RAG Meets AI Agents: Unlocking Agentic RAG with PydanticAI!

YourTechBud Codes
5.48K subscribers
Subscribe

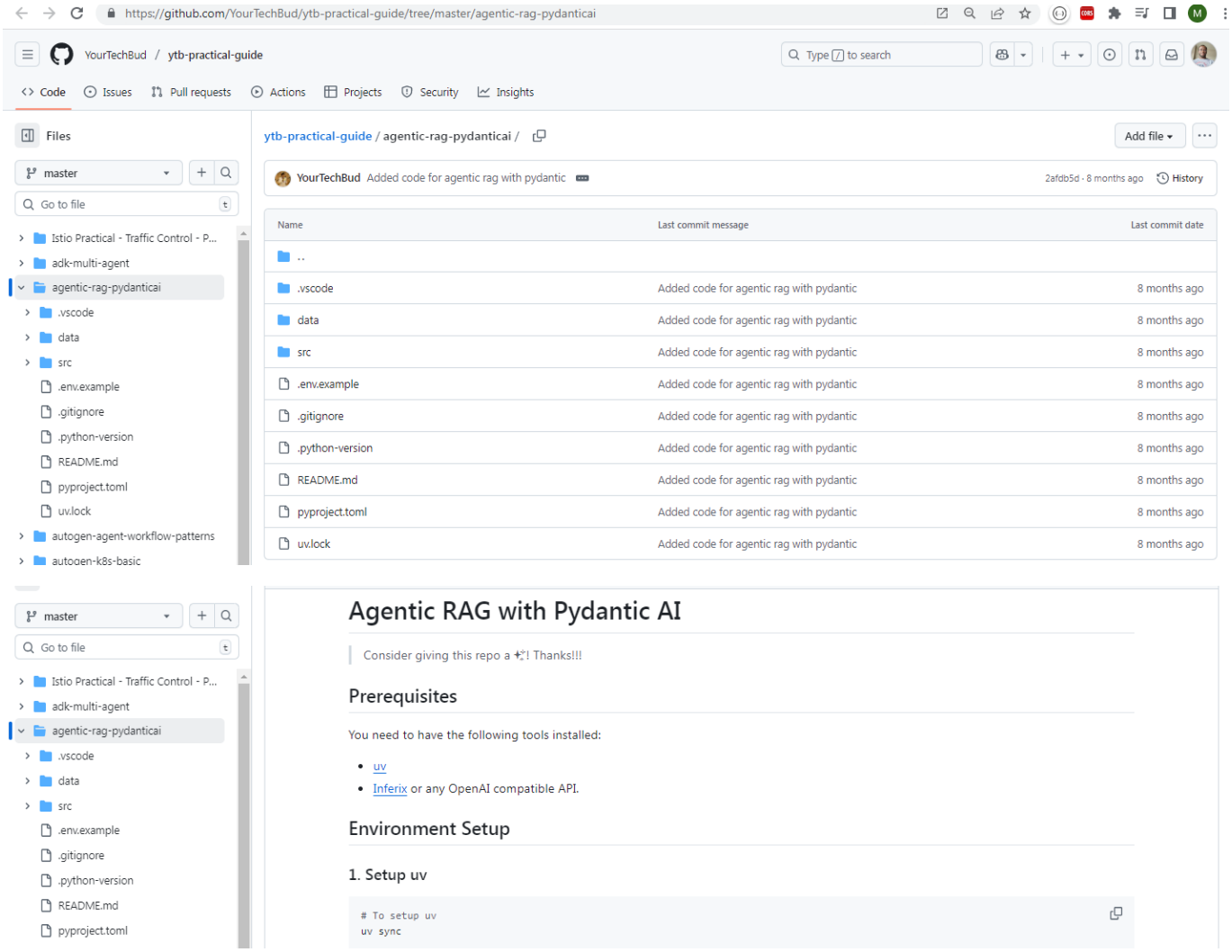
77 Share Ask Download ...

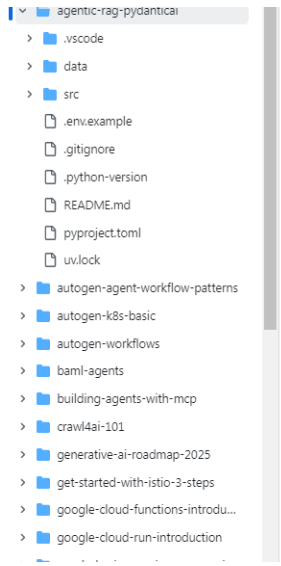
2,558 views Apr 26, 2025 #AI #GenerativeAI #GenAI
We're combining the magic of Retrieval-Augmented Generation (RAG) with the adaptability of agent-based AI, powered by PydanticAI. Whether you're an AI enthusiast or a pro developer, this session is packed with mind-blowing insights and real-world value! 🥊

Here's the Agenda:
1️⃣ Spot the Limitations of RAG: Not every problem can be solved with standard RAG pipelines. Learn how to pinpoint those challenging use cases where a little "agentic thinking" can save the day.
2️⃣ Discover Agentic RAG Awesomeness: Find out how combining RAG with agentic principles opens up a world of possibilities. Trust us, it's a game-changer!
3️⃣ Real-World Action: Dive into a live use case and watch Agentic RAG in action as we push its boundaries in a practical scenario.

References:
1] PydanticAI Documentation: https://ai.pydantic.dev/
2] Code: https://github.com/YourTechBud/ytb-pr...

Chapters:
00:00 - Introduction
00:17- A RAG Recap
03:44 - Where naive RAG falls short?
06:01 - Improving our RAG pipeline
11:12 - Why use agents for RAG?
13:55 - Using PydanticAI for RAG?
31:03 - Building our agents
35:39 - The final agentic RAG pipeline
39:13 - Seeing it in action
41:47 - Making it even harder





2. Install Inferix (OpenAI compatible API)

> Note: You can use any OpenAI compatible backend or simply use OpenAI itself.

- Feel free to use any OpenAI compatible API.
- Make sure you have Ollama installed.
- Make sure you have pulled a model. I recommend Qwen 2.5 32B.
- Use this guide to setup inferix to host a OpenAI compatible API capable of function calling.

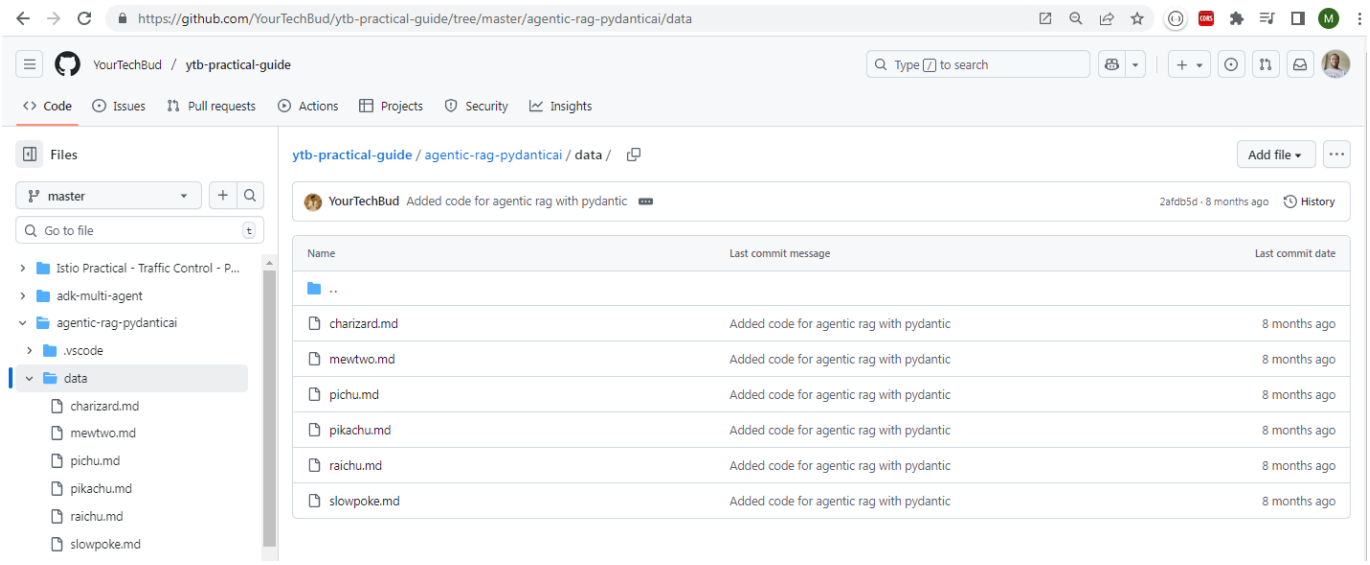
3. Set the environment variables

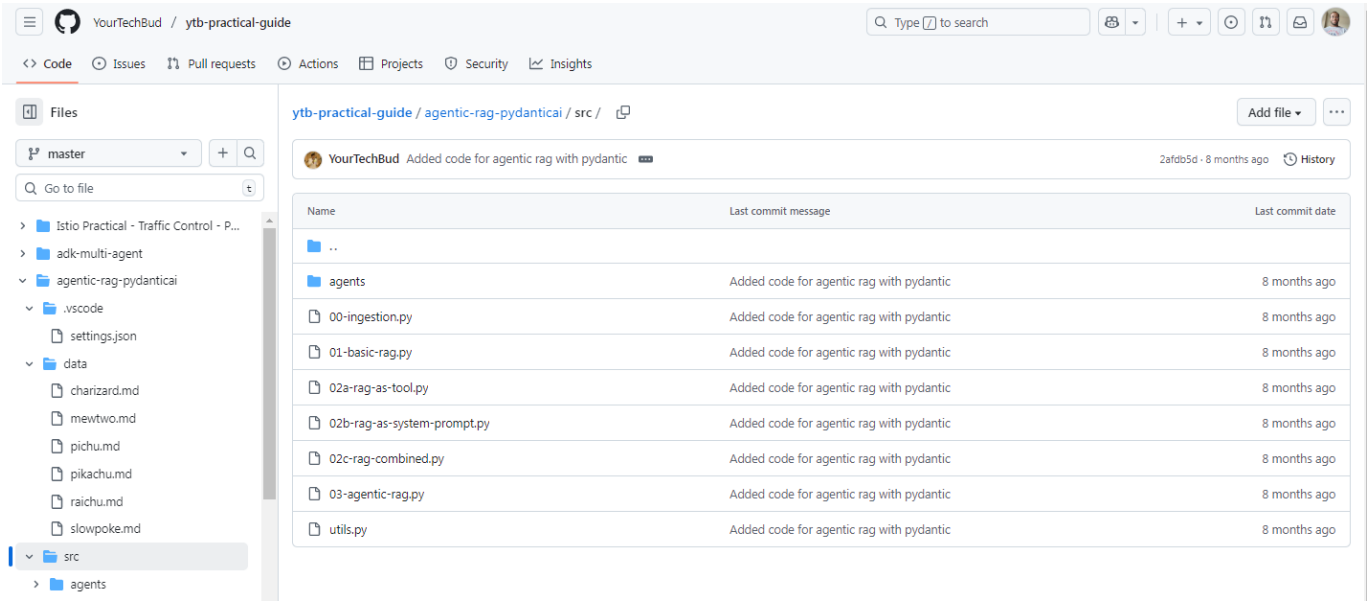
- Checkout the example dotenv file at `.env.example`.
- Create a new `.env` using the example one as a template.
- Replace the variables as necessary.

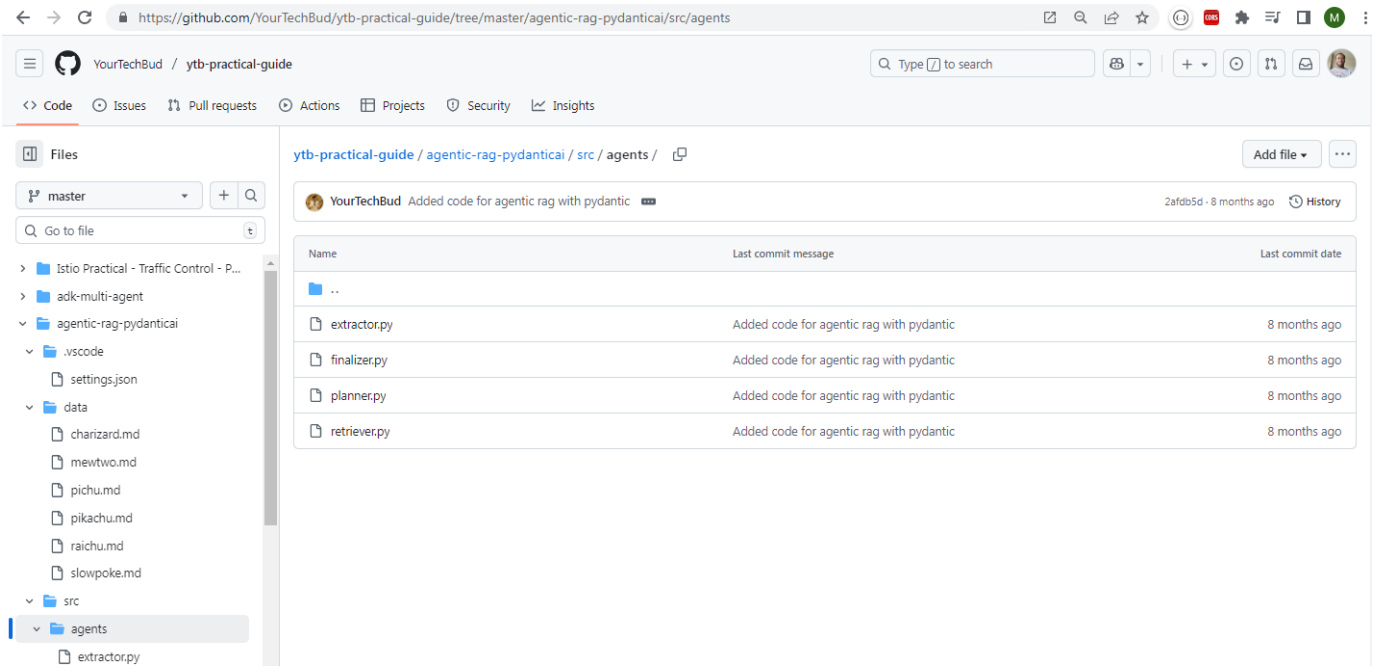
Run the app

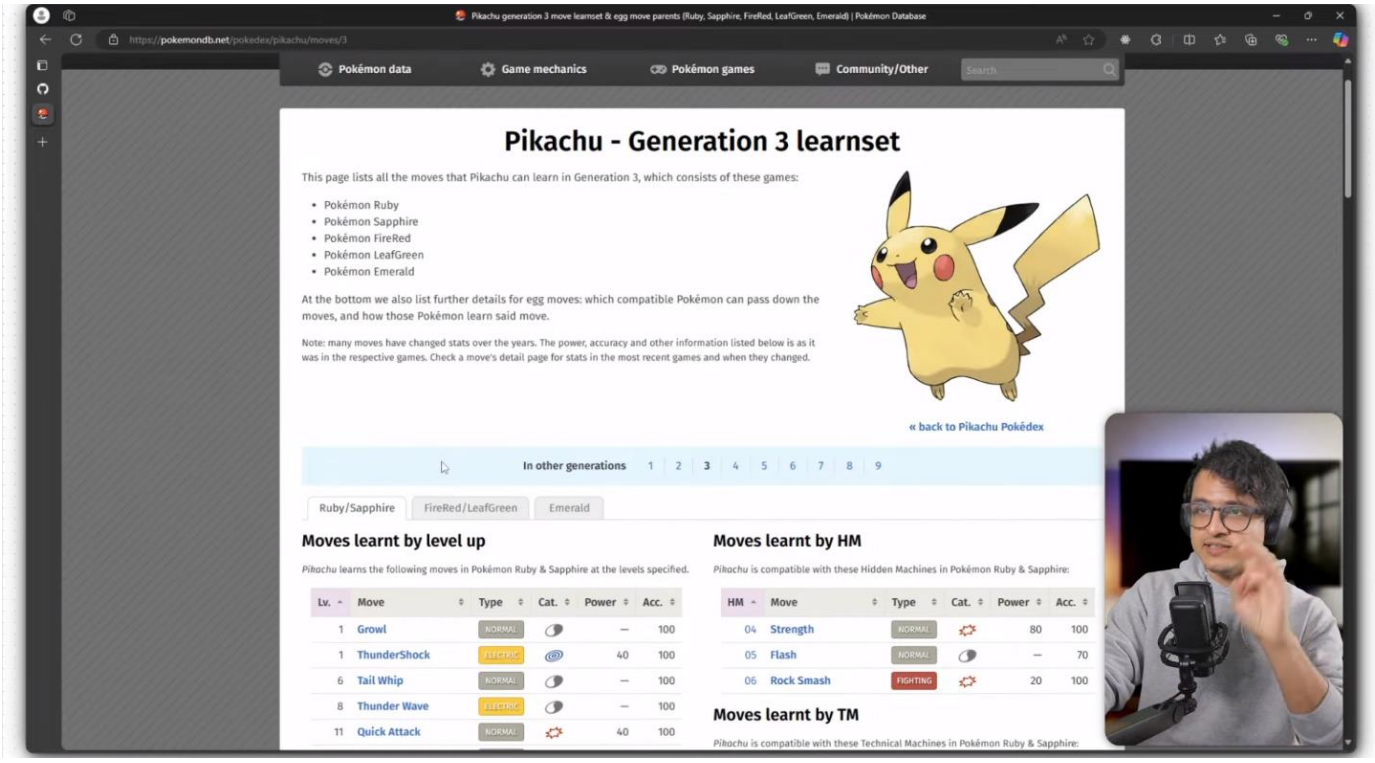
We've got a few scripts in this project:

1. `uv run src/00_humble_call.py` - Runs the ingestion script to create the embeddings and full text search index.
2. `uv run src/01-basic-rag.py` - Demonstrates a basic RAG workflow.
3. `uv run src/02a-rag-as-tool.py` - Demonstrates how you can agents can perform RAG by means of a tool call.
4. `uv run src/02b-rag-as-system-prompt.py` - Demonstrates how you can use Pydantic AI's system prompt hook to perform RAG.
5. `uv run src/02c-rag-combined.py` - Demonstrates how you can combine the two approaches to perform RAG.
6. `uv run src/03-agentic-rag.py` - Demonstrates how you can use Pydantic AI's agentic workflow to perform RAG.

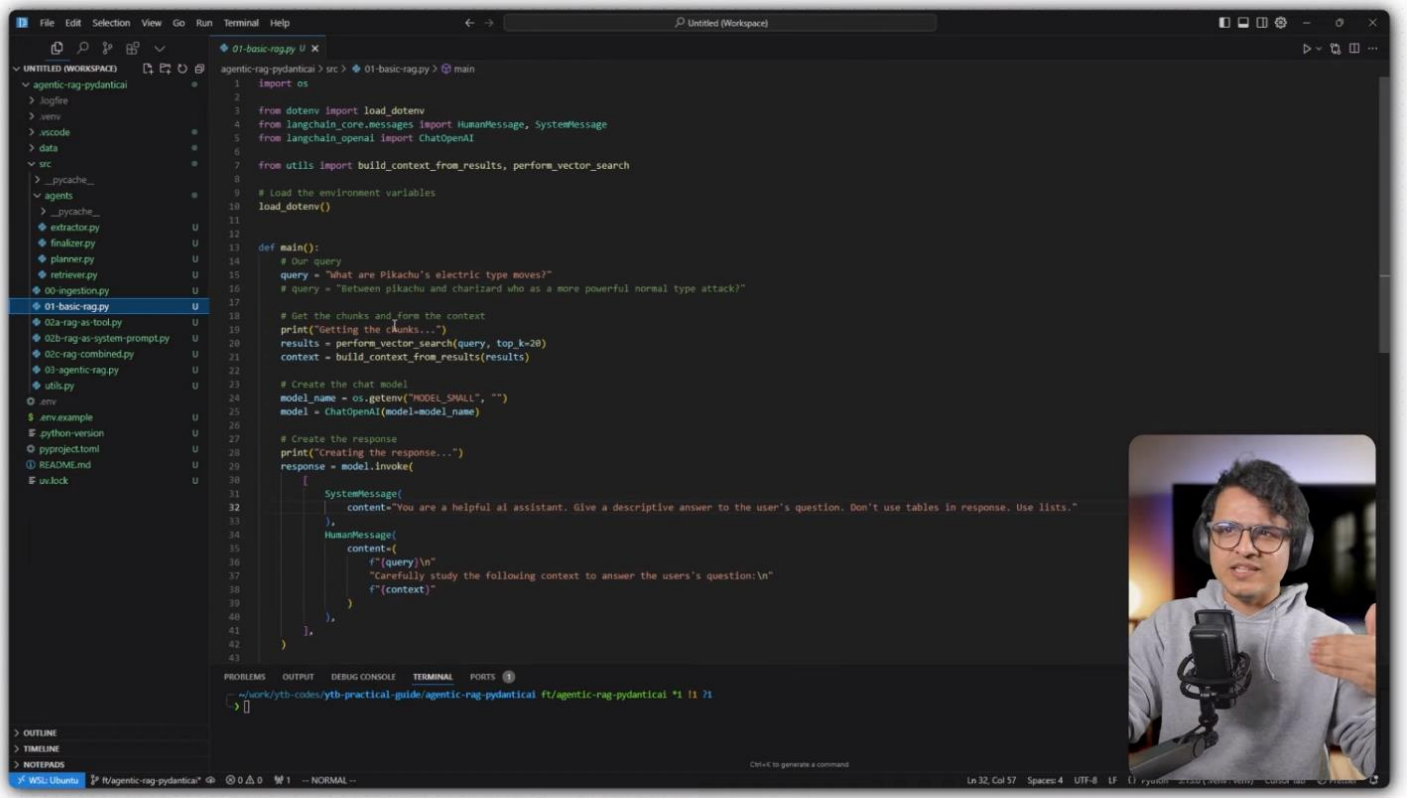


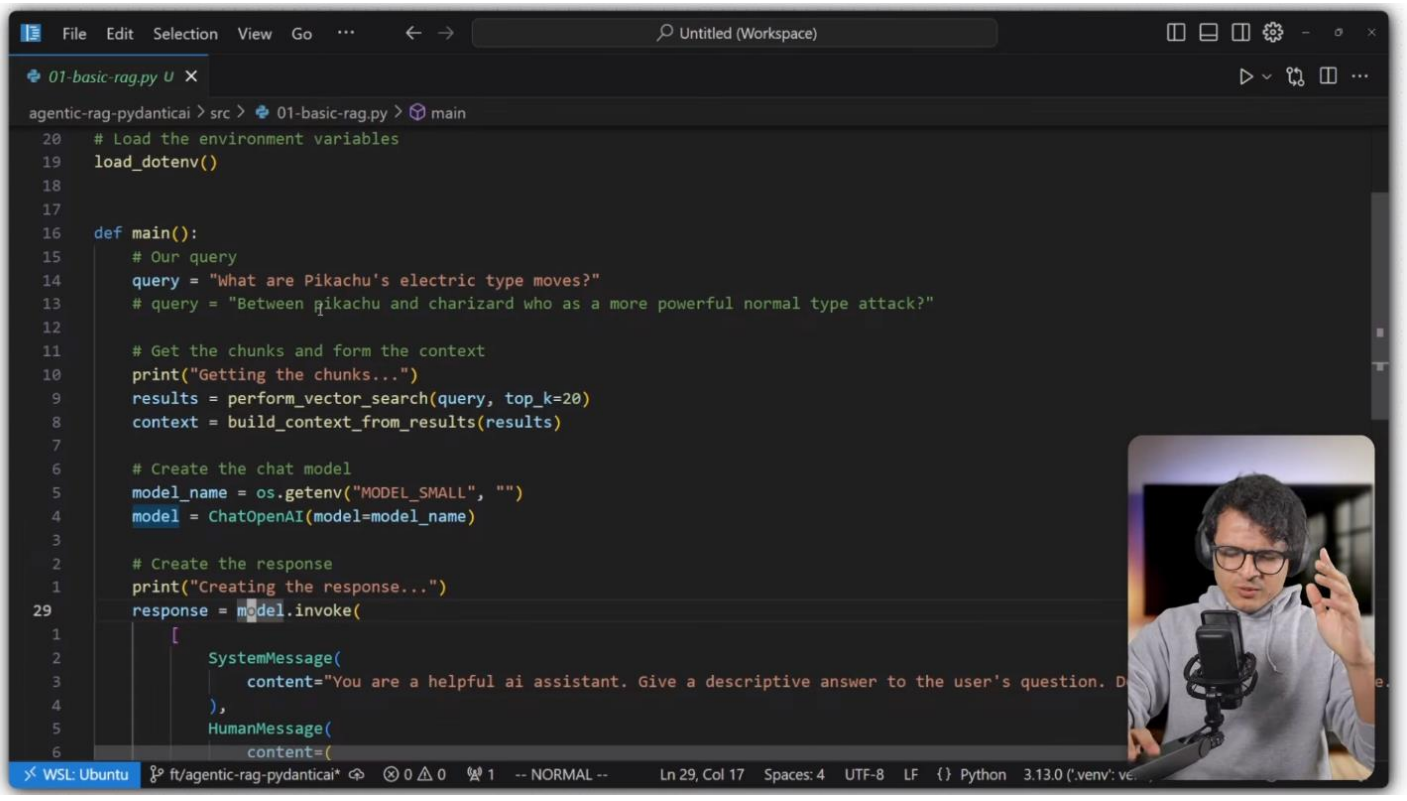


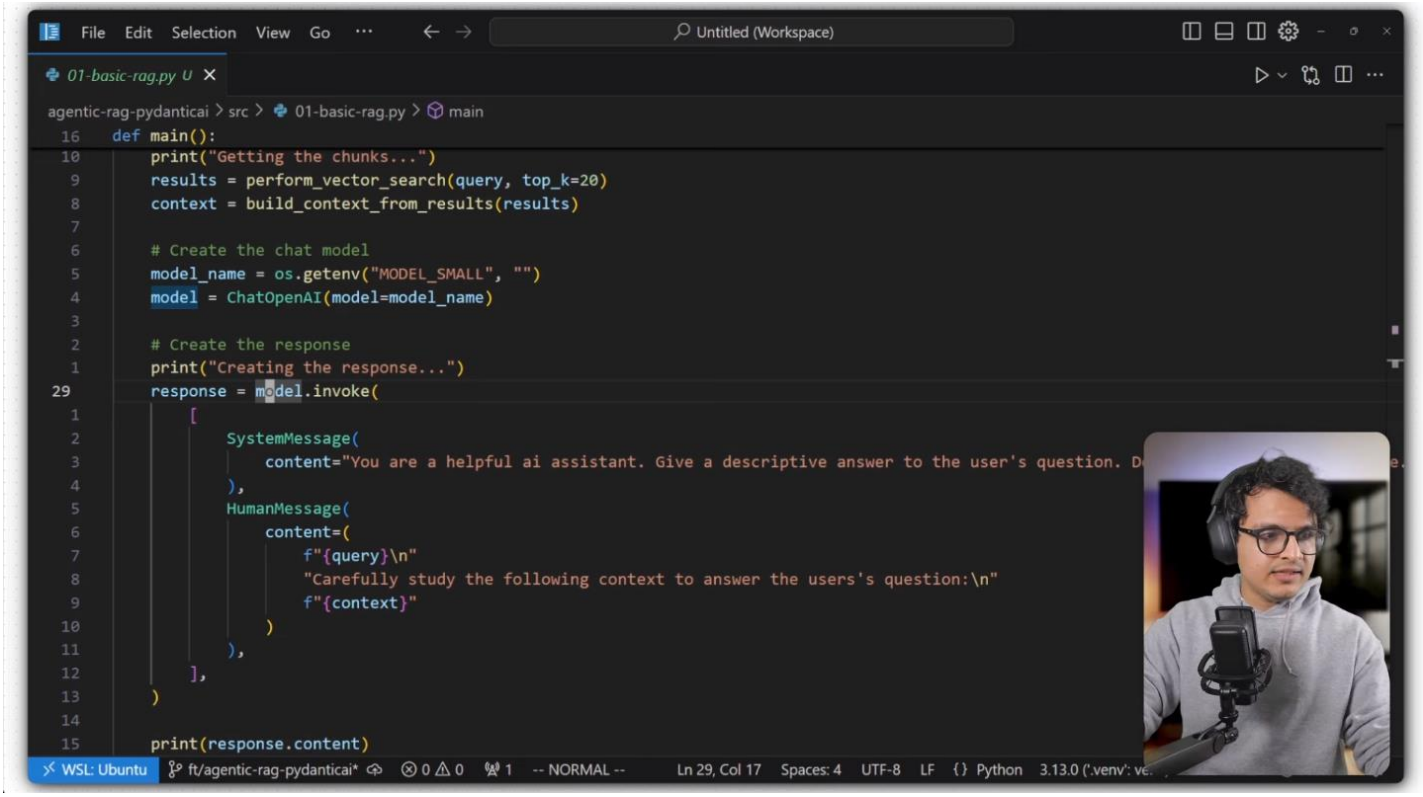


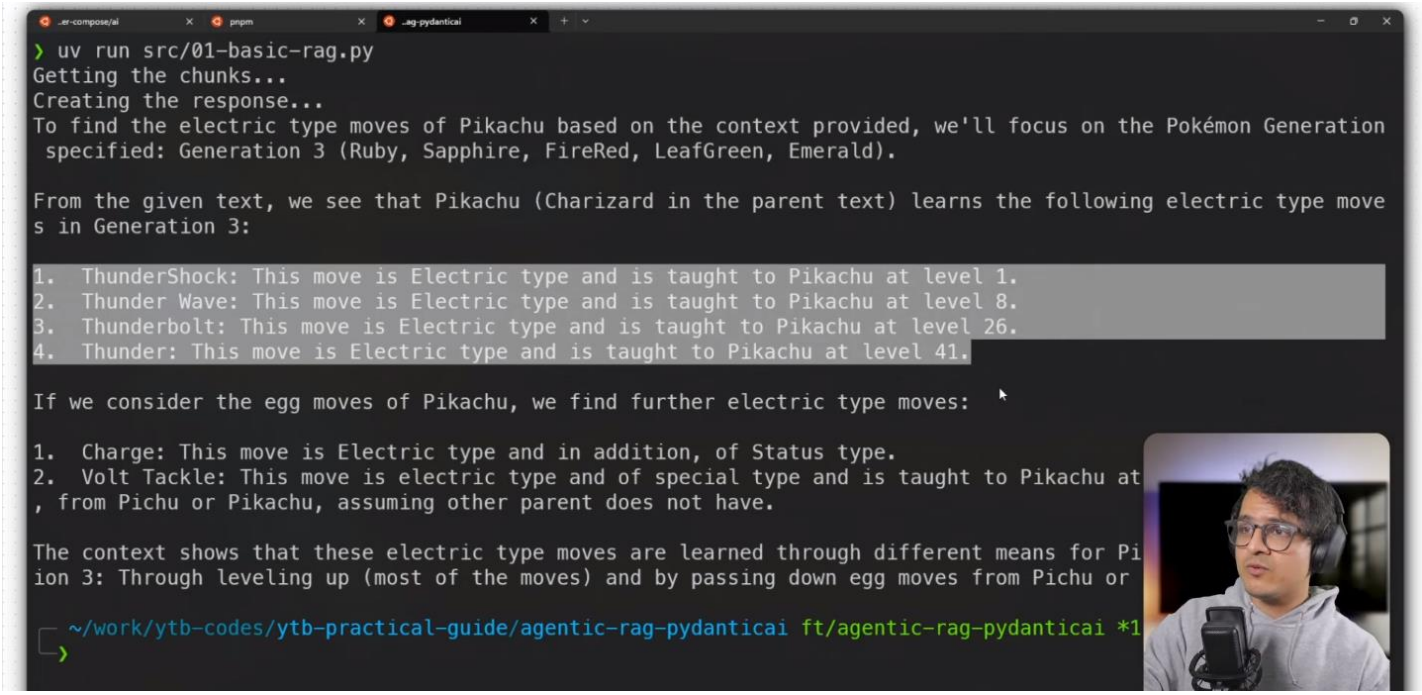


This is our dataset in markdown format

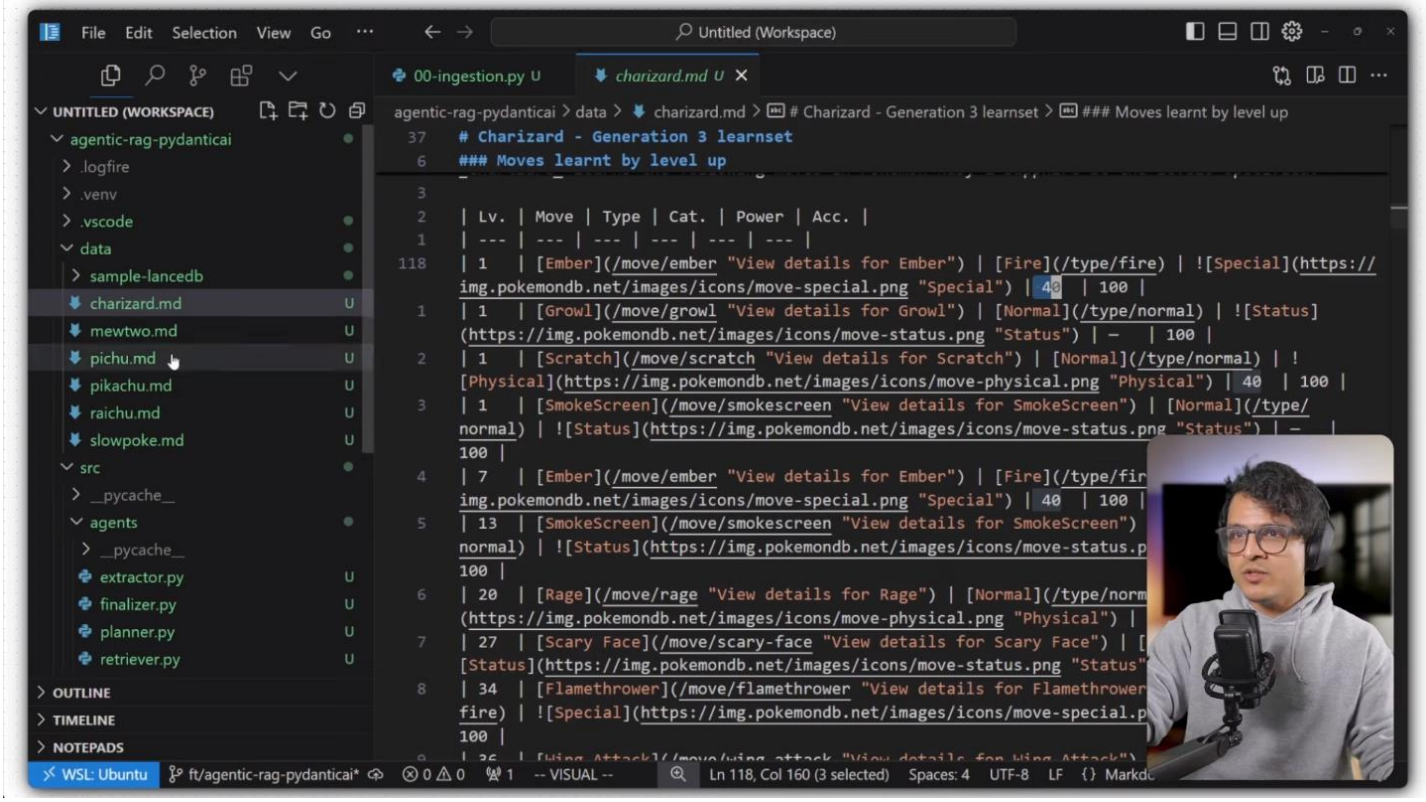

File Edit Selection View Go Run Terminal Help
Untitled (Workspace)
01-basic-rag.py U
agentic-rag-pydanticai > src > 01-basic-rag.py > main
UNTITLED (WORKSPACE)
agentic-rag-pydanticai
.logfire
.venv
.vscode
data
src
__pycache__
agents
__pycache__
extractor.py
finalizer.py
planner.py
retriever.py
00-ingestion.py
01-basic-rag.py
02a-rag-as-tool.py
02b-rag-as-system-prompt.py
02c-rag-combined.py
03-agentic-rag.py
utils.py
.env
.env.example
.python-version
pyproject.toml
README.md
uv.lock
import os
from dotenv import load_dotenv
from langchain_core.messages import HumanMessage, SystemMessage
from langchain_openai import ChatOpenAI
from utils import build_context_from_results, perform_vector_search
# Load the environment variables
load_dotenv()
def main():
# Our query
query = "What are Pikachu's electric type moves?"
# query = "Between pikachu and charizard who as a more powerful normal type attack?"
# Get the chunks and form the context
print("Getting the chunks...")
results = perform_vector_search(query, top_k=20)
context = build_context_from_results(results)
# Create the chat model
model_name = os.getenv("MODEL_SMALL", "")
model = ChatOpenAI(model=model_name)
# Create the response
print("Creating the response...")
response = model.invoke(
[
SystemMessage(
content="You are a helpful ai assistant. Give a descriptive answer to the user's question. Don't use tables in response. Use lists."
),
HumanMessage(
content=(
f"{query}\n"
"Carefully study the following context to answer the users's question:\n"
f"{context}"
)
),
],
)
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
~/work/yth-codes/ytb-practical-guide/agentic-rag-pydanticai ft/agentic-rag-pydanticai *1 !1 ?1
OUTLINE
TIMELINE
NOTEPADS
WSL: Ubuntu
ft/agentic-rag-pydanticai*
-- NORMAL --
Ln 32, Col 57
Spaces: 4
UTF-8
LF

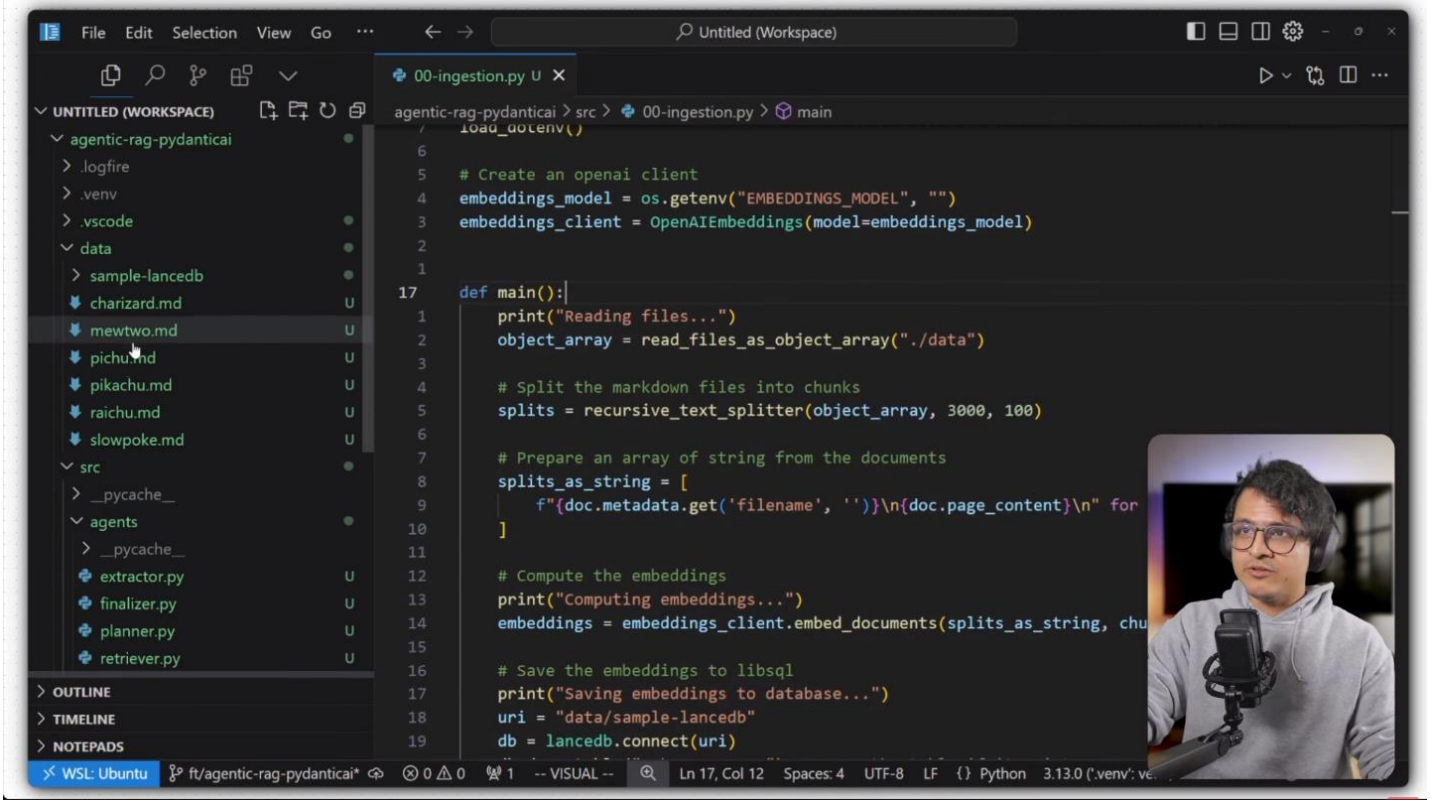

File Edit Selection View Go
Untitled (Workspace)
01-basic-rag.py U
agentic-rag-pydanticai > src > 01-basic-rag.py > main
# Load the environment variables
load_dotenv()
def main():
# Our query
query = "What are Pikachu's electric type moves?"
# query = "Between pikachu and charizard who as a more powerful normal type attack?"
# Get the chunks and form the context
print("Getting the chunks...")
results = perform_vector_search(query, top_k=20)
context = build_context_from_results(results)
# Create the chat model
model_name = os.getenv("MODEL_SMALL", "")
model = ChatOpenAI(model=model_name)
# Create the response
print("Creating the response...")
response = model.invoke(
[
SystemMessage(
content="You are a helpful ai assistant. Give a descriptive answer to the user's question. D
),
HumanMessage(
content=(
WSL: Ubuntu
ft/agentic-rag-pydanticai*
-- NORMAL --
Ln 29, Col 17
Spaces: 4
UTF-8
LF
Python
3.13.0 ('.venv': ve

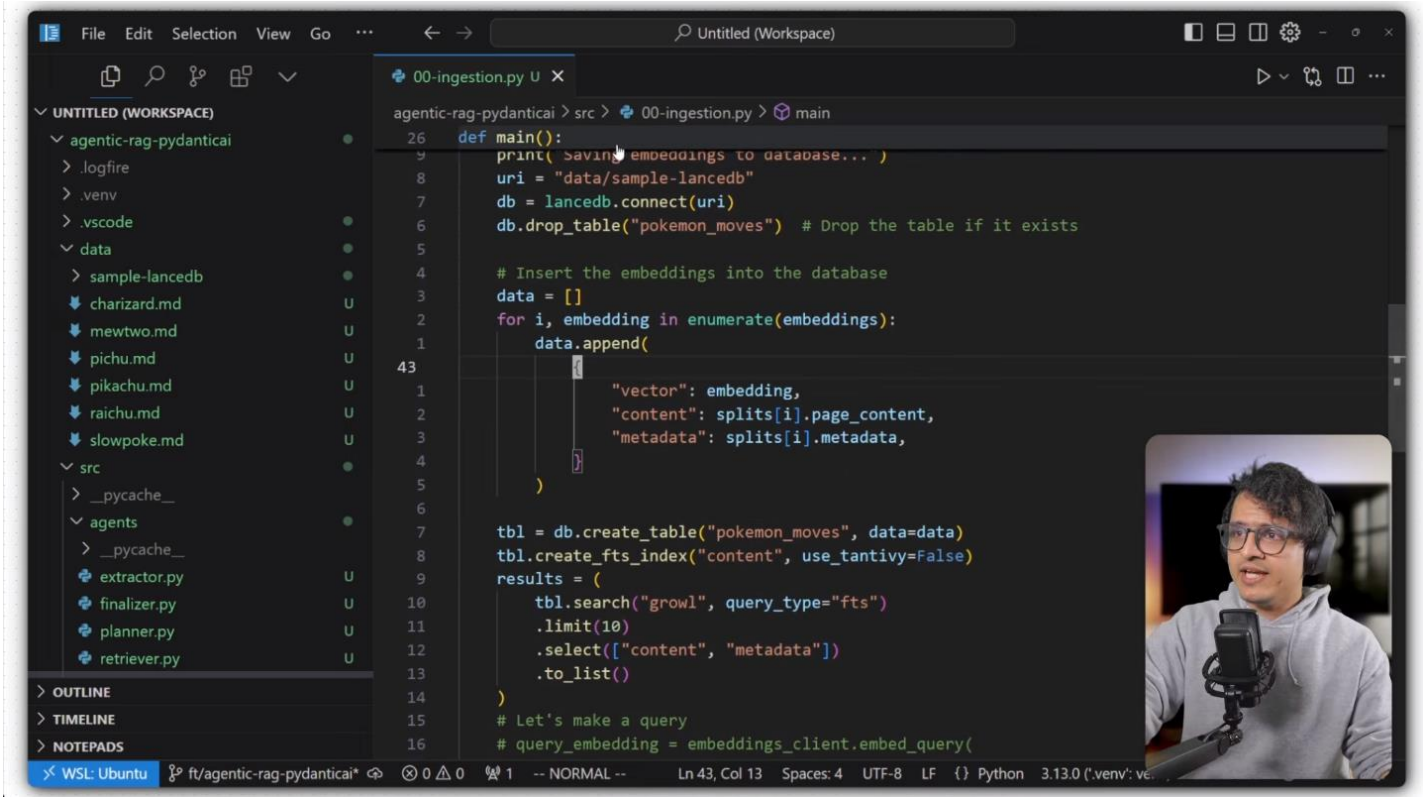


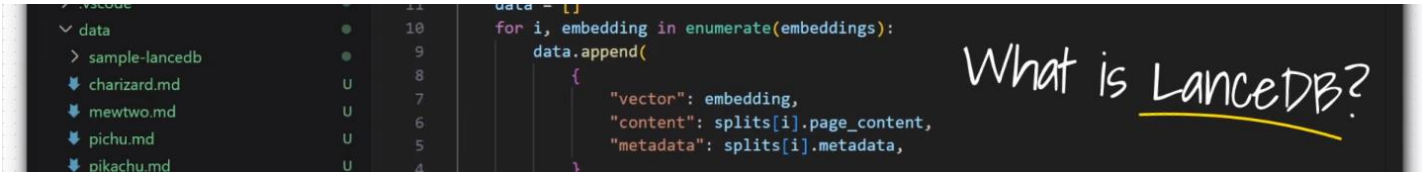


This is now an answer that is grounded in real document facts!

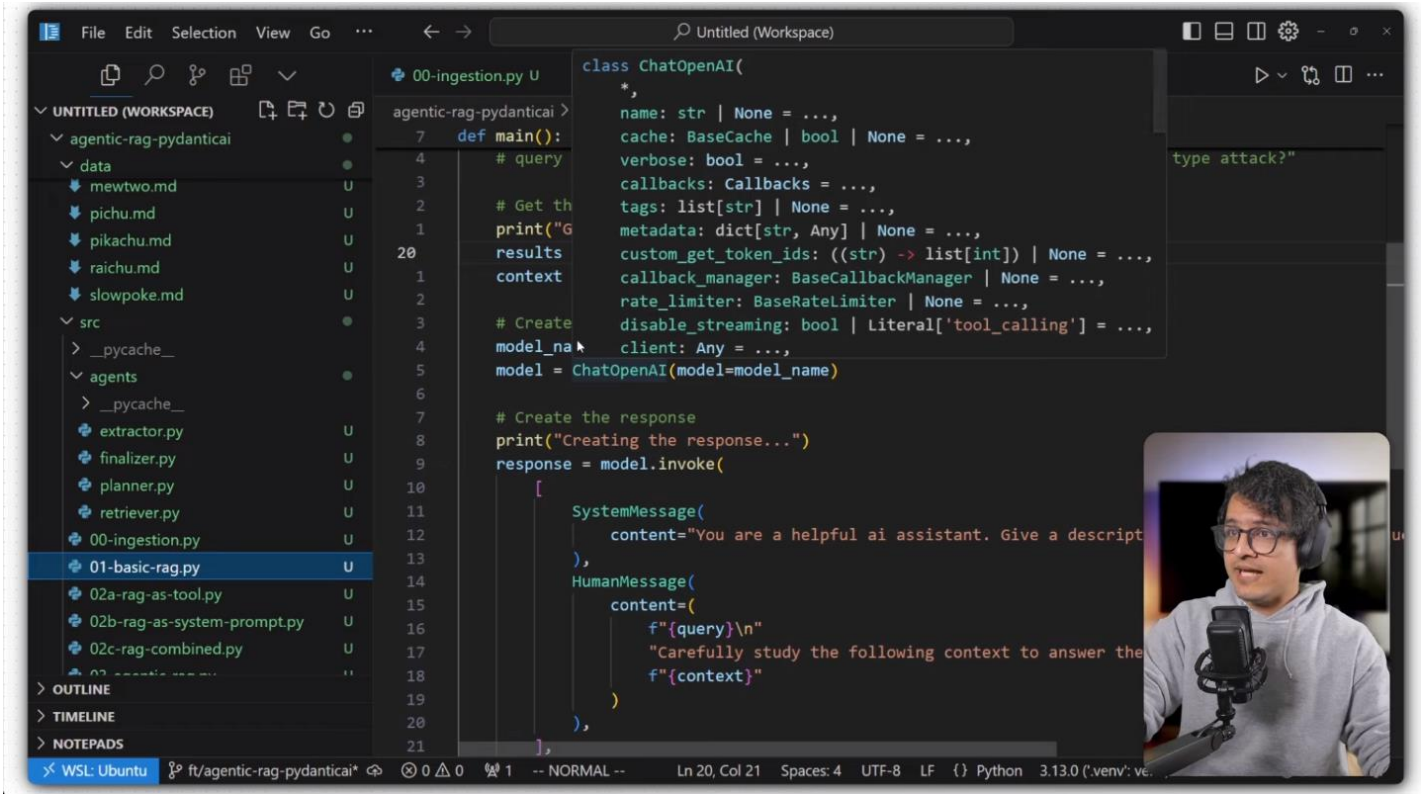

File Edit Selection View Go
Untitled (Workspace)
00-ingestion.py U
charizard.md U
agentic-rag-pydanticai > data > charizard.md > # Charizard - Generation 3 learnset > ### Moves learnt by level up
UNTITLED (WORKSPACE)
agentic-rag-pydanticai
.logfire
.venv
.vscode
data
sample-lancedb
charizard.md
mewtwo.md
pichu.md
pikachu.md
raichu.md
slowpoke.md
src
__pycache__
agents
__pycache__
extractor.py
finalizer.py
planner.py
retriever.py
OUTLINE
TIMELINE
NOTEPADS
Charizard - Generation 3 learnset
Moves learnt by level up
Lv.	Move	Type	Cat.	Power	Acc.
1	[Ember](/move/ember "View details for Ember")	[Fire](/type/fire)	![Special](https://img.pokemondb.net/images/icons/move-special.png "Special")	40	100
1	[Growl](/move/growl "View details for Growl")	[Normal](/type/normal)	![Status](https://img.pokemondb.net/images/icons/move-status.png "Status")	–	100
1	[Scratch](/move/scratch "View details for Scratch")	[Normal](/type/normal)	![Physical](https://img.pokemondb.net/images/icons/move-physical.png "Physical")	40	100
1	[SmokeScreen](/move/smokescreen "View details for SmokeScreen")	[Normal](/type/normal)	![Status](https://img.pokemondb.net/images/icons/move-status.png "Status")	–	100
7	[Ember](/move/ember "View details for Ember")	[Fire](/type/fire)	![Special](https://img.pokemondb.net/images/icons/move-special.png "Special")	40	100
13	[SmokeScreen](/move/smokescreen "View details for SmokeScreen")	[Normal](/type/normal)	![Status](https://img.pokemondb.net/images/icons/move-status.png		100
20	[Rage](/move/rage "View details for Rage")	[Normal](/type/normal)	(https://img.pokemondb.net/images/icons/move-physical.png "Physical")		
27	[Scary Face](/move/scary-face "View details for Scary Face")	[Status](https://img.pokemondb.net/images/icons/move-status.png "Status"			
34	[Flamethrower](/move/flamethrower "View details for Flamethrower	fire)	![Special](https://img.pokemondb.net/images/icons/move-special.p		100
WSL: Ubuntu ft/agentic-rag-pydanticai* 0 0 1 -- VISUAL -- Ln 118, Col 160 (3 selected) Spaces: 4 UTF-8 LF {} Markd

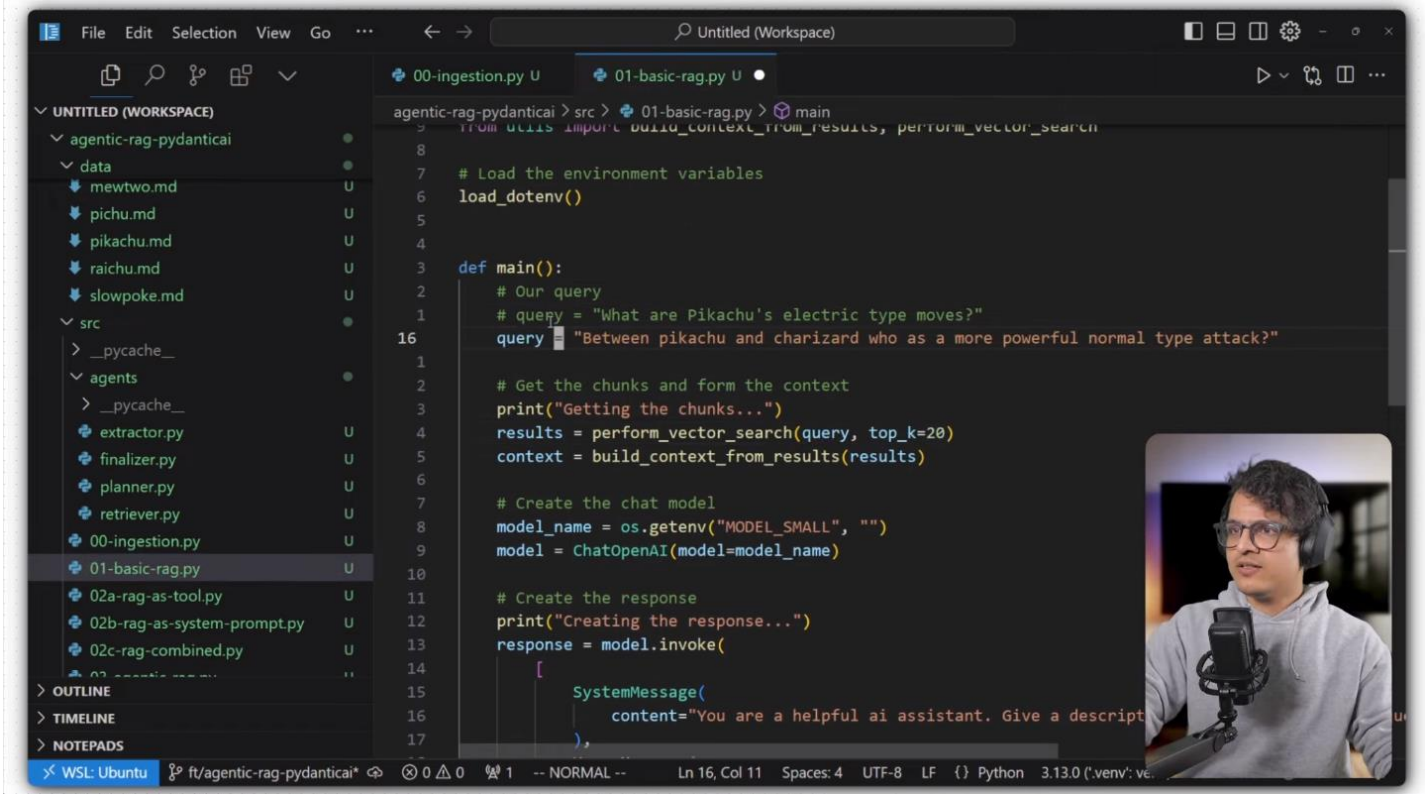

File Edit Selection View Go
Untitled (Workspace)
00-ingestion.py U
agentic-rag-pydanticai > src > 00-ingestion.py > main
UNTITLED (WORKSPACE)
agentic-rag-pydanticai
.logfire
.venv
.vscode
data
sample-lancedb
charizard.md
mewtwo.md
pichu.md
pikachu.md
raichu.md
slowpoke.md
src
__pycache__
agents
__pycache__
extractor.py
finalizer.py
planner.py
retriever.py
OUTLINE
TIMELINE
NOTEPADS
load_dotenv()
# Create an openai client
embeddings_model = os.getenv("EMBEDDINGS_MODEL", "")
embeddings_client = OpenAIEmbeddings(model=embeddings_model)
def main():
print("Reading files...")
object_array = read_files_as_object_array("./data")
# Split the markdown files into chunks
splits = recursive_text_splitter(object_array, 3000, 100)
# Prepare an array of string from the documents
splits_as_string = [
f"{doc.metadata.get('filename', '')}\n{doc.page_content}\n" for
]
# Compute the embeddings
print("Computing embeddings...")
embeddings = embeddings_client.embed_documents(splits_as_string, chu
# Save the embeddings to libsql
print("Saving embeddings to database...")
uri = "data/sample-lancedb"
db = lancedb.connect(uri)
WSL: Ubuntu ft/agentic-rag-pydanticai* 0 0 1 -- VISUAL -- Ln 17, Col 12 Spaces: 4 UTF-8 LF {} Python 3.13.0 ('.venv': ve



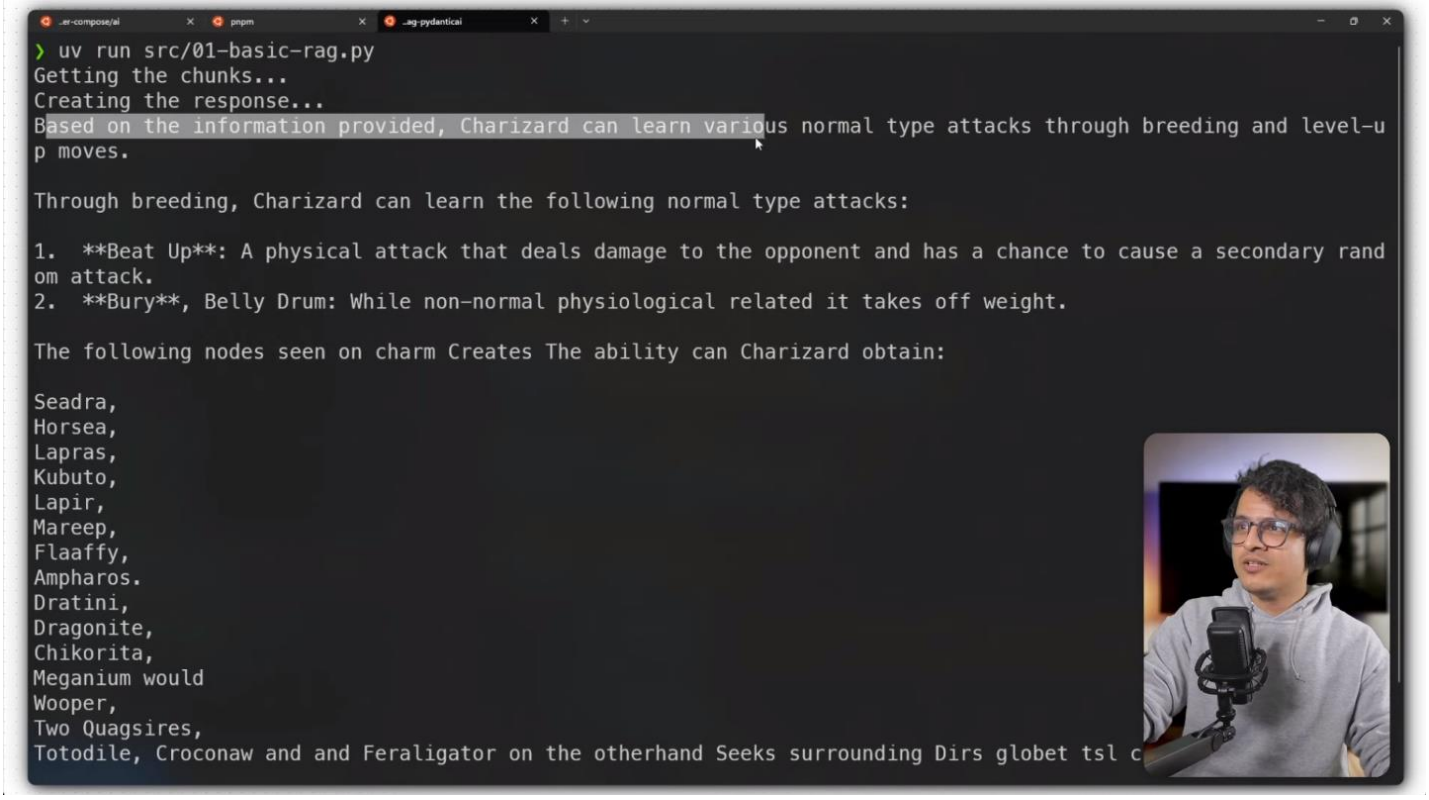


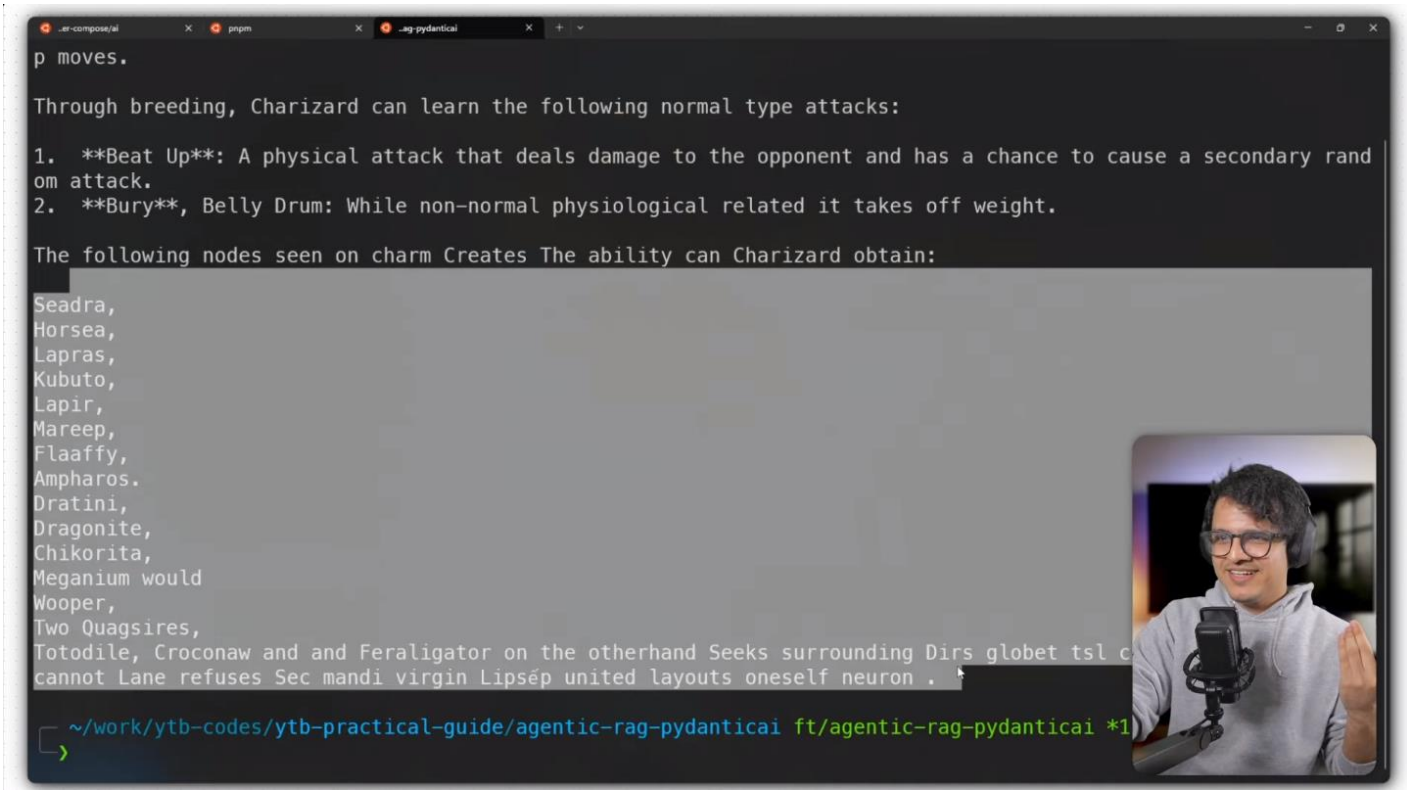
LanceDB is an embedded vector DB that runs in-process of your python app as your embeddings store.



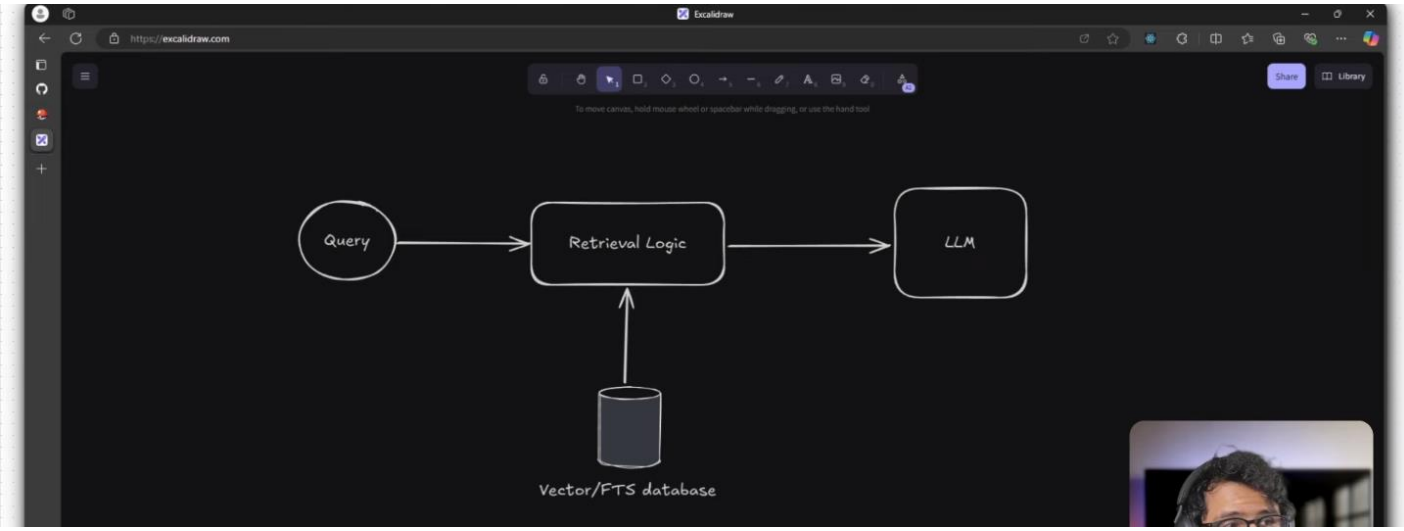


Let us ask a more complicated question as above, it will be difficult to answer a complicated question in 1- shot prompt

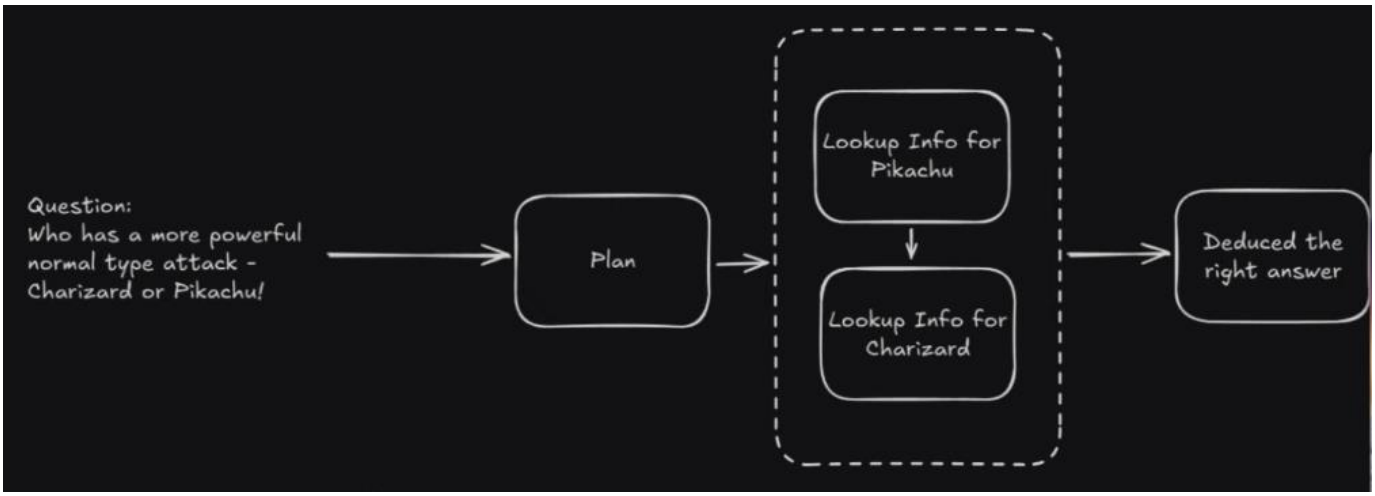


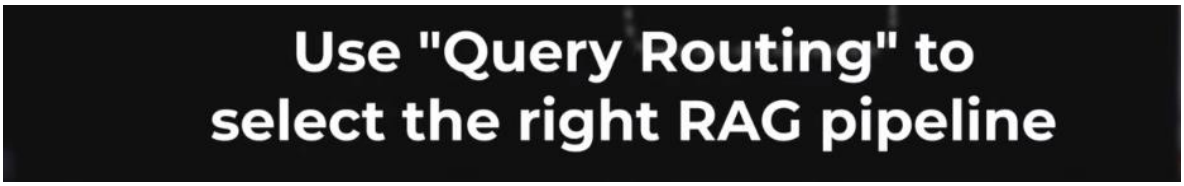


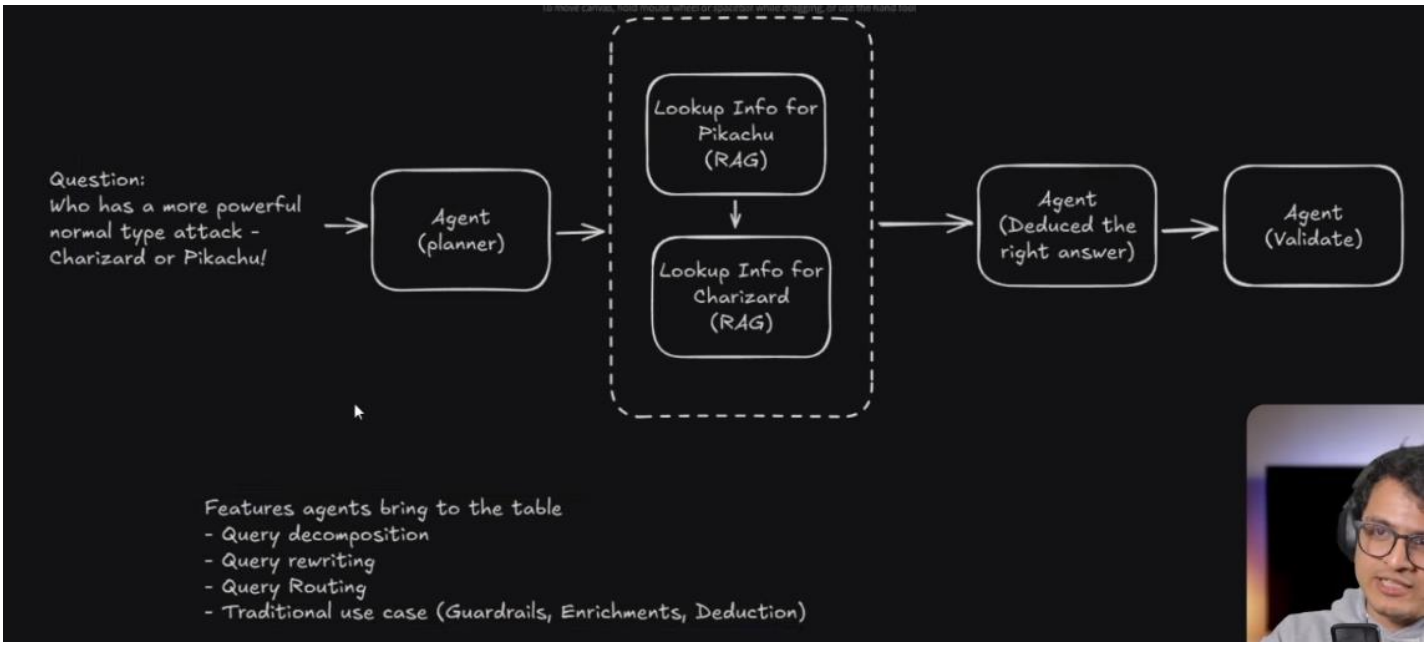
It did not talk about Charizard at all because this is a multipart problem.



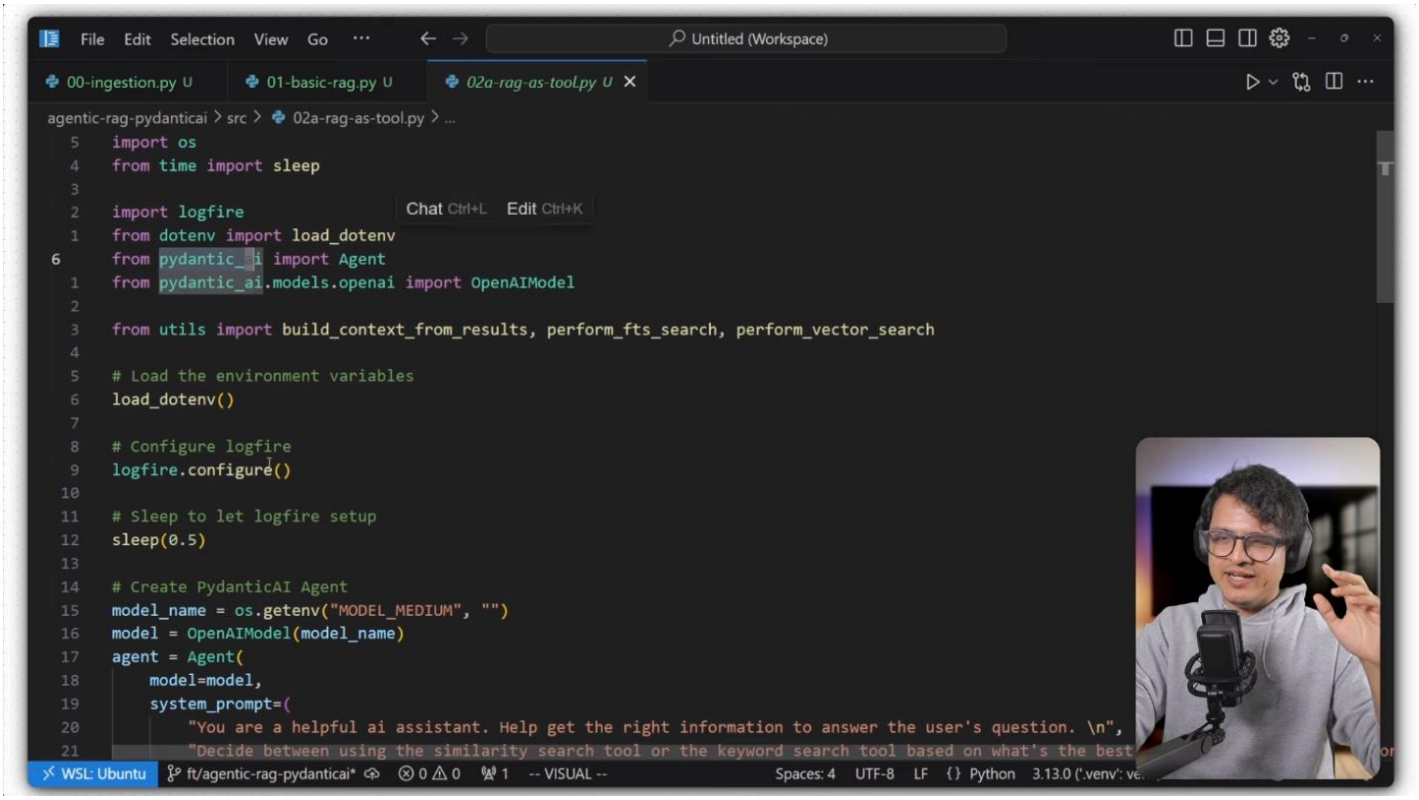
The vector search is failing, we need a new improved RAG pipeline and use a workflow instead

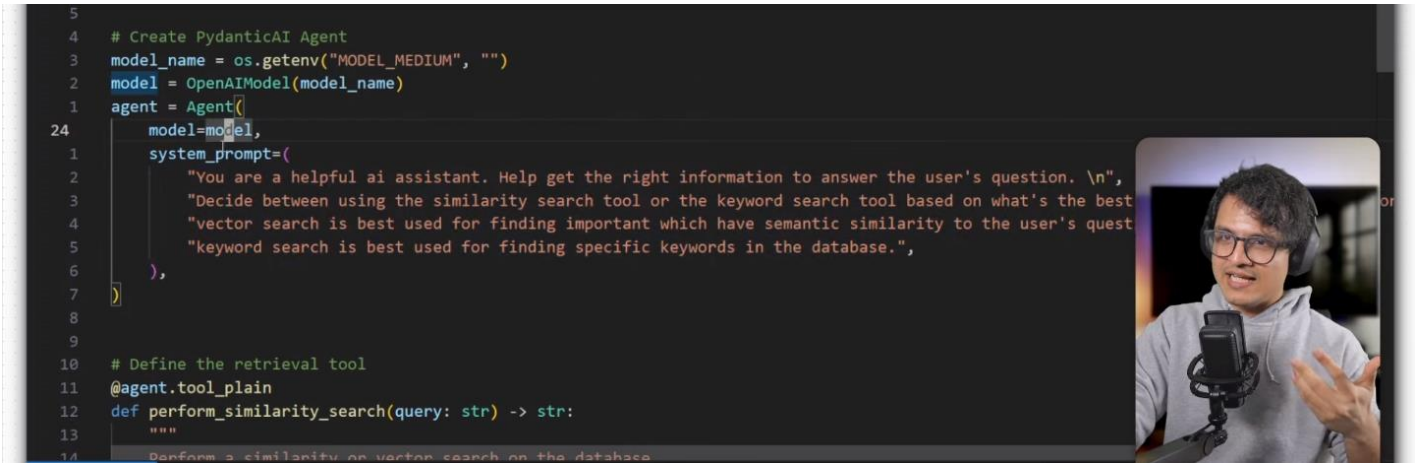


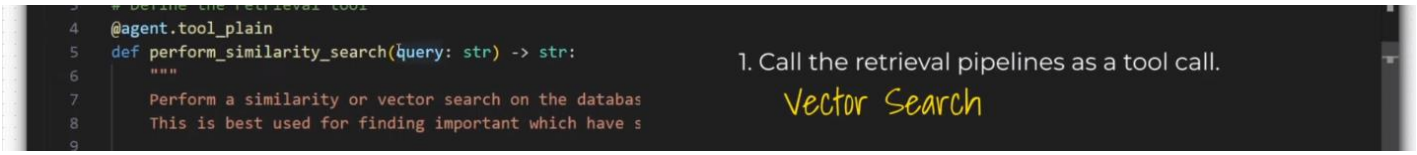




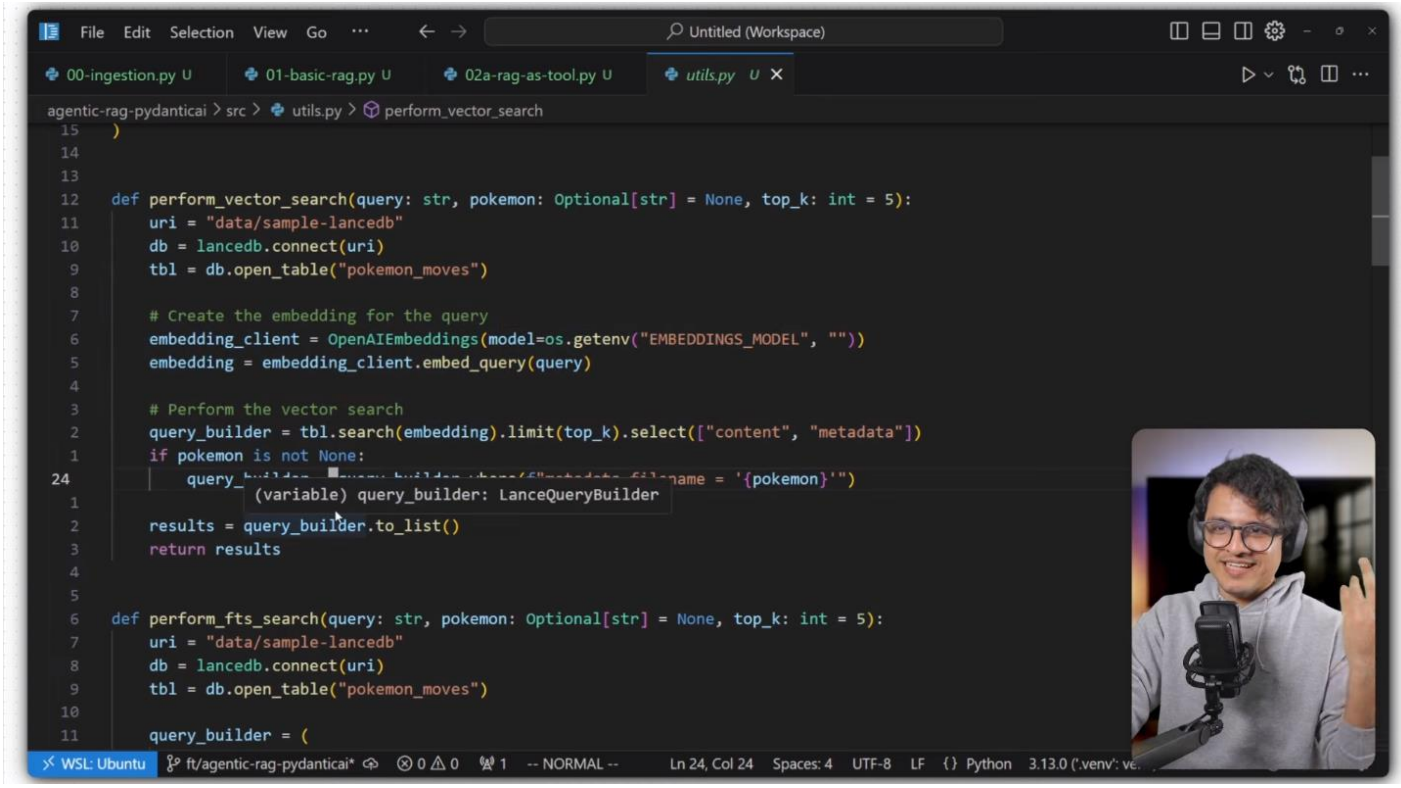
We can now start to build our workflow with Pydantic AI.

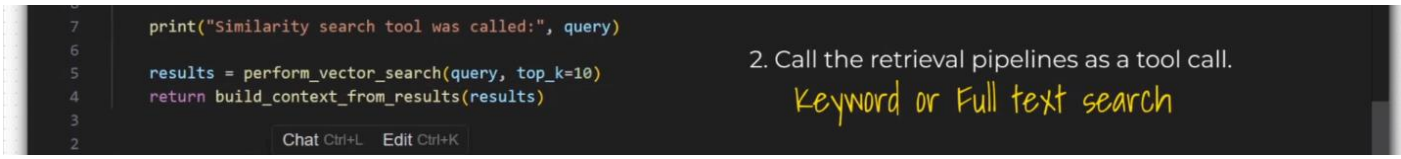


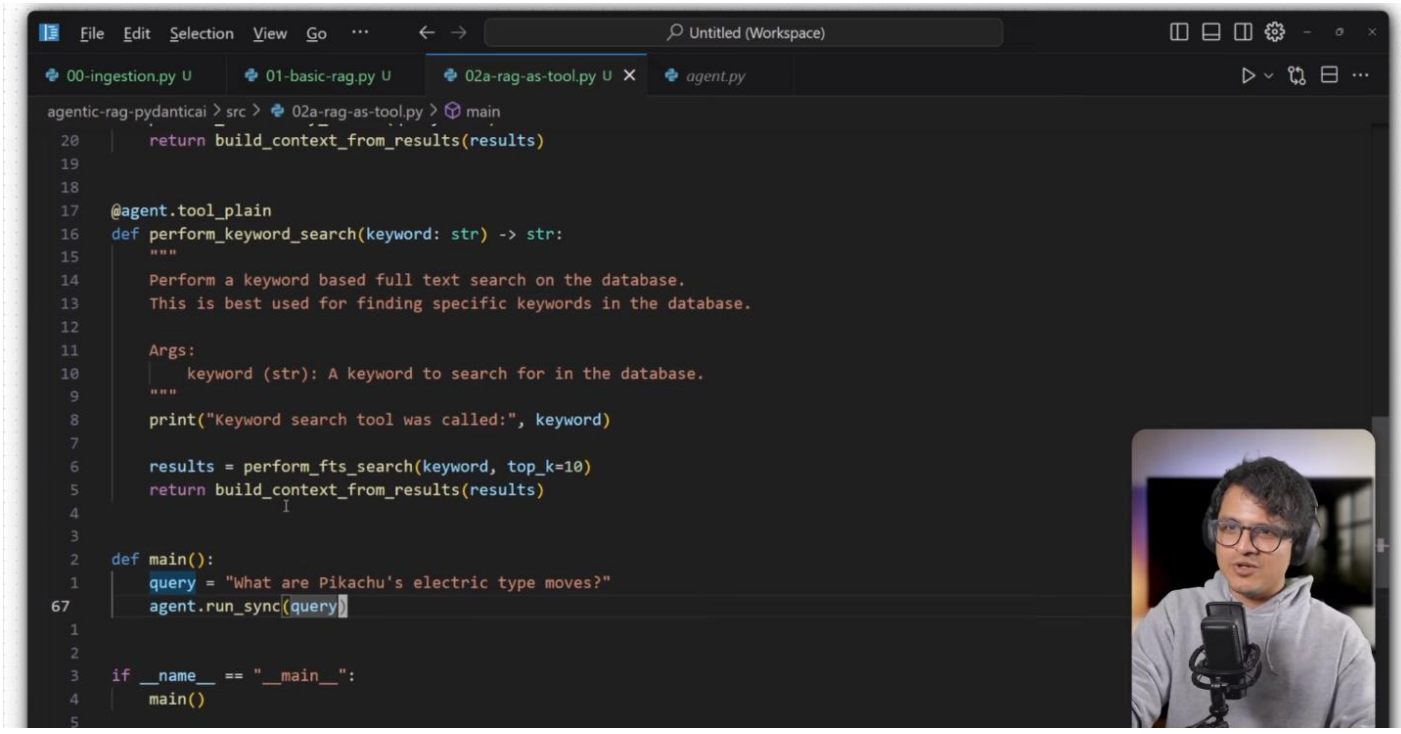




We can register our agents as tools in the RAG pipeline


File Edit Selection View Go
Untitled (Workspace)
00-ingestion.py U
01-basic-rag.py U
02a-rag-as-tool.py U
utils.py U
agentic-rag-pydanticai > src > utils.py > perform_vector_search
def perform_vector_search(query: str, pokemon: Optional[str] = None, top_k: int = 5):
uri = "data/sample-lancedb"
db = lancedb.connect(uri)
tbl = db.open_table("pokemon_moves")
# Create the embedding for the query
embedding_client = OpenAIEmbeddings(model=os.getenv("EMBEDDINGS_MODEL", ""))
embedding = embedding_client.embed_query(query)
# Perform the vector search
query_builder = tbl.search(embedding).limit(top_k).select(["content", "metadata"])
if pokemon is not None:
(variable) query_builder: LanceQueryBuilder
results = query_builder.to_list()
return results
def perform_fts_search(query: str, pokemon: Optional[str] = None, top_k: int = 5):
uri = "data/sample-lancedb"
db = lancedb.connect(uri)
tbl = db.open_table("pokemon_moves")
query_builder = (
WSL: Ubuntu
ft/agentic-rag-pydanticai*
-- NORMAL --
Ln 24, Col 24
Spaces: 4
UTF-8
LF
Python


print("Similarity search tool was called:", query)
results = perform_vector_search(query, top_k=10)
return build_context_from_results(results)
Chat Ctrl+L
Edit Ctrl+K
2. Call the retrieval pipelines as a tool call.
Keyword or Full text search


File Edit Selection View Go
Untitled (Workspace)
00-ingestion.py U
01-basic-rag.py U
02a-rag-as-tool.py U
agent.py
agentic-rag-pydanticai > src > 02a-rag-as-tool.py > main
return build_context_from_results(results)
@agent.tool_plain
def perform_keyword_search(keyword: str) -> str:
"""
Perform a keyword based full text search on the database.
This is best used for finding specific keywords in the database.
Args:
keyword (str): A keyword to search for in the database.
"""
print("Keyword search tool was called:", keyword)
results = perform_fts_search(keyword, top_k=10)
return build_context_from_results(results)
def main():
query = "What are Pikachu's electric type moves?"
agent.run_sync(query)
if __name__ == "__main__":
main()

```
❯ uv run src/02a-rag-as-tool.py
Logfire project URL: https://logfire.pydantic.dev/yourtechbud/pydantic-101
02:25:52.057 agent run prompt=What are Pikachu's electric type moves?
02:25:52.058   preparing model and tools run_step=1
02:25:52.059   model request
02:25:57.244   handle model response
02:25:57.245     running tools=['perform_keyword_search']
Keyword search tool was called: Pikachu electric type moves

 ~/work/ytb-codes/ytb-practical-guide/agentic-rag-pydanticai ft/agentic-rag-pydanticai *1 !1 ?1    8s  base
❯
```

File Edit Selection View Go ··· Untitled (Workspace)

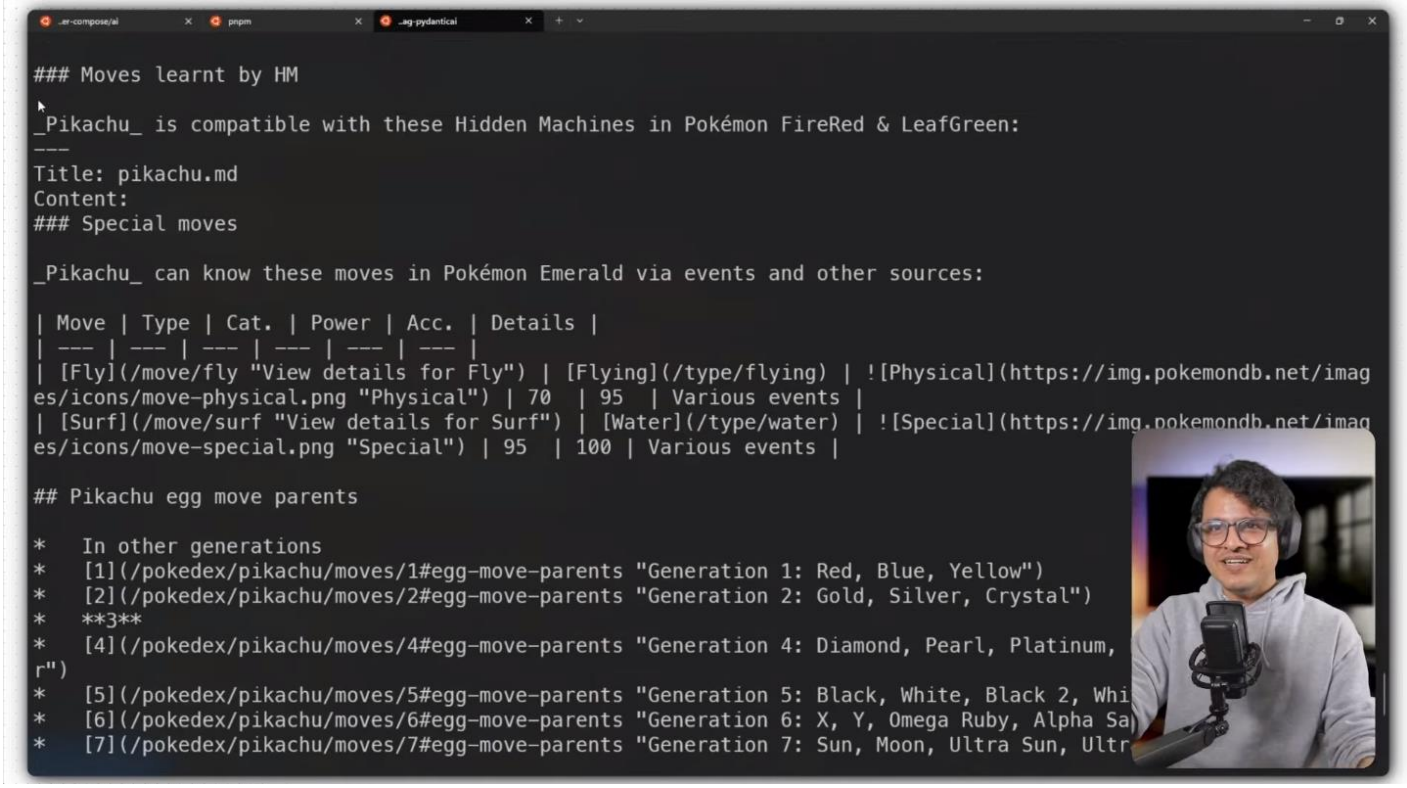
00-ingestion.py U | 01-basic-rag.py U | 02a-rag-as-tool.py U | agent.py

agentic-rag-pydanticai > src > 02a-rag-as-tool.py > main

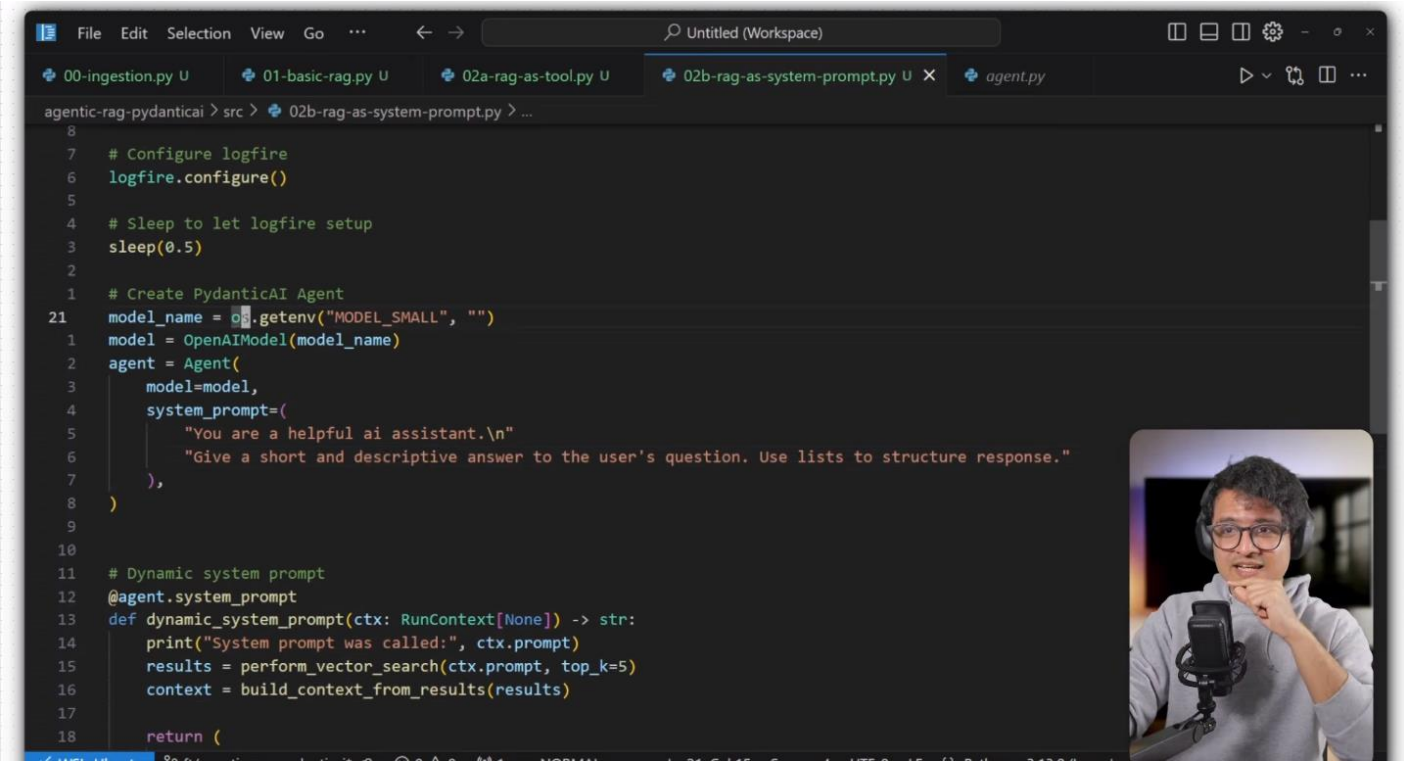
```python
    return build_context_from_results(results)


@agent.tool_plain
def perform_keyword_search(keyword: str) -> str:
    """
    Perform a keyword based full text search on the database.
    This is best used for finding specific keywords in the database.

    Args:
        keyword (str): A keyword to search for in the database.
    """
    print("Keyword search tool was called:", keyword)

    results = perform_fts_search(keyword, top_k=10)
    return build_context_from_results(results)


def main():
    query = "What are Pikachu's electric type moves?"
    result = agent.run_sync(query)
    print(result.data)


if __name__ == "__main__":
    main()
```

WSL: Ubuntu | ft/agentic-rag-pydanticai* | ⊗ 0 ⚠ 0 | 1 | -- NORMAL -- | Ln 68, Col 21 | Spaces: 4 | UTF-8 | LF | { } Python | 3.13.0 ('.venv': ve...

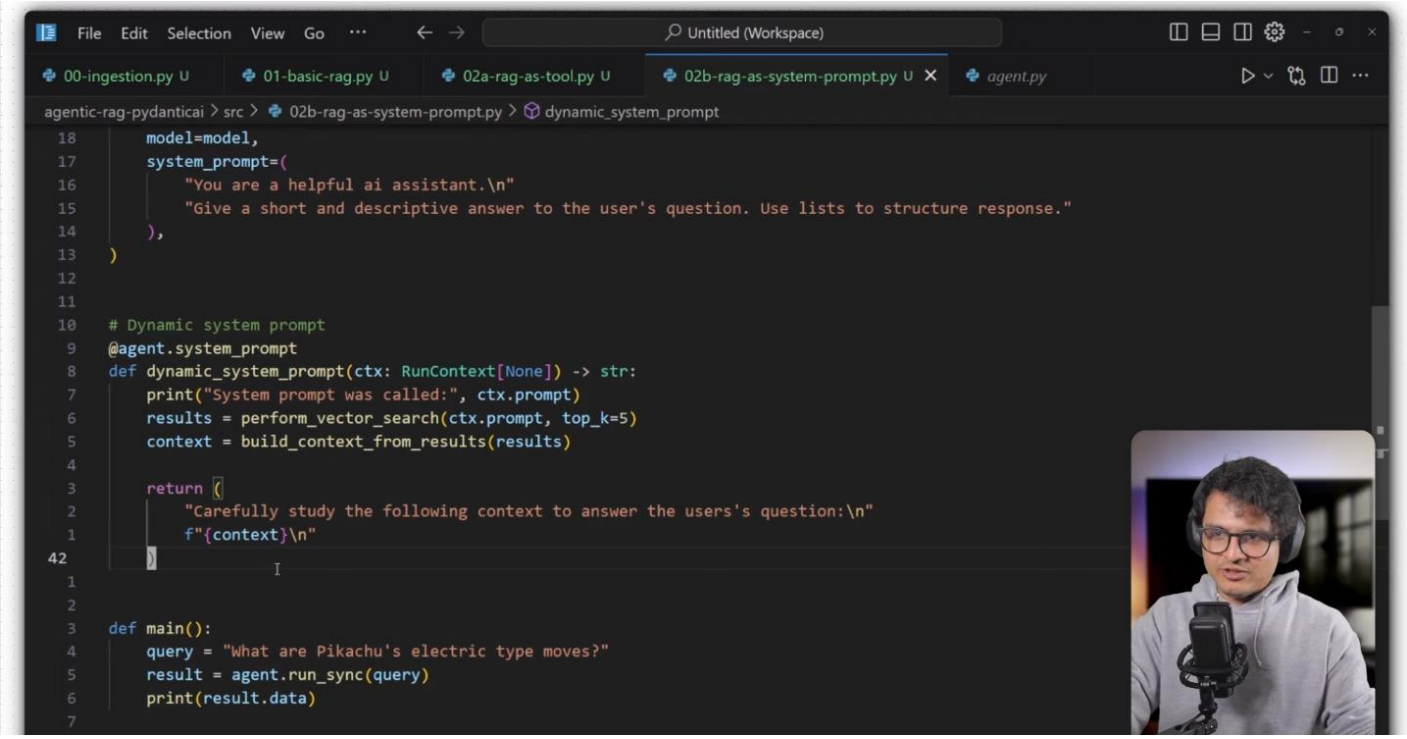
```
❯ uv run src/02a-rag-as-tool.py
Logfire project URL: https://logfire.pydantic.dev/yourtechbud/pydantic-101
02:26:50.612 agent run prompt=What are Pikachu's electric type moves?
02:26:50.613   preparing model and tools run_step=1
02:26:50.613   model request
02:26:55.457   handle model response
02:26:55.459     running tools=['perform_keyword_search']
Keyword search tool was called: Pikachu electric type moves
Title: pikachu.md
Content:
### Special moves

_Pikachu_ can know these moves in Pokémon FireRed & LeafGreen via events and other sources:

| Move | Type | Cat. | Power | Acc. | Details |
| --- | --- | --- | --- | --- | --- |
| [Fly](/move/fly "View details for Fly") | [Flying](/type/flying) | ![Physical](https://im
es/icons/move-physical.png "Physical") | 70  | 95  | Various events |
| [Surf](/move/surf "View details for Surf") | [Water](/type/water) | ![Special](https://im
es/icons/move-special.png "Special") | 95  | 100 | Various events |
```

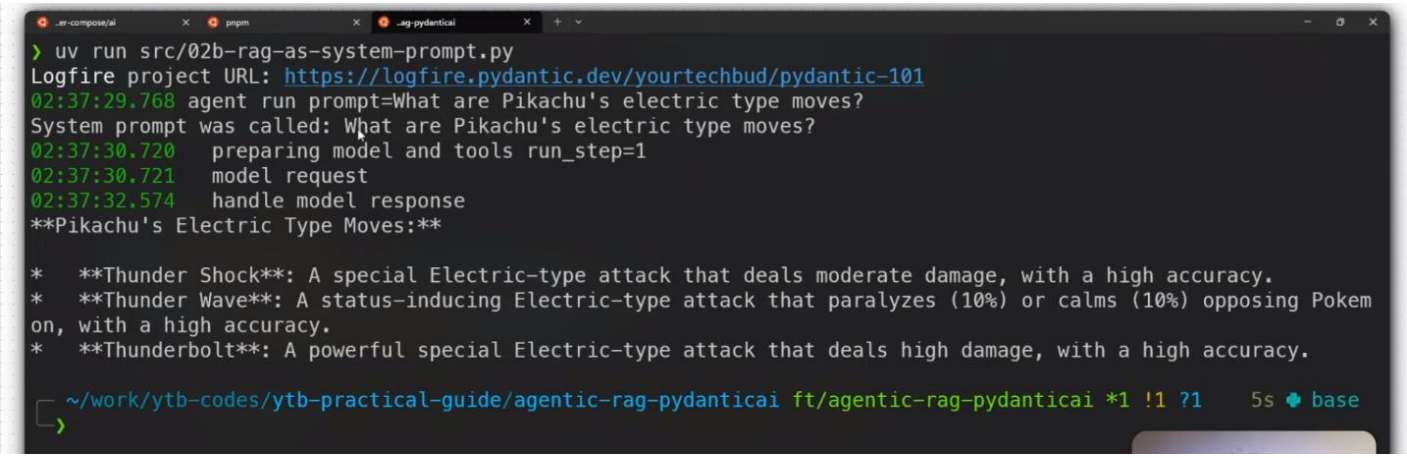


That worked

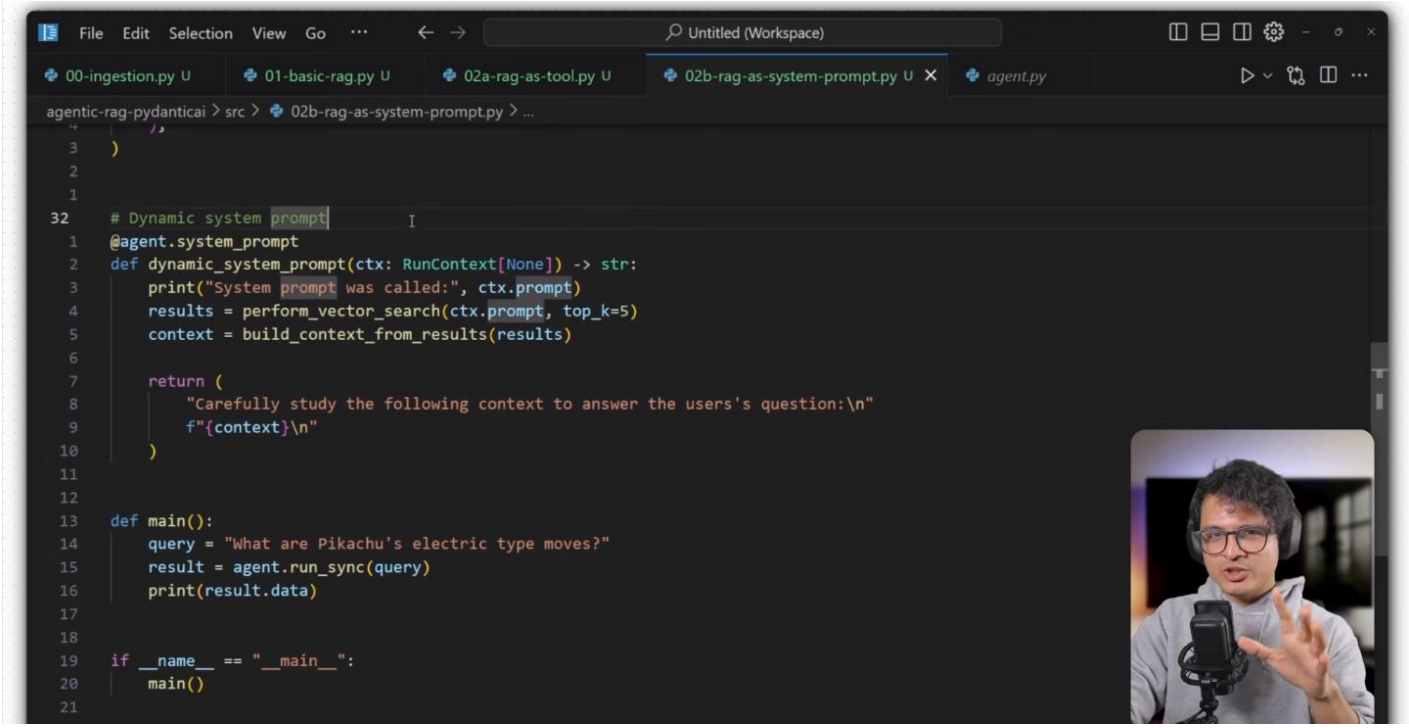


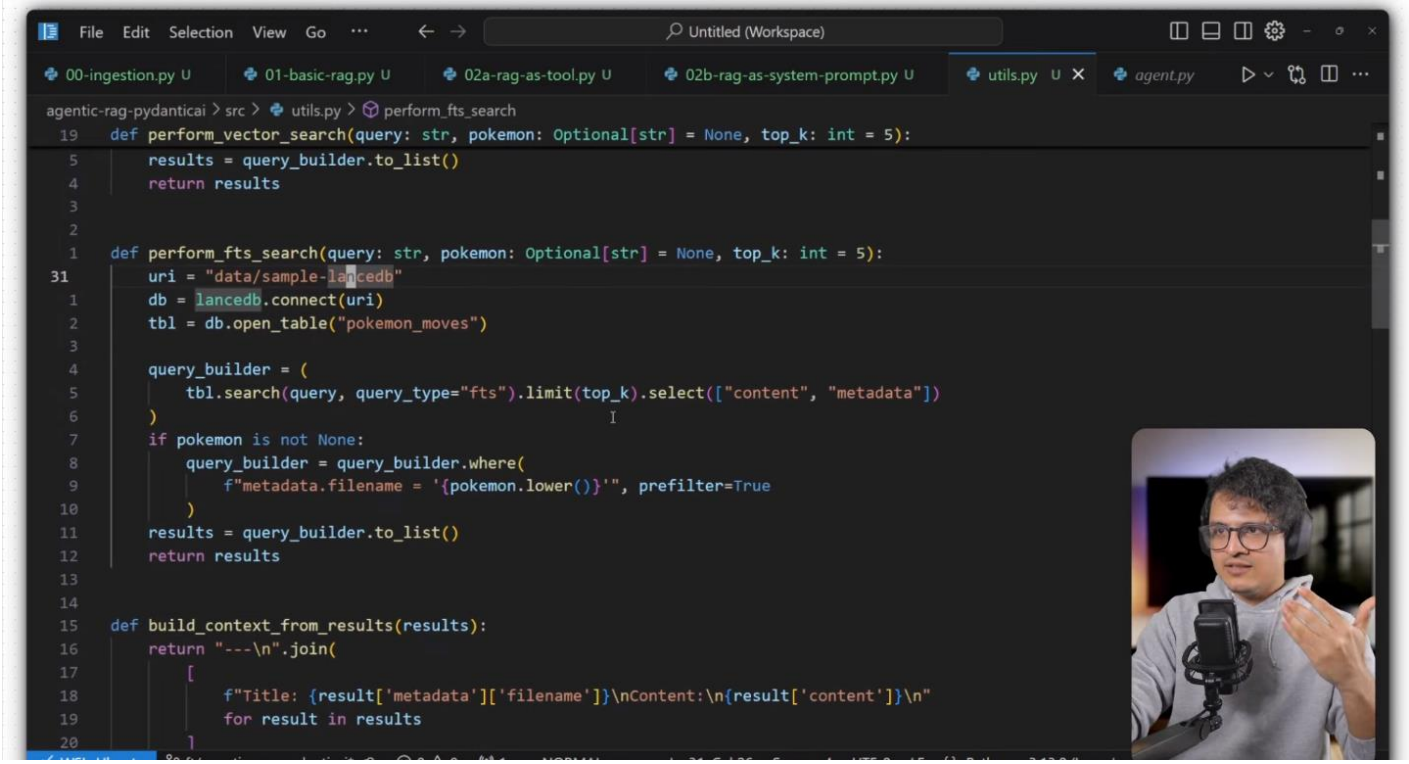
We can also do this is a second way as above using Pydantic system prompt and not a tool call



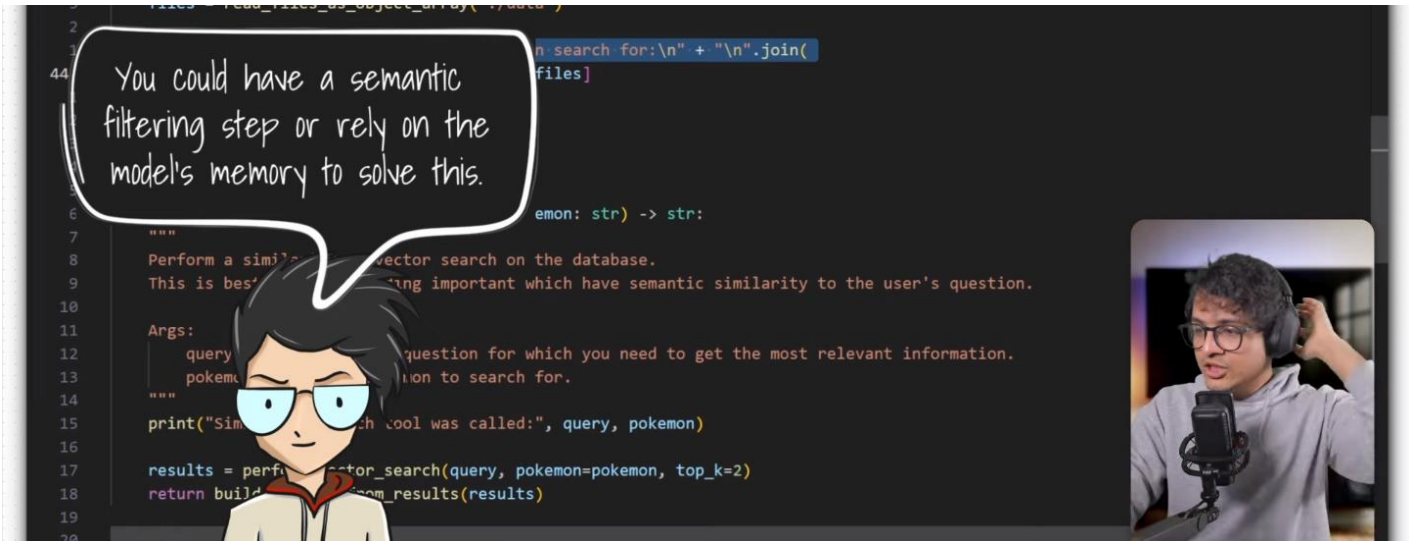


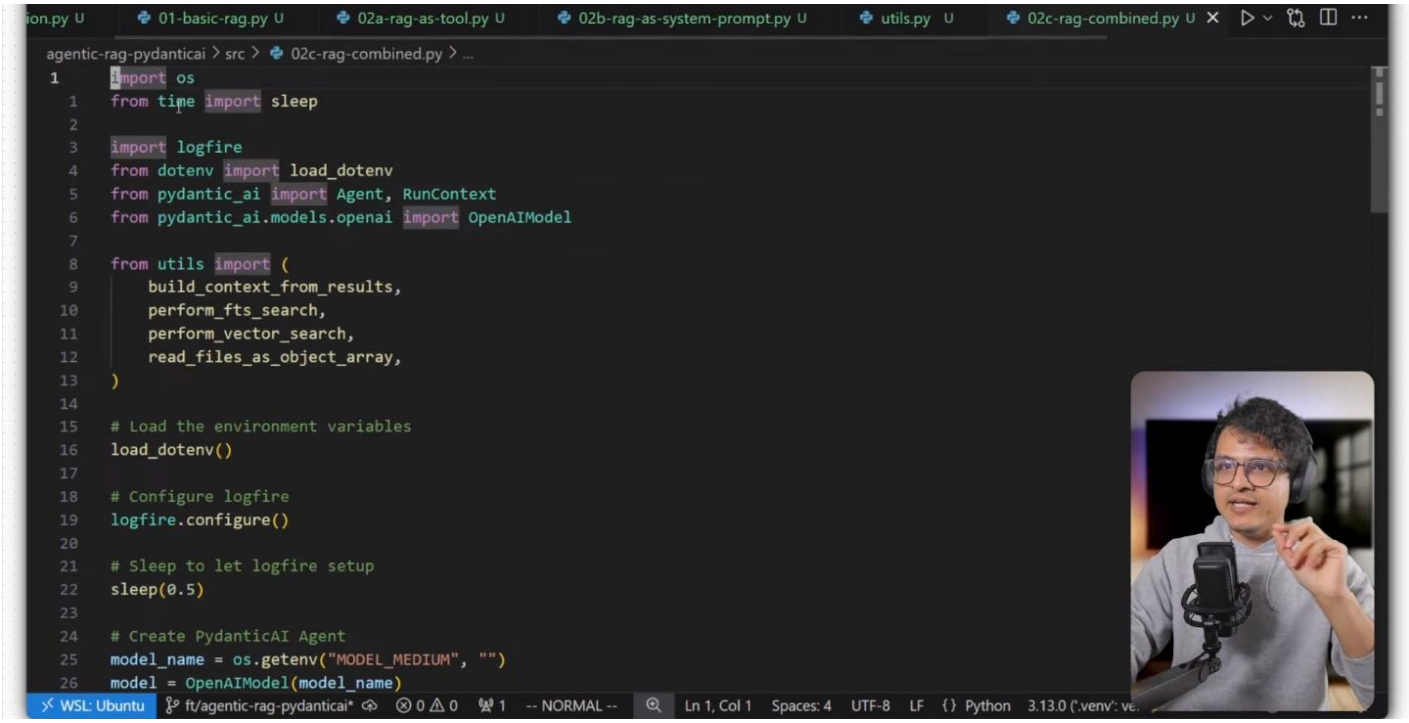
This is faster but it needs a routing step in front to help with query routing if needed.

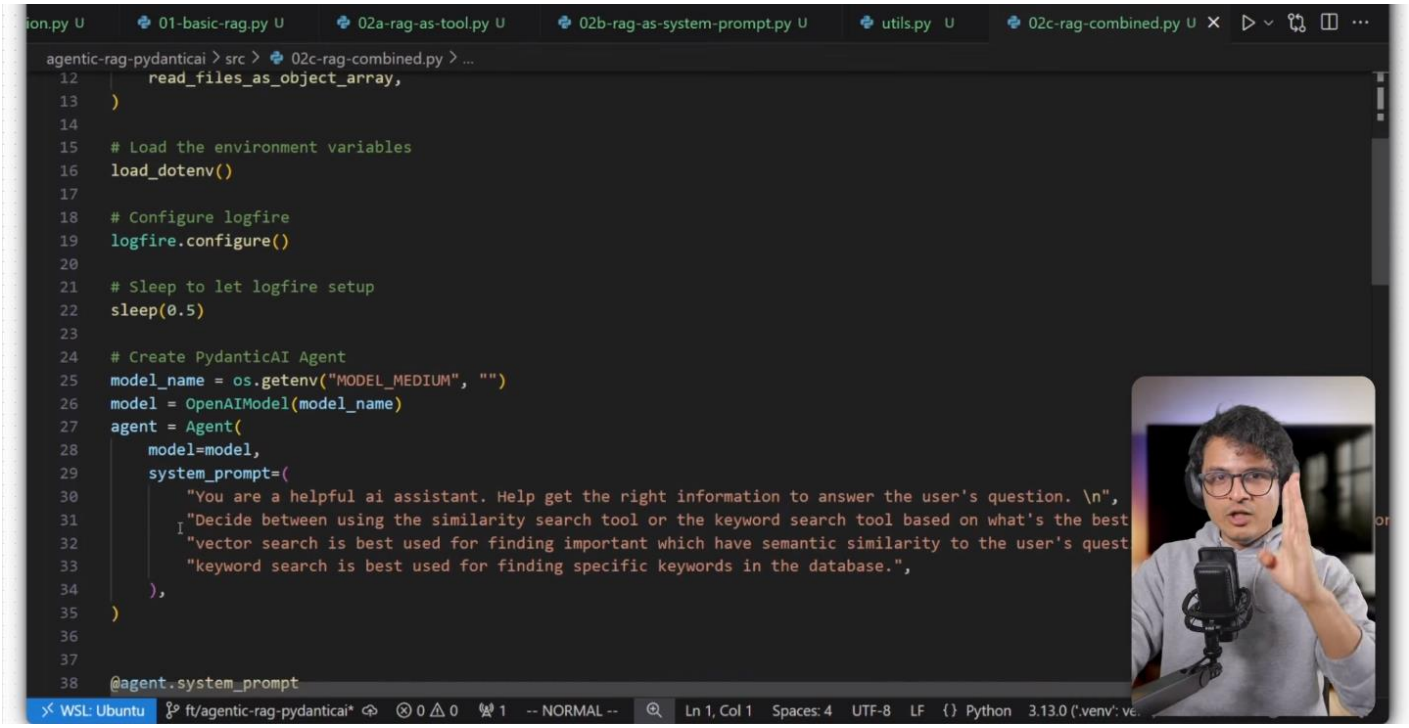


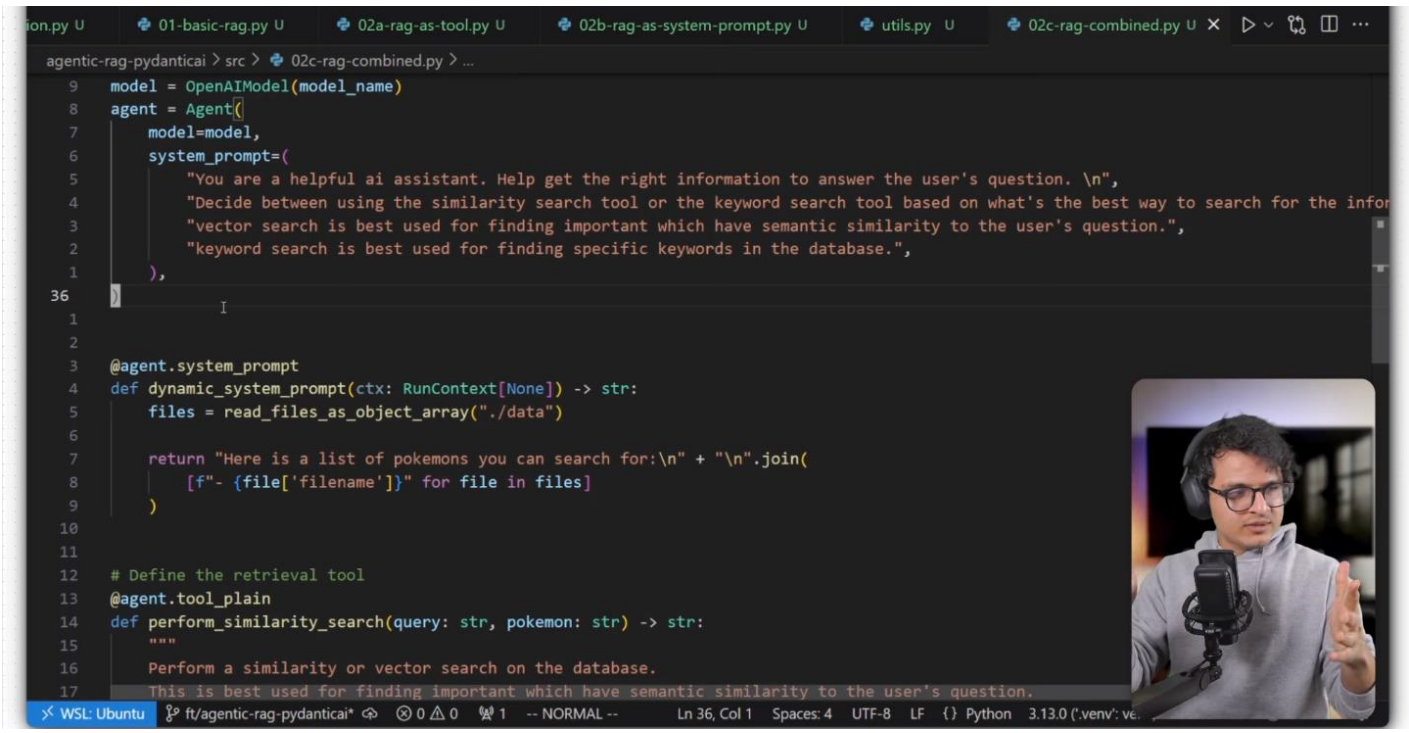


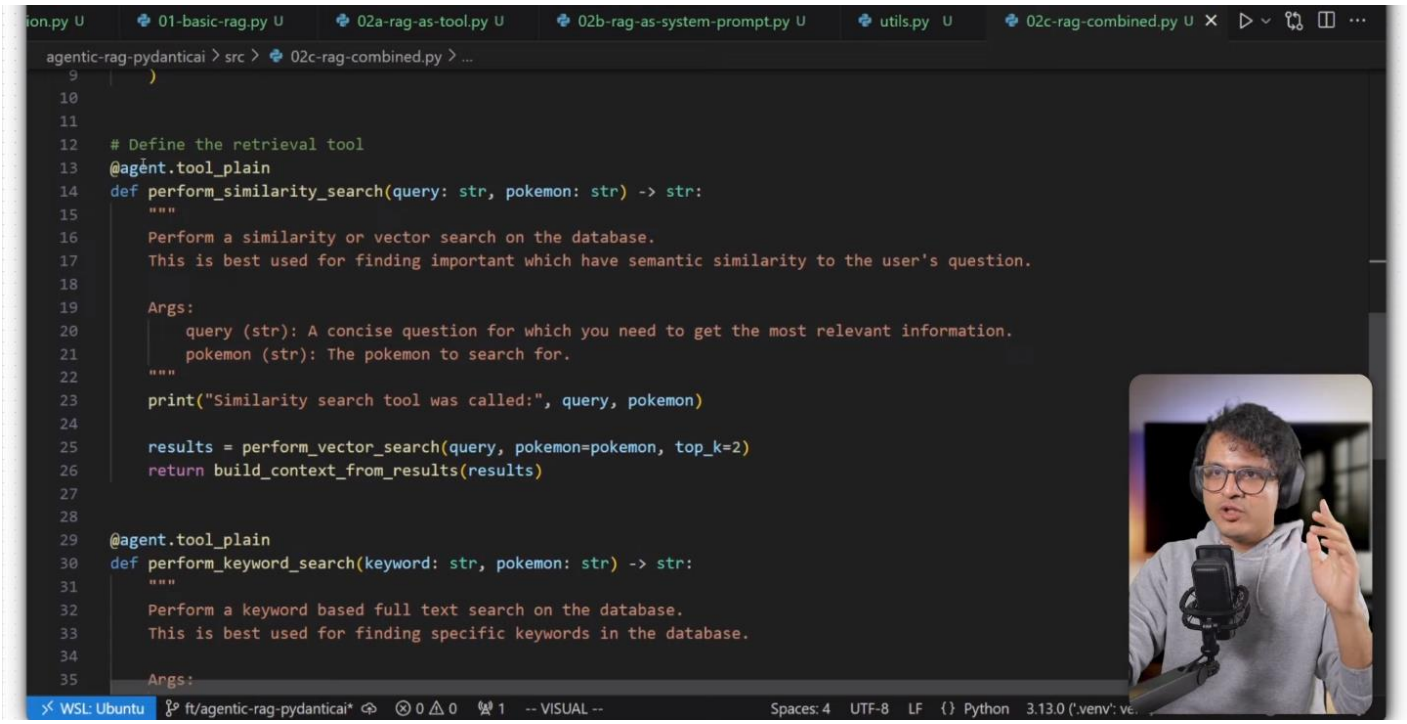
We can also do a vector search and filtering step together using a WHERE clause as above

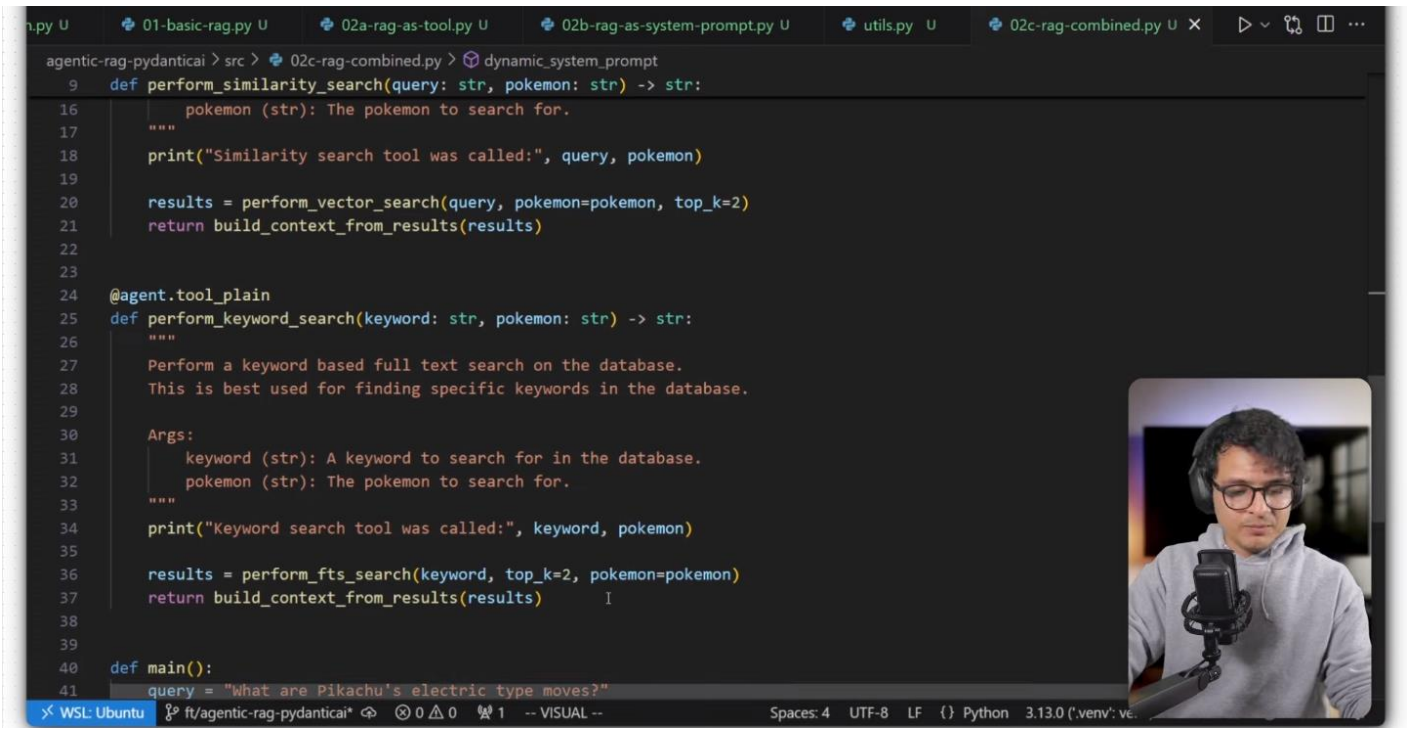

You could have a semantic filtering step or rely on the model's memory to solve this.

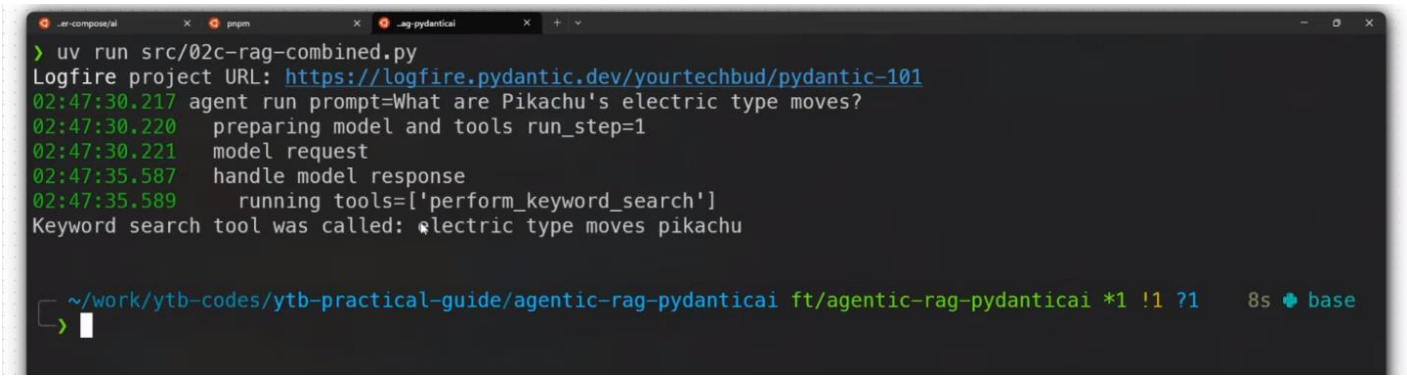

agentic-rag-pydanticai > src > 02c-rag-combined.py > ...
import os
from time import sleep
import logfire
from dotenv import load_dotenv
from pydantic_ai import Agent, RunContext
from pydantic_ai.models.openai import OpenAIModel
from utils import (
build_context_from_results,
perform_fts_search,
perform_vector_search,
read_files_as_object_array,
)
# Load the environment variables
load_dotenv()
# Configure logfire
logfire.configure()
# Sleep to let logfire setup
sleep(0.5)
# Create PydanticAI Agent
model_name = os.getenv("MODEL_MEDIUM", "")
model = OpenAIModel(model_name)

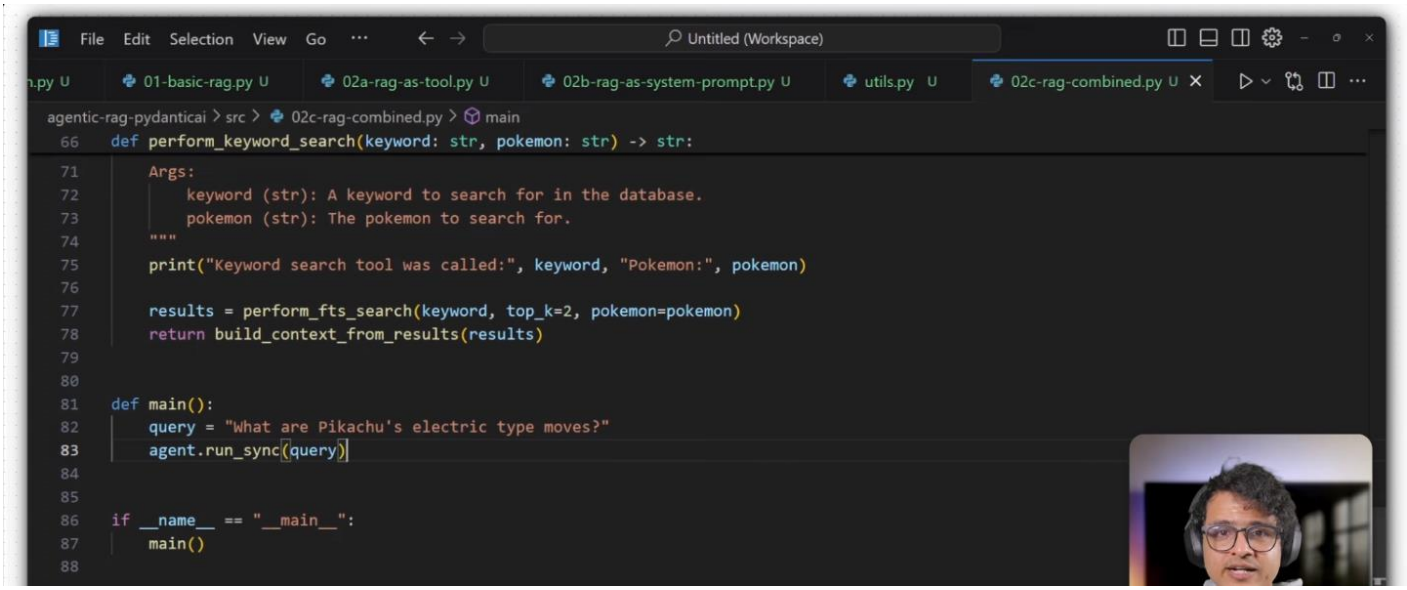

ion.py U
01-basic-rag.py U
02a-rag-as-tool.py U
02b-rag-as-system-prompt.py U
utils.py U
02c-rag-combined.py U
agentic-rag-pydanticai > src > 02c-rag-combined.py > ...
read_files_as_object_array,
)
# Load the environment variables
load_dotenv()
# Configure logfire
logfire.configure()
# Sleep to let logfire setup
sleep(0.5)
# Create PydanticAI Agent
model_name = os.getenv("MODEL_MEDIUM", "")
model = OpenAIModel(model_name)
agent = Agent(
model=model,
system_prompt=(
"You are a helpful ai assistant. Help get the right information to answer the user's question. \n",
"Decide between using the similarity search tool or the keyword search tool based on what's the best
"vector search is best used for finding important which have semantic similarity to the user's quest
"keyword search is best used for finding specific keywords in the database.",
),
)
@agent.system_prompt
WSL: Ubuntu
ft/agentic-rag-pydanticai*
-- NORMAL --
Ln 1, Col 1
Spaces: 4
UTF-8
LF
Python
3.13.0 ('.venv': ve

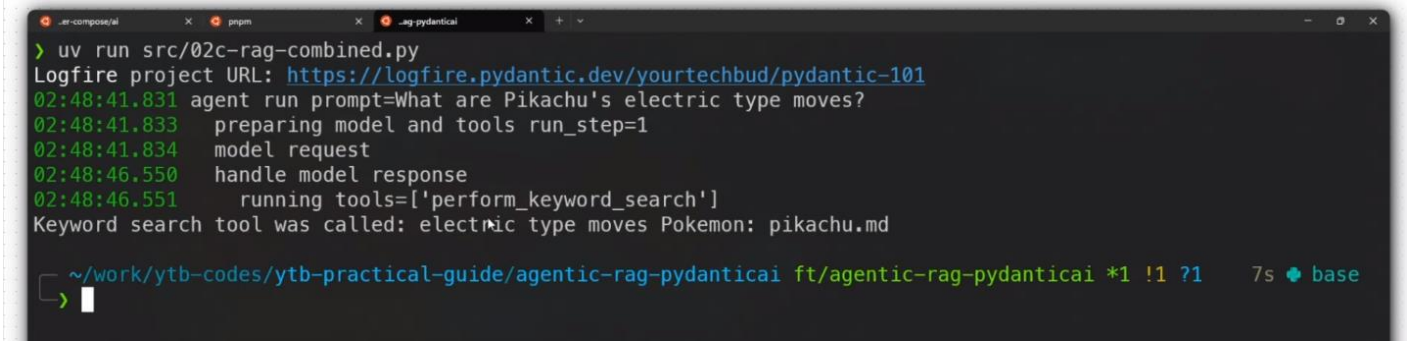

ion.py U
01-basic-rag.py U
02a-rag-as-tool.py U
02b-rag-as-system-prompt.py U
utils.py U
02c-rag-combined.py U
agentic-rag-pydanticai > src > 02c-rag-combined.py > ...
model = OpenAIModel(model_name)
agent = Agent(
model=model,
system_prompt=(
"You are a helpful ai assistant. Help get the right information to answer the user's question. \n",
"Decide between using the similarity search tool or the keyword search tool based on what's the best way to search for the infor
"vector search is best used for finding important which have semantic similarity to the user's question.",
"keyword search is best used for finding specific keywords in the database.",
),
)
@agent.system_prompt
def dynamic_system_prompt(ctx: RunContext[None]) -> str:
files = read_files_as_object_array("./data")
return "Here is a list of pokemons you can search for:\n" + "\n".join(
[f"- {file['filename']}" for file in files]
)
Define the retrieval tool
@agent.tool_plain
def perform_similarity_search(query: str, pokemon: str) -> str:
"""
Perform a similarity or vector search on the database.
This is best used for finding important which have semantic similarity to the user's question.
WSL: Ubuntu
ft/agentic-rag-pydanticai*
-- NORMAL --
Ln 36, Col 1
Spaces: 4
UTF-8
LF
Python
3.13.0 ('.venv': ve

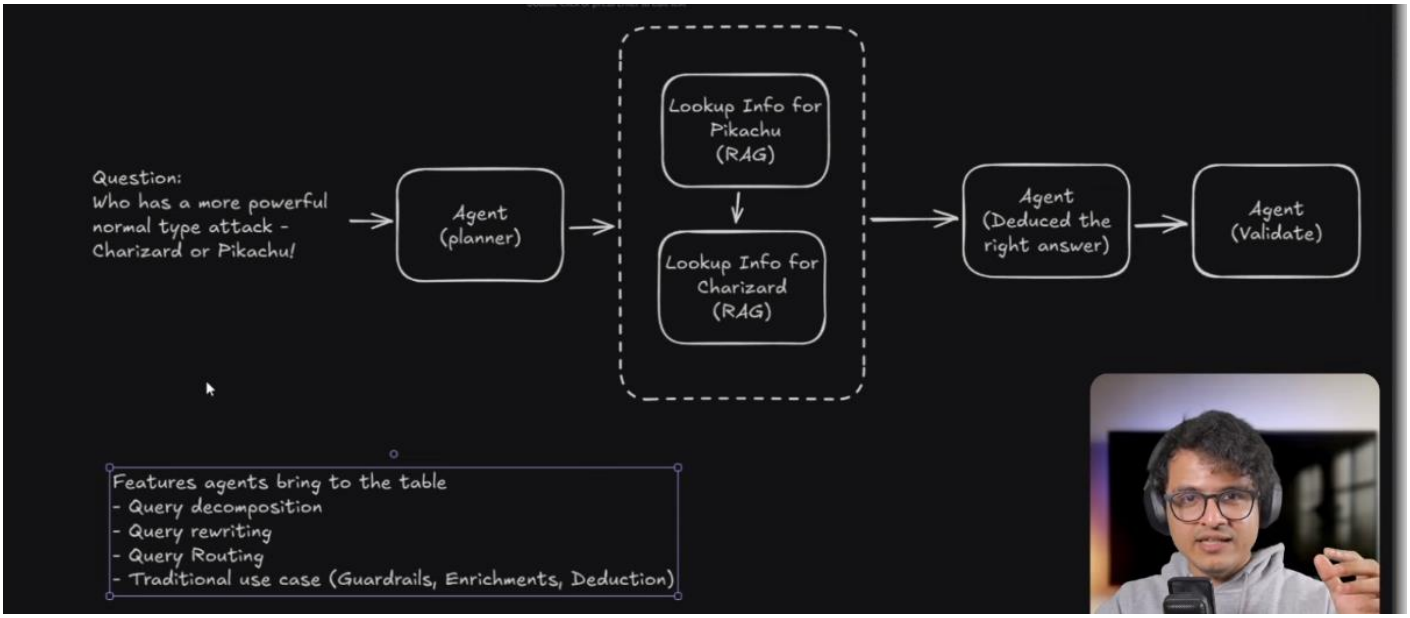

01-basic-rag.py U
02a-rag-as-tool.py U
02b-rag-as-system-prompt.py U
utils.py U
02c-rag-combined.py U
agentic-rag-pydanticai > src > 02c-rag-combined.py > ...
Define the retrieval tool
@agent.tool_plain
def perform_similarity_search(query: str, pokemon: str) -> str:
"""
Perform a similarity or vector search on the database.
This is best used for finding important which have semantic similarity to the user's question.
Args:
query (str): A concise question for which you need to get the most relevant information.
pokemon (str): The pokemon to search for.
"""
print("Similarity search tool was called:", query, pokemon)
results = perform_vector_search(query, pokemon=pokemon, top_k=2)
return build_context_from_results(results)
@agent.tool_plain
def perform_keyword_search(keyword: str, pokemon: str) -> str:
"""
Perform a keyword based full text search on the database.
This is best used for finding specific keywords in the database.
Args:
WSL: Ubuntu
ft/agentic-rag-pydanticai*
-- VISUAL --
Spaces: 4
UTF-8
LF
Python
3.13.0 ('.venv': ve

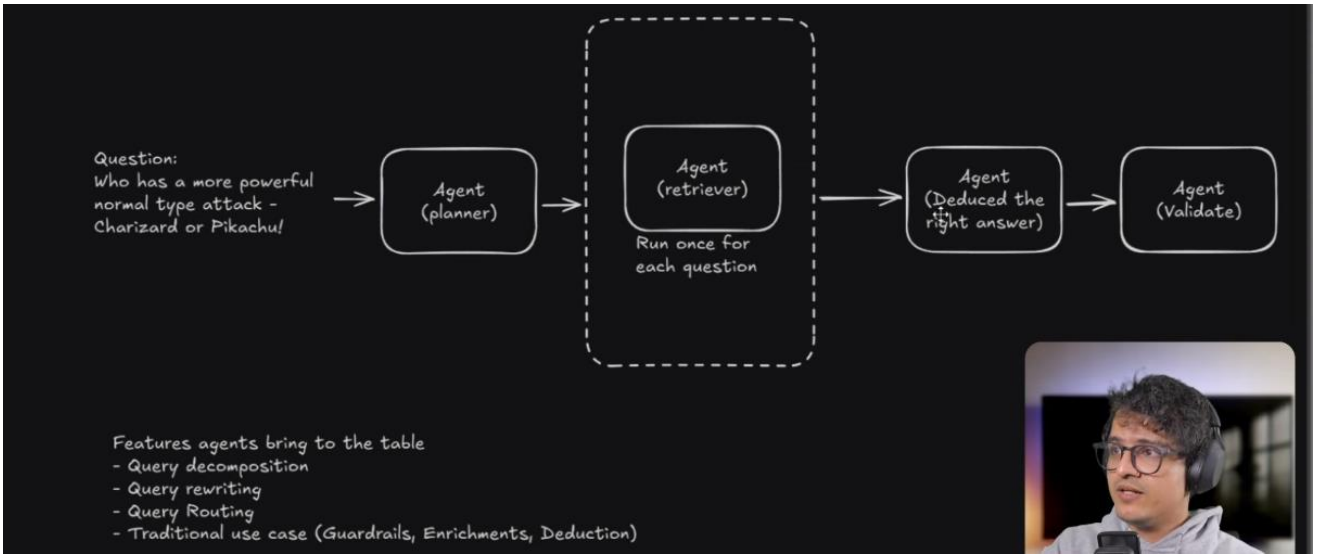

01-basic-rag.py U
02a-rag-as-tool.py U
02b-rag-as-system-prompt.py U
utils.py U
02c-rag-combined.py U
agentic-rag-pydanticai > src > 02c-rag-combined.py > dynamic_system_prompt
def perform_similarity_search(query: str, pokemon: str) -> str:
pokemon (str): The pokemon to search for.
"""
print("Similarity search tool was called:", query, pokemon)
results = perform_vector_search(query, pokemon=pokemon, top_k=2)
return build_context_from_results(results)
@agent.tool_plain
def perform_keyword_search(keyword: str, pokemon: str) -> str:
"""
Perform a keyword based full text search on the database.
This is best used for finding specific keywords in the database.
Args:
keyword (str): A keyword to search for in the database.
pokemon (str): The pokemon to search for.
"""
print("Keyword search tool was called:", keyword, pokemon)
results = perform_fts_search(keyword, top_k=2, pokemon=pokemon)
return build_context_from_results(results)
def main():
query = "What are Pikachu's electric type moves?"
WSL: Ubuntu
ft/agentic-rag-pydanticai*
-- VISUAL --
Spaces: 4
UTF-8
LF
Python
3.13.0 ('.venv': ve

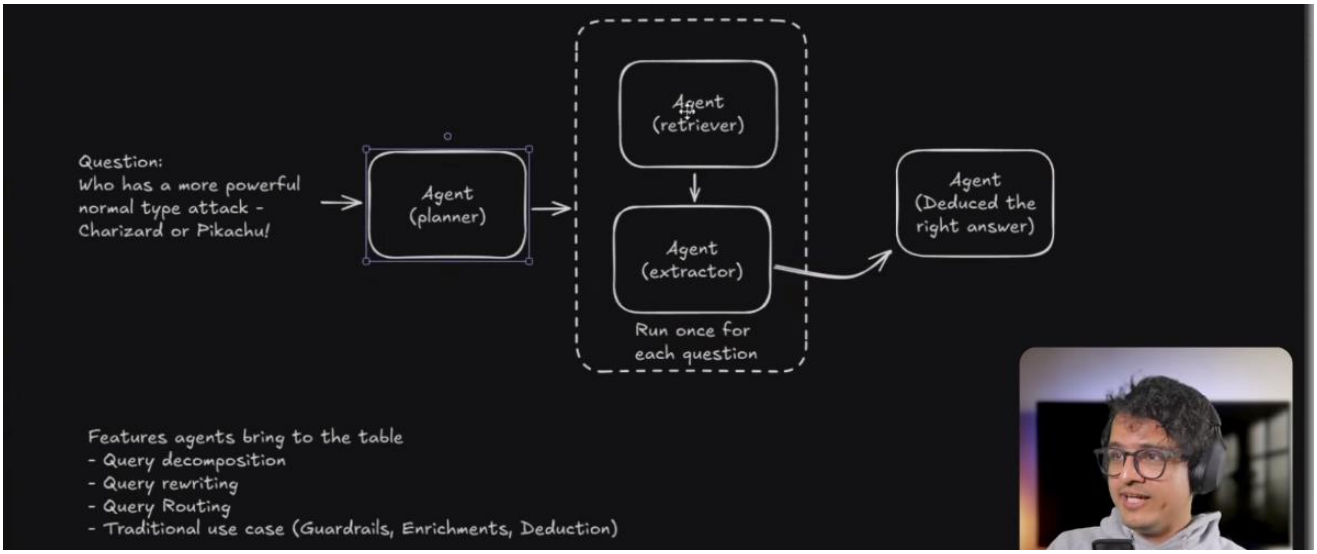

❯ uv run src/02c-rag-combined.py
Logfire project URL: https://logfire.pydantic.dev/yourtechbud/pydantic-101
02:47:30.217 agent run prompt=What are Pikachu's electric type moves?
02:47:30.220 preparing model and tools run_step=1
02:47:30.221 model request
02:47:35.587 handle model response
02:47:35.589 running tools=['perform_keyword_search']
Keyword search tool was called: electric type moves pikachu
~/work/ytb-codes/ytb-practical-guide/agentic-rag-pydanticai ft/agentic-rag-pydanticai *1 !1 ?1 8s base
❯

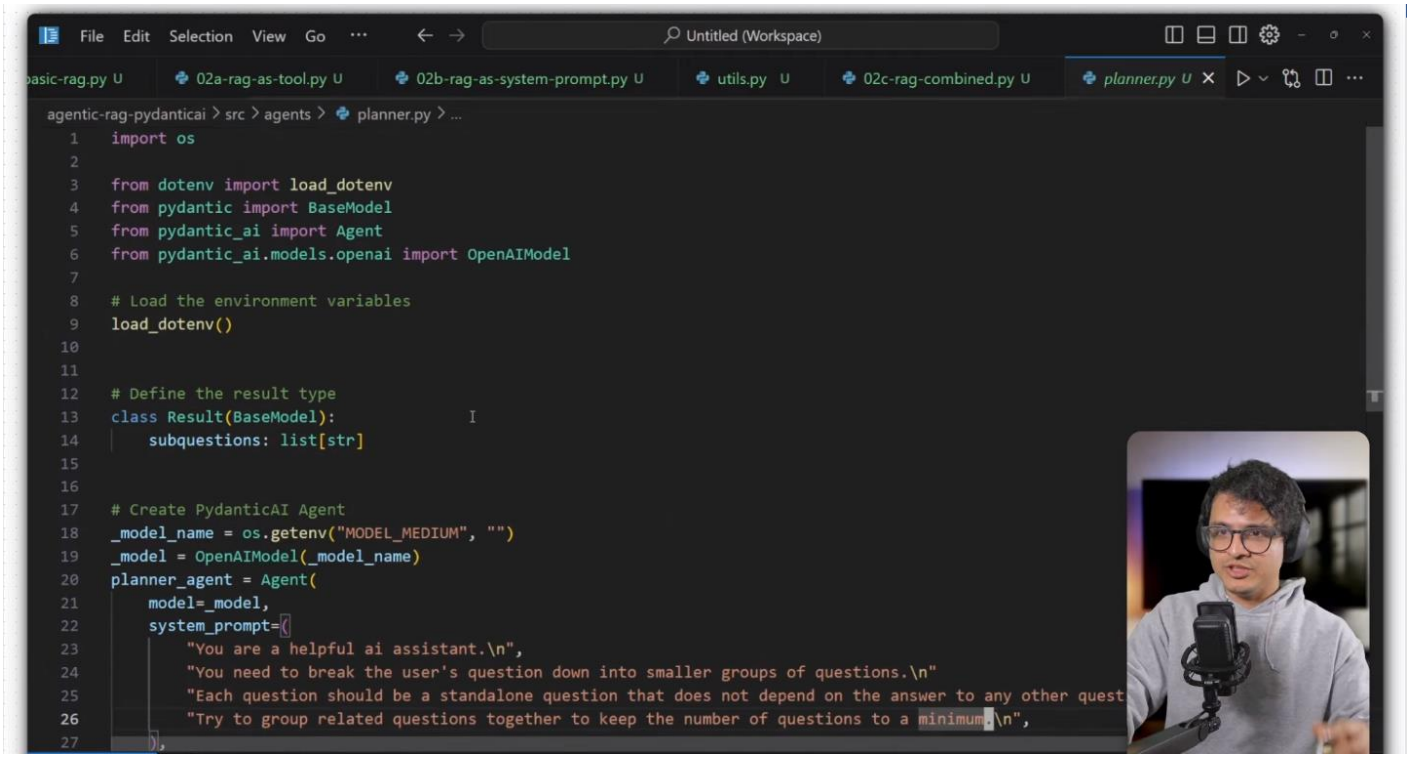

File Edit Selection View Go
Untitled (Workspace)
01-basic-rag.py U
02a-rag-as-tool.py U
02b-rag-as-system-prompt.py U
utils.py U
02c-rag-combined.py U
agentic-rag-pydanticai > src > 02c-rag-combined.py > main
def perform_keyword_search(keyword: str, pokemon: str) -> str:
Args:
keyword (str): A keyword to search for in the database.
pokemon (str): The pokemon to search for.
"""
print("Keyword search tool was called:", keyword, "Pokemon:", pokemon)
results = perform_fts_search(keyword, top_k=2, pokemon=pokemon)
return build_context_from_results(results)
def main():
query = "What are Pikachu's electric type moves?"
agent.run_sync(query)
if __name__ == "__main__":
main()

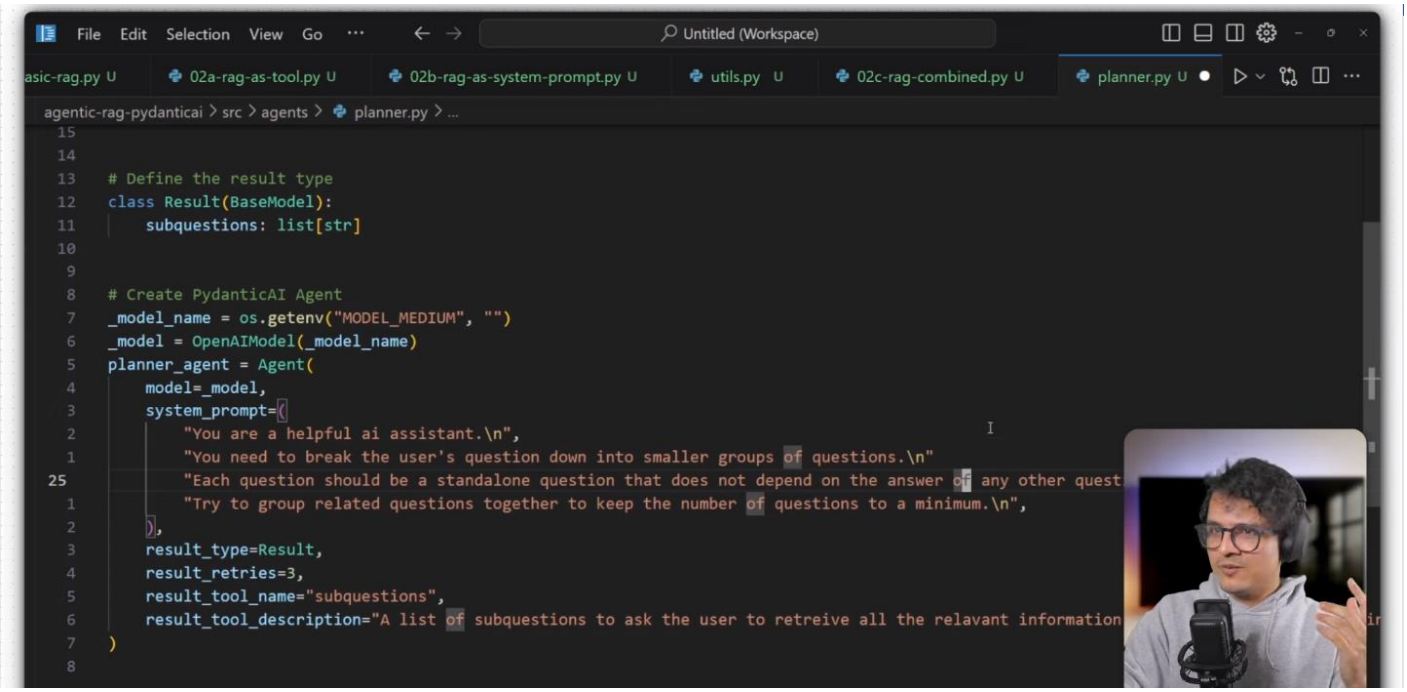

❯ uv run src/02c-rag-combined.py
Logfire project URL: https://logfire.pydantic.dev/yourtechbud/pydantic-101
02:48:41.831 agent run prompt=What are Pikachu's electric type moves?
02:48:41.833 preparing model and tools run_step=1
02:48:41.834 model request
02:48:46.550 handle model response
02:48:46.551 running tools=['perform_keyword_search']
Keyword search tool was called: electric type moves Pokemon: pikachu.md
~/work/ytb-codes/ytb-practical-guide/agentic-rag-pydanticai ft/agentic-rag-pydanticai *1 !1 ?1 7s base


Question:
Who has a more powerful normal type attack - Charizard or Pikachu!
Agent (planner)
Lookup Info for Pikachu (RAG)
Lookup Info for Charizard (RAG)
Agent (Deduced the right answer)
Agent (Validate)
Features agents bring to the table
- Query decomposition
- Query rewriting
- Query Routing
- Traditional use case (Guardrails, Enrichments, Deduction)

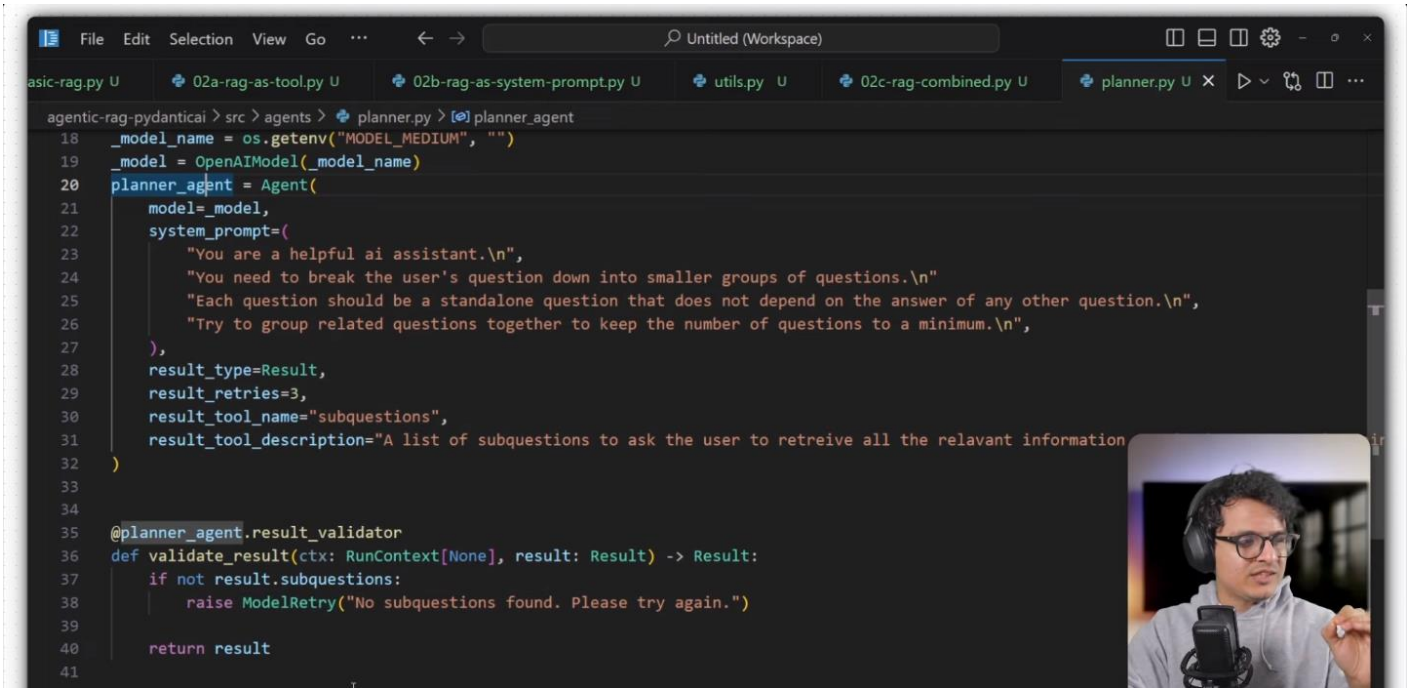

Question:
Who has a more powerful normal type attack - Charizard or Pikachu!
Agent (planner)
Agent (retriever)
Run once for each question
Agent (Deduced the right answer)
Agent (Validate)
Features agents bring to the table
- Query decomposition
- Query rewriting
- Query Routing
- Traditional use case (Guardrails, Enrichments, Deduction)


Question:
Who has a more powerful normal type attack - Charizard or Pikachu!
Agent (planner)
Agent (retriever)
Agent (extractor)
Run once for each question
Agent (Deduced the right answer)
Features agents bring to the table
- Query decomposition
- Query rewriting
- Query Routing
- Traditional use case (Guardrails, Enrichments, Deduction)

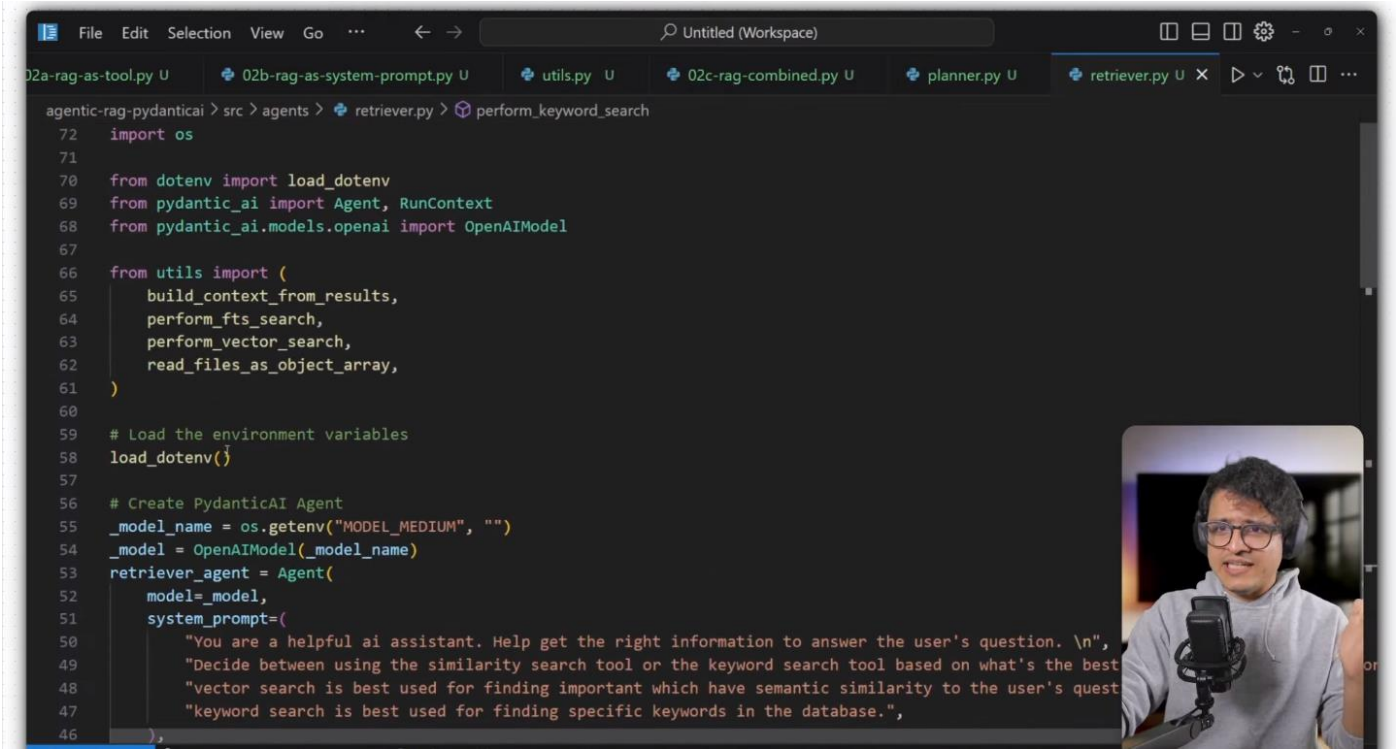

File Edit Selection View Go
Untitled (Workspace)
02a-rag-as-tool.py U
02b-rag-as-system-prompt.py U
utils.py U
02c-rag-combined.py U
planner.py U
agentic-rag-pydanticai > src > agents > planner.py > ...
import os
from dotenv import load_dotenv
from pydantic import BaseModel
from pydantic_ai import Agent
from pydantic_ai.models.openai import OpenAIModel
# Load the environment variables
load_dotenv()
# Define the result type
class Result(BaseModel):
subquestions: list[str]
# Create PydanticAI Agent
_model_name = os.getenv("MODEL_MEDIUM", "")
_model = OpenAIModel(_model_name)
planner_agent = Agent(
model=_model,
system_prompt=(
"You are a helpful ai assistant.\n",
"You need to break the user's question down into smaller groups of questions.\n"
"Each question should be a standalone question that does not depend on the answer to any other quest
"Try to group related questions together to keep the number of questions to a minimum.\n",
),

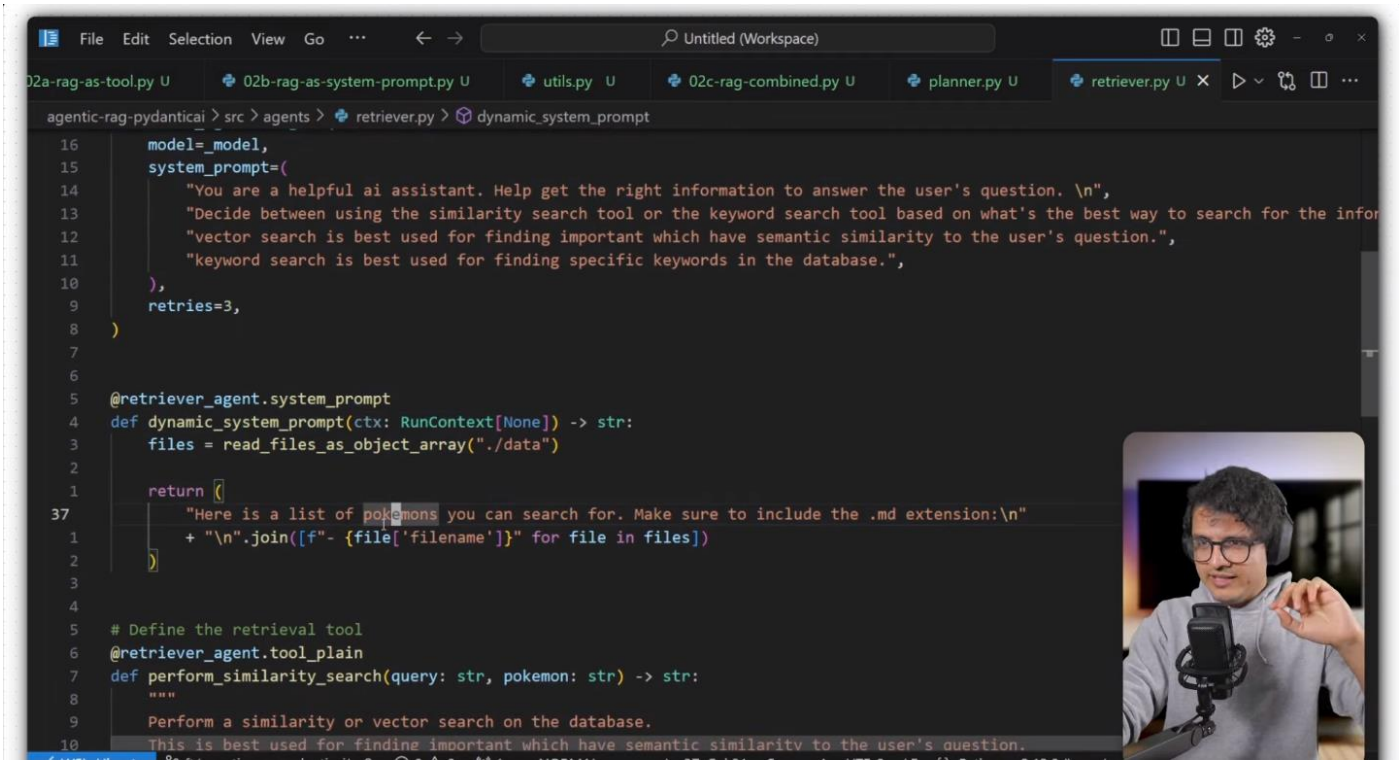


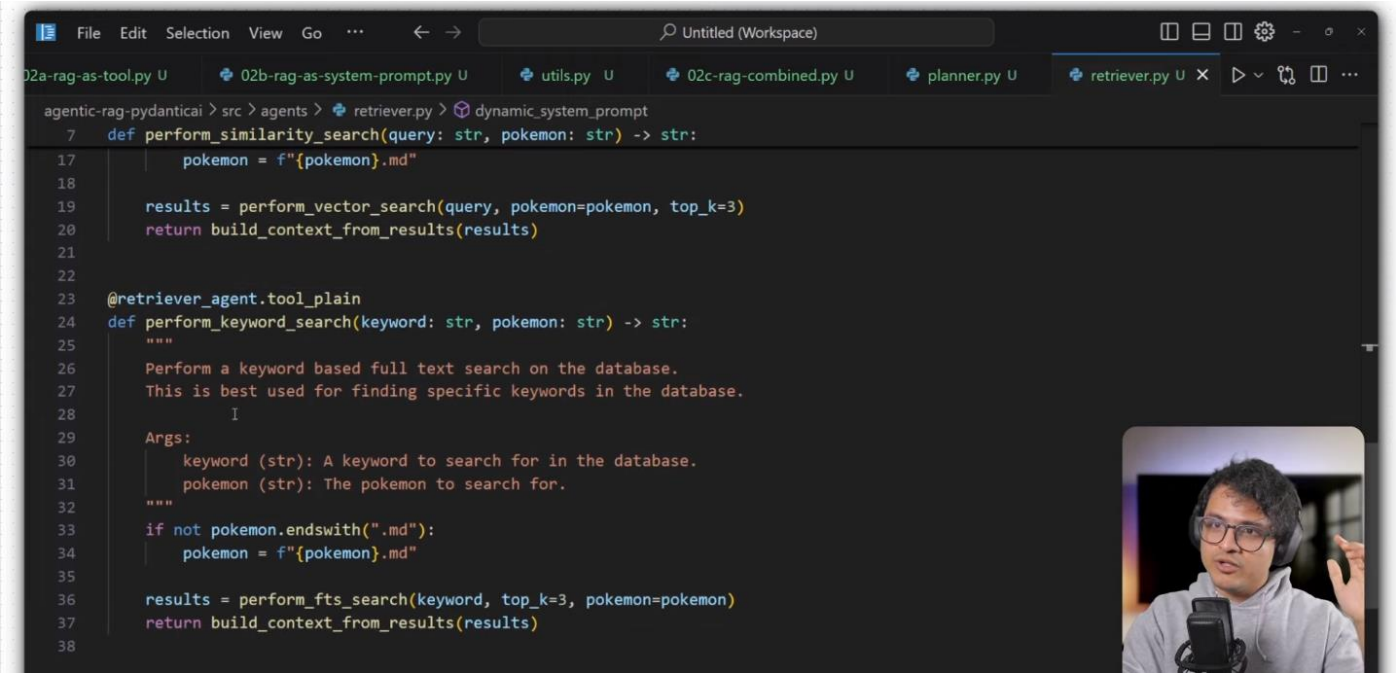
This agent will give us a structured output response



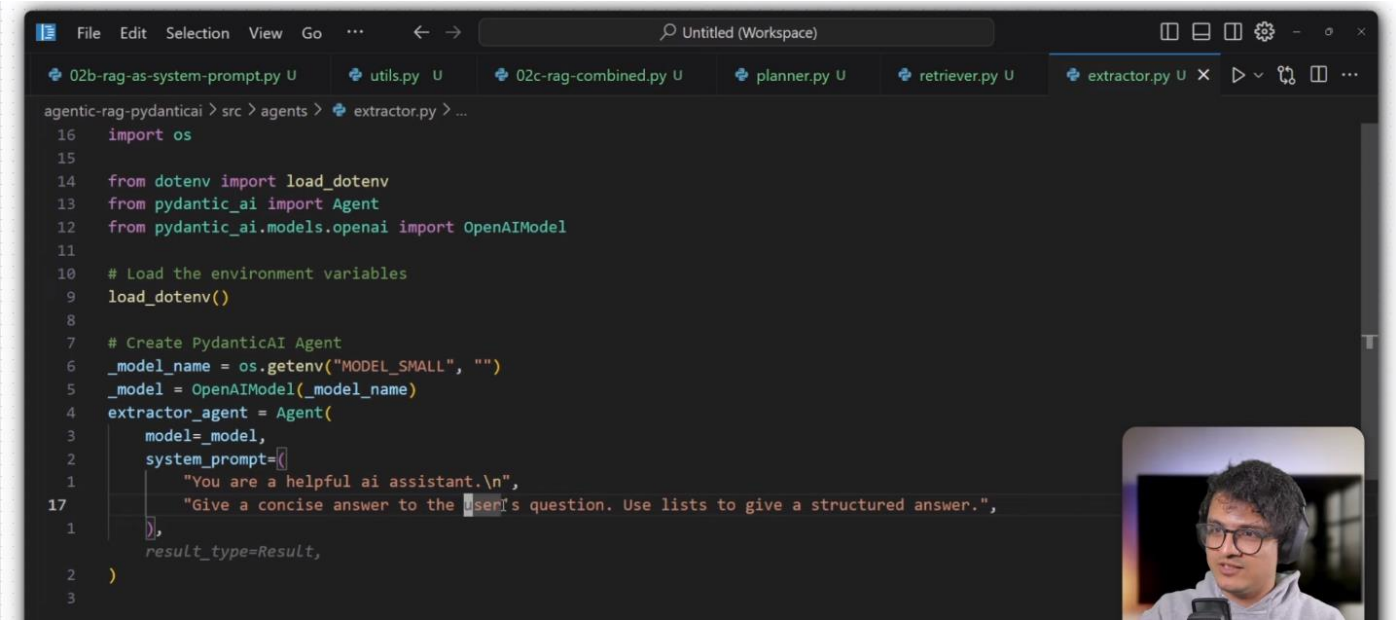
We can also do result validation in Pydantic as above


File Edit Selection View Go
Untitled (Workspace)
02a-rag-as-tool.py U
02b-rag-as-system-prompt.py U
utils.py U
02c-rag-combined.py U
planner.py U
retriever.py U
agentic-rag-pydanticai > src > agents > retriever.py > perform_keyword_search
import os
from dotenv import load_dotenv
from pydantic_ai import Agent, RunContext
from pydantic_ai.models.openai import OpenAIModel
from utils import (
build_context_from_results,
perform_fts_search,
perform_vector_search,
read_files_as_object_array,
)
# Load the environment variables
load_dotenv()
# Create PydanticAI Agent
_model_name = os.getenv("MODEL_MEDIUM", "")
_model = OpenAIModel(_model_name)
retriever_agent = Agent(
model=_model,
system_prompt=(
"You are a helpful ai assistant. Help get the right information to answer the user's question. \n",
"Decide between using the similarity search tool or the keyword search tool based on what's the best
"vector search is best used for finding important which have semantic similarity to the user's quest
"keyword search is best used for finding specific keywords in the database.",
),


File Edit Selection View Go
Untitled (Workspace)
02a-rag-as-tool.py U
02b-rag-as-system-prompt.py U
utils.py U
02c-rag-combined.py U
planner.py U
retriever.py U
agentic-rag-pydanticai > src > agents > retriever.py > dynamic_system_prompt
model=_model,
system_prompt=(
"You are a helpful ai assistant. Help get the right information to answer the user's question. \n",
"Decide between using the similarity search tool or the keyword search tool based on what's the best way to search for the infor
"vector search is best used for finding important which have semantic similarity to the user's question.",
"keyword search is best used for finding specific keywords in the database.",
),
retries=3,
)
@retriever_agent.system_prompt
def dynamic_system_prompt(ctx: RunContext[None]) -> str:
files = read_files_as_object_array("./data")
return (
"Here is a list of pokemons you can search for. Make sure to include the .md extension:\n"
+ "\n".join([f"- {file['filename']}" for file in files])
)
Define the retrieval tool
@retriever_agent.tool_plain
def perform_similarity_search(query: str, pokemon: str) -> str:
"""
Perform a similarity or vector search on the database.
This is best used for finding important which have semantic similarity to the user's question.

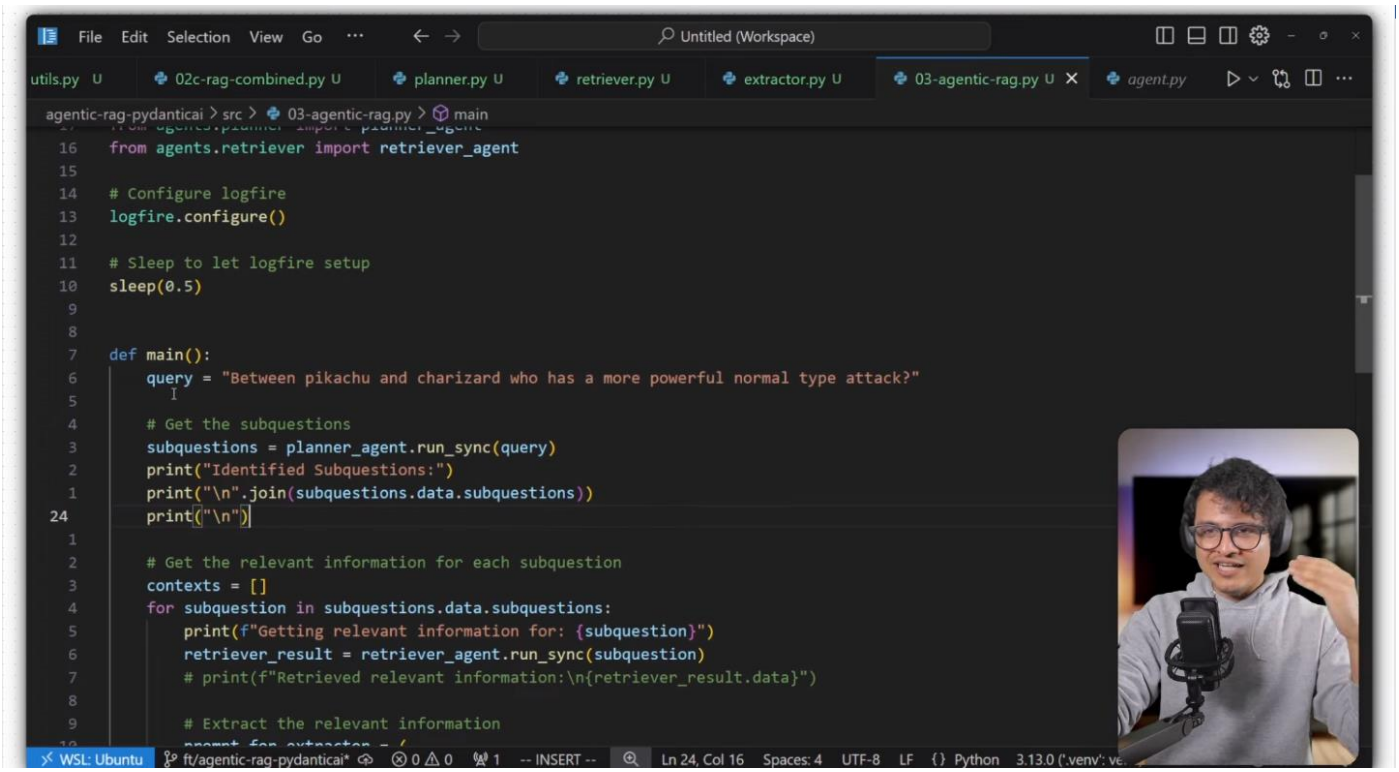


This is our previous retriever code

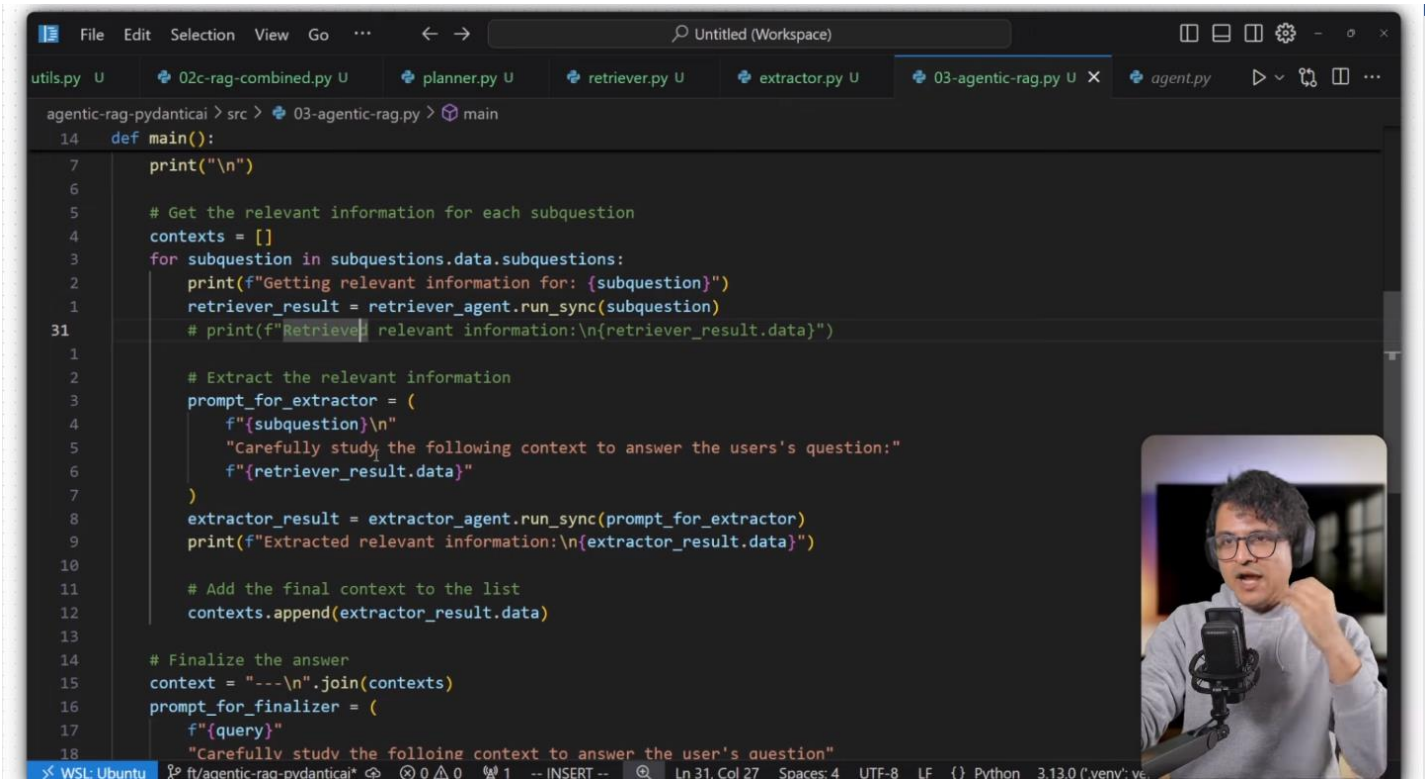


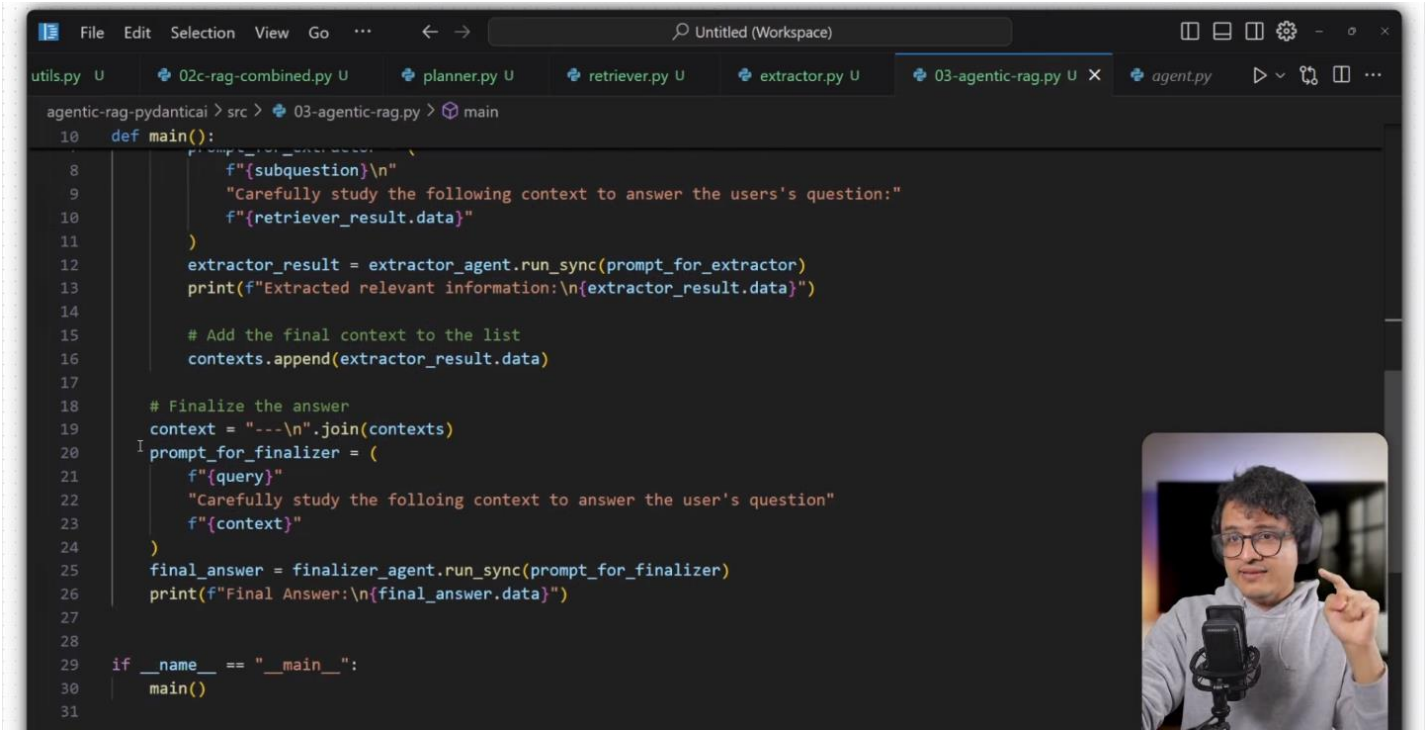
Then we have our extractor code

This is the deducer code



Now we can start the workflow code itself as above


File Edit Selection View Go
Untitled (Workspace)
utils.py U
02c-rag-combined.py U
planner.py U
retriever.py U
extractor.py U
03-agentic-rag.py U
agent.py
agentic-rag-pydanticai > src > 03-agentic-rag.py > main
def main():
print("\n")
# Get the relevant information for each subquestion
contexts = []
for subquestion in subquestions.data.subquestions:
print(f"Getting relevant information for: {subquestion}")
retriever_result = retriever_agent.run_sync(subquestion)
# print(f"Retrieved relevant information:\n{retriever_result.data}")
# Extract the relevant information
prompt_for_extractor = (
f"{subquestion}\n"
"Carefully study the following context to answer the users's question:"
f"{retriever_result.data}"
)
extractor_result = extractor_agent.run_sync(prompt_for_extractor)
print(f"Extracted relevant information:\n{extractor_result.data}")
# Add the final context to the list
contexts.append(extractor_result.data)
# Finalize the answer
context = "---\n".join(contexts)
prompt_for_finalizer = (
f"{query}"
"Carefully study the folloing context to answer the user's question"
WSL: Ubuntu
-- INSERT --
Ln 31, Col 27
Spaces: 4
UTF-8
LF
Python

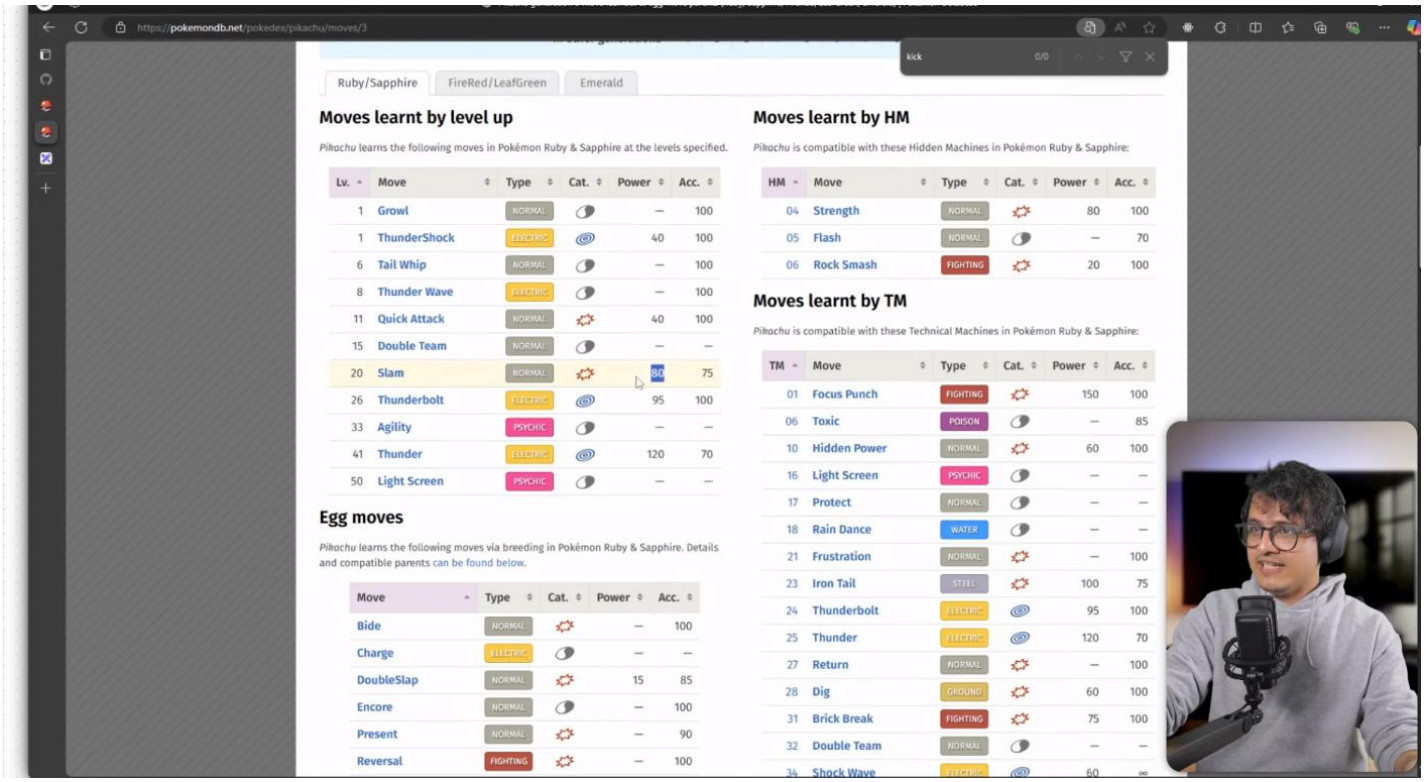

File Edit Selection View Go
Untitled (Workspace)
utils.py U
02c-rag-combined.py U
planner.py U
retriever.py U
extractor.py U
03-agentic-rag.py U
agent.py
agentic-rag-pydanticai > src > 03-agentic-rag.py > main
def main():
f"{subquestion}\n"
"Carefully study the following context to answer the users's question:"
f"{retriever_result.data}"
)
extractor_result = extractor_agent.run_sync(prompt_for_extractor)
print(f"Extracted relevant information:\n{extractor_result.data}")
# Add the final context to the list
contexts.append(extractor_result.data)
# Finalize the answer
context = "---\n".join(contexts)
prompt_for_finalizer = (
f"{query}"
"Carefully study the folloing context to answer the user's question"
f"{context}"
)
final_answer = finalizer_agent.run_sync(prompt_for_finalizer)
print(f"Final Answer:\n{final_answer.data}")
if __name__ == "__main__":
main()

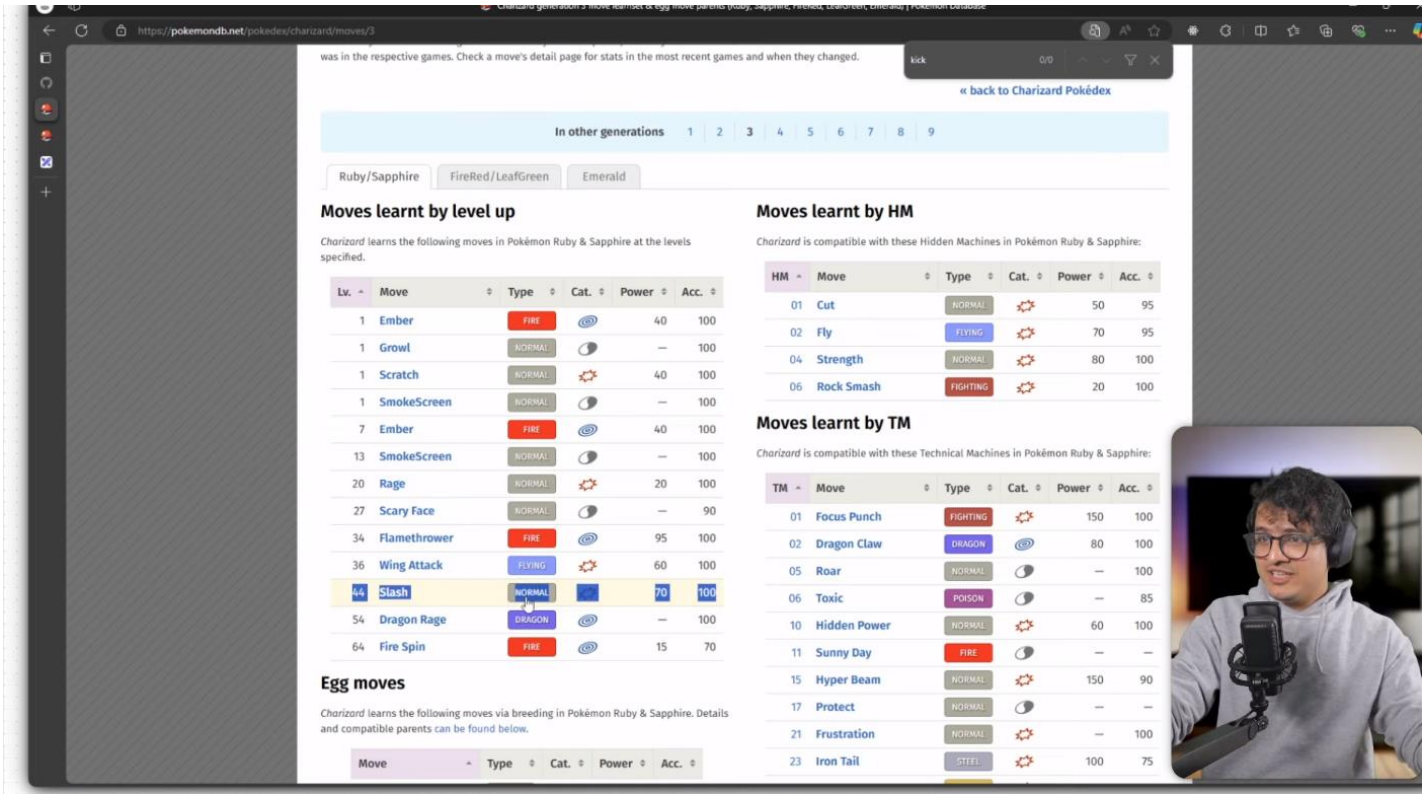
```
❯ uv run src/03-agentic-rag.py
Logfire project URL: https://logfire.pydantic.dev/yourtechbud/pydantic-101
03:24:58.708 planner_agent run prompt=Between pikachu and charizard who has a more powerful normal type attack?
03:24:58.709   preparing model and tools run_step=1
03:24:58.710   model request
03:25:06.214   handle model response
Identified Subquestions:
What normal-type attack does Pikachu have?
What is the power of Pikachu's normal-type attack?
What is the highest level at which Pikachu can learn a normal-type attack?
What normal-type attack does Charizard have?
What is the power of Charizard's normal-type attack?
What is the highest level at which Charizard can learn a normal-type attack?


Getting relevant information for: What normal-type attack does Pikachu have?
03:25:06.215 retriever_agent run prompt=What normal-type attack does Pikachu have?
03:25:06.216   preparing model and tools run_step=1
03:25:06.217   model request
03:25:11.271   handle model response
03:25:11.272     running tools=['perform_keyword_search']
03:25:11.351 extractor_agent run prompt=What normal-type attack does Pikachu have?
Carefully study the...le with these Hidden Machines in Pokémon FireRed & LeafGreen:

03:25:11.353   preparing model and tools run_step=1
03:25:11.353   model request
03:25:13.799   handle model response
Extracted relevant information:
Based on the given context, here is a list of normal-type attacks that Pikachu has:
```

```
Identified Subquestions:
What is the power of Pikachu's strongest normal type attack?
What is the power of Charizard's strongest normal type attack?


Getting relevant information for: What is the power of Pikachu's strongest normal type attack?
03:30:10.566 retriever_agent run prompt=What is the power of Pikachu's strongest normal type attack?
03:30:10.567   preparing model and tools run_step=1
03:30:10.567   model request
03:30:15.643   handle model response
03:30:15.645     running tools=['perform_keyword_search']
03:30:15.670 extractor_agent run prompt=What is the power of Pikachu's strongest normal type attack?
C...d compatible parents [can be found below](#egg-move-parents).

03:30:15.672   preparing model and tools run_step=1
03:30:15.672   model request
03:30:19.485   handle model response
Extracted relevant information:
Based on the given context, the strength of a Pikachu's Normal-type attack can be evaluated

### Normal-type attacks

+ **Growl**:   Not applicable (configures status conditions rather than dealing damage)
+ **ThunderShock**:   40 points of attack power (but it is electric-type move so it is not
+ **Tail Whip**:  Not applicable (configures status conditions rather than dealing damage)
+ **Thunder Wave**:  Not applicable (configures status conditions rather than dealing damag
+ **Quick Attack**:  40 points of attack power
+ **Double Team**:   Not applicable (deals no damage by itself; possibly puts the target to
 documented)
```

```
+ **Double Team**:  Not applicable (deals no damage by itself; possibly puts the target to sleep, which is not
 documented)
+ **Slam**:  80 points of attack power
+ **Agility**:  Not applicable (a status move with no direct attack power; as described in the Pokémon games, t
his move can lower an opponent's speed but that is purely mana-based)
+ **Thunder**:  (Electric-type, not used for the answer)
+ **Light Screen**:  Not applicable (a status move with no direct attack power).

The most powerful Normal-type attack available and learned by Pikachu in Pokémon Ruby & Sapphire and Pokémon Fi
reRed & LeafGreen at the given levels is **Slam** which is 80 points of attack power, followed by **Quick Attac
k** which is also 40 points of attack power.
Getting relevant information for: What is the power of Charizard's strongest normal type attack?
03:30:19.488 retriever_agent run prompt=What is the power of Charizard's strongest normal type attack?
03:30:19.489   preparing model and tools run_step=1
03:30:19.490   model request
03:30:24.442   handle model response
03:30:24.443     running tools=['perform_keyword_search']
03:30:24.452 extractor_agent run prompt=What is the power of Charizard's strongest normal t
atible parents [can be found below](#egg-move-parents).

03:30:24.454   preparing model and tools run_step=1
03:30:24.454   model request
03:30:26.087   handle model response
Extracted relevant information:
**Strongest Normal Type Attack of Charizard:**
- Charizard's strongest normal type attack is **Slash**.
- Slash's power is **70**, which is the highest power among Charizard's normal type attacks
03:30:26.090 finalizer_agent run prompt=Between pikachu and charizard who has a more powerf
 power among Charizard's normal type attacks in Pokémon.
```

```
03:25:58.935   model request
03:26:10.106   handle model response
Final Answer:
To answer the user's question regarding which of Pikachu or Charizard has a more powerful normal-type attack, I
'll compare the information provided for the normal-type moves of each Pokémon.

**Pikachu:**
- Normal-type attacks listed include: Body Slam (Power: 85), Double-Edge (Power: 120), Endure (no damage), Mega
 Kick (Power: 120), Mega Punch (Power: 80), Mimic (no damage), Substitute (no damage), Bide (no specified power
), and Slam (Power: 80).
- The most powerful normal-type attack Pikachu can learn is **Mega Kick** or **Double-Edge**, both with a power
 of **120**.

**Charizard:**
- Normal-type attacks listed include: Body Slam (Power: 85), Double-Edge (Power: 120), Endure (no damage), Mega
 Kick (Power: 120), Mega Punch (Power: 80), Mimic (no damage), Sleep Talk (no damage), Snore (Power: 40), Subst
itute (no damage), Swords Dance (no damage).
- The most powerful normal-type attack Charizard can learn is **Mega Kick** or **Double-Edg
er of **120**.

**Conclusion:**
Both Pikachu and Charizard have the same most powerful normal-type attack, which is **Doubl
ick**, both with a power of **120**. Therefore, they have equally powerful normal-type atta
l.
❯
❯

~/work/ytb-codes/ytb-practical-guide/agentic-rag-pydanticai ft/agentic-rag-pydanticai *1
❯
```


https://pokemondb.net/pokedex/pikachu/moves/3
kick
0/0
Ruby/Sapphire
FireRed/LeafGreen
Emerald
Moves learnt by level up
Pikachu learns the following moves in Pokémon Ruby & Sapphire at the levels specified.
Lv. Move Type Cat. Power Acc.
1 Growl NORMAL — 100
1 ThunderShock ELECTRIC 40 100
6 Tail Whip NORMAL — 100
8 Thunder Wave ELECTRIC — 100
11 Quick Attack NORMAL 40 100
15 Double Team NORMAL — —
20 Slam NORMAL 80 75
26 Thunderbolt ELECTRIC 95 100
33 Agility PSYCHIC — —
41 Thunder ELECTRIC 120 70
50 Light Screen PSYCHIC — —
Egg moves
Pikachu learns the following moves via breeding in Pokémon Ruby & Sapphire. Details and compatible parents can be found below.
Move Type Cat. Power Acc.
Bide NORMAL — 100
Charge ELECTRIC — —
DoubleSlap NORMAL 15 85
Encore NORMAL — 100
Present NORMAL — 90
Reversal FIGHTING — 100
Moves learnt by HM
Pikachu is compatible with these Hidden Machines in Pokémon Ruby & Sapphire:
HM Move Type Cat. Power Acc.
04 Strength NORMAL 80 100
05 Flash NORMAL — 70
06 Rock Smash FIGHTING 20 100
Moves learnt by TM
Pikachu is compatible with these Technical Machines in Pokémon Ruby & Sapphire:
TM Move Type Cat. Power Acc.
01 Focus Punch FIGHTING 150 100
06 Toxic POISON — 85
10 Hidden Power NORMAL 60 100
16 Light Screen PSYCHIC — —
17 Protect NORMAL — —
18 Rain Dance WATER — —
21 Frustration NORMAL — 100
23 Iron Tail STEEL 100 75
24 Thunderbolt ELECTRIC 95 100
25 Thunder ELECTRIC 120 70
27 Return NORMAL — 100
28 Dig GROUND 60 100
31 Brick Break FIGHTING 75 100
32 Double Team NORMAL — —
34 Shock Wave ELECTRIC 60 ∞


https://pokemondb.net/pokedex/charizard/moves/3
was in the respective games. Check a move's detail page for stats in the most recent games and when they changed.
kick
0/0
« back to Charizard Pokédex
In other generations 1 2 3 4 5 6 7 8 9
Ruby/Sapphire
FireRed/LeafGreen
Emerald
Moves learnt by level up
Charizard learns the following moves in Pokémon Ruby & Sapphire at the levels specified.
Lv. Move Type Cat. Power Acc.
1 Ember FIRE 40 100
1 Growl NORMAL — 100
1 Scratch NORMAL 40 100
1 SmokeScreen NORMAL — 100
7 Ember FIRE 40 100
13 SmokeScreen NORMAL — 100
20 Rage NORMAL 20 100
27 Scary Face NORMAL — 90
34 Flamethrower FIRE 95 100
36 Wing Attack FLYING 60 100
44 Slash NORMAL 70 100
54 Dragon Rage DRAGON — 100
64 Fire Spin FIRE 15 70
Egg moves
Charizard learns the following moves via breeding in Pokémon Ruby & Sapphire. Details and compatible parents can be found below.
Move Type Cat. Power Acc.
Moves learnt by HM
Charizard is compatible with these Hidden Machines in Pokémon Ruby & Sapphire:
HM Move Type Cat. Power Acc.
01 Cut NORMAL 50 95
02 Fly FLYING 70 95
04 Strength NORMAL 80 100
06 Rock Smash FIGHTING 20 100
Moves learnt by TM
Charizard is compatible with these Technical Machines in Pokémon Ruby & Sapphire:
TM Move Type Cat. Power Acc.
01 Focus Punch FIGHTING 150 100
02 Dragon Claw DRAGON 80 100
05 Roar NORMAL — 100
06 Toxic POISON — 85
10 Hidden Power NORMAL 60 100
11 Sunny Day FIRE — —
15 Hyper Beam NORMAL 150 90
17 Protect NORMAL — —
21 Frustration NORMAL — 100
23 Iron Tail STEEL 100 75

```
03:30:26.090 finalizer_agent run prompt=Between pikachu and charizard who has a more powerful normal t...ighest
 power among Charizard's normal type attacks in Pokémon.
03:30:26.090   preparing model and tools run_step=1
03:30:26.090   model request
03:30:33.093   handle model response
Final Answer:
To determine which Pokémon has a more powerful Normal-type attack between Pikachu and Charizard, we need to com
pare their strongest Normal-type moves:

1. **Pikachu's Normal-type Attacks**:
   - **Quick Attack**: 40 points of attack power.
   - **Slam**: 80 points of attack power.

   Therefore, Pikachu's most powerful Normal-type attack is **Slam**, with an attack power of **80**.

2. **Charizard's Normal-type Attacks**:
   - **Slash**: 70 points of attack power.

   Therefore, Charizard's most powerful Normal-type attack is **Slash**, with an attack pow

Comparing these two moves:
- **Pikachu's Slam**: 80 points of attack power.
- **Charizard's Slash**: 70 points of attack power.

Hence, **Pikachu** has a more powerful Normal-type attack with **Slam**, which has 80 point
ompared to Charizard's **Slash**, which has 70 points of attack power.

~/work/ytb-codes/ytb-practical-guide/agentic-rag-pydanticai ft/agentic-rag-pydanticai *1
❯
```

File Edit Selection View Go ··· Untitled (Workspace)

utils.py U | 02c-rag-combined.py U | planner.py U | retriever.py U | extractor.py U | 03-agentic-rag.py U | agent.py

agentic-rag-pydanticai > src > agents > retriever.py > retriever_agent

```python
retriever_agent = Agent(
    model=_model,
    system_prompt=(
        "You are a helpful ai assistant. Help get the right information to answer the user's question. \n",
        "Decide between using the similarity search tool or the keyword search tool based on what's the best way to search for the infor
        "vector search is best used for finding important which have semantic similarity to the user's question.",
        "keyword search is best used for finding specific keywords in the database.",
    ),
    retries=3,
)


@retriever_agent.system_prompt
def dynamic_system_prompt(ctx: RunContext[None]) -> str:
    files = read_files_as_object_array("./data")

    return (
        "Here is a list of pokemons you can search for. Make sure to include the .md extension:\n"
        + "\n".join([f"- {file['filename']}" for file in files])
    )


# Define the retrieval tool
@retriever_agent.tool_plain
def perform_similarity_search(query: str, pokemon: str) -> str:
    """
    Perform a similarity or vector search on the database.
```

utils.py U | 02c-rag-combined.py U | planner.py U | retriever.py U | extractor.py U | 03-agentic-rag.py U | agent.py

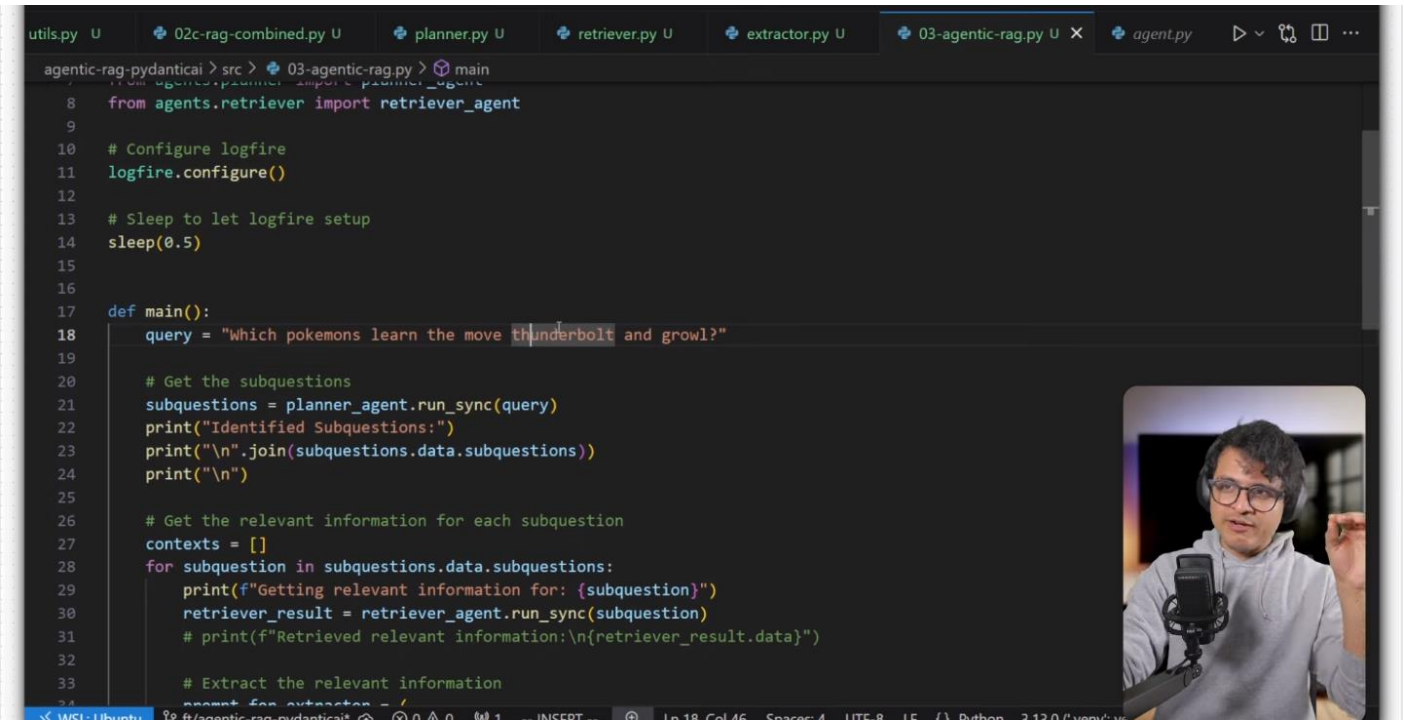
agentic-rag-pydanticai > src > agents > retriever.py > perform_keyword_search

```python

# @retriever_agent.system_prompt
# def dynamic_system_prompt(ctx: RunContext[None]) -> str:
#     files = read_files_as_object_array("./data")

#     return (
#         "Here is a list of pokemons you can search for. Make sure to include the .md extension:\n"
#         + "\n".join([f"- {file['filename']}" for file in files])
#     )


# Define the retrieval tool
@retriever_agent.tool_plain
def perform_similarity_search(query: str) -> str:
    """
    Perform a similarity or vector search on the database.
    This is best used for finding important which have semantic similarity to the user's question.

    Args:
        query (str): A concise question for which you need to get the most relevant information.
    """

    results = perform_vector_search(query, top_k=3)
    return build_context_from_results(results)
```

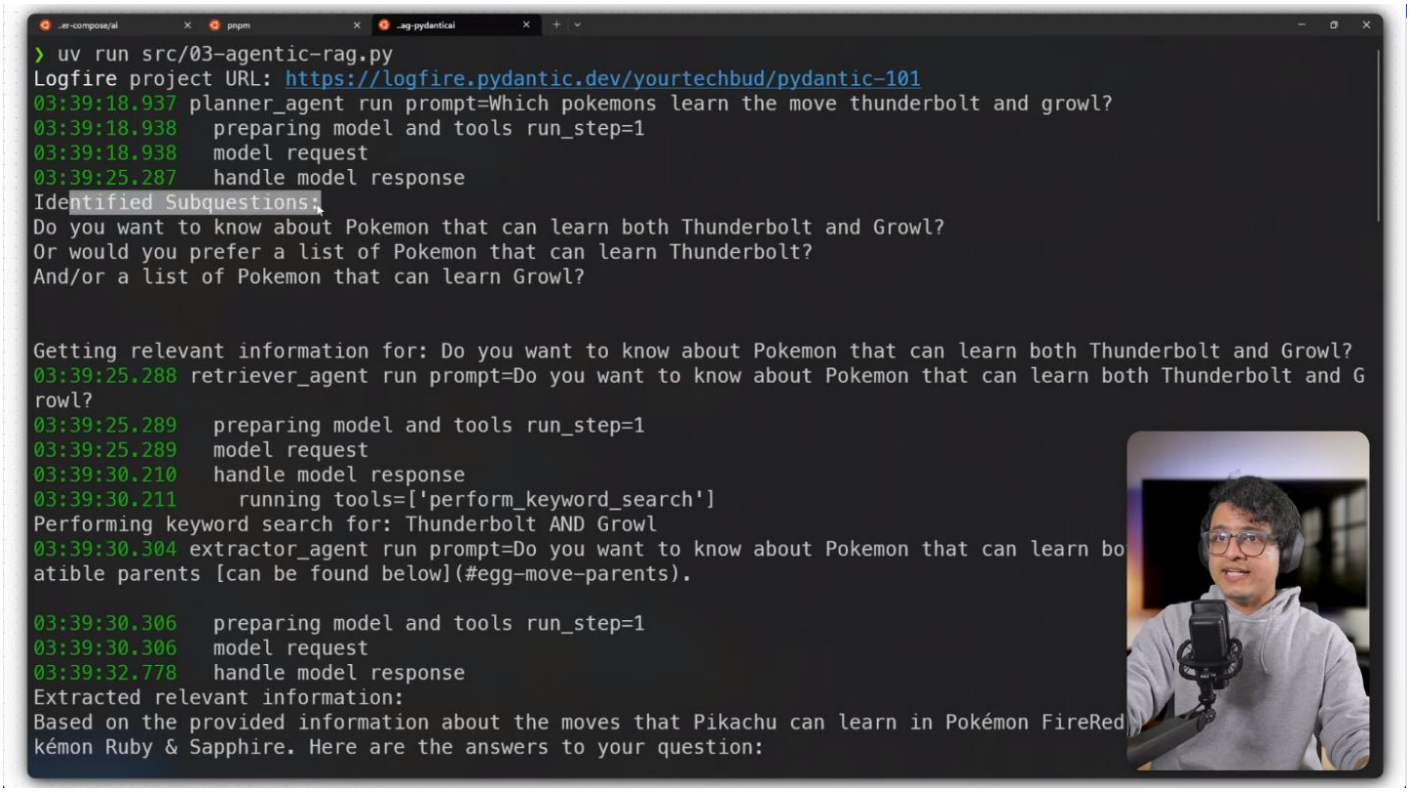
utils.py U | 02c-rag-combined.py U | planner.py U | retriever.py U | extractor.py U | 03-agentic-rag.py U | agent.py

agentic-rag-pydanticai > src > agents > retriever.py > perform_keyword_search

```python
def perform_similarity_search(query: str) -> str:
    This is best used for finding important which have semantic similarity to the user's question.

    Args:
        query (str): A concise question for which you need to get the most relevant information.
    """

    results = perform_vector_search(query, top_k=3)
    return build_context_from_results(results)


@retriever_agent.tool_plain
def perform_keyword_search(keyword: str, pokemon: str) -> str:
    """
    Perform a keyword based full text search on the database.
    This is best used for finding specific keywords in the database.

    Args:
        keyword (str): A keyword to search for in the database.
    """

    results = perform_fts_search(keyword, top_k=3)
    return build_context_from_results(results)
```



This will require scans across all pokemons

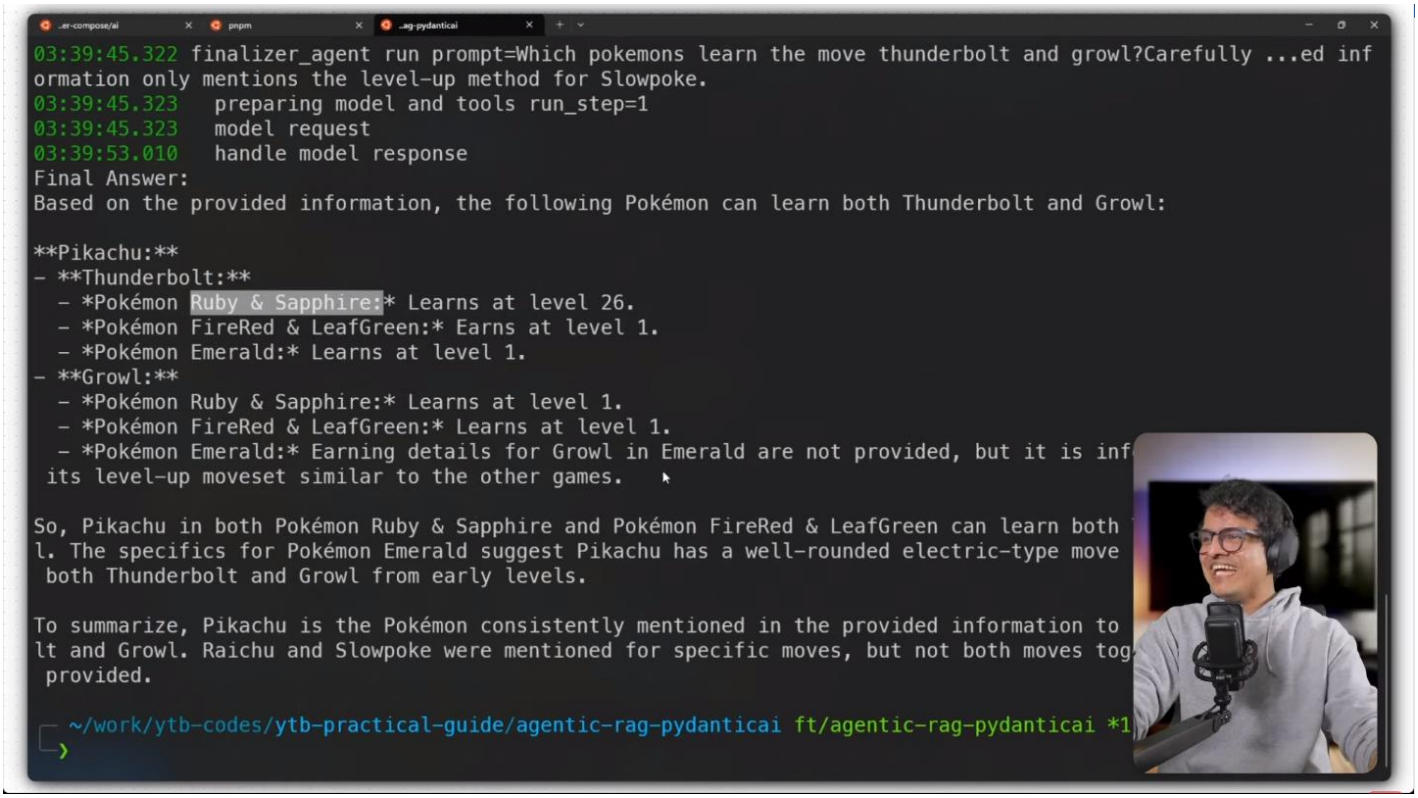


```
atible parents [can be found below](#egg-move-parents).

03:39:30.306   preparing model and tools run_step=1
03:39:30.306   model request
03:39:32.778   handle model response
Extracted relevant information:
Based on the provided information about the moves that Pikachu can learn in Pokémon FireRed & LeafGreen, and Po
kémon Ruby & Sapphire. Here are the answers to your question:

**Pokémon that can learn both Thunderbolt and Growl in Pokémon FireRed & LeafGreen:**

* Pikachu

**Reason:** According to the information provided, Pikachu learns Thunderbolt at level 26 and Growl at level 1
in Pokémon FireRed & LeafGreen.

**Pokémon that can learn both Thunderbolt and Growl in Pokémon Ruby & Sapphire:**

* Pikachu

**Reason:** According to the information provided, Pikachu learns Thunderbolt at level 26 a
in Pokémon Ruby & Sapphire.
Getting relevant information for: Or would you prefer a list of Pokemon that can learn Thun
03:39:32.781 retriever_agent run prompt=Or would you prefer a list of Pokemon that can lear
03:39:32.782   preparing model and tools run_step=1
03:39:32.782   model request
03:39:37.217   handle model response
03:39:37.218     running tools=['perform_keyword_search']
Performing keyword search for: Thunderbolt
```

```
**Pokémon that can learn both Thunderbolt and Growl in Pokémon FireRed & LeafGreen:**

* Pikachu

**Reason:** According to the information provided, Pikachu learns Thunderbolt at level 26 and Growl at level 1
in Pokémon FireRed & LeafGreen.

**Pokémon that can learn both Thunderbolt and Growl in Pokémon Ruby & Sapphire:**

* Pikachu

**Reason:** According to the information provided, Pikachu learns Thunderbolt at level 26 and Growl at level 1
in Pokémon Ruby & Sapphire.
Getting relevant information for: Or would you prefer a list of Pokemon that can learn Thunderbolt?
03:39:32.781 retriever_agent run prompt=Or would you prefer a list of Pokemon that can learn Thunderbolt?
03:39:32.782   preparing model and tools run_step=1
03:39:32.782   model request
03:39:37.217   handle model response
03:39:37.218     running tools=['perform_keyword_search']
Performing keyword search for: Thunderbolt
03:39:37.228 extractor_agent run prompt=Or would you prefer a list of Pokemon that can lear
 tutors ([details](/mechanics/move-tutors#tutor-frlg)):

03:39:37.230   preparing model and tools run_step=1
03:39:37.230   model request
03:39:39.424   handle model response
Extracted relevant information:
Based on the provided information, here is a list of Pokémon that can learn Thunderbolt:
```

```
03:39:37.218     running tools=['perform_keyword_search']
Performing keyword search for: Thunderbolt
03:39:37.228 extractor_agent run prompt=Or would you prefer a list of Pokemon that can learn Thunderbo...m move
 tutors ([details](/mechanics/move-tutors#tutor-frlg)):

03:39:37.230   preparing model and tools run_step=1
03:39:37.230   model request
03:39:39.424   handle model response
Extracted relevant information:
Based on the provided information, here is a list of Pokémon that can learn Thunderbolt:

1. **Pikachu**:
   - In Pokémon Ruby & Sapphire: Learns Thunderbolt at level 26.
   - In Pokémon FireRed & LeafGreen: Learns Thunderbolt at level 1.
   - In Pokémon Emerald: Learns Thunderbolt at level 1.

2. **Raichu**:
   - In Pokémon FireRed & LeafGreen: Learns Thunderbolt at level 1.
   - In Pokémon Emerald: Learns Thunderbolt at level 1.
Getting relevant information for: And/or a list of Pokemon that can learn Growl?
03:39:39.427 retriever_agent run prompt=And/or a list of Pokemon that can learn Growl?
03:39:39.427   preparing model and tools run_step=1
03:39:39.427   model request
03:39:43.452   handle model response
03:39:43.453     running tools=['perform_keyword_search']
Performing keyword search for: Growl
03:39:43.459 extractor_agent run prompt=And/or a list of Pokemon that can learn Growl?
Carefully study...d compatible parents [can be found below](#egg-move-parents).
```

```
2. **Raichu**:
   - In Pokémon FireRed & LeafGreen: Learns Thunderbolt at level 1.
   - In Pokémon Emerald: Learns Thunderbolt at level 1.
Getting relevant information for: And/or a list of Pokemon that can learn Growl?
03:39:39.427 retriever_agent run prompt=And/or a list of Pokemon that can learn Growl?
03:39:39.427   preparing model and tools run_step=1
03:39:39.427   model request
03:39:43.452   handle model response
03:39:43.453     running tools=['perform_keyword_search']
Performing keyword search for: Growl
03:39:43.459 extractor_agent run prompt=And/or a list of Pokemon that can learn Growl?
Carefully study...d compatible parents [can be found below](#egg-move-parents).

03:39:43.460   preparing model and tools run_step=1
03:39:43.461   model request
03:39:45.320   handle model response
Extracted relevant information:
Based on the provided information, the Pokémon species that can learn Growl are:

1. **Slowpoke**
   - Slowpoke can learn Growl at level 6 in Pokémon Ruby & Sapphire, Pokémon Emerald, and P
fGreen.

Please note that other Pokémon species might also learn Growl through other methods, such a
ing. However, the provided information only mentions the level-up method for Slowpoke.
03:39:45.322 finalizer_agent run prompt=Which pokemons learn the move thunderbolt and growl
ormation only mentions the level-up method for Slowpoke.
03:39:45.323   preparing model and tools run_step=1
```


_er-compose/ai
pnpm
_ag-pydanticai
03:39:45.322 finalizer_agent run prompt=Which pokemons learn the move thunderbolt and growl?Carefully ...ed inf
ormation only mentions the level-up method for Slowpoke.
03:39:45.323 preparing model and tools run_step=1
03:39:45.323 model request
03:39:53.010 handle model response
Final Answer:
Based on the provided information, the following Pokémon can learn both Thunderbolt and Growl:
Pikachu:
- **Thunderbolt:**
- *Pokémon Ruby & Sapphire:* Learns at level 26.
- *Pokémon FireRed & LeafGreen:* Earns at level 1.
- *Pokémon Emerald:* Learns at level 1.
- **Growl:**
- *Pokémon Ruby & Sapphire:* Learns at level 1.
- *Pokémon FireRed & LeafGreen:* Learns at level 1.
- *Pokémon Emerald:* Earning details for Growl in Emerald are not provided, but it is inf
its level-up moveset similar to the other games.
So, Pikachu in both Pokémon Ruby & Sapphire and Pokémon FireRed & LeafGreen can learn both
l. The specifics for Pokémon Emerald suggest Pikachu has a well-rounded electric-type move
both Thunderbolt and Growl from early levels.
To summarize, Pikachu is the Pokémon consistently mentioned in the provided information to
lt and Growl. Raichu and Slowpoke were mentioned for specific moves, but not both moves tog
provided.
~/work/ytb-codes/ytb-practical-guide/agentic-rag-pydanticai ft/agentic-rag-pydanticai *1