("https://api.deepgram.com/v1/listen")

What happens:

1. Audio sent to Deepgram
2. Deepgram returns text

Example:

"Hello, how are you?"

🧠 STEP 3 – SEND TEXT TO AI

```
requests.post("https://api.openai.com/v1/chat/completions")
```

AI:

- Reads conversation
- Thinks
- Sends reply

Example reply:

"I am fine. How can I help you?"

🔊 STEP 4 – TEXT → SPEECH (TTS)

```
requests.post("https://api.deepgram.com/v1/speak")
```

✔ AI text → voice
✔ Saved as WAV (16kHz)

🔧 STEP 5 – CONVERT AUDIO FOR ASTERISK

```
sox bot.wav -r 8000 bot_8k.wav
```

❓ Why?

Asterisk needs **8kHz audio**

▶ STEP 6 – PLAY AUDIO TO CALLER

```
agi('STREAM FILE /tmp/bot_1_8k ""')
```

📞 Caller hears AI speaking

🔚 STEP 7 – END CALL

```
WAIT FOR DIGIT 2000
```

If user presses **#**
➡ Call ends

PART 9️⃣ – HOW CALL 1002 → 1001 WORKS

Example dialplan logic (simplified):

1002 dials 1001
⬇
Asterisk answers
⬇
Dialplan runs AGI script
⬇
Your Python file runs
⬇
AI answers instead of human

So:

- **1001 is NOT a person**
- **1001 is an AI agent**

PART 🔟 – FILE INSTALL COMMANDS

```
chmod 755 voice_agent_stt.py
ln -s /home/... /var/lib/asterisk/agi-bin/
```

These:

- Give permission
- Link script to Asterisk AGI folder

🧩 SIMPLE REAL-LIFE EXAMPLE

📞 You call **1001**

AI says:

> "Hello, how can I help you?"

You say:

> "What is your name?"

AI thinks 🤔
AI replies 🔊
Conversation continues

You built:

- ☎ Phone system
- 🎤 Voice recognition
- 🧠 AI brain
- 🔊 AI speech

****

`VERBOSE` is **very important in Asterisk**, especially for beginners.

📢 WHAT IS `VERBOSE`?

`VERBOSE` is an **Asterisk AGI command** used to:

👉 **Print messages on the Asterisk CLI (terminal)**
👉 **Help you see what your program is doing**

It is mainly used for:

- Debugging
- Understanding call flow
- Seeing errors or values

🧠 SIMPLE DEFINITION

VERBOSE = “Print this message on Asterisk console”

It does **NOT**:
❌ Talk to the caller
❌ Play audio
❌ Affect the call

It is **only for YOU (developer)**.

📌 WHERE DOES IT SHOW?

When you run:

asterisk -rvvv

You will see messages like:

VOICE AI AGENT STARTED
TURN 1
USER SAID: Hello
BOT REPLY: How can I help you

All these come from `VERBOSE`.

🧾 BASIC SYNTAX

VERBOSE "message" level

Example:

agi('VERBOSE "HELLO WORLD" 1')

🔍 BREAKDOWN

1️⃣ `VERBOSE`

➡️ Asterisk command

2️⃣ `"HELLO WORLD"`

➡ Message you want to print

3️⃣ 1 (verbosity level)

➡ How important the message is

📊 VERBOSITY LEVELS (VERY SIMPLE)

Level	Meaning
1	Normal info
2	More details
3	Debug
4+	Very detailed

👉 Most people use **1**

🧪 YOUR CODE EXAMPLES

Example 1

```
agi('VERBOSE "VOICE AI AGENT STARTED" 1')
```

📺 Console shows:

```
VOICE AI AGENT STARTED
```

Example 2

```
agi(f'VERBOSE "TURN {turn}" 1')
```

📺 Console shows:

```
TURN 1
TURN 2
TURN 3
```

Example 3

```
agi(f'VERBOSE "USER SAID: {user_text}" 1')
```

📺 Console shows:

```
USER SAID: Hello
```

Example 4

```
agi(f'VERBOSE "BOT REPLY: {bot_text}" 1')
```

📺 Console shows:

```
BOT REPLY: How can I help you?
```

🚫 COMMON BEGINNER MISTAKE

❌ Thinking `VERBOSE` speaks to the caller

❌ Thinking `VERBOSE` is mandatory

✔ It is **only for logging**

🎧 DIFFERENCE: VERBOSE vs STREAM FILE

Command	Who hears it?
VERBOSE	Developer (CLI)
STREAM FILE	Caller

🧠 REAL-LIFE ANALOGY

Think of:

- 🎤 **Caller** = customer
- 📞 **Asterisk** = call center

- 🧾 **VERBOSE** = CCTV monitor for admin

The customer never sees it.

🧰 WHEN SHOULD YOU USE VERBOSE?

✔ To print:

- Variable values
- API responses
- Errors
- Flow checkpoints

Example:

```
agi(f'VERBOSE "API STATUS: {stt_resp.status_code}" 1')
```

✅ FINAL ONE-LINE ANSWER

> **VERBOSE prints debug messages to the Asterisk console, allowing you to see what your AGI script is doing.**

📌 Asterisk Debugging – Short & Document-Ready

1️⃣ Enable More Verbosity

```
asterisk -rvvvvv
```

- `-r` → connect to running Asterisk
- `v` → verbosity level
- More `v` = more debug output

2️⃣ NoOp() vs VERBOSE

Feature	NoOp()	VERBOSE
Used in	Dialplan (`extensions.conf`)	AGI scripts (Python)
Purpose	Debug dialplan flow	Debug AGI logic
Plays audio	❌ No	❌ No
Visible on CLI	✅ Yes	✅ Yes

Examples

NoOp(Call started)

agi('VERBOSE "Call started" 1')

3️⃣ Log to File

Asterisk logs (recommended)

Edit:

/etc/asterisk/logger.conf

[logfiles]

full => notice,warning,error,verbose

Reload:

asterisk -rx "logger reload"

Logs saved at:

/var/log/asterisk/full

Python file logging

```
import logging

logging.basicConfig(

    filename="/var/log/asterisk/voice_agent.log",

    level=logging.INFO

)

logging.info("User said something")
```

✅ Summary

- `-rvvvvv` → maximum console debug
- `NoOp()` → dialplan logging
- `VERBOSE` → AGI logging
- `logger.conf` → persistent logs

****

```
# ---------------- AGI HELPER ----------------
def agi(cmd):
    sys.stdout.write(cmd + "\n")
    sys.stdout.flush()
    return sys.stdin.readline()
```

👉 **YES** — this function is the **MAIN CORE CONNECTION** between **your Python script** and **Asterisk**.

Without this function:
❌ Asterisk cannot talk to Python
❌ Python cannot control the call

🧠 WHY THIS FUNCTION IS SO IMPORTANT

This line is the bridge:

sys.stdout.write(cmd + "\n")
It sends commands **from Python → Asterisk**

This line listens back:

sys.stdin.readline()
It reads responses **from Asterisk → Python**

So this function creates a **two-way communication channel**.

🔌 HOW ASTERISK TALKS TO AGI (VERY SIMPLE)

When Asterisk runs an AGI script:

- **Asterisk = Parent process**
- **Python script = Child process**
- Communication happens via:
  - `STDIN` (input)
  - `STDOUT` (output)

This is built into the Asterisk AGI protocol.

🧩 BREAK DOWN THE FUNCTION LINE BY LINE

def agi(cmd):

➡ Define a function named `agi`
➡ `cmd` = Asterisk command as text

sys.stdout.write(cmd + "\n")

➡ Sends command **to Asterisk**

Example:

agi("ANSWER")

Actually sends:

ANSWER

---

sys.stdout.flush()

➡ Forces Python to send immediately
➡ Without this, Asterisk may **wait forever**

⚠️ VERY IMPORTANT LINE

---

return sys.stdin.readline()

➡ Waits for **Asterisk response**

Example response:

200 result=0

🔄 FULL COMMUNICATION FLOW

Python → STDOUT → Asterisk
Python ← STDIN ← Asterisk

🧪 REAL EXAMPLE

Python code:

r = agi("ANSWER")

What happens:

Python sends:
ANSWER
Asterisk replies:
200 result=0

`r` contains that reply

📞 COMMON COMMANDS USING THIS FUNCTION

Code	Meaning
`agi("ANSWER")`	Answer call
`agi("VERBOSE \"Hello\" 1")`	Log message

`agi("RECORD FILE ...")`	Record voice
`agi("STREAM FILE ...")`	Play audio
`agi("WAIT FOR DIGIT 2000")`	Wait for the key

All of these **go through this function**.

❓ IS THIS THE ONLY WAY TO CONNECT?

Other AGI styles exist:

Method	Language
AGI (STDIN/STDOUT)	Python, PHP
FastAGI	TCP socket
EAGI	Audio streaming
ARI	REST API

But in **your code**:
👉 **This function is THE CORE**

🧠 REAL-LIFE ANALOGY

Think of:

- 📞 Asterisk = Manager
- 🧠 Python script = Assistant
- 📢 `sys.stdout` = Talking
- 👂 `sys.stdin` = Listening
- 🔁 `agi()` = Telephone between them

❗ VERY IMPORTANT NOTE

This function is **NOT part of Python standard library** for Asterisk.

👉 You **created it yourself**
👉 It follows **Asterisk AGI protocol**

✅ FINAL ANSWER (ONE LINE)

> **Yes — this `agi()` function is the main core that connects your Python script with Asterisk using AGI protocol.**

Response	Meaning
`200 result=0`	Command succeeded
`200 result=1`	Digit pressed / true
`200 result=-1`	Call hung up / error