Python and AI Power-Up Program Offline Class-20251003_192708-Meeting Recording

October 3, 2025, 1:57PM

1h 24m 8s

Tirth started transcription

Tarun Jain 0:08

OK so this were the pending task before we proceed to agents. So we thought of using fast API and how to create endpoints with fast API and 2nd is how to use post. Gray SQL.

To start.

Next.

OK.

So we'll start with fast API. Uh, so if you look at this repo Python.

Then.

Workshop uh scripts.

Raghas Langraph. So we'll utilize the same code. So I'll just share this repo if in case you don't have this right and if I have to check this files, let me open the VS code. Let's use VS code instead of collab for the first one. So we have this client dot PY. Is this familiar? So what are we trying to do? Whenever we create any code base, it's better to create in multiple files. Client dot PY has your LLM and embedding model. Then we have config dot PY. If anything needs to be changed, let's suppose I want to change model name. I'll come to config dot PY and then we had eval dot PY. Main dot PY was like this where you're just using a simple for loop. So here what we will do today is instead of running it in the state graph builder.

We will run it via API, so here we will.

Perform inference on API endpoint and how will we create this API endpoint using fast API?

And rest remaining files are same. You have state dot PY then utils dot PY we didn't touch because there was no error handling that we had right? So app dot PY was not there. Even if you check in this repo we don't have app dot PY. So let's create app dot PY. That is where we'll create our fast.

API embeddings.

No same repo. We'll just create a new file app dot PY. So we have main dot PY but

we don't have app dot PY. So now we need two new libraries. The first library is pip install.
Fast API and 2nd is pip install.
UV con. So UV con is the server where you will execute your fast API. Do you guys know what is this ESGI server?
Asynchronous gateway inference.
It should be asynchronous. It's called asynchronous server gateway inference. So usually whenever we work with this libraries of Python, you have Django, you have Flask and you have uh fast API. So there are different servers using which you will run it. One is.
You have ESGI. I'm not completely sure how this works, but usually it is mainly used for web applications whenever you are working with web application and then you also have something called as WSGI. We use WSGI for the rest of our implementation so far with PHP and everything.
So in WSGI you have this UV con. So what we did is we did this UV con. So once you write fast API code, how do you run it? You need a server to run it, right? So for that we'll be using this WSGI.
Can we do all about poetry? I'm just asking is poetry going to poetry? You can't know. I like poetry a lot. It was very simple and it so like poetry is like once you build your own library, right? During that time you'll use poetry a lot. So to those who don't know the context of poetry.
We have requirements dot DXT right? Just like that. Let's suppose you have any library. So what library we have used? Land chain PIP UI.
So whenever you want to create this version here, if you see 0.3.27 and now if you want to create a new release, so you have a file called pi toml file TOML. So you might have seen this TOML file in most of the open source frameworks.
He.
So uh very simple context. If we have component or Jason or if we have package dot Jason, we have this file dot UML where you can do poetry add and then you can just add it by by identic or anything you want to add. Huh. Same same thing.
It is very similar to that. OK, because that's the log file you can say it is simply uh, composite or Jason file. Uh, OK, because composite or Jason file similar. OK, so usually it's a replacement for requirements not because not every time in when you're hosting your libraries, you'll use it.
So poetry can use it. So when you use poetry you'll typically have these two new files.

One is poetry lock and one more is pipe project dot tamil file and and if you want to build your libraries.
During that time we'll have these two files which is mandatory with requirement requirements dot txt will be there. So with that you'll have these two new files. That's it. So now coming back to the code, let's first do quick start and once we have the quick start then probably we'll go with app dot PY.
And here I'll show you 3 examples. One is get method, one more is post method, and one more post method using pydentic because this is what we'll be using a lot instead of just base post command. So first we'll install it from fast API import fast API.
And there are two ways to test it. The first way is we'll use the local host and fast API provides something called a swagger UI. So swagger UI basically you have the endpoints and what is the?
My default. So first step is just import. Then you have app equals to fast API and then I'll just use a simple lab dot get and I'll have health check.
I'll check and then that.
I'll check status equals to OK, so it's a simple thing. You have a function which is nothing but health check and then you are retaining a Jason format. The only thing is whenever you're using fast API, just add a decorator. So for decorator, whatever REST APIs command you have, whether it is post put, you can use it here.
Now if I do app dot getl check it should show me status OK and I'll add one more dummy command.
App dot get.
This should work. So instead of read read route, I'll just keep it as index. Then you'll have message.
And welcome to this endpoint. All right, so now if you want to run this, there are two ways you can run it. First thing is you can directly run in your terminal which is called UV for.
And now the syntaxes. First thing is your file name. So what is the file name here? It will be quick start.
Then whatever variable you're using here. So what variable are we using here? It's app. Then will be our host. If you want to define your host, you can define your host. If you want to define your port, you can define the port and then reload.
So by default it will run on 8000 port, so I'll just copy this command.
So I hope you're following the ones who are on online. So we just have to import fast

API. The command is pip install fast API and then typing. As of now we are not using but we'll use it later.
Then you have app equals to fast API, just that's it. These are the two initial lines.
Then simple function. Before you define function you just have to add your REST API. This thing get or anything and this is your route. What route you did.
And in order to run it, first method is this thing. If not, what you can do is second method is I'll show. So now I'll come back to terminal. I will stop all this.
And here you just start to run UV con. So can you see this autocomplete? So you have UV con, then you have app. App is nothing but the file name. Then you have host, host, you have 000, OT is 3000, then reload, right? So now I'll just do UV con. Quick start app and then reload.
So this will point a local host which is running on 8000 if I just copy this.
So you have welcome to this endpoint. Then you have a check status. OK, now here if you just write docs, this is your Flagger UI and probably I guess this is something you might be very familiar with. So here you just click on.
No, we we still have to work on this because the output model is not there. The schema it's not there. If you which is called on the you see the example when you it needs to be the schema.
That we had that. So this is good. This is better than the configuration that we do with node JS and PHP. They still need a bit of refinement to see what kind of output you want, but this is still good.
So what we do is whenever we write documentation for API reference, we just copy paste. We don't do much of the index, then again click on try it out, then execute. You have your request URL call comment.
And.
That's it. Now let's suppose I want to test this.
Oh 3.1 oh S specification 3.1.
So I can try it. So once you get this base URL right, you can use all the route. If you use docs that will redirect you to swagger UI.
And if you want to test this in Python, what you can do is you can create a simple API dot PY file. So we created two files, one is app dot PY and now I'm creating one more. It's called API dot PY. I will just do.
Import requests.
And uh URL.
There's nothing but I'll paste it and I'll just use health check then response equals to.

You have requests dot get add your URL.
And then just print response dot Jason.
So this is I'm adding in API dot PY.
OK, we created 3 files, one is app dot PY, one more is quickstart dot PY and then you have API dot PY. Now this URL is running so I'll just come back to terminal.
Create a new terminal and I'll just write Python API dot PY.
So it should be status OK.
All right, so let's quickly move to post command.
Or probably you can try out the Swagger UI until then.
Actually you uh help me with one thing I might not be careful fully for this one. Can you send this folder and share it with us? Which one? It's everybody right now only check out the. It does not have everything that we're doing right now. Yeah this we are writing now. We did it live.
Right. Can you flip it and select so I can be on the same page? Oh, OK, I think you want me to not on the same page.
OK.
No, the order, the order. OK, I think all the time.
I have some additional files here, but that should be fine because I was testing it.
OK, why is it showing 800 MB? OK, I'll have to remove this pipe UI.
Date also I don't need it.
Yeah.
So we just import it and once we import what we are supposed to do is app dot get back slash. This is your decorator function is same. You just have to return in Jason.
And I'll also show second approach to run it. So first approach was UV con filename app is nothing but you gave a app. What if I give this as app two? Let's suppose I make this as app two. I come to terminal. This should not work. Now if you see it's not working.
Error loading EAGI app. EAGI is nothing but your asynchronous survey gateway.
Attribute app is not found, so this should be app which is your second keyword and then reload. So now let's use different approach.
We have name equals to.
Main here we just have to import UV con then UV con app host you can define 000 or if you want different I'll keep it 0.0.01 and port I'll keep it as 3000. So now what I'll do is I'll create a new file.
Instead of running UV con command now what you can do is you can just run

python dot quick start.
Not PY. Now why? Because I already have that.
You know.
Yeah.
So now just copy this.
Sam.
Recommended we uh UV convert hides this particular because you'll usually not add this in most of the what you call code.
It's better to use the terminal command, but there are these two ways. One is you add this name function where you import UV con. Once you import UV con, you just have to run it, give the app name UV con.
So you're not proxing it with NGX or anything else or not? Oh no, we can do. You can. Yeah, you can run a fast API on NGX. Yeah, for API for SSL and everything, because it would require essential computations for hand schedule.
OK, that I'm not sure. OK, OK, so usually we run it on the NGX. If not, we use swagger UI. If not, you'll be gone. Apart from this, I've not tested anything else because this is not what we write.
OK.
OK, I'm not sure. You're not. You think that as of now. No, I usually don't work on this. OK, OK, So what we do is we write the endpoints. Once we write the endpoints, we just share this code. Then the DevOps team handles whatever fixes they need to do.
But this is something that I don't usually work upon, but I'm since fast API I wanted to share some details like how do you create endpoints. This is will be useful.
One thing what I'm sure is this is running in ESGI. If you want to convert this into web server thing we usually use UV con that understanding you have but the SSL part I'm not sure.
OK, so let's look at few more examples here only. One is what if you have input variables and apart from input variables, if there is any optional keywords then how will you use it? So app dot get.
Items here I'll add items ID.
Then dev.
Read items, that's it. Item ID calls to item ID. If in case you have any dictionary, you can just use the dictionary and then read it. So now if you see here what is the input that we are expecting? Integer. So you have to give integer. In most of the cases

what we'll do is here we will define pydantic.
Instead of it for validation.
The talks will be updated accordingly, huh? Then off will be updated accordingly, yeah.
So if I come to terminal, this is currently running.
One second, let me remove this port because this is not automatically loaded.
So instead of 3000, this will be 8000.
Read item. So here I'll add one more.
So let me import typing from typing import optional.
What if you want to give additional query? So I'll just copy the same code.
And here I'll keep it items only and I'll keep this as get items query and new parameter Q. I'll keep it same optional optional string which is none and here I'll just add.
Query SQ. So we added two new comments, one is read item and one more is read item query.
You.
But the you information you know the you want to with comma. You would have to add somewhere Q. Yeah like what is the query string over here actually due to the same endpoint that's I I don't think so.
The documentation get item, read item. This is different but the endpoint is same and what is same so if I do.
Read items.
If I do 2, we have two and here it will be question mark Q equals to two notebooks. OK.
The commented on the the first one. Now you think commented on the first one. If I do item now, OK, that's a different end point. Now if I come back to docs, you have four end point.
Like Q is the additional parameter if you want to give anything. Yeah so.
So if I do item, then you have two, then question mark whatever input variable you have. OK, the second parameter will always be the query string.
So here if I give something different it will be wrong.
Because you are expecting the input variable to be cubed.
Thank.
But if I need to do that, we could do this and that you can do. Then you can just do.
You don't need a colons here. What you can do is.

This is Q. Then you can do add then the second variable E equal to A E equals to A.
OK, here, let's keep it something. We have a new comma. OK, OK.
Right. OK, page int int OK to then here I'll just add.
Page, page and now P equals to five.
OK, this should be item or items? Item. No, there is an error.
Uh, no, it's correct query notebooks page. Why is it 1?
OK, this should be changing. This would be optional. STR. Yeah, OK.
So now if you see this should be not page, this should be P because here if you see here I've added it as P So if I make this as page then it will reflect.
So no, it didn't give error. It took the default. So earlier what I had was the default.
Yeah no it was not. It was like this page item one. Now if I refresh this you have page one.
This should be page. Now it is fine but here we made mistake. Usually you'll not write like this. OK you will have optional then hint and here basically we'll keep it as null.
Now this page should match with this thing.
So if I keep it as one, I mean if I make this as P Hmm.
It should not take one, rather it should take none, none, none, right.
Right. Same thing. We can try with some other like a user and you can have a age and name.
We have the next. I'm not good. We are thinking like PSP and.
Users read users.
You also have this example STR. Mm-hmm. Uh, language for like frequent guest.
Then age is optional age.
End none.
Till here we got it. It's the same thing. I'm adding one more new thing. Users. I want to greet users by the name. If name is not there, I'll just keep it as a guest. Yeah, we have not moved.
I've got was not for the story. Oh correct, this will be pushed.
So here it will be written.
Please.
Message.
Welcome.
New.
OK, maybe we don't need to add function. What add function here?
All right. Yes. Yeah. Yeah. So you have name, you have age.

And now if you want to run this again, you have to come back to API URL equals to. It will be users.
And then response request.
So dot post here you have a parameter called Jason equals to data. Let me add new data.
Print. That's it. So same URL that you have data. This is nothing but your payload basically.
Then Jason equals to payload, then Jason. That's it.
No Python at Pi dot PY. OK, this is wrong.
And the H it should be a H34.
No.
Doesn't make sense.
Yeah, I think.
OK, what is the mistake here? App dot post it is correct.
The name is guest because if it won't, it won't get the guest and you are not here.
Which one? So post is fine. I'll get the default message. User it is not getting the parameters.
What does the doc show? Doc will show something there.
OK.
But I think it is excepting not in Jason format, but in uh form data. OK, wait, this should be 00. That is fine. That is more than fine. One OK 7 dot 0.0.1 is equivalent on your local at 0.0.0.
So if I test it here.
Why is it blocked?
It's very Yeah, you are 24 years old. They're actually converting not very strictly, but application is fine.
No, no, it's not running the place. I can see that because you had it.
What of it? It's getting converted to query string name equal to Tarun. This is this is what they're calling as a query string. So the parameters are going as a query string as a get request, yes.
So what should be edited here?
We we should use square brackets. I no, no, not here. EPM are those specific changes. We are doing it with the docs type. So the way we are doing it in app dot PY, let it go to app dot PY.
We have uh.

We have like an input input or something which input like in the like we are using an input that from the we are taking some inputs from the user and then OK like that yeah.
So that will happen here, right? You want the input to be taken here? Yeah, that should be possible. Name equals to input. This should be integer. Yeah, this will be int.
And now payload will be.
Dave that this will be age.
Wait, let me check if this is Jason or this is data.
No, this is fine. This is still fine. The issue is not over here. The issue is how we are defining in app dot PY.
8.
This one, Yeah, the function. So let's do one thing. Let's just create a pedantic last and this get the user details in that one. It'll work this.
OK, let us create the identity things. Yeah, less what any of the user base model.
It's not and then name an HCMM. Um I import.
Race model.
The very first parameter we see is the dictionary which you can receive within a body. You have to specify that it is a body dot dot dot, but with my identity it will just work. So this is fine.
Base model you do have user. User is nothing but whatever I've defined here and then return just user. Yeah, you know you can format it.
OK, yeah, like message. And we were giving that map like dot, yes.
We can add that message first.
It's a user part.
Do you need to come? We've got name and we got it.
Because we have identity class, it knows that we're going to get it from the body because this is updated now. Yeah, now if you see the docs first.
The document it won't write query at the end. So now now you are getting it as a Jason instead of that query. So now if you try.
Here as well and in your code as well, we'll just know all the blocks.
The video inputting data was different. Now it is shipping OK and not here as well when you were running on API dot PY it will work fine.
I just need to remove the include, so we did. We did something there. Yeah, just to conclude.

And it's.
OK, now it works. Yeah, for the way you collect data in your dot PY is how it will consider identity class or else dictionary with body. It body is. I mean that comes from.
Uh, pass API, then it will work. So yet if I just give name, let's suppose I'll keep data data. It should be date.
Uh, give it as a dictionary. OK, yeah, yeah, no, I got it. So this will be dict. Yeah. And then you. So this will be data dot name, yes.
And this will be data dot page page and then you have to give body equal to equal to the default value equal to B capital body.
BODY because it is coming from body B capital.
There is some cloud, no, yes, yes, body this one exactly and this comes from fast API. Then you could tell it like OK, this is what you could do. This comes from fast API. Then then you don't need to use the class.
So it should know that it's coming from body.
So if you come in just out now, so it will accept uh any type of input you any type of I don't know with identity you can make it work proper structure with structure like so let me make this as users too.
And here it will be users to.
Yeah, I.
You have to do the like what kind of body? If you don't give it to the body, then I think from dictionary it might understand if you just give it a dict and you don't need to give bodies. But body is what it say that it should be. OK, what if I don't define what it works? Yeah.
But you have to use a dictionary rather than anything. But that might be the binary. You know the schema that you usually use. We usually use only because data validation blindly just use instead of typing.
So this was the new example. Remaining was already added. Yeah, so this is for. So this is already done and these two approaches we already saw.
All right, so is there any other examples left? OK, we'll directly go with app dot PY. So let's come back to app dot PY. And here we need the rag app, right? So from fast API, fast API.
App because of the fast API here we can define name description and all.
Description.
Because whatever data I saved inside main dot PY, this was that one itself. Now what

you need to do is whatever you are defined in state graph builder that will come inside app dot PY. So for that I'll create.
App dot get.
Instead of this, I'll keep it health check.
Help.
Status. So this will be OK.
Which one that we are looking at now? It feels like we are working with combine some of ESP and JavaScript for which one? For the two minutes if you can look at the.
The.
But a fast API is very easy. Fast API with flask. Again, this my lines of code. If I get better documents, actually move everything to.
Exactly the documentation you want we copy from same URL. I guess there's something called as docs.
We had somewhere API.
If you see that this was we didn't even spend time here. We created the app and blindly we just copy pasted. So this was from swagger only.
This actually this format is from the open API specification. Yeah, yeah, this format comes from that. Yeah, yeah, this one. Yes, this format. That is what it comes from. And there are multiple tools like this of scribe that was scribe. Yeah scribe we have used for document. Yeah so that will use this URL and then it will basically the data in the format setting point.
The syntax is familiarized, right? All all very good for me here. OK, so I want if you have question here, it's very hard for me to answer.
Because this is not my area of interest, I know how to use this function. Once you give me a point, I know how to test it. So yeah.
So I'm I'm just removing few files. Now we just we just need main dot PY, API dot PY. I don't need it because that was used for testing main dot PY state. We don't need it. Utils, we don't need it. OK, app dot PY, main dot PY. So in app dot PY usually that logic will come which is our state builder.
So just copy this.
And I will paste it here. And 2nd what we need, we need pydantic. So from pydantic import base model and I'll define class.
Query which is correct because typically you'll just ask query. So fast API this is done, parenting is done. Then you have state which is your graph builder. Then you have

query here. What I'll do is I will define my graph graph equals to.
Graph builder and what does this graph builder do? If we have to check that logic state we have line graph code. This is your rack search node. Then we have answer node. Then we have graph builder. Graph builder is just starting the node, compiling it and returning it.
So if I want to use this, I just have to use invoke. So come back to app dot PY, create a new node app dot post. I'll create an endpoint rag.
That def if you have to create people making for you create uh which one? Like I want to create this route in multiple places. I can use the same. This should be in config. OK, OK OK because in config is your place. Yeah from config app OK OK.
Either it can be in config, it can be in.
In the documentation.
OK, doc, not keep on it.
Uh, next level.
This will be query.
Query colon and here we'll have response equals to graph dot invoke.
Ready.
And return.
Small dot content.
Block query.
Context.
Response dot context OK then answer.
This this also we can use.
So let's define last response.
Base model query context answer and here I'll just return.
What should I response and this will be. This is keyword argument so I can directly add keyword argument response. That's it. There should be some parameter here for request model.
Uh, where is that request model?
Request model request response. That's it.
So we created two base model. One is for the input and output because I have multiple Jason files so it doesn't. I mean there is no need to have different ways to use response dot query. Instead what we can do is just have pydentic and use a keyword argument. Why? Because I know you have keyword based.
Dictionary response and just return and yeah, so now what I'll do is I'll use a different

function here. So instead of 8000, I'll keep it 5000.
Wait, this should be post correct Python.
R.py.
What is the parameter for scene?
So then 5000 already used because it's a 9000, it's 5001.
No, no, no. It's not here. Yeah. This one would, yeah.
That better look.
The.
And you scroll down and click on the schema apart from the response running this one, yeah.
Oh.
I get it. Yeah. I need to. You can slow down a bit more. I don't need to be able to.
OK. No, no, good, good. This is.
OK, so I'll come back to main dot PY now. So here I've commented the code import request.
URL.
We should be able to use it from here as well directly. How do we use it? You're you're trying to post the data on that one. OK, with in docs, yes.
Yeah, and then.
Thank you.
No, there isn't been something like.
It will be near central.
Yeah.
Troubleshooting Python invalid error. Wait a second. What are we returning here?
App dot py queries query.
It is showing all that uh response response model is response.
Return response.
Oh, this is like this.
Query query and it will not come from. It will be my not my.
Yeah, this is.
Response model is response which is correct. This is correct and this is coming from Lang chain. This is issue from Lang chain.
Where is that logs?
On 9th error it is expected as we give it in a string.
No, it's already running reload. OK, wait, we are running. OK.

OK, try it out.
OK.
Yeah.
Yeah, it is not coming on the.
It's not hot removing. It is not hot removing.
Yeah, now this speaking time it will work.
OK.
What is putting the? So there is some other, right? There is not failed.
Should be very string environment oh.
Please play.
Now this spider. Sometimes spider is good, but sometimes when you see a.
OK.
Meanwhile, I'll just copy this code and send it so app dot PY we wrote in main dot PY we didn't write anything because we are testing it down.
It's the same anywhere. Let's go to the output.
And I can do a second clear so much during task which search are equal to this.
OK, but why is it?
Let's test out main dot PY if the existing app is working or not.
Or that.
Just pick.
Here if you get error that means.
OK, this is working.
Maybe something like causing issue in the response. So let me come back to app dot PY.
So let me remove this.
F dot PO is correct. Response. So main dot PO is working. Yeah, main dot PO is working. I'm just question answer STR. Let us see the response other than, you know, KRC. Just you get that we get it.
I think what you will do. You just print it and then you will without this thing. Yeah, with each and everything. OK, so so we have dot invoke. So in the dictionary we will have.
Uh, basically in this module.
I'm just checking the spelling mistakes. This is query.
Which one? We need to just the let's say let's say response. We give one value 1 by 1 each and every value respectively. So query equal to result dot query result dot

query. Yes like this.
What they've argument? No, I want to just a but the.
So remove this. Let us do not need 1000.
Yeah.
Yeah, do you recall?
The hub slash app only because I don't want to reload multiple times.
Dogs.
Write out.
We didn't know anything different. Uh, we are listening past. Mm-hmm. But what is the error?
Dict object as no attribute. Are we using query or are we using question right? Uh, very. Mm-hmm.
So let's do this something request.
What? No, no, I think it's fine. Can you just print the what you in fact print results? And see the what? Uh, how to discuss that? Yeah.
No, this we already know, no?
What did it come expected on the the return part or yeah?
OK, that is the sharing the planning.
Oh, there's something. Oh wait, no, there is. So we've got the output. So if you see this is query, this is context.
And you also have the answer.
But why is internal error like uh later uh that won't tell anything. There is I mean future so uh if this is async Q then do we need to say async define uh then Q capital Q.
No, in this small queue. What? Async. No, Async. No, no, we didn't use Async. Yeah, like as an array.
What you mean like this?
I don't know.
Do you need to the response only? Would it not convert it to response by itself if you give additional if you just return. So usually what I did was when I tested it out I just give OK what I do if you just return the return it should still work fine now.
Do it because if I think fast EPS you already given the response model, we'll convert it to the response model. That is what I'm thinking. So we can just return return it, yeah.
It it should convert by default. OK, this is auto reload.
OK, it said OK, but what is the output?

But I think you I think the problem is you also specified the response type like response model equal to response and you're also returning response.
So it is doing wise I guess. So either you remove the response model and then do it response below with KRS that should be fine.
I can think of a.
OK.
So we removed request response model model and then we are adding response and result then it should work fine and this is it twice. If it is doing it twice then I think it should.
Yes, thank you. Oh, you don't need something like a encoding thing. So we're doing encoding twice, but if you're doing it as a response model as well. So internally your response make. I don't see it.
OK, OK, thank you for that. It was good. Yeah, we're still fine. Yeah, fine. We know how this works. I tested with this is the code that I had used response model equals response. OK.
And this worked actually. So if I come back to main dot PY OK.
I didn't write it on swagger, so let's just check here. So here the URL is.
This one.
Rag.
Very equals to.
Payload equals to.
Maybe query and then just I'll remove this graph builder. OK, I'll remove this entire thing because I don't need it, OK.
Import request so in the same code, but the thing what I'm doing here is I'm repeating two times which technically it's not required.
And here main dot PY endpoint drag contact to mail at the query.
It do terminal Python.
It works here, but if I try this again.
But works now. OK, now what did we do? We didn't do it. So so what? Yeah, this is very weird. Nothing. The same thing. Response model equals to response.
See something in response model. Maybe the spelling of response model. Mm-hmm.
OK, let me copy this demo work on a paste it somewhere safety.
Yeah.
Right, right. Response model I've used and this was working. Yeah, so let me copy this.

Response. No, I didn't do any spelling mistake. Anything. Yeah, this is Python.
But yeah, we don't have to do that. The best method is result, response model goes to response, return result and this will take this format. So the first error which we made was list we just.
Wrote it as STR. Mm-hmm. But this was supposed to be STR list STR. Yeah, that's it.
So we just did quick start of what we call. We started with quick start where we used to get in post and then just added in point. OK, the code was same config meet.
So I'll just share this app.py.
Nobody else. Sorry guys, I have to go. Thank you Karan. So I'll share this hybrid search now. This was also pending last time that we discussed so.
Fast API is done. We'll quickly look at this hybrid search and then the Postgres SQL.
So for Postgres SQL I've created a entire documentation which you can follow. So do you guys use PG admin? PG admin? OK, so that means the installation step is reduced now.
So this is done, right? I'll open a different folder. Is this done right? We are OK, yeah, so I'll just open a different folder.
OK, so when it comes to Postgres SQL, if in case you want to use that as a vector database, because again when it comes to vector database, if you use quadrant PV8, if you're hosting that on server, hardly it will take $2.00 or $20 based on your server cost.
But if you're using quadrants cloud, probably you'll have to pay some amount for the quadrant cloud. But when it comes to PG vector, again, since it's running on your own server, it will be based on what server cost it cost. So mainly if you want to decide what vector database to choose.
It depends first on cost. Second, it depends on your search quality. If in case you're utilizing too much of memory quantization, those details memory quantization, it's not there in PG vector. PG vector is just base vector database.
So whatever HNSW you have, you can use HNSW. But apart from that I've not added anything else up till now. So if you want to use that you have something called as PG vector. So PG vector is mainly used for Postgres. So first thing what you need to do is you need to clone this repo. Once you clone the repo you just have to export the.
So this is for Mac but for Linux this line will be different. Is this clear? And then we have direct installation if in case you don't want to do clone and once that is done you have to start the server. Once the server is running then what you can do is you can open your PGR.

Oh, this is for Postgres. OK, so till here it is PG vector. So till here it is PG vector, you're installing it, you're adding it and once you do that next thing is you have to verify. So I'll just copy this PSQL Postgres.
And.
This is required like we need to have code RAM for this or good GPU for this. No. Why do you do GPU RAM? Yeah RAM RAM also you have GP, but if you're running in server, it's not gonna be in the memory for server if you want you can.
Use Oracle. Oracle.
So that can help.
Lifetime free server 24 GB.
Oracle has this offering of 24 GB RAM instance. OK, I think.
Are there any limitations for that? No limitation. I've not heard of it. So you just have to test this once. Once you're inside that version SQL that it's working fine. But there are two major things that you will face errors in. The first error is.
You'll find this vector extension to be empty. So basically when you are installing it, you just have to verify if you have vector extension available or not. So all this algorithms that you have HNSW it is inside this folder. So if you just copy this. Unless you have to verify if it is there or not.
So if you see you have the SQL file. So this is where all your vector vector extensions is available and if you want to again verify it, you just have to open this again PSQL. PSQLI have a user, so user is nothing but Tarun Jain and my database is Vector DB and now if I run.
The second comment DX vector.
It should show one row and why is it showing that one row? Because you have the HNSW enabled. So the only thing is these two lines you have to be very careful of. So when you're doing installation, first check if this is available or not. Second then what you can do is if this is not available.
For new vector database you can just do create vector extension. Then once this is done sorry this is sub bedding not a command. So create extension then create extension vector is the command and this is to list. That's it.
Probably this is the only area where you'll face errors. Yeah, this is the only area where you'll feel feel error. Apart from that, all these are very simple steps because I guess this is already installed in your.
And yeah, extension because this is what line chain or any other PG vector is using and once this is done that's it. And this command is also if in case you don't have PG

admin, this command will install PG admin.
OK, so after.
So in new version of PG-18, Yeah, I'll put this 80. OK, I guess I'm using the 16th version. Yeah, the old one.
So now when we come back to PG Admin, we have to create a new server. So when you create new server you will need your name.
Password and port number. Port number again is default 5400 something and when you do that 5432 is the port number, port, username, password and vector database name where you actually want to save your entire vectors. So these four variables if you have you have your connection string so that is your input.
So for quadrant VV 8 you have endpoint URL and you have API key. But when it comes to Postgres SQL you don't need those, you just need a connection string. Connection string along with your collection name. Collection name is default that will be there for your all the vector databases.
Which binary quantization? Like all the quantization techniques it's currently not available. MMR you can use because directly you're using it with Lang chain. So MMR is there with hybrid search you can use.
But The thing is specific sparse embedding models is not there like you you'll have to only use beyond 25 because beyond 25 is universal sparse models. If there is any new advancement, let's suppose anyone releases a better sparse models now and if you want to switch to that.
You have to check compatibility of what you call PostgreSQL, whether that supports that embeddings or not. If HRSW doesn't support those embeddings, it won't work. It will just do vector search. Even though it tells it will do hybrid search, it won't do it right. And now if we come to guest dot PY, it's the same thing.
The code is same so if you see it you have username, you have password hosted is localhost port 5432 vector database and this code is same. You have URLs, web-based URLs then load it, chunk it and this is what you need.
So this is your input variable when you are defining PG vector. So earlier what did we do? We define quadrant vector store. We have embeddings, we had collection name and here we had client instead of client. Now we have connection, that's it. And then just add documents.
So add documents only when you're in guest ticket. So when you do inference, let's suppose I create app dot PY now. Yeah, then I don't need to add here. What you need to do is just import this PG vector and what else do we need?

We need fast embed embeddings. Yeah, yeah, yeah. And here just define your PG vector. That's it.
And connection string will be there. Connection string only these three variables. That's it.
The search.
And now whatever user query you have, you can define your user query like.
I had something related to policy documents. Here you just have to do.
Vector store dot we have maximum marginal relevancy and then just pass your query query K equals to three. So this is one approach. If not what you can do, you just have to define your vector store.
Retriever equals to vector store.
dot as retriever. Here you can define your search type to be MMR then keyword arguments K to B3 then invoke. You can use this to approach but the major thing is this connection string. So instead of client you'll use connection string.
But apart from that, your entire logic of engaged dot PY is same. The results will be same as well. The results depends on the algorithms that you're seeing. OK, so HNSW everyone have different ways they have implemented.
So you have M if you remember I told how depth you're going. Yeah right. Many people have their own batching algorithm. So when it comes to quadrant they have their base they have implemented. PVA they have implemented different in majorly trial and error. In PG vector it's trial and error because they're not the main players of vector databases. So.
I'm not sure how well they're dedicated in this feature, but in terms of feature, I mean the number of features which VG Vector supports. Yeah, that's very limited in. OK, but if you just want to use it for your internality purpose, if you're not, you have the database already and you're receiving it for normal, we don't want to.
Mainly for cost. If in case you want to cut down the cost and if you still want rag during that time, since you already have VG vector, you are using it. So this is not additional dependencies, it's just you are reusing it. But this is the only change remaining everything is same.
You just have to comment this line when you when you do the inference.
OK this I'll push the code along with the README file. I do have this README file available and the last thing is this hybrid search for Lama index. Lama index we quickly saw how you can build a rack solution so you have.
Directory reader. Then you have storage context. So what does storage context take?

It will take your vector store as a input variable. Once you have storage context, next to have vector index. So vector store index will have your node. Now what is node in lambda index?
Notes.
Just chunks. So in chunk yeah, then you will have your storage context. That's it. Now when you look at storage context, embedding is default, right? So what do you need to do? You need to use settings, settings dot, embed model, whatever embed model you need.
And service engine which is your query engine, query engine as LLM by default which is open AI AI. So if you want to change it you have to use settings dot LLM equals to whatever LLM you read.
So now what we'll do is when you define your vector store, before you give it to your storage context, you have to enable hybrid search. So when do we apply reranking? Before indexing or after indexing? After indexing. So what will happen after indexing? You'll have 10 documents which are retrieved.
Out of this 10 now what you need to do is you need to cut short with five which are the most 5 relevant which have to give to the user. So there are two ways you can use re ranking. One is using open source model, one is using close source. For open source you have cross encoders.
You have flash rerank. When it comes to close source, you have coyer, which is the best player when it comes to relanking.
And this will happen post the indexing. So post indexing in lumb index is nothing but your query engine. So when you define your query engine you are inferencing it. So you define your storage context along with storage context you have to define. Preprocessor, which is your node. That node is nothing but your core here. So we'll try this out. I'll share this notebook. So the remaining code is same. Hardly when it comes to Lama index, you will only change five or six lines of code. That's the most. But if you're building your own UI, if you're customizing it, which is very rare cases, in that rare cases also you'll have cookbooks. So Alarm Index have too many cookbooks, but the only issue with it is it's not updated, so sometimes you might face error.
If you get error then it takes time in Laman text. They do have cookbooks. They have very very cookbooks than Lang chain. But the only thing is these two players are like rapidly they build features. After two days you'll find a new version. Now that new version might not support older version.

So those cookbooks are not updated. That's the only issue with alarm index.
OK, so the steps are same from LAMA index we have to install it. Then you have quadrant, then you have fast and red. So import statements is same. We are not sending an import statement. Then why are we using this? Since it doesn't support us ensure in collab we are using it.
And LAMB index typical users essential when you are doing query query then we are defining that what you call Google Gemini API keys and we have to operate the settings. So now if you look at the settings we are changing the embed model by fast embed embeddings which is taking Gino model.
Then LLM is your Gemini 2.5 and document loading is same. Till here we don't have any new line of code. The only new line of code is when you are defining the vector store. So when you define vector store basically you have two kind of embeddings, one is dense, one is parse.
Dense by default you have your storage context as embedding model. That embedding model is your Gina. Gina is a dense embedding model. So now if you need sparse, you just have to define fast embed sparse model equals to quarter BN 25.
So again, if you see you don't have to define create collection, then define dense, then define sparse that is already defined in Lama index. So Lama index are like how much lines of code you write. We'll make sure we concise it and then you have.
A new parameter called enable library to be true. So you also have to ensure when you're doing inference this is enabled as true.
So these are the two new parameters. Why batch we have added. Mostly when you're using both dense and sparse, it might take too much of time. So even if you add batches, the time will be same. But the only thing is you will not hit the RAM at Max, so there will be some delay after every single batch.
So then you define vector store. Once you have vector store by default storage context is in memory. We are appending the from defaults to quadrant which is in memory to quadrant and then you are defining the indexing which happened only once.
Where you're defining the documents. Once the indexing is done, that's it. Your data is saved. We are starting with inference. So inference you define client collection name and hybrid which is enable hybrid to be true. And I'm not defining embedding because embedding is by default in lum index.
So now vector store index from vector store not from documents because data is

saved. So now we are saving your inference vector store. Then when you define query engine just define these two parameters. One is your similarity top K then sparse top K.
Then you are vector store query mode. So in line chain what did we had? We had retriever mode to be hybrid. So did you observe that? So if I open line chain quadrant.
You have something called as retriever mode. On the left hand side you just have to click on hybrid vector search and here if you see you have a retriever mode and when you're defining your vector store you have to define your retriever mode to be. Hybrid. So in love index what are you doing? You're just waiting this as enable hybrid to be true, but the keyword is same.
Yeah, right. So yeah, embedding is not there. Emedding is default when it comes to lum index. The only thing is parse embedding model is not defined. So you have to define this parse embedding model.
Dense we don't have any option that we have to that's it's already there no. Yeah like here. So dense will be in your settings dot embed model. Embed model is your dense nice sparse is nothing but your first this thing. So now what if you're using mill voice? What if you're using what do you call?
VB8 they already have beyond 25, so quadrant has fast embed separately because you want embedding model also to be very fast. So they build fast embed separately so that the entire ecosystem that they're creating it is fast enough for your inference. Bill OS have BM 25 within their what you call their vector database so you can directly use it. What is BM 25? I'm not sure if they support BM 42. I have not seen any application with BM 42 but BM 25 it is common Uber it's there and there is one more.
More companies uses VM 25, I guess booking.com. They also uses VM 25.
The syntax is same. You're defining your inference, then you're defining query engine. Ask any input from your database. Now how do you get the response? Response dot response till here. What you're trying to do is you're just working with hybrid search.
Now how do we do reranking? Whatever query engine is there, reranking happens post indexing. So for that you have something called as post preprocessor. So from Llama index install post preprocessor coyere rerank.
Then import it and what was the model name we used last time? It was the Rerank English V30. So this is the model name which Kuya provides, which is the latest one.

You need to add API key. I remember last time I shared my API we didn't create new accounts.
And once you do that, same syntax. The only thing is whenever you're using reranking, make sure the K value is more.
So if you see a similarity top K before it was 4, now I increased it with six. You can keep this as 10 as well. You can keep this 10 then your top end can be 4. So now how much is your K value which is going to LLM 4? It is 4. Why?
Those are your top results which are defining in coyer. This syntax is same. You just define your coyer relank. When you're doing your query engine, you're adding this new model. That's it. Oh yeah, this new line. And then this syntax again query engine dot query.
So now we just three lines of additional lines. You have both. I did search and re ranking so if you want to change.
Courier rerank to cross encoders. You can use cross encoders. So you have cross encoders, you have flash rerank and you have courier, but cross encoders will be very slow. But if you are fine with that latency then it's fine.
That's it.
And very rarely reranking will improve your performance. Very rarely you will just have .2 difference, .3 difference, not more than that. We can use confidence for confidence. Yeah, the confidence course of reranking will be useful, but I hit most of the times you're just reranking it based on.
The documents that you already have, right? So reranking was good when the limited model context length was there. Now all models have good too much of context length, right? So the concept of reranking was good enough when you had limited LLM context. Now there is not an issue even if you pass K equals to 10.
It won't sum the LLM performance. Yeah, until if K value of every chunk is more than 8000 in that cases then you will need re ranking. We should rather we have some smaller like coding use cases.
But coding use cases, you'll use grep. Yeah, grep in the sentence directly to get you the context. Better get the grep grep with EST. So nowadays you don't need relunking tools. Huh. Where tools you said grep is a tool grep log, huh?
That's it. So tomorrow probably we'll directly jump to agents tomorrow. Oh, yeah, sorry. I mean, Monday. Yeah, no worry.
How much registration do we have for workshop? Registration for workshop, like chairs coming 400 people above workshop. It's very hard to manage the audience.

Oh, if it is 300-400, no, no, it's working.
Working in works. No, no, no, no. I usually do ask. No, no, no, 1:50. I have two hours. I usually I until they don't get output, I won't. And the session is last only, so don't worry. It's just tea after you. Yeah, I just hope they're not OK. Even if they're sleepy, I have swags.
That's why you are last. OK, you won't let them sleep. I know I brought a additional specs as well for those who answered. Yeah, not because it's in the the luggage.
Matt, Darvin and Karan, he's asleep.
Yeah.
So the we can connect again for once you have the flow chart, I'll create the flow chart and then confirm it with you, yeah.

◉ **Tirth** stopped transcription