# Python and AI Power-Up Program Offline Class-20251009_184819-Meeting Recording

October 9, 2025, 1:18PM

1h 12m 20s

**Hardip Patel** started transcription

**Hardip Patel** 0:07
Hello.
Hello.

0:09
Uh, hello.

**Tarun Jain** 0:10
Uh, hello.

**Hardip Patel** 0:15
Take a second.

**Ajay Patel** 0:33
Hello.

**Hardip Patel** 0:33
Hello. Huh

**Ajay Patel** 0:34
Yeah, everyone is joining. Just give them 5 to 10 minutes.

**Tarun Jain** 0:38
Yeah, OK. OK.

**Hardip Patel** 0:40
Yeah.

Today we are not continuing, right? We are doing something. Today's not the fine tuning.

**Tarun Jain** 1:12
Uh, no, no. So the agent T crack in memory is pending, right? So today we'll just complete that.

**Hardip Patel** 1:18
OK, OK, it won't be able to come.
Here's a little bit of workload added.

**Ajay Patel** 1:26
Yes, he's occupied in another meeting, so he won't be joined.

**Hardip Patel** 1:30
So.

**Tarun Jain** 1:30
OK, so the memory part is there, right? That is similar to what I conducted on the first. So I guess uh already executed memory.

**Hardip Patel** 1:39
Oh, OK.

**Tarun Jain** 1:40
So agentic rag is something that I'll share the code. It's pretty straightforward that also I guess has already executed, but there are just two changes that I wanted to say. That's it.

**Ajay Patel** 1:41
OK.

**Hardip Patel** 1:45
Mhm.

**Ajay Patel** 1:45
Mhm.

**Hardip Patel** 1:54
OK.

**Tarun Jain** 1:57
So if you are using fast API rate fast embed for agentic rack, it is not supported directly. So if you see here we define a knowledge inside knowledge we are directly defining fast embed. So if we define Gina AI and 768 that is not working because this is hardcoded with three.

**Hardip Patel** 1:58
Oh.
Mhm.
Right.
Yeah.

**Tarun Jain** 2:16
84.
So that bug fix I've done just now.
So if you see this is 34 minutes ago.

**Hardip Patel** 2:27
OK.

**Tarun Jain** 2:28
So apart from this, most of the codes is same what we discussed. So yeah.
Well, let me know when we need to start.

**Hardip Patel** 2:42
Um.

**Ajay Patel** 2:43
Ishan is there.

**Hardip Patel** 2:46
No, Ishan is not yet joined.

**Ajay Patel** 2:47
Yeah.
He will be joining. Just a second. Let me ask him.

**Hardip Patel** 2:54
Also there is ram and let me ask her.
Yeah.

**Ajay Patel** 3:21
Ishan is in conference room, so yeah, we can continue.

**Hardip Patel** 3:24
Hmm.

**Tarun Jain** 3:26
Yeah, OK. Uh.

**Ajay Patel** 3:27
And we are recording this session. OK, who is recording? OK, Hardeep.

**Tarun Jain** 3:31
Uh, is my screen?

**Hardip Patel** 3:32
Yeah.

**Tarun Jain** 3:35
OK, I'm sharing the screen only.

**Ajay Patel** 3:37
No, you have stopped sharing, OK.

**Tarun Jain** 3:40
Uh, no.

**Ajay Patel** 3:41
Yeah, now it is visible.

**Hardip Patel** 3:43
Yeah.

**Tarun Jain** 3:43
Yeah, OK. So we'll start off with the memory part today. If you look at this particular diagram which I showed earlier, so majorly when it comes to agents, we start off with tools, right? So if you ask any prompt based on the prompt, which?
Tool is the most relevant to take the action. So here we have calendar, calculator, code interpreter and search. So we looked at some of the examples like tably and also we looked at CSV and all. And then for planning part you have LM which is your Gemini model and every single LLM model. There are two parts of reasoning.

**Hardip Patel** 4:09
Yeah.

**Tarun Jain** 4:19
First task creation, second task execution, right? So what do we do in task creation? In task creation you are just doing the planning part for which you are using COT which is chain of thoughts prompting. Then for task execution you need some kind of feedback.
So this is where we have prompt technique like React.
And React is nothing but reasoning and action. So let me drag this aside. So you have task creation, you have task execution. This is prompt technique. This is prompt technique and you're combining this with an LLM. This is where you have your. Planning part. So the one part which was currently pending was memory and in

memory we have two parts, one is short term memory and one more is long term memory. So let's take an example when you are using if you ask any question. Whatever is saved earlier, right? And if you want to ask what I asked earlier, that is where you use short term memory which is only within that session. Then you have long term memory which is for a longer period of time. So let's try to understand this with some examples.

So this is how our existing solution looks like, which is stateless, right? So what happens in stateless? You ask a query, you have an agent and then agent will generate the response. So over the period of time you are not updating anything, right? So it's just stateless. There is no form of.

Of any context that you are saving. So if you want to build a stateful kind of agentic application that will learn over the period of time, this is where you use memory right? So if you see here you have user then query agent and then from the agent if you see you have write then it learns from the interaction and memory.

So there are also scenarios where you have to retrieve the information. So here it is adding and the second feature is search. So based on my current preference, what is the solution that I need to extract? So let me simplify this so that it becomes very easy to understand.

So let's understand this architecture in very simple terms. So for long term memory we use something called as MEM zero. So MEM zero is a memory layer for the agent application. So I'll explain this with an example.

So let's suppose you are a user and you have two preferences which is related to food and travel. Now what is happening is in your initial preference.

In your initial preference, what you're trying to do is you have preference of you need window seat when you travel.

Window seat or the corner seat. This is your preference and 2nd preference is your complimentary meal.

Complimentary meal.

Is vegetarian Jain food. OK, so let's try to understand this example. Your initial preference is nothing but your messages. So if you see here you have user, you have multiple messages. So user 1, user 2 is nothing but your user ID. So what I'm trying to do is I'm just considering the messages.

**Hardip Patel** 7:33

Mhm.

**Tarun Jain** 7:43

For the user ID 1, user ID one and the initial preference are these two. Now what is happening is after certain amount of time. So let's suppose after two months.
You are again traveling. Now this time what you're trying to do is you're doing inference. During inference you will ask a question. Hey, I am traveling.
To New York.
What are the places?
Should I visit? So now if you look at it, so instead of what are the places, I'll just add what are the food places should I visit? So considering you have long term memory here, whenever you ask this question natively, let's suppose if you ask this question directly to ChatGPT, it will give you some generic.
Responses.
But now if you use your agent.
With the memory, memory in the sense the preference that you have. Now what will happen is whenever you do inference, when it says you need to travel to New York and when you do anything related to food places, here the recommendations will be the places where they serve Jain food.
You understood. Yeah, like whenever you save your initial preferences you will tell what you like. Like it can be window seat, corner seat, any meal that you are doing. So this is nothing but any interactions. Then anytime movement, right? No matter what the session you're using, if you're reusing that agent again.

**Hardip Patel** 9:12

Mm.

**Tarun Jain** 9:23

What are you trying to do? Whenever there is any inference, you are getting a context. You are getting a context. That context is nothing but your memory. Now instead of giving you very generic responses, you will have a personalized response.

**Hardip Patel** 9:27

Hmm.

**Tarun Jain** 9:38

So this is where you call it as persist. So whenever it comes to long term memory, you are persisting your data over a longer period of time to build it stateful agent application. So let's take one more example. Now let's suppose.
When I add complimentary meal Jain food in flight, I didn't like it, so my preference got changed to only vegetarian Hindu food.
So now what will happen next time? Whenever I ask any question, hey, I need to travel to New York, what are the food places should I visit? Instead of preferring Jain, it will now pick Hindu.
So this kind of task updation is called update and here you're using add.
So initially whenever you add your interaction it will be add. Now if your same interaction is changing over a period of time then you will use update. So this is happening automatically. So this is what it meant here. So if you see here whenever you ask a user query you are extracting your information.
And summary generator is nothing but whatever your preferences is there, it will create a summary of it which will be given to the LLM and then you have update phase. Update phase whenever you are using search right based on the given current preference, it will automatically pick add, update or delete.
So I'll also show the code for it. So when we see code it will be much simple. But in simple understanding there are two state. The first state is your extraction phase which uses LLM to generate your preferences and then second you have your update phase.
So based on the given current preferences, you're updating your memory. So how is it updating? You're using graph memories. So if you remember earlier I spoke something about EST. So what kind of EST search it uses? It uses node which is nothing but your graph.
Here also they are using graph memories.
So we'll look at the code, and once you look at the code, it will be much more simple.
And yeah, this is it with the memory part. What we'll do is we'll get to the code and all these slides, whatever you have so far, you have everything here, whether it is agents, RAG, evals, MCP and also memory.

**Hardip Patel** 11:54
Yes.


**Tarun Jain** 12:05

So once you look at the code right, I've taken some of the simple example itself so that when you see the output it will be very clear like what is update, what is add and when do when do we do inference and how memory matters when you're doing inference.

**Hardip Patel** 12:08
Hello.
OK.

**Tarun Jain** 12:35
So now coming to mem 0 part mem 0 either you can run in two ways.
Mem 0 is completely open source, so there are three components we need in order to implement memory. So memory without.
Mem 0 API key.
Then memory with Mem 0 API. So there are majorly 3 components. The first component is LLM.
The second one is vector store. So I said right when you are saving your data. So let's suppose you say you need a window seat, you need a complementary meal. Where is it getting stored? It is getting stored in a vector store, right? So instead of store, we call it as indexing.
So you have LLM, you have vector store. So whenever you have vector store, what details do we need? What is the dependency of vector store?
So LLM vector store and in order to save in vector store you need data and.

**Hardip Patel** 13:43
Mm.

**Ajay Patel** 13:47
Metrics. No, not metrics.

**Hardip Patel** 13:50
The the the.

**Tarun Jain** 13:51

You're close. You need metrics, but what is that called in neural network terminology?

**Ajay Patel** 13:52

I'm getting.

I'm waiting. No, not in a waiting. I'm waiting. OK, OK.

**Tarun Jain** 13:58

Correct. So it's embeddings.

**Hardip Patel** 14:00

Yeah.

**Tarun Jain** 14:01

So instead of metrics, we call it vectors and vectors are stored. Vectors are the formation of your embeddings, right? So if you don't want to use Mem 0 API key, you have to specify your LLM vector store and embedding right? So by default.

And by default your LLM is open AI.

And vector store by default it is graph.

And embeddings by default again is open EI.

But if you if you want to override this, what we are doing now is we have to create configuration. In this configuration you will provide your LLM and tell who is the provider of it. It can be anything. If you want to use Olama, you can use Olama. If you want to use Gemini, you can use Gemini.

Anything is fine and then you have vector store. Now why are we overriding the vector store? Most of the time when you are embed when you are using embeddings you have to change the dimension of it. So Open AI has 1536 as there.

Embedding dimension and not all the embedding supports 1536, so you have to keep it as 1024 or 768 JINA use 768 and there are some models which uses 1024 if you want to define these parameters.

You have to define that inside vector store and here again you can use any vector store provider. It can be quadrant, it can be middleverse, it can be VV8 and then you have embedding and embedding. Also you can have any provider.

In our case it will be fast embed. Is it clear so?

**Hardip Patel** 15:47
Yes.

**Tarun Jain** 15:47
Mem 0 is open source, but there are some defaults which we need to override. Like LLM is open AI, vector store is graph, then embeddings is open AI. We are overriding it by defining the configuration and suppose if you want to use their own what you call API key and use their dashboard then you can use.
API key and then instead of defining configuration you can directly define memory and in the code that I've showed, I've shown you both the example how to use mem 0 with without API key and with API key.
OK, let's start with the installation.
Opic we can probably ignore. Opic it is just that whenever.
If you look at the memory result, right, you have memory, you have ash, you have metadata, you have scores. What I'm trying to do is I'm saving all these details inside a OPIC metadata so that we can trace. So if you remember in OPIC what we usually have is we have trace.
We have feedback score.

**Hardip Patel** 16:54
Mm.

**Tarun Jain** 16:55
And we have metadata. Trace is nothing but whatever input output you ask, it will be traced here. And during evals we used feedback scores and metadata during function calling. We used it like what was the input that went to function calling and what was the output. Same thing here. Also what is the data that went into?

**Hardip Patel** 16:56
OK.

**Tarun Jain** 17:14
Memory and what did it it retrieve so that information will go in metadata.
I hope that this made sense why we need long term memory and how to define

MEM 0 and without MEM 0 API key.
Uh, any doubts anyone in this part?

**Ajay Patel** 17:36
No.

**Hardip Patel** 17:36
No, not currently.

**Ayush Makwana** 17:38
Oh.

**Tarun Jain** 17:38
OK, so when you see examples, it will be much clearer and we can take questions once again if in case you feel any issues. So are you on the same collab?

**Ajay Patel** 17:49
Yeah, just a mute.

**Ayush Makwana** 17:49
Yes, yes.

**Tarun Jain** 17:50
OK.
So we are using AGNO also. After the end of the code I'll also show you how we can use memory within the AGNO framework as well. Memzero EI is the library that we are using which is again open source. Then we have Opic for embedding model. We'll be using Langching because.
We don't have any other providers if you want to use fast embed, right? So if you want to use fast embed you have to use it from langchain and this also support I added yesterday but it's not yet merged.
I'm not sure. I have added this yesterday but it's still yet to be merged. So if you see fast embed embeddings for local embeddings, this is for MEM 0. But as of now natively it doesn't support fast embed. So what what are we trying to do is we're using it from line chain.

And if in case you want to know these details, I'll come back here. They have the documentation.

**Ayush Makwana** 18:46

Mm.

**Tarun Jain** 18:54

Where is the documentation?

Docs dot mem zero OK dot docs dot mem zero dot AI. If you Scroll down for embedding models you have supported embedding models. You have Open AI, Azure Open AI, Olama, Hugging Face. Then here you have langching.

So we'll just use langchains, open AI and then override it inside the provider. So if you see in configuration you have to define embedded provider is a langchain, so the syntax will be same.

So we'll be using line chain for embeddings. We have to install these two things.

OK.

Uh, we can create the copy of this. To those whoever are running it, you can create the copy.

3.

Mem 0 will take some time to install the libraries. Hardly it should be like 40 to 50 seconds.

OK, it shows 14 seconds.

**Ajay Patel** 20:55

Took me 53 seconds.

**Tarun Jain** 20:57

This one? Yeah, even I thought it should take 40, but it showed 14.

So this line should usually take around 40 to 50 seconds.

All right and now what we can do is we can click on runtime then restart session.

**Ajay Patel** 21:20

Mhm.

**Tarun Jain** 21:22

Now coming back to the drawing board, here if you see LLM, if I'm using Gemini, I need the LLM key. Then vector store if I'm defining quadrant, I'll save it locally which is in memory. So I don't need any API key and embedding. I'm using Langchain. From Langchain I'm using fast embed. Fast embed doesn't need any API.
So what we will do is we will directly import only Google API key.
Till here it's the same syntax what we have worked so far. Now here if you see from mem zero we are importing memory. So when you import memory here you are supposed to you are.

**Hardip Patel** 21:56
Mm.

TJ **Tarun Jain** 22:07
Supposed to define your configuration.
If you use API key from M0 instead of memory, you will do it memory client which will show later and then from linechain dot embeddings we are importing fast embed embeddings same syntax.
And the model that we are using again is same Gina and the Max length is 768 which is our embedding size.
Uh, I don't need any token, so I'll just hit cancel.

**Ajay Patel** 22:43
It is a hugging face token.

TJ **Tarun Jain** 22:44
But fast embed doesn't need any this thing. So there are some models which requires licensing. If that model, I mean let's suppose if you remember when we had a Mistral model, we were supposed to provide permission.

**Ajay Patel** 22:59
Mhm.

TJ **Tarun Jain** 23:02
And once you provide the permission then only you can use that model. So if you

have those models then only you have to provide HF token. But this is very open model so you don't need any licensing. So we don't even have to pass HF token.

**Hardip Patel** 23:07
Um.

**Tarun Jain** 23:18
But HF is hugging face, yeah.
Uh, till here is it done past embed embeddings.

**Hardip Patel** 23:24
Yes.

**Ajay Patel** 23:25
Yeah, yes.

**Ayush Makwana** 23:25
Yes, yes.

**Tarun Jain** 23:27
OK so here we have config and for config we have only three things, LLM, vector store and embedder. So provider here is and make sure you are using very light model because this is not your main LLM. So whatever you are using here it is for this step.

**Hardip Patel** 23:34
Hmm.

**Tarun Jain** 23:45
So if you see in the slides.
You have your user preferences. Based on the user preferences, what you're trying to do is you're trying to create a summary for your preferences, so you don't need reasoning model to do this. Even if you have a lightweight model, that should be more than sufficient.
So that's the reason why for this one I'm only using the light version, but once you

get preferences, So what is memory as a preference? Memory as a preference is nothing but a context.
So if you see here all these things, whatever you're passing.
All these things is just a context. So once you have the context then you will use an LLM. That LLM can be a Gemini 2.5 Pro or Gemini 2.5 Flash. But in order to create this memory you can use a lighter lighter model which is Gemini 2.5 Flash light.

**Ajay Patel** 24:33
Hmm.

**Tarun Jain** 24:41
And then you have vector store. Vector store is nothing but quadrant and here for configuration you have to define collection name. If you are defining it in in memory it will be path. What if I want to define it in cloud then what will change here?
This we can keep it as long term.
So if I want to define cloud then what will be the logic here?
Instead of path, what are the other two parameters we have to use?

**Hardip Patel** 25:10
The URL.

**Ajay Patel** 25:13
Oh.

**Hardip Patel** 25:14
Yeah, yeah, the workspace.

**Ajay Patel** 25:15
URL.

**Tarun Jain** 25:16
Correct one will be URL which is your HTTP and it will end with 6333 and then you will have your API key.

**Hardip Patel** 25:17
Yeah, project name and workspace.

**Ajay Patel** 25:18
Oh.
Yeah, with the port number, yeah.

**Hardip Patel** 25:26
Oh, sorry, that is for.

**Tarun Jain** 25:27
OK, so if you want to define cloud, you can define this too, but we'll use in memory.

**Ajay Patel** 25:33
Mm.

**Tarun Jain** 25:35
So LLM is done. Vector store is done. Now for embedder we are defining provider which is line chain. Then config model is nothing but the embeddings which is this one.
Once you have the configuration, we just have to append our memory. So from memory dot from config define config.

**Ajay Patel** 26:02
So the application warning.

**Tarun Jain** 26:05
Oh, that is fine. It's just the date, date, time one.

**Ajay Patel** 26:07
Date time, yeah.

**Tarun Jain** 26:13
So daytime is nothing but one of the metadata that the users.

At what time did you save your memory? So for that they're using this date time.
Till here is done.

**Ajay Patel** 26:27
Yeah.

**Ayush Makwana** 26:27
Yes.

**Tarun Jain** 26:29
OK, so now if you see I'm saving some of my initial preferences like you have role user content, what is the must try food in Baroda? Then you have some information. Then I'm not into street food. I prefer Gujarati talis. So now what will happen is when you have this interaction.
And when you are adding it into client, So what phase is this?
So what kind of face is this?

**Hardip Patel** 26:54
In ingestion.

**Tarun Jain** 26:57
It's ingestion, but in terms of memory this will be called as expansion phase. But in simple words when you're using vector store, this is called indexing.

**Hardip Patel** 27:00
Open.
OK.

**Tarun Jain** 27:07
Which is correct. So you have client dot add, you're adding messages. Every single preferences that you add, you should have user ID. This is must. So user ID can be anything. Let's suppose here you are defining user one.

**Hardip Patel** 27:09
Yes.

**Tarun Jain** 27:24

So user one has this preference, then you'll have user 2, user 2 will have its preference. So user ID and messages are must keyword. Metadata is optional. So why do we add metadata whenever you see in drag or also in memory?
If you have metadata defined by your end, this can be used for filtering. So when I do inference and if I need to extract details only from category foot then I will only use this information and if you scroll below you also have message too.
Here it is very specific to traveling. So now we have travel. So whenever I do search, if I want to apply filtering, I can use this metadata. So is this clear? Metadata in simple words, it's used for filtering.

**Ajay Patel** 28:09

And.

**Tarun Jain** 28:13

So I have messages where I have some information then result then result one. So now if you see prefer is Gujarati talis over street food. Did I enter this message in my prompt?

**Ajay Patel** 28:28

No.

**Tarun Jain** 28:29

No, right. This is what it meant. Summary.

**Ajay Patel** 28:31

No.

**Tarun Jain** 28:34

Is this clear?

**Ajay Patel** 28:35

Mm-hmm.

**Ayush Makwana** 28:36
Yes.

**Tarun Jain** 28:36
So where are we currently? We have used, I mean we have messages, we have expansion phase. Now if you look at expansion phase you have summary and here it's using LLM. So it doesn't matter if you want to use have any heavy LLM or just.
I mean don't use any AVLLM, just use very lightweight. So you have Mister models. If you want to use Ollama, you can use Ollama. If anyone is using Azure, you can use O4 mini or GPT 4 mini. This is very simple. You don't have to use any heavyweight model and here we used.
Gemini 2.5 flashlight. So this was 1 memory and then the second memory is Mandap in Baroda is famous for authentic Gujarati thalis. I didn't enter this message. This was interpreted by LLM based on my given preference.
Is this clear to clear?

**Ayush Makwana** 29:31
Yeah.

**Hardip Patel** 29:31
Yes.

**Ayush Makwana** 29:32
Yes.

**Tarun Jain** 29:32
OK, so now we'll do the same thing. So if you see here here I'm adding more preferences that I need a window seat, window seat or person seat and then Hindu vegetarian meal. So these are the two preferences that it needs to save same user ID. So if you see client dot add is for expansion phase which is our indexing user ID is same but the category is different. One is food and one more is travel.
And now if I print results planning to travel to Hong Kong from Bangalore needs Hindu vegetarian meal and it is also saving my preferences. These two are very important.

**Ajay Patel** 30:14
Mm.

**Tarun Jain** 30:14
So now considering all this, what do you think is the best use case of whatever you just saw now?
Like where do you think this will be very useful?

**Ajay Patel** 30:26
No.

**Hardip Patel** 30:27
MM mm.

**Ajay Patel** 30:30
Useful use case would be.
Like for.

**Hardip Patel** 30:35
I think uh, no taking assistant could be one thing.

**Ajay Patel** 30:38
Oh.

**Tarun Jain** 30:40
OK, no taking assistant.

**Ajay Patel** 30:40
Oh.
Not taking decisions, but.

**Ayush Makwana** 30:47
List.

**Tarun Jain** 30:47
So Nalal, suppose if you have notion right and in notion if you have some unfinished task there might be your. Let's suppose previously you use certain approach and you got certain result. You can save all those interactions.

**Ajay Patel** 31:01
Mm.

**Tarun Jain** 31:03
One more can be a recommendation system.

**Ajay Patel** 31:07
Oh yeah, and that's a nice use case.

**Tarun Jain** 31:09
So if you like this particular thing you it will also refer OK, so you like order. These are the recommendation for you. So now order is your preference.

**Ajay Patel** 31:21
Mhm.

**Tarun Jain** 31:22
And then to build your trip planner, right? So let's suppose if you are traveling to New York now and your memory knows that you prefer Hindu vegetarian meal and you prefer window seater. So this is not required, but for trip planner this is very important detail.

**Ajay Patel** 31:27
Hm.
OK.

**Tarun Jain** 31:41
This one and this one. So next time whenever you use a chatbot.

**Ajay Patel** 31:43
We can use it here. Stock prediction also not for stock prediction also not this.
We can also use forest of prediction.

**Tarun Jain** 31:50
Huh. Stop prediction. Also it's there because it is time series.

**Ajay Patel** 31:54
Hmm.

**Tarun Jain** 31:56
So wherever you think timing is very critical time, that is where you use memory.

**Ajay Patel** 32:00
Mhm.

**Tarun Jain** 32:04
And there is booking.com something uh booking.com open AI trip planner.
So here they built something called as AI trip planner and if you scroll below you'll find some information. So if you see you have smart filters, this is like traditional search relied on drop down. So usually what you do is.
Instead of drop down, now they're just asking you the query. For example if you just enter sunset views it will give you some recommendation of hotels and if you have great gym then it will give you some preferences. So this is preference if you see great gym sunset views.
And based on your preferences, you're using smart filters. Smart filters is nothing but again, your memory.
Is this clear?

**Ajay Patel** 32:59
Mhm.

**Tarun Jain** 33:00
So there are two important details one needs to understand. Memory is not rag.

**Hardip Patel** 33:00
Yes.

**Ayush Makwana** 33:01
Yes.

**Ajay Patel** 33:07
OK.

**Tarun Jain** 33:10
Even though memory is indexing and using search, it is different right? The logic is same. But if you see how are you adding, it is altering your information. In drag your data will never be altered. It is knowledge transfer.
But here you are changing your preferences in a personalized tone. So this is the difference.

**Ajay Patel** 33:32
OK.

**Tarun Jain** 33:35
But in simple words, the working is same. The working is you index and you use search, but the way you index is different. You're not using HNSW here.
So HNSW is a nearest algorithm that is used for rag. Here you are using graph where you are creating a relationship node with LLM, right? So in indexing did we ever used an LLM?

**Hardip Patel** 33:55
OK.

**Tarun Jain** 33:59
For rag, no, but for memory it's using.

**Hardip Patel** 34:00
No.

**Ajay Patel** 34:05
OK.

**Tarun Jain** 34:07
All right, so till here are we done? I hope you got the responses.

**Ajay Patel** 34:12
Yeah.

**Hardip Patel** 34:13
Yeah.

**Ayush Makwana** 34:14
Yeah.

**Tarun Jain** 34:14
OK, so now what we'll do is we'll ask a new question. Based on this new question, you will use client dot search, give the user query and user ID and by default limit is under and I just need limit to be 30.
So I have added two queries, but let's just keep it one.
So if you see on traveling to Baroda, suggest me dish to try and recommend place to visit. So here it is doing search.
And what is this called? This is called context.
Are we clear to clear?

**Hardip Patel** 34:58
Hmm.

**Tarun Jain** 34:59
So, so far we are not using any LLM. You are saving your data, you are searching it. Now this information, whatever you have here, memory, memory, memory, memory. You will combine this into a context and give it to the LLM so that LLM can generate the response.

**Ayush Makwana** 34:59
Hmm.

**Hardip Patel** 34:59
Yes.

**Tarun Jain** 35:17
So far it is just indexing and search. So now if you see context I'm using what is this called list comprehension from for M in memories of results. So if I just print memories of results.
You have a list O if I do length of list.

**Ayush Makwana** 35:35
Mm.

**Tarun Jain** 35:39
It will be 5 or six. You have 5. Now what are you trying to do? You want to look through entire follow and you only want to extract memory and then once you have memory this is still in list. I'm using join and then I'm creating it in a dictionary.
Uh, string. So now if I print this context, this is a pure string.

**Hardip Patel** 35:55
Mm.

**Tarun Jain** 36:01
Print context.
Is this clear? This will go to the LLM now.

**Ayush Makwana** 36:10
Hmm.

**Ajay Patel** 36:11
OK.

**Tarun Jain** 36:12
Let me know till here if you have completed.

**Ayush Makwana** 36:17
Yes.

**Ajay Patel** 36:18
Yeah.

**Tarun Jain** 36:20
OK.
So once you have the context, what is the next part? We have to define our LLM and OPIC. I guess let's avoid OPIC because that was just to show some difference. I'll comment this line.
But we know how to use Opic right everyone. OK, so here if you see I'm just defining track whatever function I'm defining here will be traced. So this one I'll comment and if I want to append the traces Opic context dot update current trace.

**Hardip Patel** 36:38
Yes, yes.

**Ajay Patel** 36:39
Yeah.
Yeah.

**Tarun Jain** 36:54
If you use feedback score it will update feedback score. If you define metadata it will upend metadata. So what is the format for metadata? It is dictionary. So if you see here whatever memresult I had right memresult I'm saving scores. I'm saving category which is my metadata memory and ID. So all this.

**Ajay Patel** 37:04
Oh.

**Tarun Jain** 37:13
Information.
So you're already seeing this. This will be saved in tracing so that if anything goes wrong you can debug it. That is the main agenda of tracing.
Here I just need LLM which is from Google dot Genai import Genai sorry client and I don't have to run this because all these three are for Opic. I will directly run LLM client as client.
So now if you see this is the role play and I'm taking the example of travel assistant.
So when it comes to travel assistant there is a rule called executive.

**Ayush Makwana** 37:52
Yes.

**Tarun Jain** 37:55
Assistant.
How many of you know there is a job role of executive assistant?

**Ajay Patel** 38:03
Executive assistant. This is something new for me because.
Anyone aware?

**Tarun Jain** 38:09
OK, so executive assistant is like a manager, personal manager.

**Hardip Patel** 38:10
Yeah, they're from there first right now.

**Ajay Patel** 38:14
Oh, personal assistant.

**Ayush Makwana** 38:15
OK.

**Tarun Jain** 38:16
Huh. So like, if you know Shahrukh Khan, Pooja Dadlani.

**Ajay Patel** 38:17
B.
OK, OK.

**Tarun Jain** 38:21
So Pooja Datnali is the executive assistant of Shah Rukh Khan. So whatever Shah Rukh Khan does, she plans that.

**Ajay Patel** 38:31
OK.

**Tarun Jain** 38:33
Right. So executive assistant is like your personal manager. So here also what we are trying to do is we are building an agent who understands your preferences and that will make a decision. So that decision is like if you are going New York, which hotel you have to visit.

**Ajay Patel** 38:37
Hmm.

**Ayush Makwana** 38:45
Children.

**Tarun Jain** 38:48
And uh, what food to try? So those kind of queries.

**Hardip Patel** 38:54
Mm.

**Tarun Jain** 38:55

So you can check out this prompt. It's related to that itself. So if you see trusted advisor, so executive assistant role is to make decision as on on your on your behalf.

**Ajay Patel** 39:10
The complex.

**Tarun Jain** 39:19
Here is a done system prompt.

**Ajay Patel** 39:21
Hmm.

**Ayush Makwana** 39:22
Yes.

**Hardip Patel** 39:22
Yes.

**Tarun Jain** 39:22
Alright, so here if you see the starting 3 lines of code is same.
So what did we do here? We already exposed that particular lines here. First, whenever you encounter new user query, you have to extract your relevant memories. Once you have the memories, what are you trying to do? You're getting the context. So I just wrote these three lines 1st and then I copy pasted here.

**Ayush Makwana** 39:38
8.
I mean.

**Tarun Jain** 39:47
So client dot search you have the query query then user ID then limit. This was the same line. Now from memories I need to extract memories result and then from memories result all the memory I have to combine it and then convert it into a context.
So these three lines are same. Now you have system prompt and when you have

system prompt you also need to have a user prompt. All the input variables will go in user prompt just like rag. So in rag you had context. That context will also be here. But now this is your memory.
So we can also have your rag response also here. So this is your context. So instead of context you can make it your preference.
Preference.
Reference reference and if you have context you can define context.
So now this context is coming from rag.
Is this clear?

40:41
Hmm.

**Hardip Patel** 40:43
Yes.

**Tarun Jain** 40:43
OK, so one thing always keep this in mind, never use input variables in system prompt, only use that in user prompt. Once you have user prompt, the last step is to define your response which is LLM client dot models dot generate response.

**Ayush Makwana** 40:43
M.

**Ajay Patel** 40:52
OK.

**Ayush Makwana** 40:52
Mhm.

**Tarun Jain** 41:01
Here if you see now I'm using different LLM. This LLM will generate the final response which is Gemini 2.5 flash and your input is nothing but your user prompt and inside configuration you can define your system instruction. So this is simple how you configure your.

**Ajay Patel** 41:04
OK.

**Ayush Makwana** 41:07
Yeah.

**Tarun Jain** 41:19
Gemini LLM and Opic I will remove it.
That's it. So extract your context, define the prompt, use it to the LLM.
And now if you see I'm asking I need food place recommendation for the food in New York. So when it gives recommendation in New York, I need the places which has Gujarati food. Why? Because that was saved in my preferences.

**Ajay Patel** 41:46
Mhm.

**Tarun Jain** 41:51
Since we are running locally, this will take around six to seven seconds at Max, but if you use quadrant cloud it will be a bit faster.
So if you see it has generated some recommendation, where is the place? Certainly given your preferences Vatan Vatan. So I'll just copy this Vatan Indian restaurant.

**Ajay Patel** 42:09
One and up.
Akin to call it in Baroda and your desire for.

**Tarun Jain** 42:19
And I will.
So if you see what that is in New York.

**Ajay Patel** 42:26
Hmm.
Do you?
It is.

**Tarun Jain** 42:38
OK, it's authentic Indian cuisine. You have a then they have two to three branches which is from New York. So here what you can do is make sure you define it as print so that you can see it very clearly. But I hope you got the response.

**Ajay Patel** 42:52
Yeah, similar to you.

**Tarun Jain** 42:53
It's very detailed. What happened?

**Ajay Patel** 42:56
No, similar to your response is similar to you.

**Tarun Jain** 42:59
Now if I copy this to pure Gemini.
It is very generic. It doesn't know what my preferences are now. So if it gives Vatan, then probably Gemina is good. Let's see.
OK, it has taken landscape.

**Ajay Patel** 43:24
So correct me if I'm wrong Tarun. So I guess MEM zero they are using a lot. I guess ChatGPT is using MEM zero behind the scene whenever we start a new chat.

**Tarun Jain** 43:34
Oh no.
Why will ChatGPT use a competitor? They might have their own memory logic which we are not sure. So this is very important. So if you look at ChatGPT right, have you ever thought this like when you use ChatGPT?

**Ajay Patel** 43:42
OK, OK, OK, OK.
I think.
Hmm.

**Tarun Jain** 43:52
And when you upload a PDF.


**Ajay Patel** 43:54
Hmm.

**Tarun Jain** 43:55
Is it using rag?


**Ayush Makwana** 43:59
It is fine.


**Ajay Patel** 44:01
I don't think so.

**Tarun Jain** 44:04
Yeah, because they never exposed it. So they might use vector store, they might not use any vector store or there might be very limited for free users. We are not sure. But in the recent recent release they also mentioned that when you are creating a new chat we.


**Ajay Patel** 44:07
OK.
Mhm.

**Tarun Jain** 44:20
Use the chat from previous history as well, but they're not exposed what they're using. So ChatGPT will have their own memory, will have their own multimodal. They might also have their own drag, but we don't know. ChatGPT is different than GPT.


**Ajay Patel** 44:26
OK.

**Tarun Jain** 44:37

So GPT is a base model, ChatGPT is a product. So how they defined all these details, we are exactly not sure like thinking how did they build they they're not shared those details.

**Ajay Patel** 44:37
Hmm.

**Ayush Makwana** 44:41
OK.

**Ajay Patel** 44:45
Yeah.
Hmm.

**Tarun Jain** 44:53
So it might be, but I don't think Memzero, I mean Open AI and Memzero are collaborating.

**Ajay Patel** 44:53
OK, but I mean.
Mhm.
OK.

**Tarun Jain** 45:04
Because when it comes to default, right, all the frameworks that you have seen so far will have Open AI as defaults like Llama Index, Agno, Agno. I'm sure because I've worked and I've collaborated in some of the projects with Agno and when I met their owner, I'm pretty much sure they don't have any links.

**Ajay Patel** 45:10
Hmm.

**Tarun Jain** 45:23
With open AI and there is something called a swarm. Did I tell this word earlier swarm?

**Ayush Makwana** 45:29
Yes, yes.

**Ajay Patel** 45:29
Well.

**Tarun Jain** 45:30
So if you just search for swarm, open AI.

**Hardip Patel** 45:31
Yes, yes, that is.

**Tarun Jain** 45:35
And check their framework how they have built it.
So you import swarm, you have agent name, instruction and instead of tools they have functions. So isn't this very similar to Agno?

**Ayush Makwana** 45:50
Mhm.

**Ajay Patel** 45:50
Mm.

**Tarun Jain** 45:51
So Agno came first, then they came.

**Ayush Makwana** 45:52
Yeah.

**Tarun Jain** 45:56
Same goes for Google EDK.

**Ajay Patel** 46:00
Agent Development Kit.

**Tarun Jain** 46:02

So even if you see the syntax get started.
So you just have to learn one framework. 2nd framework is very easy to quick, I mean very easy to pick up.
So if you see here you have agent, agent name, model is also same, description same variable instruction is also same, tools is also same. So here if you see tools you are defining custom.
So when you're defining your custom tool, you have to have your doc string and you need to have a data type.

**Ajay Patel** 46:41

Mm.

**Tarun Jain** 46:41

So if you see, that's it. If you learn, you have learned Google ADK by default.
The only thing Google ATK does different is your deployment, which is our agent engine, Cloud Run and GK.

**Ajay Patel** 46:56

OK.

**Tarun Jain** 46:57

Till here is a done.

**Ayush Makwana** 47:00

Yes.

**Tarun Jain** 47:01

OK, so now whatever we did so far, we use our own configuration. What if you want? You don't want to define that. You can directly use mem 0 API key and from M0 you will import memory client instead of memory and you will save your mem zero API key.

**Ajay Patel** 47:01
Yes.

**Ayush Makwana** 47:17
I should.

**Tarun Jain** 47:18
Then the syntax again is same. So instead of doing memory dot from config you will directly define your memory client. Then you will do mem client dot add new message user ID. Then again message new message user ID. This is the same thing.

**Ayush Makwana** 47:34
Mhm.

**Tarun Jain** 47:35
If you see the syntax is same and then for searching also the syntax is same. You give the query and you define the user ID.

**Ayush Makwana** 47:37
OK.
Thank you.

**Tarun Jain** 47:43
And here if you see you are getting all these details time. Earlier we didn't get time. Why? Because they're using some deprecated time framework.

**Ayush Makwana** 47:49
Yes.

**Ajay Patel** 47:50
Mm.

**Tarun Jain** 47:58

But that can be fixed. That is not an issue. We can just clone the repo and then use instead of doing pip install mem 0 because it is open source.

**Ayush Makwana** 47:58

Thanks.

Mhm.

**Tarun Jain** 48:11

Oh, is it familiar this syntax? We're not doing anything new here. It's the same thing. The only change is this line.

**Ayush Makwana** 48:16

Hmm.

Yes.

**Tarun Jain** 48:21

And instead of config you have environment variable.

I'll quickly show the agno part because this is again same. If you see you have user prompt, then you have client and then you're defining your LLM response. It's not a new code.

**Ayush Makwana** 48:39

Mm.

**Tarun Jain** 48:42

Oh, any doubts before we jump to agents with memory? So far it was LLM with memory, not agents with memory.

**Ajay Patel** 48:49

OK.

**Ayush Makwana** 48:50

Mhm.

**Tarun Jain** 48:51

OK, so now I'm defining from agno dot agent, then agent, then from models I'm getting Gemini. Now if you see instead of memory, I'm defining it as a tool. So can anyone tell me why?

**Ayush Makwana** 49:10
Because the agent we need to use the into the tools like uh uh before we what we used um.

**Tarun Jain** 49:24
Oh yeah, Amitesh, continue.

**Ayush Makwana** 49:26
Uh, yeah, I guess the.
What we used last time? The uh eggnol.

**Tarun Jain** 49:34
So if you see when you have memory dot add then you had memory dot save. These are the two functions you're using.

**Ayush Makwana** 49:34
Into the we can use the tool something.
Uh.

**Tarun Jain** 49:46
Am I correct? These are the only two functions we use, right? If you see you have, where is it?

**Hardip Patel** 49:47
Yes.

**Ayush Makwana** 49:48
Yeah.
Yes, yes, yes.

**Tarun Jain** 49:54

You have add and you have search. So what are these? At the end of the day, these are nothing but your functions, right? Add is a function. Save is a function. Obviously it's called class methods because memory is a class. Add and save is.

**Ayush Makwana** 50:01
Functions.
Yeah.

**Tarun Jain** 50:10
Add and save his class methods. But what are these class methods? This is nothing but a custom function.
Correct. So when it comes to agent, if you have a custom function, what is it called as?

**Ayush Makwana** 50:19
OK.

**Tarun Jain** 50:28
So I defined def Google search.

**Ayush Makwana** 50:29
Do the best one function.
Oh.

**Tarun Jain** 50:32
Search.
So now what is this depth search when I give it to the agent?

**Hardip Patel** 50:38
Tool.

**Tarun Jain** 50:38
It is tools same you have add and save which are which are your custom function.
When you're defining with an agent, you will not define it as a memory, you will

define it as a tool. The functionality will be the same. So instead of using memory it will use tools saying that.

**Ayush Makwana** 50:39
It is a tool.

**Tarun Jain** 50:56
Add memory then retrieve from memory. So there will be something called as retrieve or search. Let me confirm you have search memory. If you see one is search and one more is add. So these are the tools which.

**Ayush Makwana** 51:07
Yeah.

**Tarun Jain** 51:11
Mem 0 provides.

**Ayush Makwana** 51:13
Hmm.

**Tarun Jain** 51:13
And the syntax is same agent you have a name name instead of personal what you call agent. It's better to use executive assistant.

**Ayush Makwana** 51:15
OK.

**Tarun Jain** 51:25
And then you have Gemini again the same thing. Then tools you have M0 tools all in the sense whatever tools it supports, use all then user ID, whichever user ID you're defining description, instruction and markdown. That's it. And the only thing what you're supposed to do is you have to define your client.

**Ayush Makwana** 51:44
Hmm.

**Tarun Jain** 51:45
Which is your mem 0.

**Ayush Makwana** 51:48
OK.

**Tarun Jain** 51:50
Any doubts in memory part? We just have 5 minutes because I have to head to airport.

**Ajay Patel** 51:51
M.
OK.

**Tarun Jain** 51:59
Till here anyone has any doubts? This is it with the memory part.

**Ayush Makwana** 52:02
No, no.

**Tarun Jain** 52:05
It's just three lines of code, message, add and then search.

**Ayush Makwana** 52:06
OK.
Food.

**Tarun Jain** 52:13
But make sure user ID is defined.

**Ayush Makwana** 52:16
Yeah, mm-hmm.

**Tarun Jain** 52:18

OK, so quickly coming to the agent T crack. So why do we need agent T crack? We discussed some issues, right?
So when it comes to rag, if you ask any questions like good morning.

**Ayush Makwana** 52:26
And.

TJ **Tarun Jain** 52:32
What are you getting the responses? You are getting the responses. I don't know.

**Ayush Makwana** 52:37
Good. Yeah.

TJ **Tarun Jain** 52:39
So now what we are trying to do is we will build agentic rag. So agentic rag, the only plus point is the decision making capability. Apart from that all the functionality is same.

**Ayush Makwana** 52:42
Yeah, mm-hmm.

TJ **Tarun Jain** 52:53
So here what you can do is you can define instructions.
And this instructions will be like if the query is from the knowledge base.
You will use knowledge base.
And then you can define if the query is not from the knowledge base.
For example, who is Virat Kohli? Virat Kohli is not there in the Atlantics website. So in this question, how do you want to tackle this as? Do you want to tell it as I don't know or do you want to keep this as a search?

**Ajay Patel** 53:21
Mm.

TJ **Tarun Jain** 53:33
So now what it will do? We had one use case called mathematical agent. So this

math agent what we had was all the textbook that we had it related to mathematics. We saved it. We saved that in a knowledge base if this mathematical question is not from the knowledge base.
But if it is available on topper.com, so you have toppr.com.
Sorry, not this one.
You have topper.com Max.
Oh.
OK, not this one also, which is the topper.com.
There is some website or maybe it is stoppers. So now what it will try to do is if the query is not from the knowledge base it will use search. So this is routing. Basically you have a query and then you're deciding should it go to the knowledge base.
Or introduce search?

**Ayush Makwana** 54:35
Mm.

TJ

**Tarun Jain** 54:36
And in order to make this decision, we are using agent required.

**Ayush Makwana** 54:40
OK.

TJ

**Tarun Jain** 54:42
So this syntax is again same. If you see you're defining your quadrant client models, points, text embeddings, parse embeddings. And if you look at this def knowledge based tool, this particular syntax is similar to what you have a state.
You have node right search node that we created for Langraff. You guys remember the Langraff search node?

**Ayush Makwana** 54:59
Mm.

**Ajay Patel** 55:04
Mm.

**Ayush Makwana** 55:14
8111.

**Tarun Jain** 55:15
So here we had state, then we had drag state.

**Ajay Patel** 55:19
Oh.

**Ayush Makwana** 55:19
And.

**Tarun Jain** 55:19
And inside the track set we had the same logic. So once you have the user query, you convert that into embeddings. Then you're using prefetch for both dense and as well as parse and then you're using query points for search. So this is nothing but your search.

**Ayush Makwana** 55:21
You.
Yeah.

**Tarun Jain** 55:36
So once you have your information, you're returning it as in form of.
Here we returned it form of list of STR. But here we are not defining a node. What are we defining? We are defining a tool. So what will be the output of tool?
It will be a string. So if you look at the output instead of returning a list, I'm combining that with join dot and now this is string. Is this clear this one?

**Ayush Makwana** 56:04
Yes, yes.

**Tarun Jain** 56:06

So this was the search node that we defined for langraf. It's the same code, the entire thing. Now you have instruction. So OK, I didn't share this collab notebook.

**Ayush Makwana** 56:13
Yeah.

TJ

**Tarun Jain** 56:29
Where is it OK?
So this is for agentic rag. The early on one was for memory, but all the cookbooks and notebooks are currently pushed on this repo. So if you open notebooks inside notebook you will find diagno. So you have agentic rag, you have memory agent and whatever multi agent we defined right the customer agent and.
The marketer agent that code is this one and then the quick start is nothing but the tool calling so you can check out notebook and Agno.
OK, if you download you will see it.

**Ayush Makwana** 57:04
Mm.

TJ

**Tarun Jain** 57:10
OK, cursor is opening. Meanwhile I'll come back to the code. So once you have the instruction, if you see here you have expert QA assistant, you have model which is Muni 2.5 Pro. Now what are the tools you have? The first tool is Tablet tool which is your search.
And then you have knowledge based tool. So if you see this knowledge based tool this is a function that you have defined here.
So this is a custom function.
Is it clear how we are defining it? So in this notebook there are two approaches. One how to use custom tool. This is what I will recommend.

**Ajay Patel** 57:38
Mm.

**Ayush Makwana** 57:40
Yeah.

**Tarun Jain** 57:48

Custom tools recommended because you'll have more flexibility to define your search, right? When it comes to Agno, it will make too much of shortcut for you so that there is no flexibility. If you want to build the flexibility, you have to build a custom component which will take too much of time.

So if you just have a custom tool, it is just a task of one function right? So first one is the custom tool for agentic rack. Define your knowledge base tool. Use an existing search tool. If you want to have your own search tool that also you can define as a simple function.

And then you can ask a simple query. So if you see a mail to contact atyantik, it's using knowledge base tool. And now if I use who is Virat Kohli instead of telling I don't know, it will go to search tool and then generate the response.

**Ajay Patel** 58:33

Oh.

**Hardip Patel** 58:37

Hmm.

**Ayush Makwana** 58:37

Hmm.

**Tarun Jain** 58:37

So this is agent integrated and I also have one demo for this, but that is in a llama index.

**Ayush Makwana** 58:39

Nice.

OK.

**Tarun Jain** 58:50

I'll show the workflow. So if you see this is how the workflow will look like. You have query router agent. If it is relevant to KB it will use KB tool. Then if it is not found

then it will use search.
Is this clear? And in search it is using Google and then you have the final response.

**Ayush Makwana** 59:04
Mhm.
Yeah.

**Tarun Jain** 59:10
There might be one more tricky condition. So let's suppose you have a PDF. So let's suppose you have a PDF that you saved inside your knowledge base.

**Hardip Patel** 59:11
Is.

**Tarun Jain** 59:25
And this PDF is related to React.
React paper, which is your agent's paper. Now in this entire paper you don't have anything specific to RAG.
So now let's suppose you have user query.
Which is related to rag. What should it use?

**Hardip Patel** 59:49
Web search.

**Tarun Jain** 59:51
But how do you get to know that is a web search? So React paper is 33 pages, correct?
We know or we don't know that rack term might be there in this 33 pages or not because you didn't read the paper yet and you're directly using an agentic rack just to see if rack concept is there or not.

**Ayush Makwana** 1:00:10
Hmm.

**Tarun Jain** 1:00:11

In that case, it's a very tricky situation. It will use knowledge base.
Why? Because agent is very specific and RAG is also very specific to same domain, right? RAG and agents comes under NLP and it will be confused and it will use knowledge base. So whenever you don't get any context from knowledge base what it will do it will use search.
So it will use both the tools in the scenarios. When you have a domain expert question which was supposed to be in which was supposed to be there in the knowledge base but it was not there, then it is using search.
So this is what this flow meant. So if you see here relevant to the KB you have S then it is checking found. If it is not found then it is using search again. So if you see search has two ways one relevant to KB no then it is again going to search.
Even if it is relevant to KB and if it was not found, again it is using search.
So that approach this one if you see it is going to KB tool then again it is going to search and that is the HCS of this one that the domain specific question was asked which was not there in the knowledge base then it use search.
Is this clear?
So you'll also see those situation where you'll have both knowledge base and web search. In this case, what you usually do is you add context.
So if you see here in my system prompt I have some context which I'm adding. So actually this is not system prompt, this is just a basic prompt which is appended. And here if you see you have the some context which you are adding as a input. Now this input will tell whether the given information will be the relevant or not and then it can decide the routing.
And I'll also share the scholar notebook so that it will be something that you can relate to. The concept is same, the import statements is same, but the only thing is the syntax. The way you import will be different.
Is it clear till here the custom tool one?

**Ayush Makwana** 1:02:22
OK.

**Hardip Patel** 1:02:23
Yes.

1:02:25
Yes.

**Tarun Jain** 1:02:25
OK, so now coming to the knowledge. So Agno also provides something called as knowledge and here we'll be using PIE PDF, PIE PDF library. So you have knowledge, knowledge and here we are using quadrant and search type is when you have to define hybrid.

**Ayush Makwana** 1:02:40
OK.

**Tarun Jain** 1:02:43
So in line graph or line chain we had something called as retriever mode.
Here you have search type.
In Agno it is called search type.
So it's like if you define IB to be true, it will download B on 25 model. So let me show you that part.
So I will run this two cell and when I define this quadrant right so I'll have to refresh. I will define this then collection name and now when you do DB write it will download the model.
OK, let me run this URL.
So here if you see search type you have defined search type dot hybrid is nothing but your BM25. This should actually download the model. So can you try in your system because I have to delete the runtime and then rerun it.
If you check this syntax, you have to define quadrant. Inside quadrant you have collection name, URL, API key, then embedder. It's nothing but your fast embedder.
And if you want to use Gina, you have to define ID, then define Gina, then dimensions.
Equals to. If you see you have 384 this has to be 368 but in their back end code they have overwritten as 384. So that's the reason why I've created this PR. So once that is merged then you can use that logic.
Yeah, this one.
So far we'll just use the 384 one, the default one. But you can change this and if you

don't define this, let's suppose if you don't define this, it will ask you to enter a Open AI API key. Why? Because by default the embedding model is Open AI if you define this.

**Ayush Makwana** 1:04:42
Mhm.

**Tarun Jain** 1:04:58
Now your model is sparse embed embeddings.
Is this clear? If you don't define this, it is open AI, but if you define it, it is what you imported. It can be cohere, it can be fast embed or it can also be a game phase.

**Ayush Makwana** 1:05:05
OK.

**Hardip Patel** 1:05:15
Yes.

**Tarun Jain** 1:05:16
Till here is it clear the vector database import statement. Then you have knowledge base. Knowledge base is nothing but your knowledge which is mainly used for agentic rag in agno. Inside this you just have to define 2 variables. You have vector store equals to DB which is your quadrant.

**Hardip Patel** 1:05:18
Yeah.

**Ayush Makwana** 1:05:19
Yes.

**Tarun Jain** 1:05:33
And instead of K you have a value called Max results. So Max results is nothing but your key that you define in vector store. Like for the given user query, how many relevance documents should I extract? By default it is 10, but we can reduce it to five or seven.

And then what you have to do is you have to do knowledgebase dot add content and you have to pass your information. This is very similar to.
Vector store dot add document. Do you remember the syntax? And then you have your chunks in line chain.

**Ayush Makwana** 1:06:08
Yeah.

TJ **Tarun Jain** 1:06:09
So instead of add document you have add content. This is where you are doing indexing and indexing needs to be done only once. Once you index it, you can directly define a knowledge base and you don't have to use this again.
So this should happen only once.

**Ayush Makwana** 1:06:27
Mm.

TJ **Tarun Jain** 1:06:28
Just once.
And once your data is saved, you don't have to index it again.
Till here is it clear this syntax?

**Ayush Makwana** 1:06:39
Yes.

TJ **Tarun Jain** 1:06:40
All right, and the final step is you just define your agent. That's it. So you have role, you have model. Now you have tools here. If you see you don't have two tools now. Why? Because my RAP pipeline now is in the knowledge base. So you have knowledge, knowledge base, then search knowledge to be true. These are the two new additional parameter.

**Hardip Patel** 1:06:41
Mhm.

**Tarun Jain** 1:07:00

Earlier for custom function you had knowledge base as a second tool, but now since we are using the existing knowledge feature instead of tool we are passing it into knowledge equals to knowledge base, search knowledge equals to true and then the instruction. So instruction again is the same.

If the user query is from the knowledge base, use knowledge base tool. If it is not from the knowledge base tool, use search. So here if you see tips on how to implement features, it's using search knowledge base and then it's using focus on implementing 1-2 feature per cycle.

Let me open the PDF.

So it's from this part.

So this is one of the copilot called Emergent and they have their own cookbook in which they tell how you can start with your incremental development strategy and for feature this is the information and this was the data that I saved inside my Vector DB.

**Ayush Makwana** 1:08:00

OK.

**Tarun Jain** 1:08:01

So if you see here you have path inside path you have content emergent build guide. It should be here and once you add your content then you can ask your user query. So it's using search knowledge base. If it is not from the knowledge base it will use search.

And here there is one syntax which I added. This was for teeth basically. If you want to use specific chunking right, what you can do is you have to define your own reader. So if you see it when you do add content, now you're adding your own reader which is your pipe PDF reader.

Inside this you are defining your chunking strategy to be agentic chunking, which is just one line.

And that also we can check the documentation dogs dot agno and it will be inside knowledge. You have chunking. Here you have fixed size which is by default. Then you have agentic chunking which which is the one which I recommended. Then you have schematic chunking. This is also much better compared to.

Fixed size. So here you have to define do you need the trade off in the time or do you need the trade off in the indexing? So if you need trade off in indexing you can use schematic chunking. So the inference will be more but the chunks that you are saving will take.
Sometime, but when it comes to agentic chunking, the inference will be more when it comes to your inference level. So let me break this down.
So you have agentic chunking.
Then you have schematic chunking.
So you have two parts. One is your indexing, one is your search.
So here it is faster when it comes to indexing, whereas here it is slower.
And agent is something when it comes to search plus inference.
Here it is slower, but schematic chunking when you do search and inference it is faster.
Or does it make sense this part?

**Ayush Makwana** 1:10:07
Yes.

**Tarun Jain** 1:10:12
So these are the two chunking approaches which Agno provides and agentic chunking is something that only these people have. Our other players will never see agentic chunking. But these are common recursive chunking we saw in line chain and as well as ramindex document chunking is your character chunking.

**Hardip Patel** 1:10:15
Yeah.

**Tarun Jain** 1:10:29
Then markdown is also very common. Schematic chunking is also there in all the frameworks. Only agentic chunking is there in Agno.
That's it. So this was I I added this in the comment because this was something that was specifically asked.
And if you want to run it into essential mode, you also have essential. If you notice I use these two lines here. I use these two lines before knowledge base because knowledge base is a event pool. So this is a event pool which is running async.

**Ayush Makwana** 1:10:52
Mm.

**Ajay Patel** 1:10:58
Mm.

**Tarun Jain** 1:11:02
And in order to do this, I'm using these two lines.
To support a synchronous programming in Collab.

**Ayush Makwana** 1:11:11
You.

**Tarun Jain** 1:11:14
Yeah, mostly this is it with agentic rag and as well as memory. So we can start with the projects and if in case we need any session in between or if you have any doubts you have my e-mail and next probably once you are ready with the projects we'll have the fine tuning meet.

**Ayush Makwana** 1:11:15
OK.

**Ajay Patel** 1:11:34
OK.

**Ayush Makwana** 1:11:34
OK, OK.

**Hardip Patel** 1:11:36
Alright.

**Tarun Jain** 1:11:37
Oh, any questions?

**Ajay Patel** 1:11:37
Project definition would be project definition would be provided by you or we can take our own.

**Tarun Jain** 1:11:43
So the continuation of the RAG project. So there were a few RAG projects which were already there. Evaluations were there, right? Evaluations. What you can do is you can see did agentic RAG improve your performance or not?

**Ajay Patel** 1:11:45
OK.
Mhm.

**Ayush Makwana** 1:11:57
Yeah.

**Hardip Patel** 1:11:59
Today.

**Ajay Patel** 1:11:59
OK.

**Ayush Makwana** 1:12:00
OK, OK.

**Tarun Jain** 1:12:02
OK. Yeah. Thanks everyone.

**Ajay Patel** 1:12:05
Thank you. Thank you.

**Hardip Patel** 1:12:05
Thank you.

**Ayush Makwana** 1:12:05
Thank you. Thank you.

**Tarun Jain** 1:12:05
Yeah, all the code is there like in the collab, the GitHub that we have.

**Ayush Makwana** 1:12:08
OK.

**Hardip Patel** 1:12:12
OK.

**Tarun Jain** 1:12:13
OK. Yeah. Thanks. Thank you.

**Ayush Makwana** 1:12:14
OK. Thank you.

**Hardip Patel** 1:12:15
Then getting here.

**RamKrishna Bhatt** 1:12:15
Thanks.

**Hardip Patel** stopped transcription