Pontificia Universidad JAVERIANA Colombia

GenAI at the Edge Laboratory

Prof. Marcelo J. Rovai

rovai@unifei.edu.br

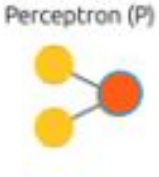
UNIFEI - Federal University of Itajuba, Brazil

TinyML4D - Academic Network Co-Chair

EdgeAIP - Academia-Industry Partnership Co-Chair

LLM / SLM

Large Language Model / Small Language Models



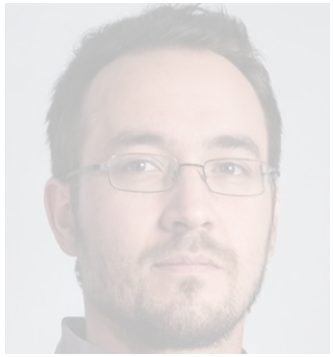
Neural Network Architectures

Vibration Analysis

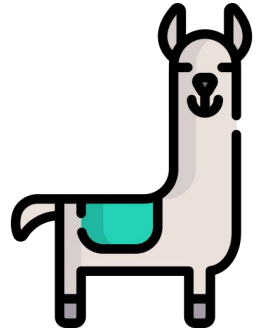
Image Classification

Text Generation

Image Generation



Large Language Models- LLMs



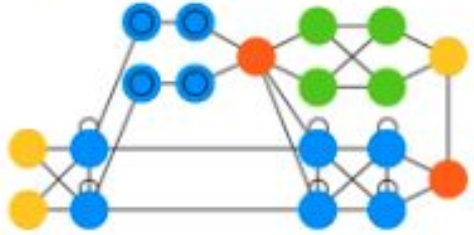
MLP - Deep Neural Network

CNN - Convolutional NN

RNN - Recurrent NN (GRU/LSTM)

GAN - Generative Adversarial N.

AN - Attention (Transformers)



Transformers to LLMs and SLMs

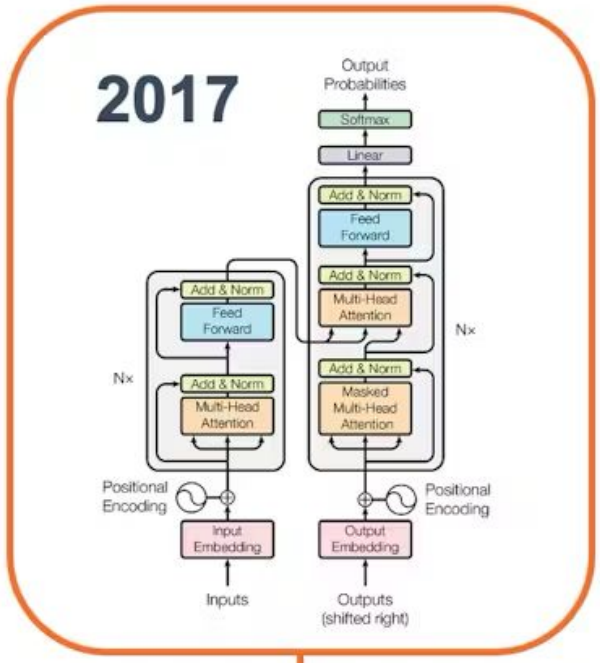
2025

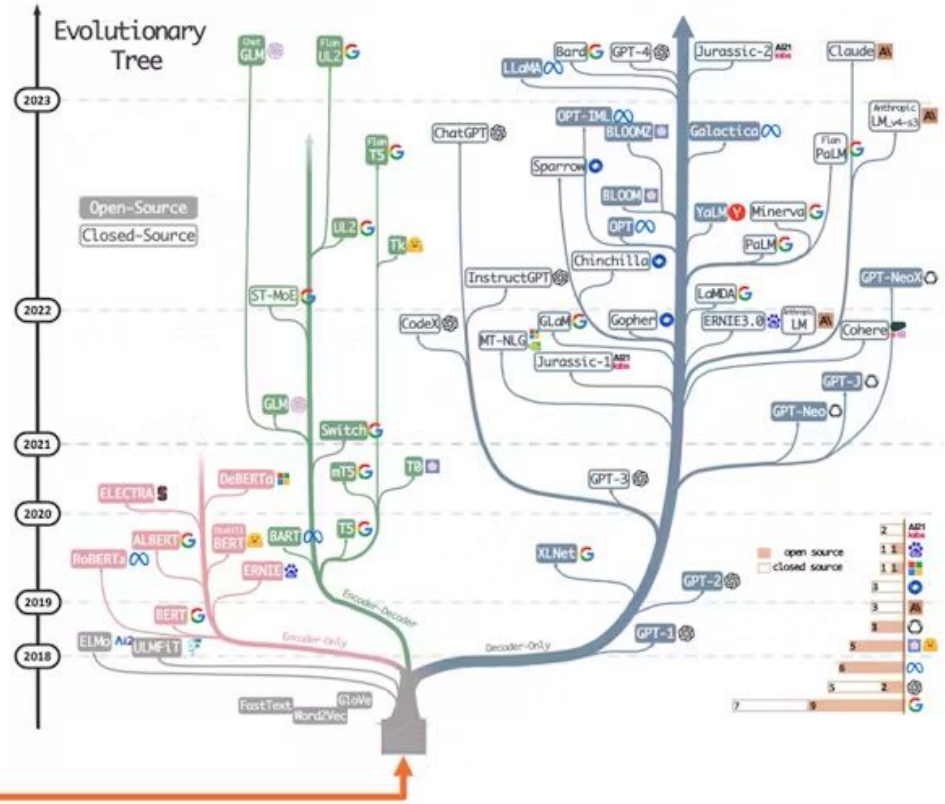
Open

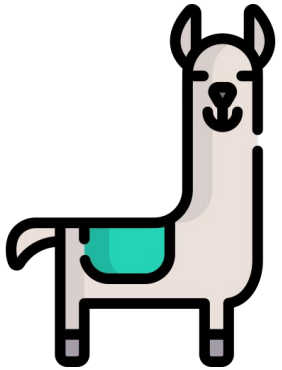
Qwen DeepSeek Phi Gemma LlaMa

Closed

Gemini GPT Claude Grok

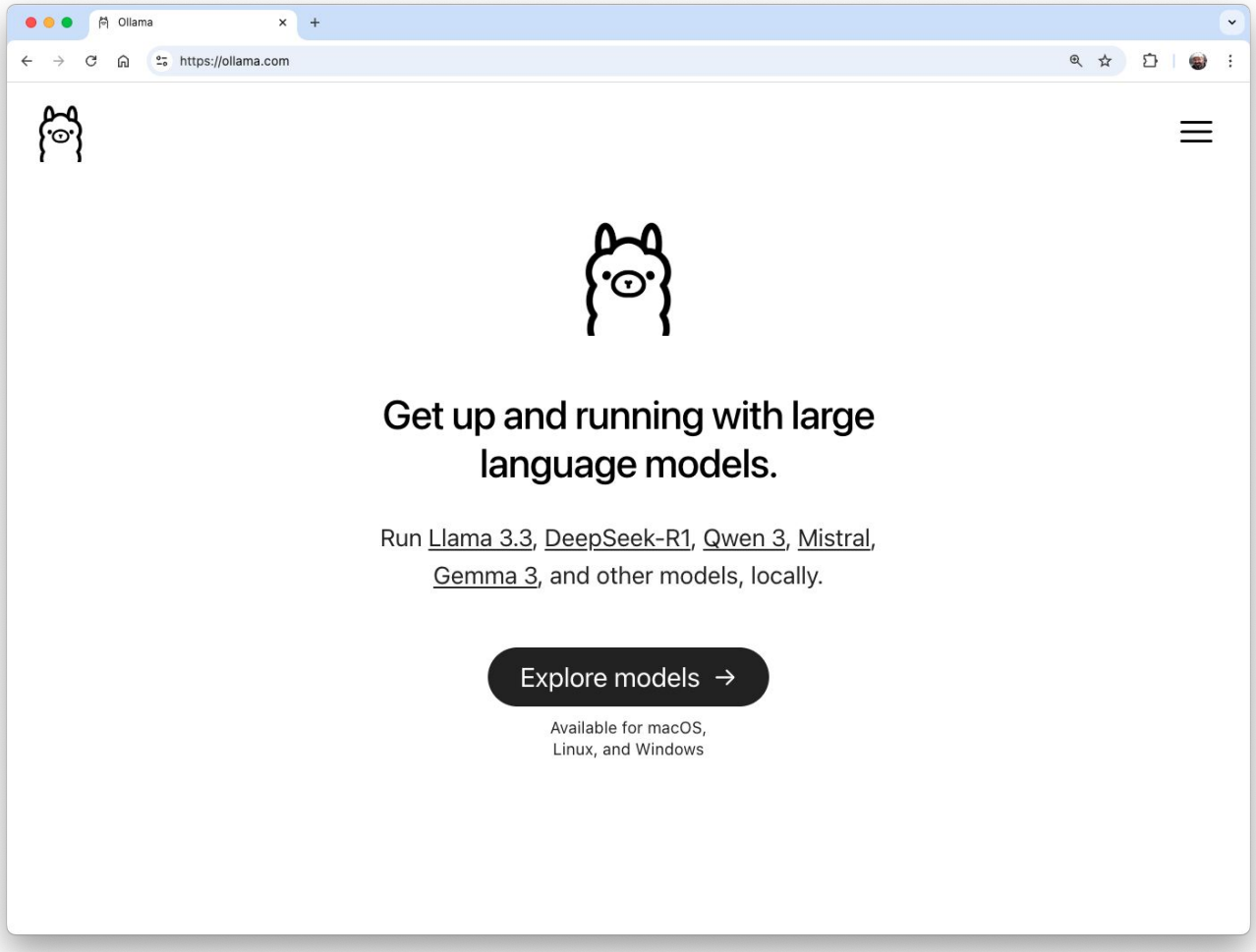






Labs – Material

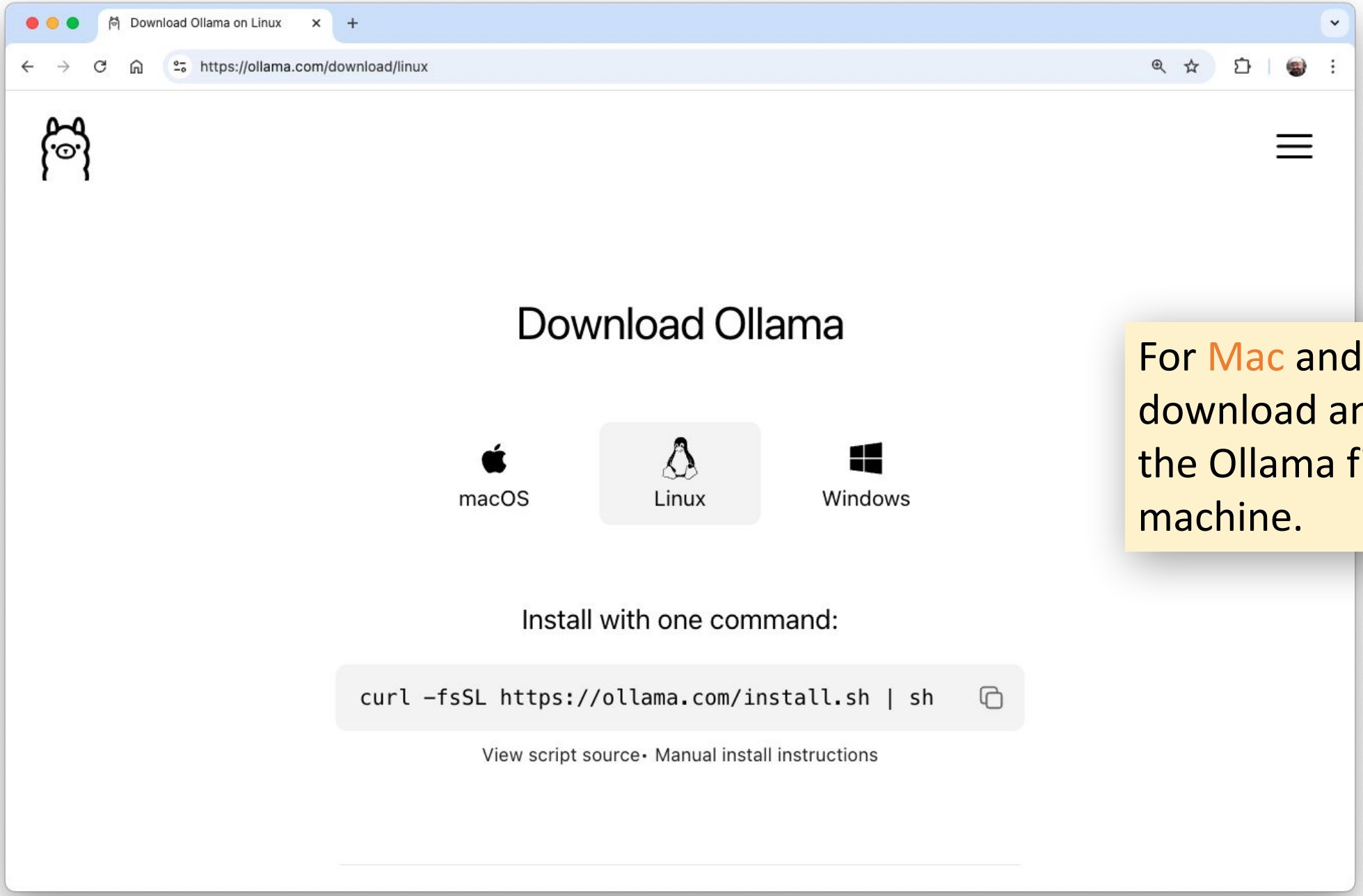
Running SLMs at the Edge



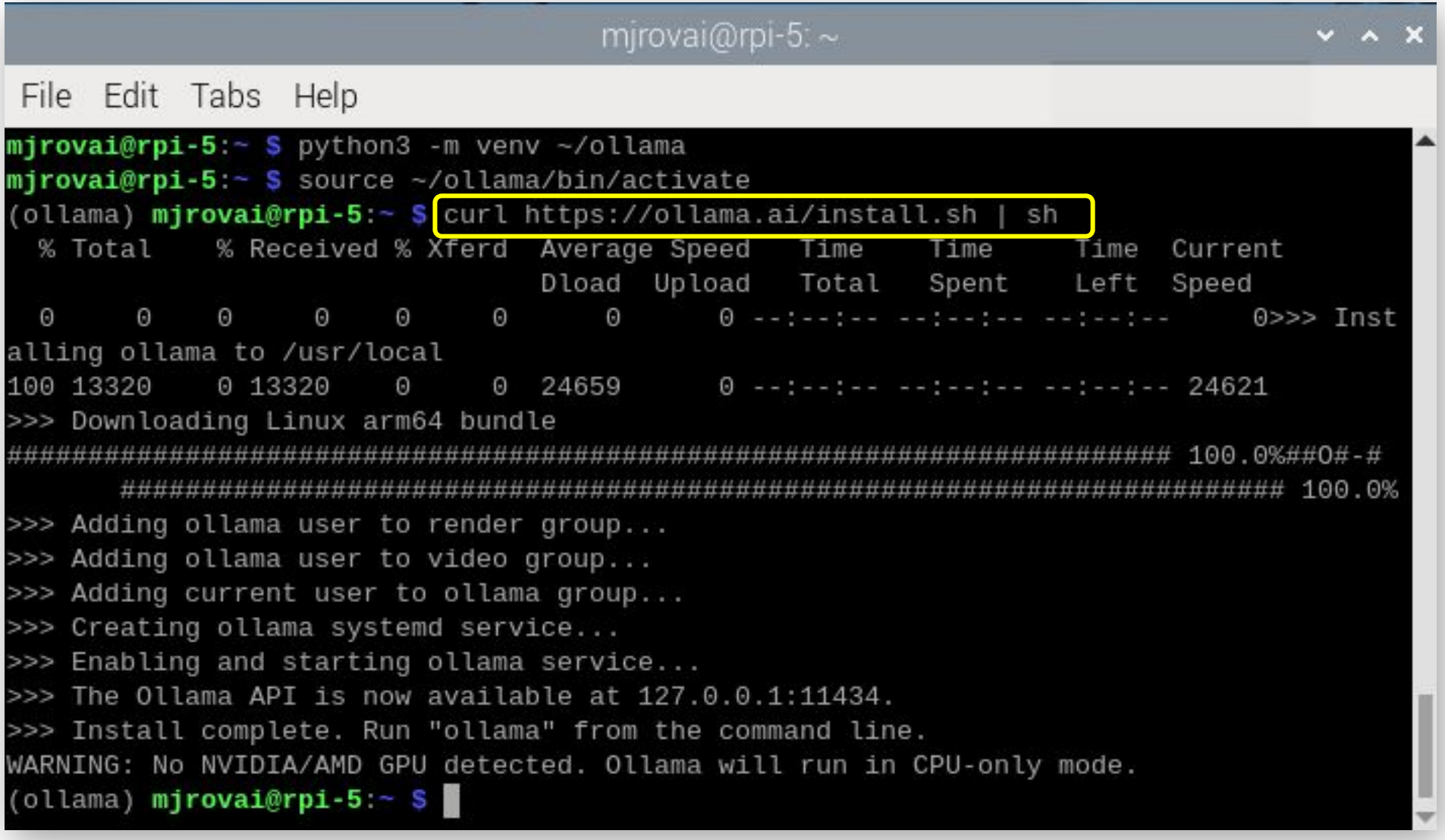
The primary and most user-friendly tool for running Small Language Models (SLMs) directly on a Raspberry Pi, especially the Pi 5, is Ollama. It is an open-source framework that allows you to install, manage, and run various SLMs (such as TinyLlama, smollm, Microsoft Phi, Google Gemma, Meta Llama, and LLaVa) locally on your Raspberry Pi for tasks like text generation and translation.

Alternatives options:

llama.cpp and the Hugging Face **Transformers** library are well-supported for running SLMs on Raspberry Pi. llama.cpp is particularly efficient for running quantized models natively, while Hugging Face Transformers offers a broader range of models and tasks, best suited for smaller architectures due to hardware limitations.

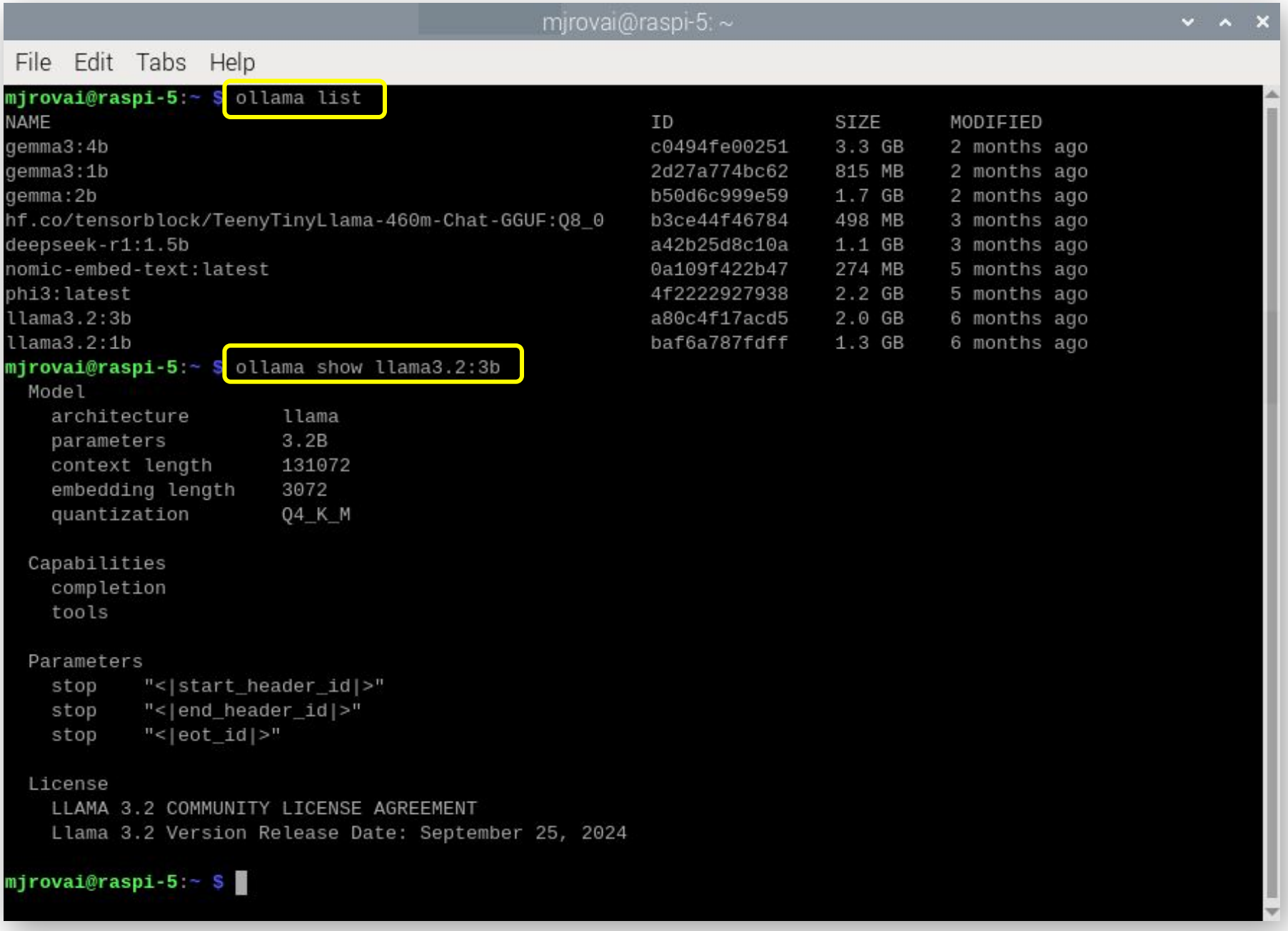


For Mac and Windows, download and execute the Ollama file on the machine.


mjrovai@rpi-5: ~
File Edit Tabs Help
mjrovai@rpi-5:~ $ python3 -m venv ~/ollama
mjrovai@rpi-5:~ $ source ~/ollama/bin/activate
(ollama) mjrovai@rpi-5:~ $ curl https://ollama.ai/install.sh | sh
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
0 0 0 0 0 0 0 0 --:--:-- --:--:-- --:--:-- 0>>> Inst
alling ollama to /usr/local
100 13320 0 13320 0 0 24659 0 --:--:-- --:--:-- --:--:-- 24621
>>> Downloading Linux arm64 bundle
100.0%##O#-#
100.0%
>>> Adding ollama user to render group...
>>> Adding ollama user to video group...
>>> Adding current user to ollama group...
>>> Creating ollama systemd service...
>>> Enabling and starting ollama service...
>>> The Ollama API is now available at 127.0.0.1:11434.
>>> Install complete. Run "ollama" from the command line.
WARNING: No NVIDIA/AMD GPU detected. Ollama will run in CPU-only mode.
(ollama) mjrovai@rpi-5:~ $

Command	Description	Example Usage
`ollama serve`	Starts the Ollama server for local API access and inference	`ollama serve`
`ollama run <model>`	Runs a specified model interactively or with input text	`ollama run smollm2:135m`
`ollama pull <model>`	Downloads a model from the Ollama registry	`ollama pull phi`
`ollama list`	Lists all models currently downloaded and available	`ollama list`
`ollama show <model>`	Displays detailed information about a specific model	`ollama show phi`
`ollama create <new_model> -f <file>`	Creates a new model from a Modelfile (customization)	`ollama create mymodel -f Modelfile`
`ollama rm <model>`	Removes a specified model from your system	`ollama rm phi`

Command	Description	Example Usage
`ollama ps`	Shows currently running Ollama processes or sessions	`ollama ps`
`ollama stop <model>`	Stops a running model session or process	`ollama stop phi`
`ollama cp <source> <target>`	Copies an existing model to a new name	`ollama cp phi mybackup`
`ollama push <model>`	Uploads a model to the Ollama registry (requires account & API key)	`ollama push mymodel`
`ollama --help`	Lists all available commands and their descriptions	`ollama --help`
`ollama --version`	Shows the current version of Ollama installed	`ollama --version`
`ollama <model> --verbose`	Run the model, showing statistics at the end	`ollama run phi --verbose`


mjrovai@raspi-5: ~
File Edit Tabs Help
mjrovai@raspi-5:~ $ ollama list
NAME ID SIZE MODIFIED
gemma3:4b c0494fe00251 3.3 GB 2 months ago
gemma3:1b 2d27a774bc62 815 MB 2 months ago
gemma:2b b50d6c999e59 1.7 GB 2 months ago
hf.co/tensorblock/TeenyTinyLlama-460m-Chat-GGUF:Q8_0 b3ce44f46784 498 MB 3 months ago
deepseek-r1:1.5b a42b25d8c10a 1.1 GB 3 months ago
nomic-embed-text:latest 0a109f422b47 274 MB 5 months ago
phi3:latest 4f2222927938 2.2 GB 5 months ago
llama3.2:3b a80c4f17acd5 2.0 GB 6 months ago
llama3.2:1b baf6a787fdff 1.3 GB 6 months ago
mjrovai@raspi-5:~ $ ollama show llama3.2:3b
Model
architecture llama
parameters 3.2B
context length 131072
embedding length 3072
quantization Q4_K_M
Capabilities
completion
tools
Parameters
stop "<|start_header_id|>"
stop "<|end_header_id|>"
stop "<|eot_id|>"
License
LLAMA 3.2 COMMUNITY LICENSE AGREEMENT
Llama 3.2 Version Release Date: September 25, 2024
mjrovai@raspi-5:~ $

Small Models

LlaMa

Gemma

```
marcelo_rovai — mjrovai@raspi-5: ~ — ssh mjrovai@192.168.4.209 — 74×12
(ollama) mjrovai@raspi-5:~ $ ollama run llama3.2:3b --verbose
>>> What is the capital of France?
The capital of France is Paris.

total duration:       1.808927736s
load duration:        39.854862ms
prompt eval count:    32 token(s)
prompt eval duration: 221.506ms
prompt eval rate:     144.47 tokens/s
eval count:           8 token(s)
eval duration:        1.506376s
eval rate:            5.31 tokens/s
```

```
marcelo_rovai — mjrovai@raspi-5: ~ — ssh mjrovai@192.168.4.209 — 67×13
(ollama) mjrovai@raspi-5:~ $ ollama run gemma2:2b --verbose
>>> What is the capital of France?
The capital of France is **Paris**.


total duration:       4.373339337s
load duration:        48.129697ms
prompt eval count:    16 token(s)
prompt eval duration: 1.968114s
prompt eval rate:     8.13 tokens/s
eval count:           13 token(s)
eval duration:        2.313284s
eval rate:            5.62 tokens/s
```

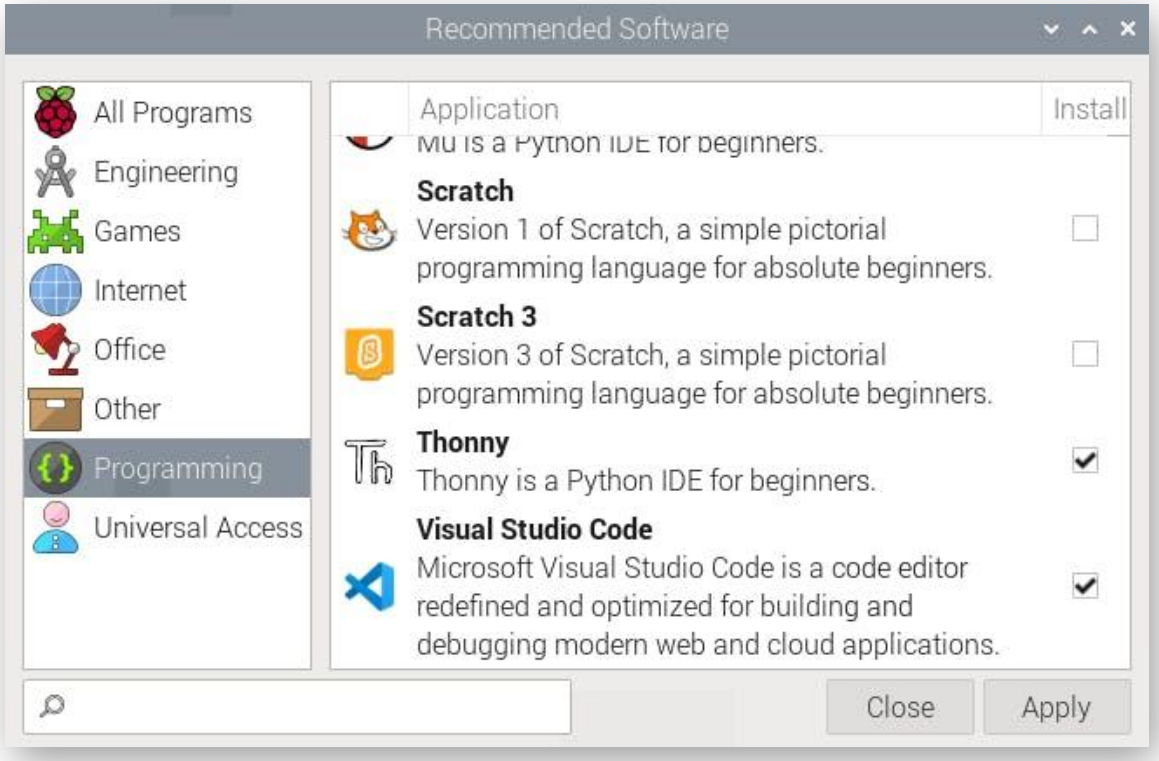
Ollama Python Library

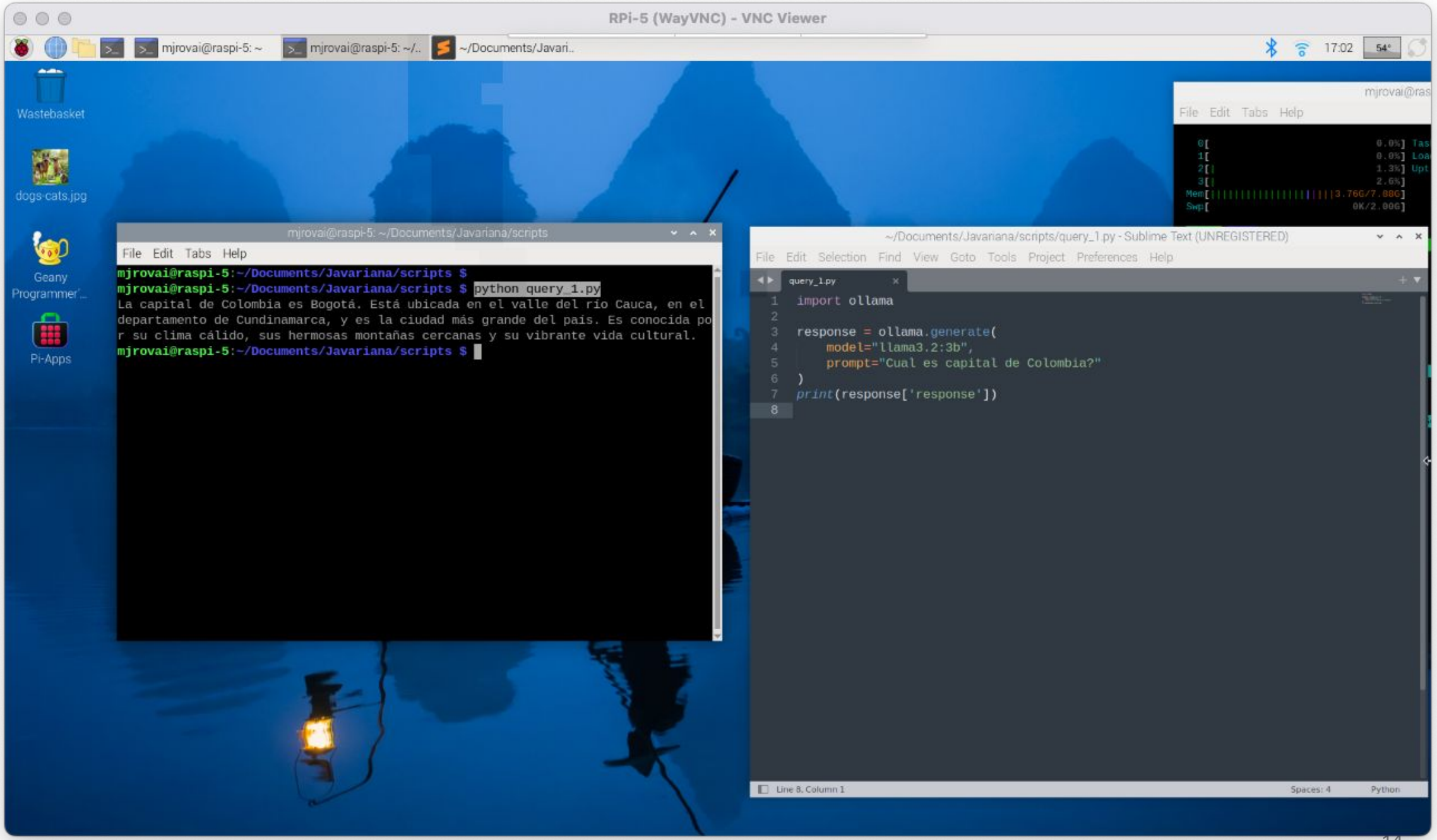
The Ollama Python library simplifies interaction with advanced LLM models, enabling more sophisticated responses and capabilities, besides providing the easiest way to integrate Python 3.8+ projects with Ollama.

```
$ pip install ollama
```

Install/use any IDE

- Nano
- Thonny
- Geany
- Sublime
- Jupyter Notebook
- Visual Studio
- …




RPi-5 (WayVNC) - VNC Viewer
mjrovai@raspi-5: ~
mjrovai@raspi-5: ~/..
~/Documents/Javari..
17:02
54°
Wastebasket
dogs-cats.jpg
Geany Programmer'...
Pi-Apps
mjrovai@raspi-5: ~/Documents/Javariana/scripts
File Edit Tabs Help
mjrovai@raspi-5:~/Documents/Javariana/scripts $
mjrovai@raspi-5:~/Documents/Javariana/scripts $ python query_1.py
La capital de Colombia es Bogotá. Está ubicada en el valle del río Cauca, en el departamento de Cundinamarca, y es la ciudad más grande del país. Es conocida por su clima cálido, sus hermosas montañas cercanas y su vibrante vida cultural.
mjrovai@raspi-5:~/Documents/Javariana/scripts $
~/Documents/Javariana/scripts/query_1.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
query_1.py
1 import ollama
2
3 response = ollama.generate(
4 model="llama3.2:3b",
5 prompt="Cual es capital de Colombia?"
6)
7 print(response['response'])
8
Line 8, Column 1
Spaces: 4
Python

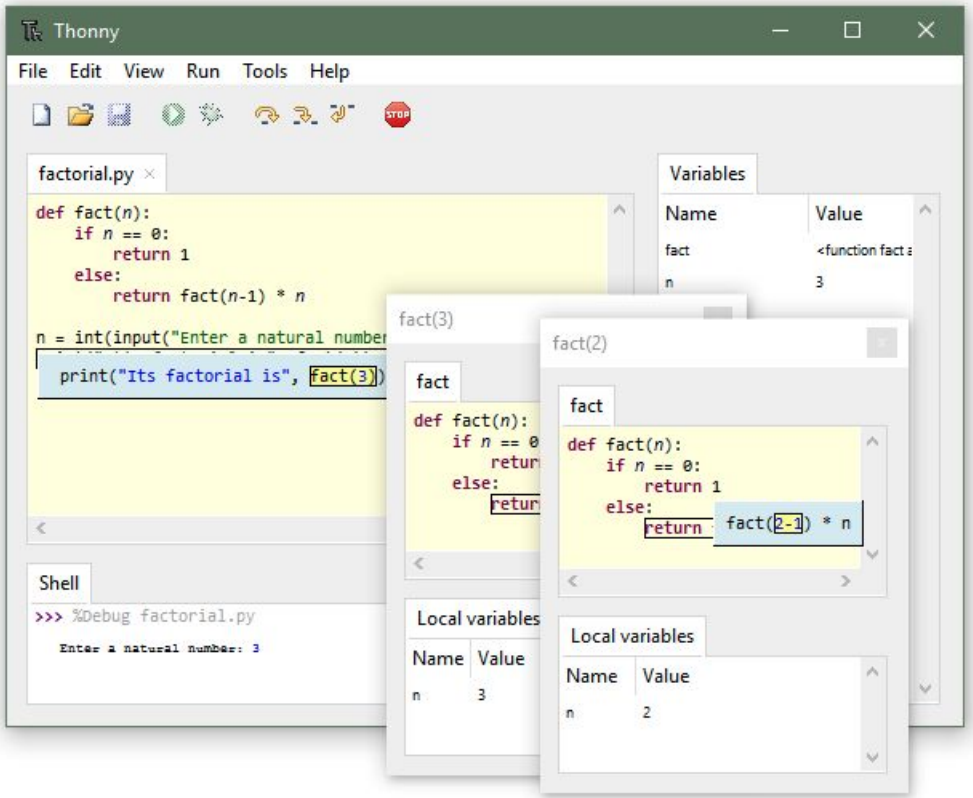
Thonny, Python IDE for begin

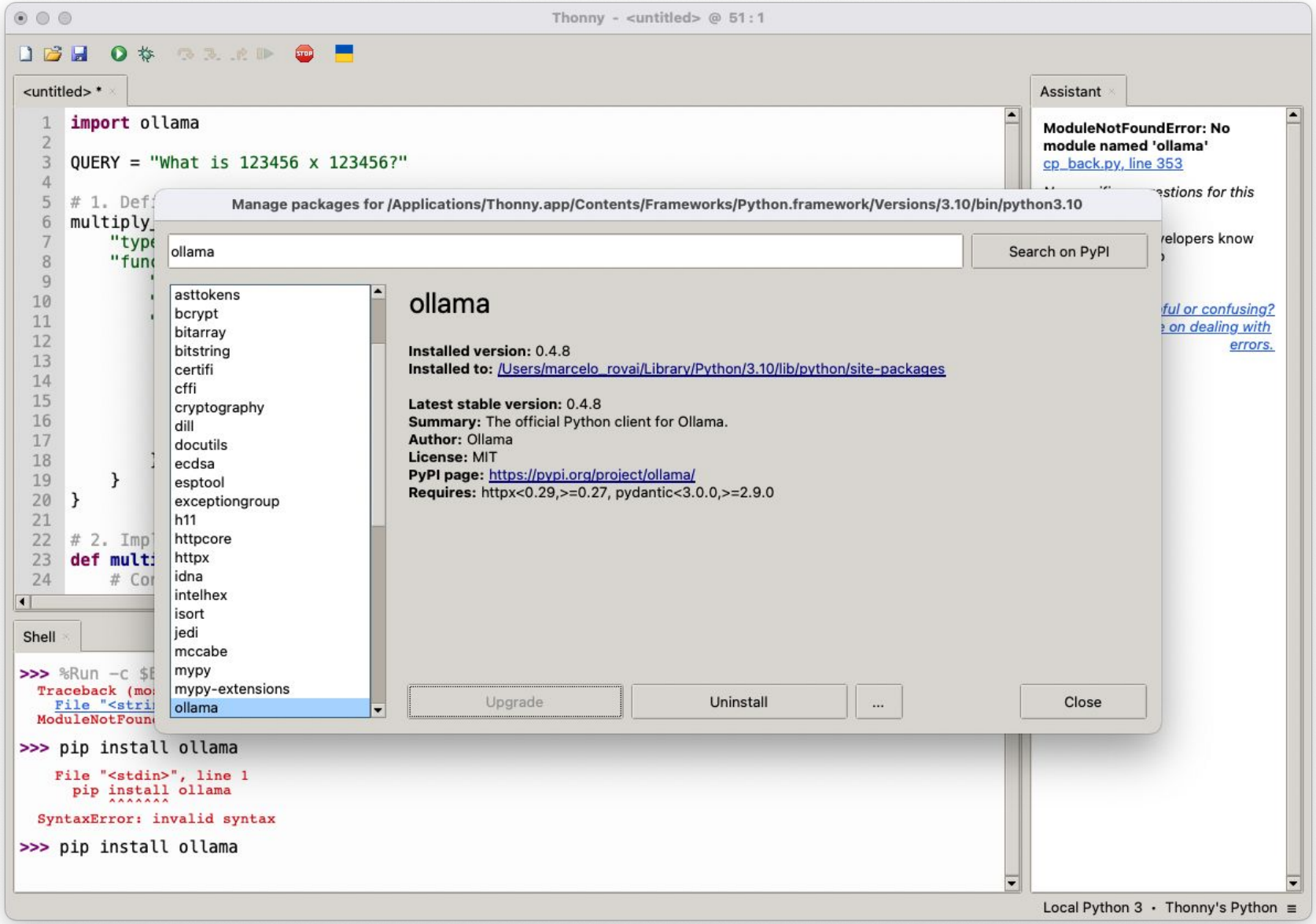
https://thonny.org

Thonny

Python IDE for beginners

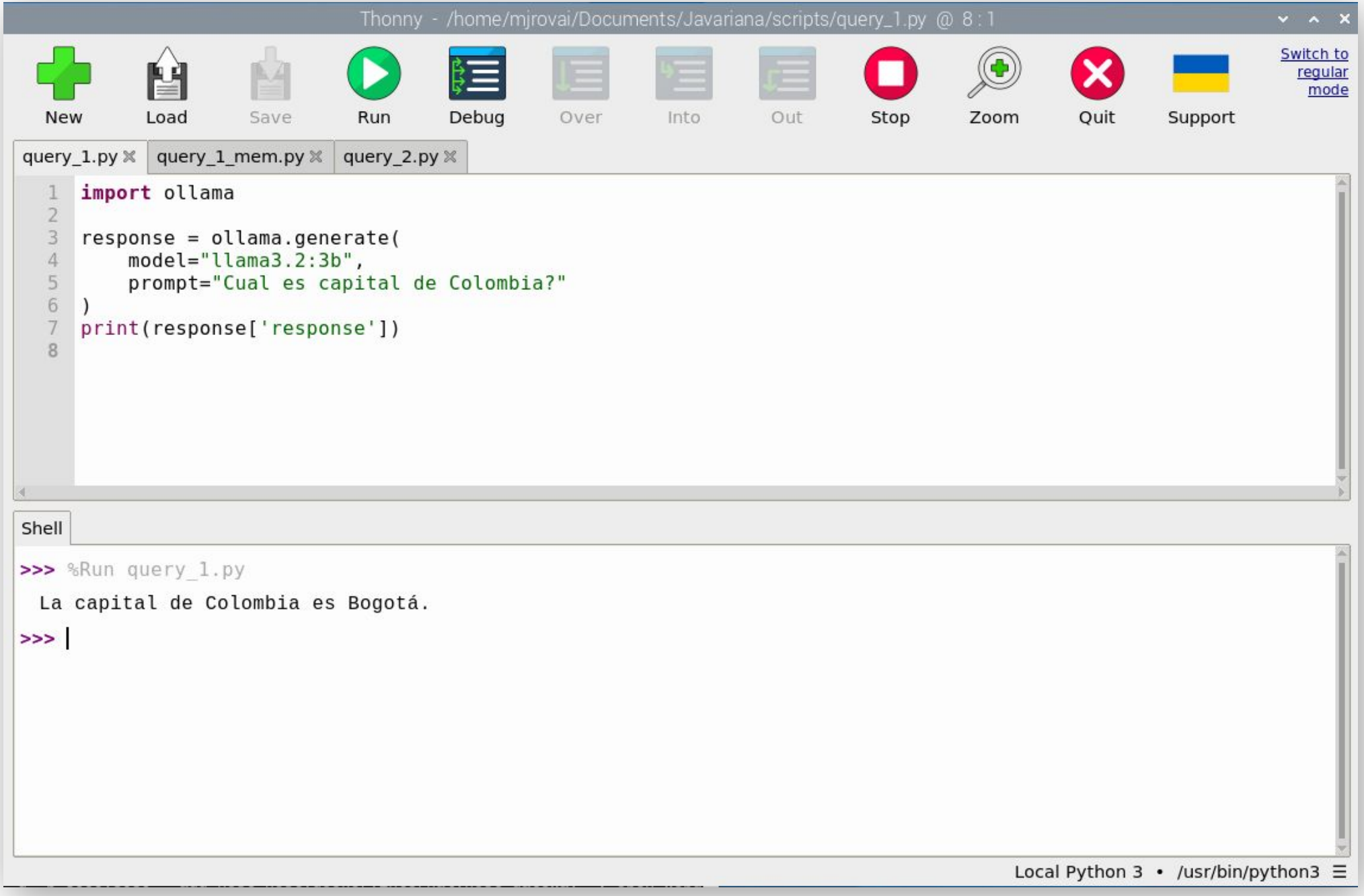
Download version 4.1.7 for Windows • Mac • Linux



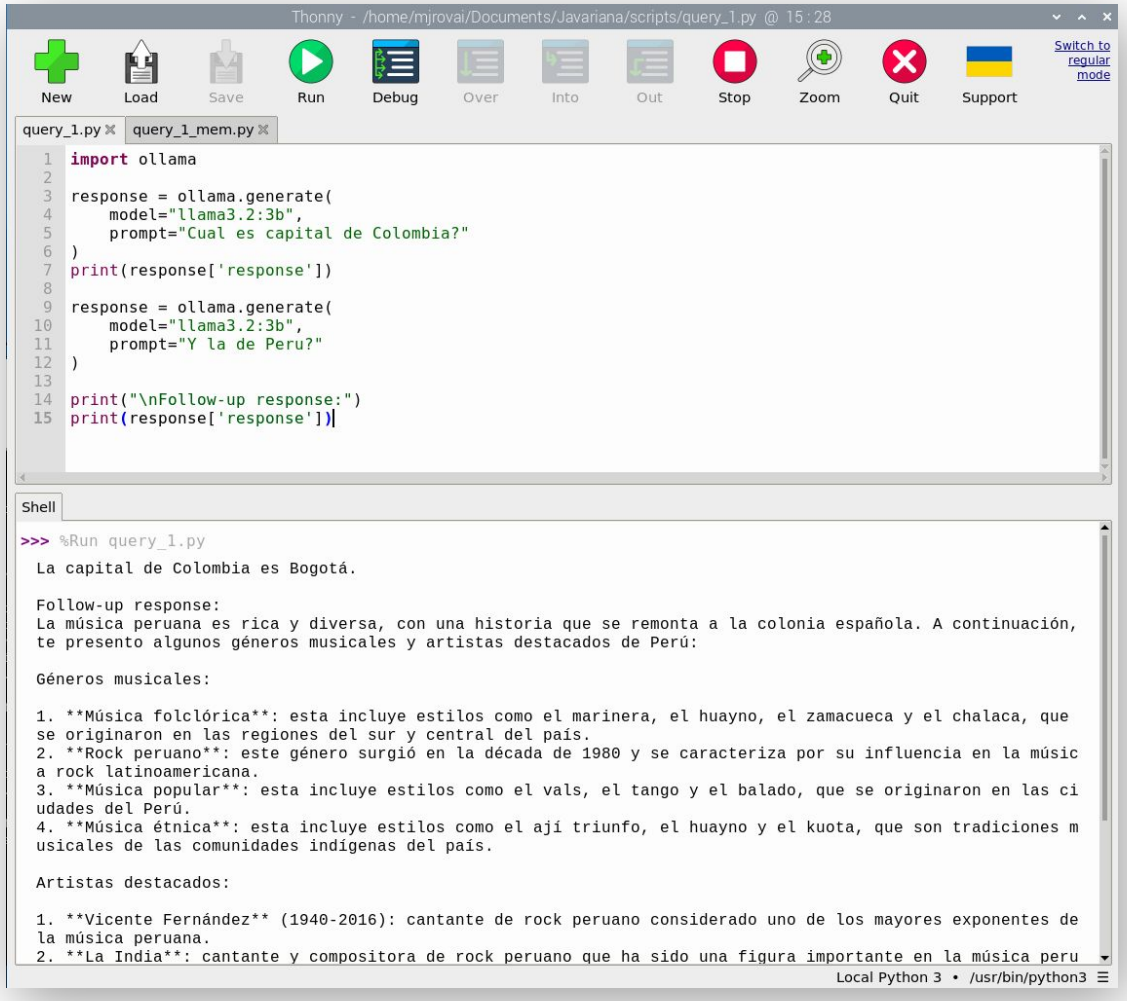


Install Ollama or any Python package on Thonny

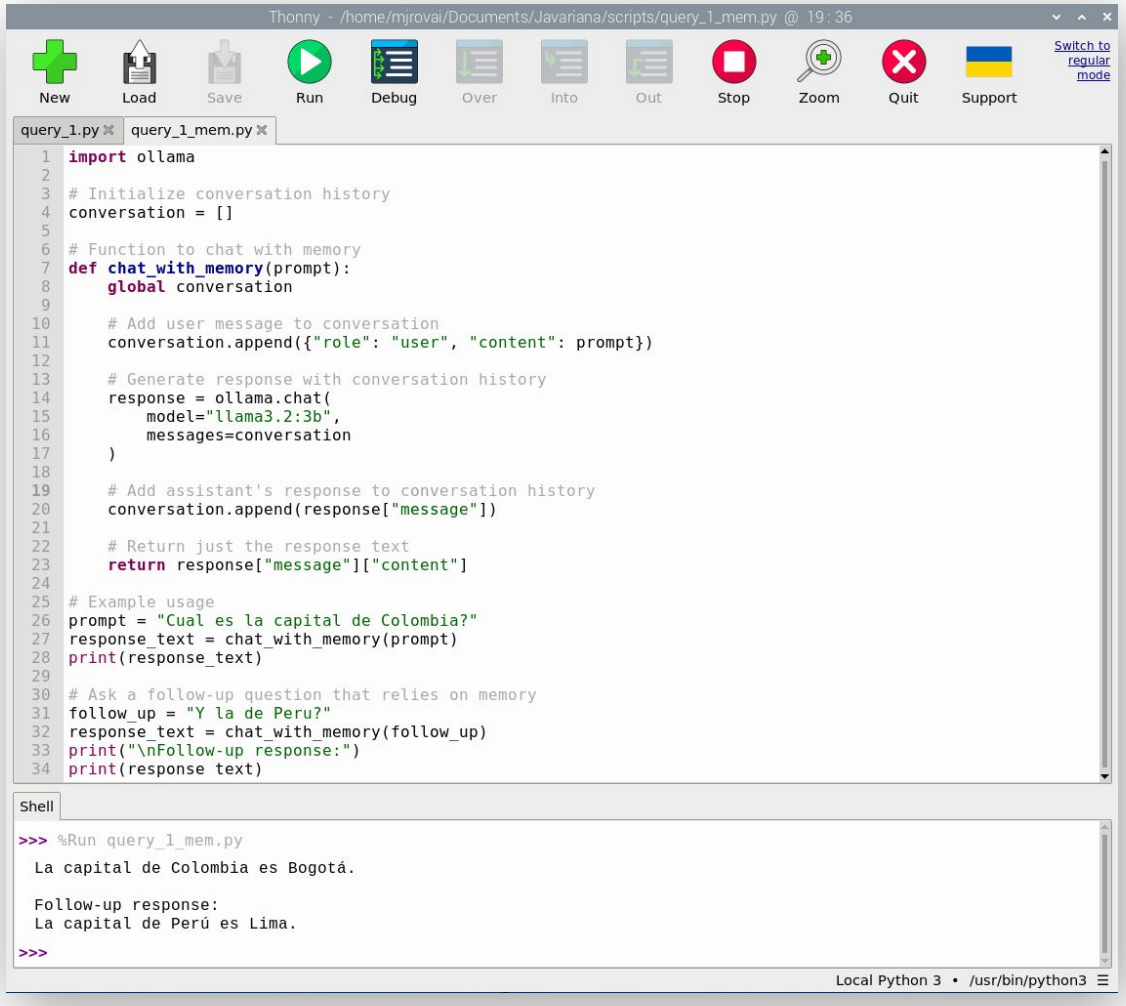
Step	Action
1	Open Thonny IDE
2	Tools → Manage Packages
3	Search for ollama
4	Ensure Ollama server is installed & running
5	Pull a model with
6	Use

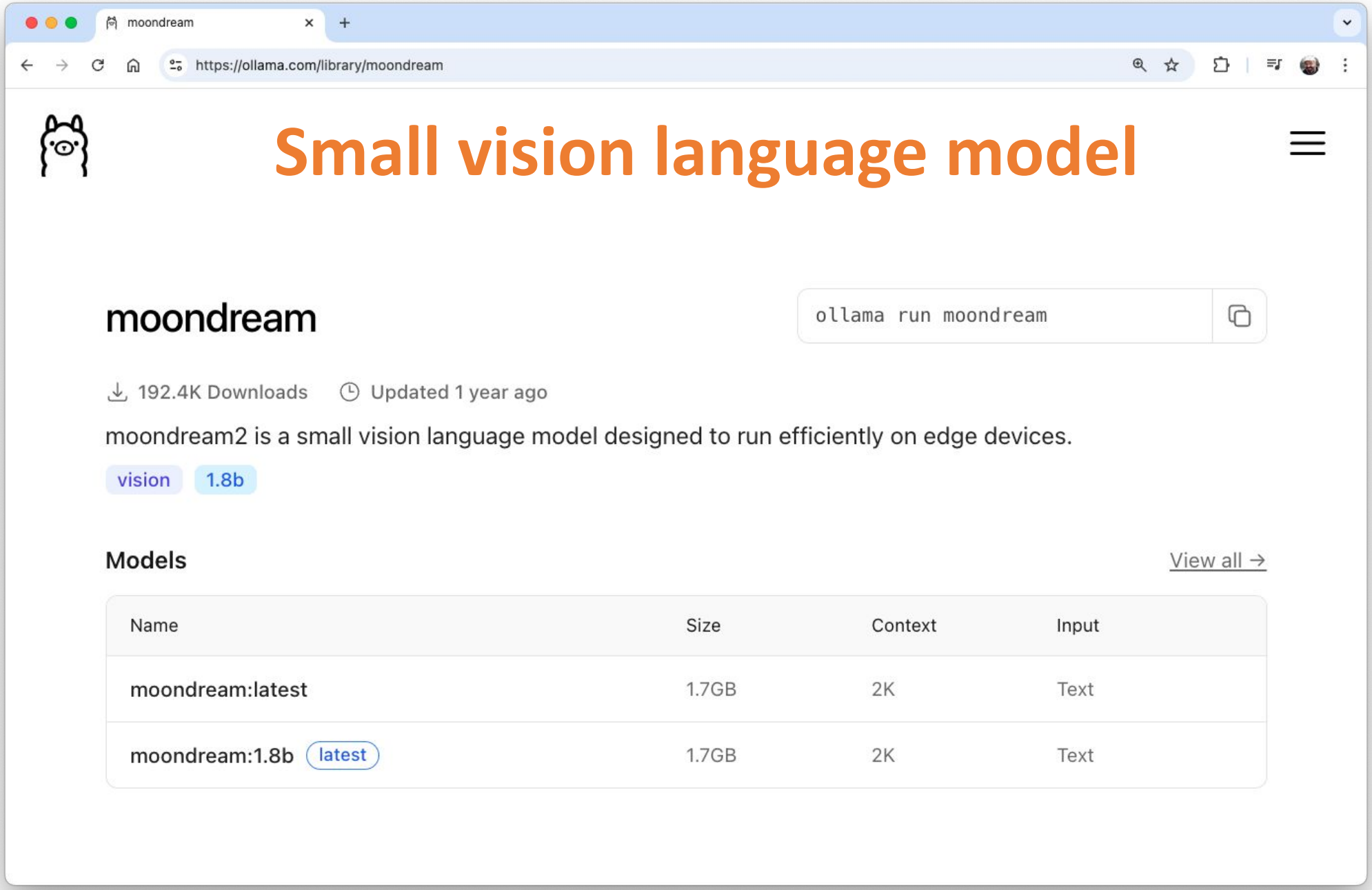

Thonny - /home/mjrovai/Documents/Javariana/scripts/query_1.py @ 8 : 1
New
Load
Save
Run
Debug
Over
Into
Out
Stop
Zoom
Quit
Support
Switch to regular mode
query_1.py
query_1_mem.py
query_2.py
1 import ollama
2
3 response = ollama.generate(
4 model="llama3.2:3b",
5 prompt="Cual es capital de Colombia?"
6)
7 print(response['response'])
8
Shell
>>> %Run query_1.py
La capital de Colombia es Bogotá.
>>>
Local Python 3 • /usr/bin/python3

ollama.generate()

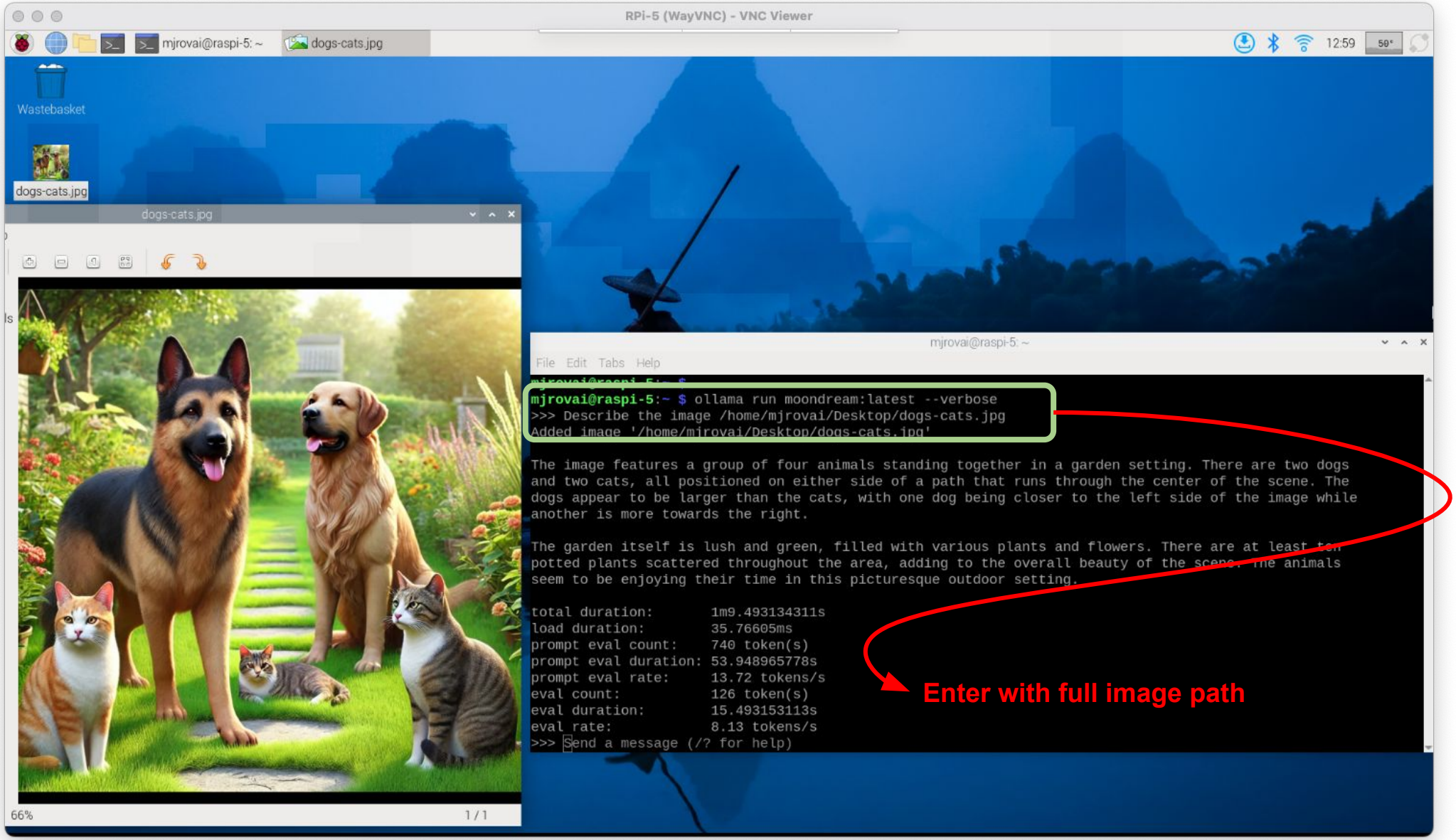


ollama.chat()




moondream
https://ollama.com/library/moondream
Small vision language model
moondream
ollama run moondream
192.4K Downloads
Updated 1 year ago
moondream2 is a small vision language model designed to run efficiently on edge devices.
vision
1.8b
Models
View all →
Name
Size
Context
Input
moondream:latest
1.7GB
2K
Text
moondream:1.8b
latest
1.7GB
2K
Text

```
mjrovai@raspi-5:~ $ ollama run moondream:latest --verbose
>>> Describe the image /home/mjrovai/Desktop/dogs-cats.jpg
```



RPi-5 (WayVNC) - VNC Viewer

mjrovai@raspi-5: ~ | dogs-cats.jpg | mjrovai@raspi-5: ~/.. | Thonny - /home/mjr.. | 13:19 | 53°

Wastebasket

dogs-cats.jpg

dogs-cats.jpg

66%

Thonny - /home/mjrovai/Documents/Javariana/scripts/img_description.py @ 4 : 27

New | Load | Save | Run | Debug | Over | Into | Out | Stop | Zoom | Quit | Support

Switch to regular mode

img_description.py | query_chat_memory.py

```python
import ollama

img_path = "/home/mjrovai/Desktop/dogs-cats.jpg"
MODEL = "moondream:latest"

def image_description(img_path):
    with open(img_path, 'rb') as file:
        response = ollama.chat(
            model=MODEL,
            messages=[
              {
                'role': 'user',
                'content': '''return the description of the image''',
                'images': [file.read()],
              },
            ],
            options = {
              'temperature': 0,
              }
      )
    #print(response['message']['content'])
    return response['message']['content']

image_description = image_description(img_path)
print(image description)
```

Shell

```
The image features a group of four animals standing together in a garden setting. There are two dogs a
nd two cats, all positioned on either side of a path that runs through the center of the scene. The do
gs appear to be larger than the cats, with one dog being closer to the left side of the image while an
other is more towards the right.

In addition to the animals, there are several potted plants scattered throughout the garden area. Some
of these plants can be seen on the left and right sides of the scene, as well as in the middle section
. The presence of both cats and dogs together creates a harmonious atmosphere within this outdoor spac
e.
```

Local Python 3 • /usr/bin/python3

LLMs: Optimization Techniques

Techniques for Enhancing SLM at the Edge

Fundamentals: Optimizing Prompting Strategies

- Chain-of-Thought Prompting
- Few-Shot Learning
- Task Decomposition

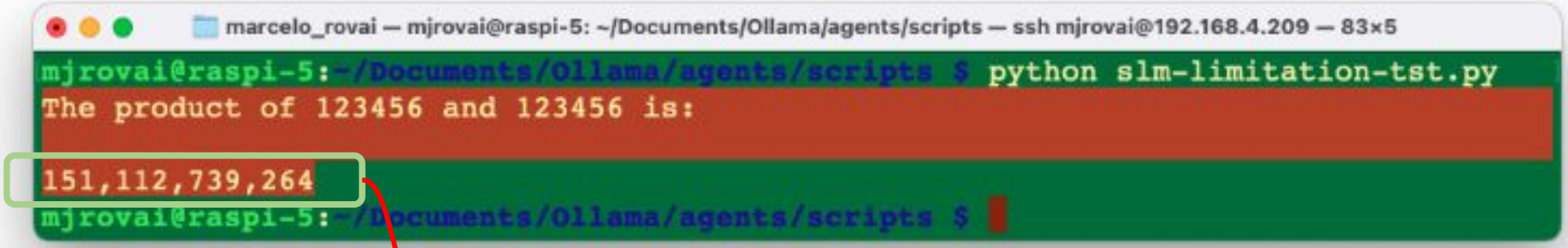
Intermediate: Building Intelligence Systems

- Building Agents with SLMs
- General Knowledge Router
- Function Calling
- Response Validation

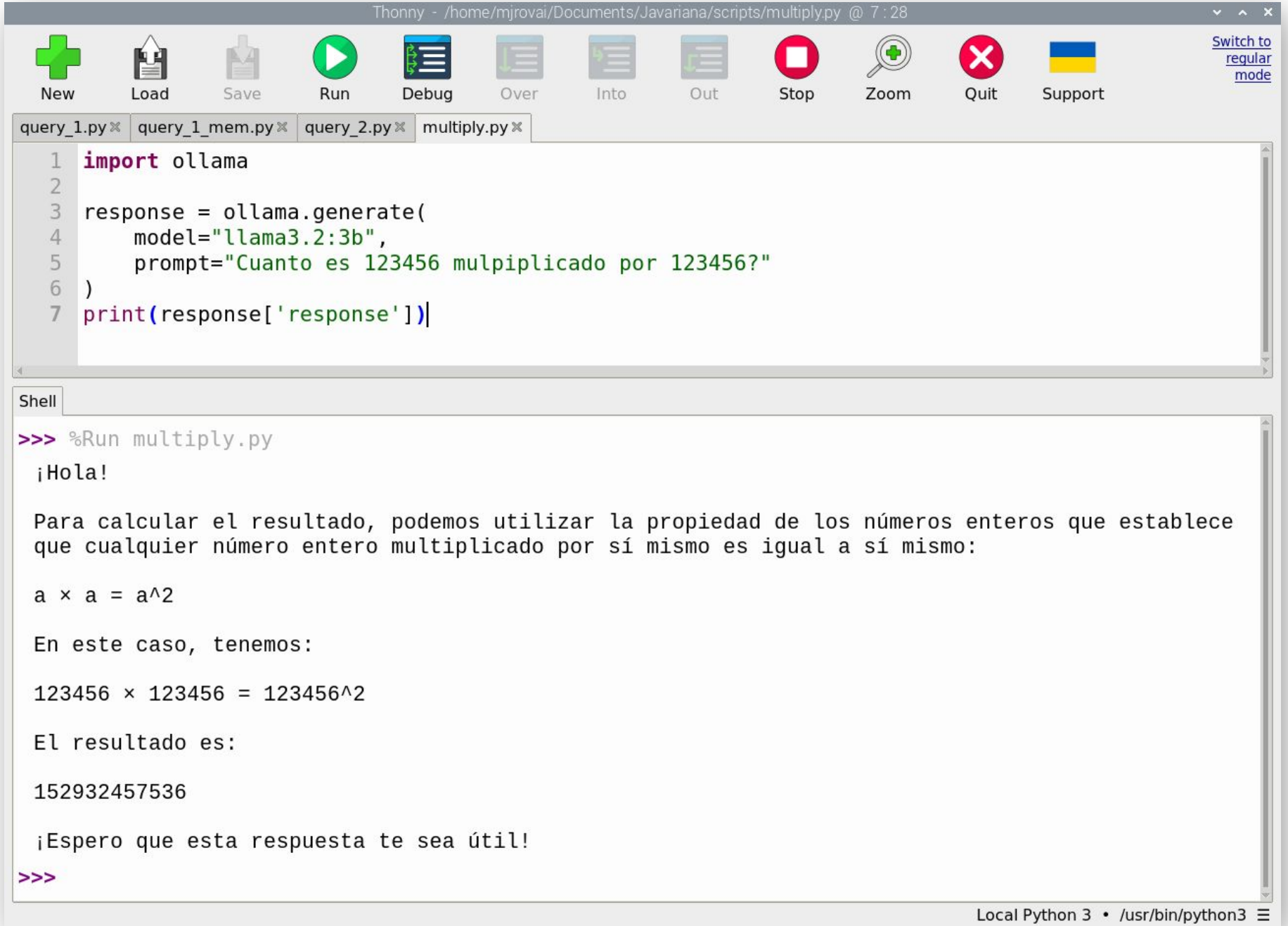
Advanced: Extending Knowledge and Specialization

- Retrieval-Augmented Generation (RAG)
- Fine-Tuning for Domain Specialization

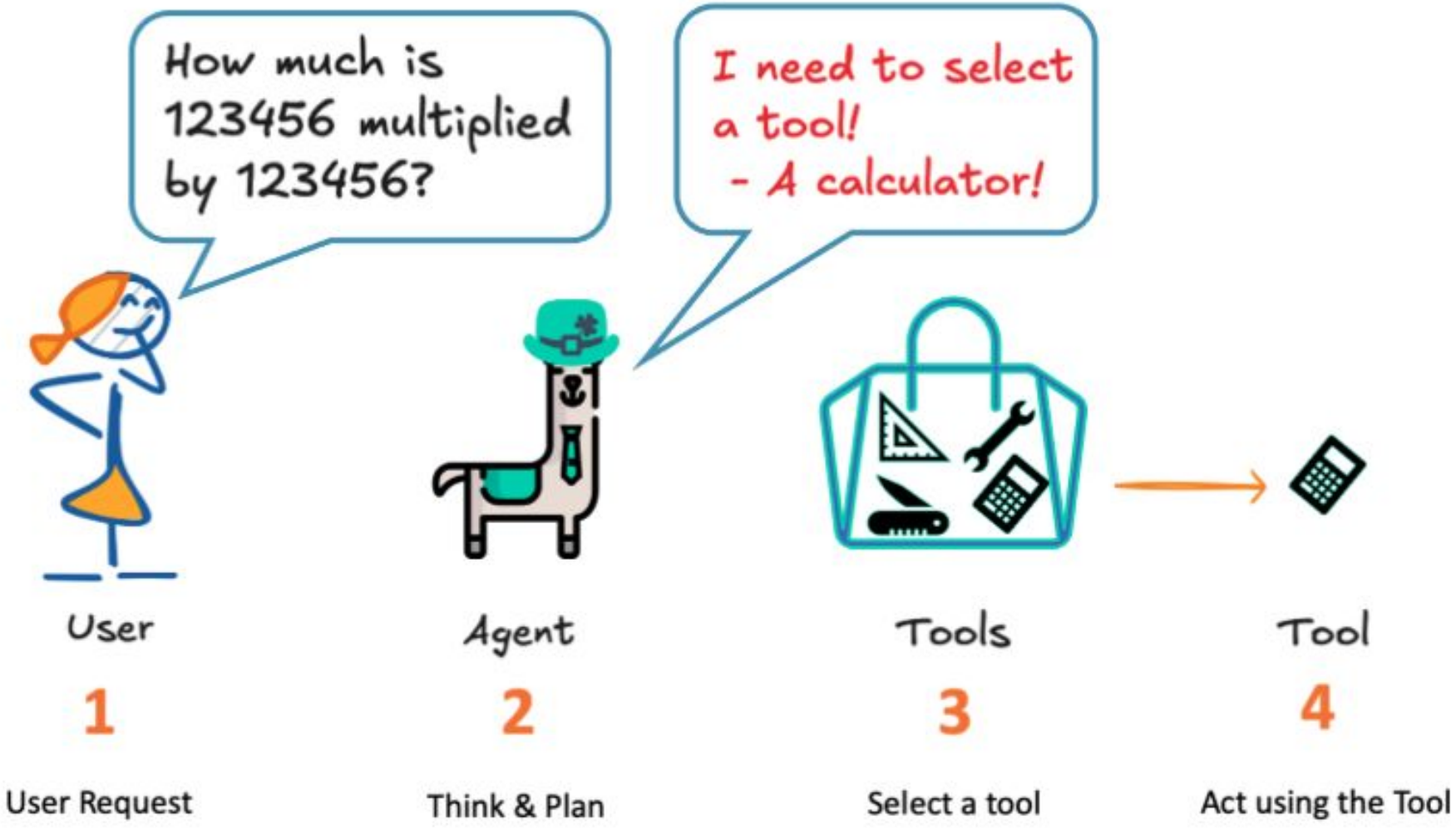
```python
import ollama

response = ollama.generate(
    model="llama3.2:3b",
    prompt="Multiply 123456 by 123456"
)
print(response['response'])
```

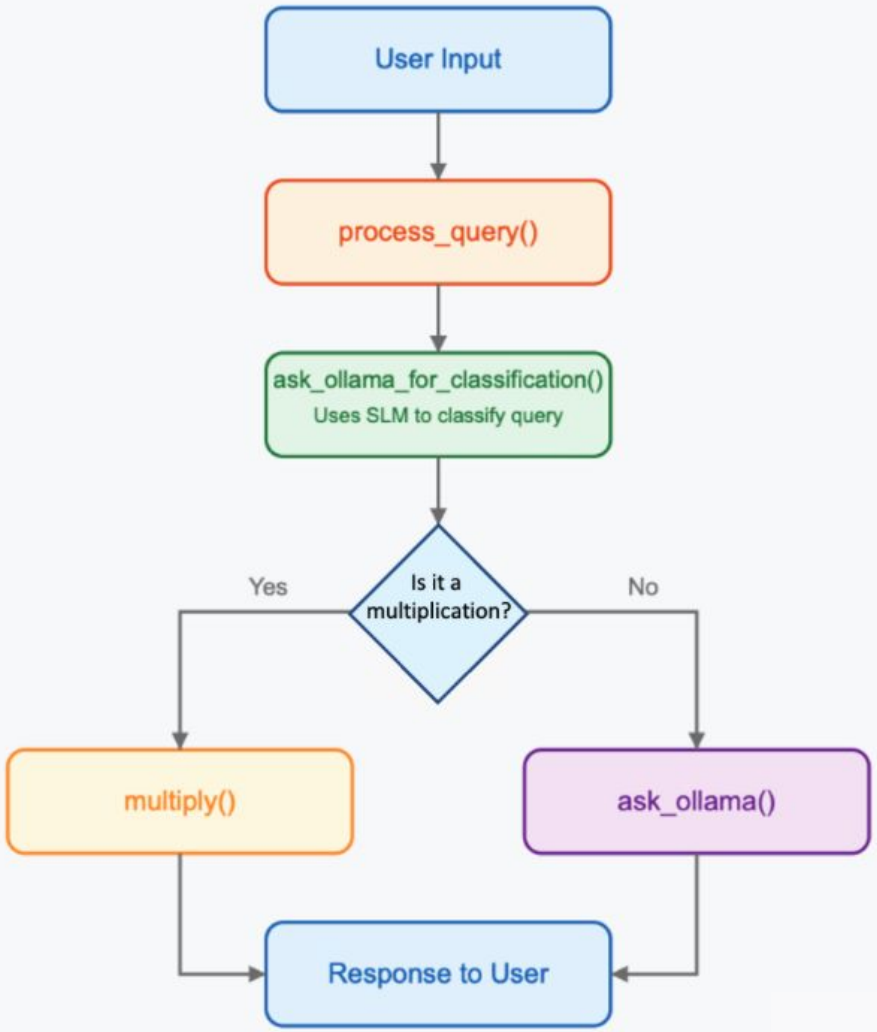


WRONG!!!!! **The correct answer -> 123,456 x 123,456 = 15,241,383,936**


Thonny - /home/mjrovai/Documents/Javariana/scripts/multiply.py @ 7 : 28
New
Load
Save
Run
Debug
Over
Into
Out
Stop
Zoom
Quit
Support
Switch to regular mode
query_1.py
query_1_mem.py
query_2.py
multiply.py
1 import ollama
2
3 response = ollama.generate(
4 model="llama3.2:3b",
5 prompt="Cuanto es 123456 mulpiplicado por 123456?"
6)
7 print(response['response'])
Shell
>>> %Run multiply.py
¡Hola!
Para calcular el resultado, podemos utilizar la propiedad de los números enteros que establece que cualquier número entero multiplicado por sí mismo es igual a sí mismo:
a × a = a^2
En este caso, tenemos:
123456 × 123456 = 123456^2
El resultado es:
152932457536
¡Espero que esta respuesta te sea útil!
>>>
Local Python 3 • /usr/bin/python3

Agents

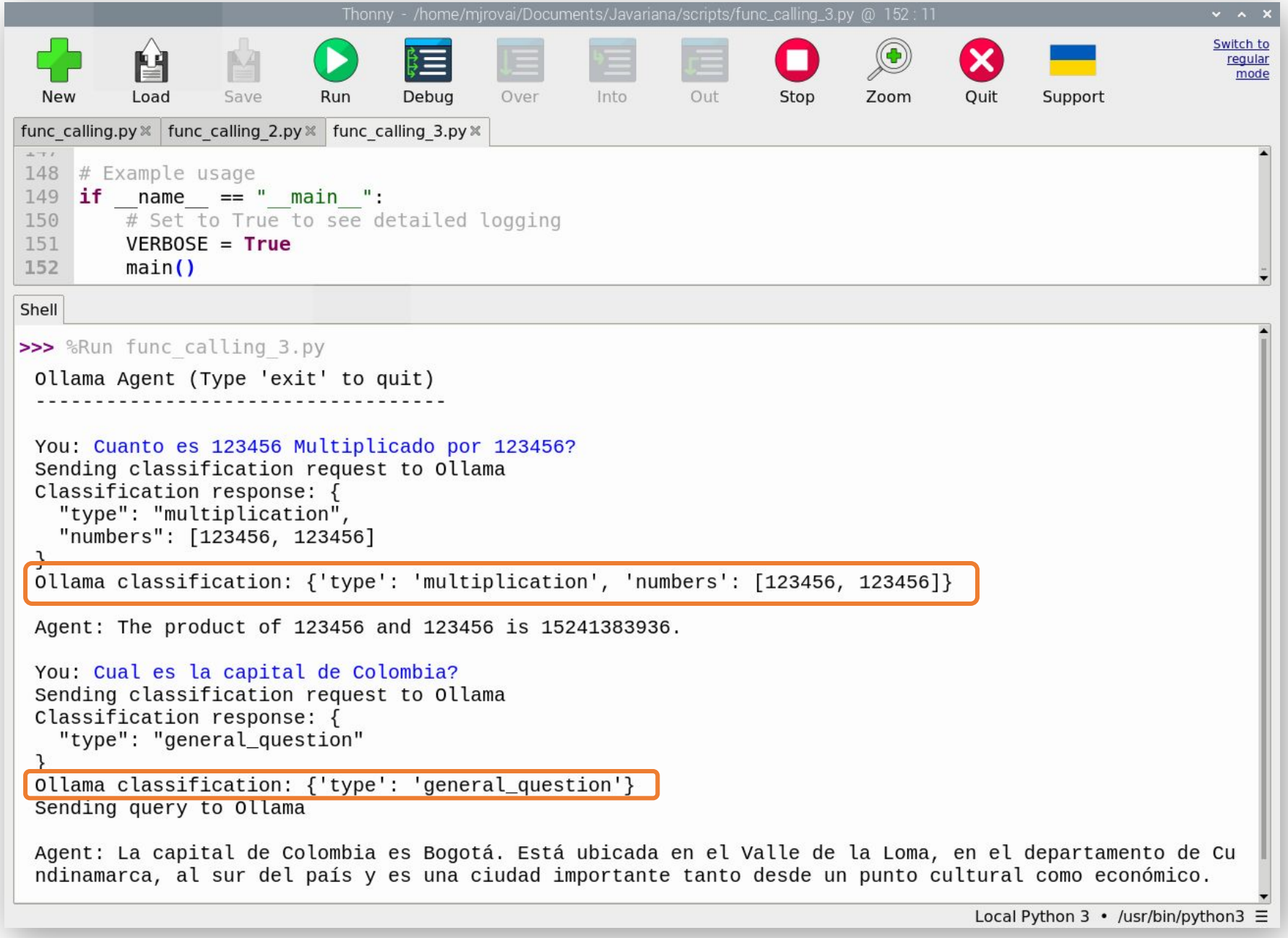

How much is 123456 multiplied by 123456?
I need to select a tool! - A calculator!
User
Agent
Tools
Tool
1
2
3
4
User Request
Think & Plan
Select a tool
Act using the Tool




```
marcelo_rovai — mjrovai@raspi-5: ~/Documents/Ollama/agents/scripts — ssh mjrovai@192.168.4.209 — 83×14
mjrovai@raspi-5:~/Documents/Ollama/agents/scripts $ python 2-simple_agent.py
Ollama Agent (Type 'exit' to quit)
----------------------------------

You: Multiply 123456 by 123456
Sending classification request to Ollama
Classification response: {
  "type": "multiplication",
  "numbers": [123456, 123456]
}
Ollama classification: {'type': 'multiplication', 'numbers': [123456, 123456]}

Agent: The product of 123456 and 123456 is 15241383936.
```

It is correct 123,456 x 123,456 = 15,241,383,936

```
marcelo_rovai — mjrovai@raspi-5: ~/Documents/Ollama/agents/scripts — ssh mjrovai@192.168.4.209 — 83×12

You: What is the capital of Brazil?
Sending classification request to Ollama
Classification response: {
  "type": "general_question"
}
Ollama classification: {'type': 'general_question'}
Sending query to Ollama

Agent: The capital of Brazil is Brasília.

You:
```


Thonny - /home/mjrovai/Documents/Javariana/scripts/func_calling_3.py @ 152 : 11
New
Load
Save
Run
Debug
Over
Into
Out
Stop
Zoom
Quit
Support
Switch to regular mode
func_calling.py
func_calling_2.py
func_calling_3.py
148 # Example usage
149 if __name__ == "__main__":
150 # Set to True to see detailed logging
151 VERBOSE = True
152 main()
Shell
>>> %Run func_calling_3.py
Ollama Agent (Type 'exit' to quit)
You: Cuanto es 123456 Multiplicado por 123456?
Sending classification request to Ollama
Classification response: {
"type": "multiplication",
"numbers": [123456, 123456]
}
Ollama classification: {'type': 'multiplication', 'numbers': [123456, 123456]}
Agent: The product of 123456 and 123456 is 15241383936.
You: Cual es la capital de Colombia?
Sending classification request to Ollama
Classification response: {
"type": "general_question"
}
Ollama classification: {'type': 'general_question'}
Sending query to Ollama
Agent: La capital de Colombia es Bogotá. Está ubicada en el Valle de la Loma, en el departamento de Cu
ndinamarca, al sur del país y es una ciudad importante tanto desde un punto cultural como económico.
Local Python 3 • /usr/bin/python3

Agents Running on SLMs: Overview

Agents powered by Small Language Models (SLMs) are autonomous or semi-autonomous software components that leverage compact, efficient language models to perform specialized tasks, interact with users, or orchestrate workflows. These agents are increasingly favored for their lower computational requirements, faster response times, and suitability for deployment in resource-constrained or privacy-sensitive environments where Large Language Models (LLMs) are impractical.

Key Traits of SLM-Powered Agents

- Efficiency & Cost-Effectiveness: SLMs require less memory and compute, making agents cheaper to run and easier to deploy at scale or on edge devices.
- Specialization: SLMs can be fine-tuned for specific domains, allowing agents to excel at focused tasks (e.g., compliance, finance, healthcare) without carrying the overhead of generalist LLMs.
- Autonomy & Collaboration: Multiple SLM agents can collaborate, each handling a segment of a workflow, sharing results, and adapting to context for complex, modular automation.
- Adaptability: With techniques like retrieval-augmented generation (RAG) and chain-of-thought prompting, SLM agents can plan, reason, and refine their actions in dynamic environments.

Next page, recommended Packages and Frameworks:

Framework / Package	Description	SLM Support	Key Features
smolagents	Lightweight Python library for building agentic systems with SLMs or LLMs.	Yes	Simple API, tool integration, supports Hugging Face and LiteLLM models, easy function/tool wrapping.
CrewAI	Agent-based framework for multi-agent orchestration.	Yes	Multi-agent workflows, UI tools, monitoring, extensibility, and integrates with many LLM/SLM providers.
Agno	Python framework for converting LLMs/SLMs into agents; supports multiple providers.	Yes	Built-in agent UI, AWS/cloud deployment, database/vector store integration, multi-agent orchestration.
OpenAI Swarm	Open-source multi-agent orchestration framework.	Yes	Lightweight, agent handoff architecture, privacy-focused, built-in retrieval/memory.
Autogen	Open-source framework for multi-agent collaboration and LLM/SLM workflows.	Yes	Cross-language support, local/remote agents, async messaging, scalable, pluggable components.
Arcee Orchestra	Commercial end-to-end agentic AI platform built on SLMs.	Yes	Intelligent model routing, security, compliance, on-prem deployment, fine-tuned SLMs for automation.
LangGraph	Graph-based agent orchestration framework from LangChain for complex, stateful, and multi-agent workflows.	Yes	Cyclic graph workflows, stateful memory (short/long-term), human-in-the-loop, parallelization, subgraphs, reflection/self-correction, visual IDE (LangGraph Studio), persistence, streaming, robust debugging and deployment tools135[7].

~/Documents/Javariana/scripts/agent_calc.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

RAG-1-Create-Persistent-Vector-Database.py | RAG-2-Query-the-Persistent-RAG-Database.py | agent_calc.py

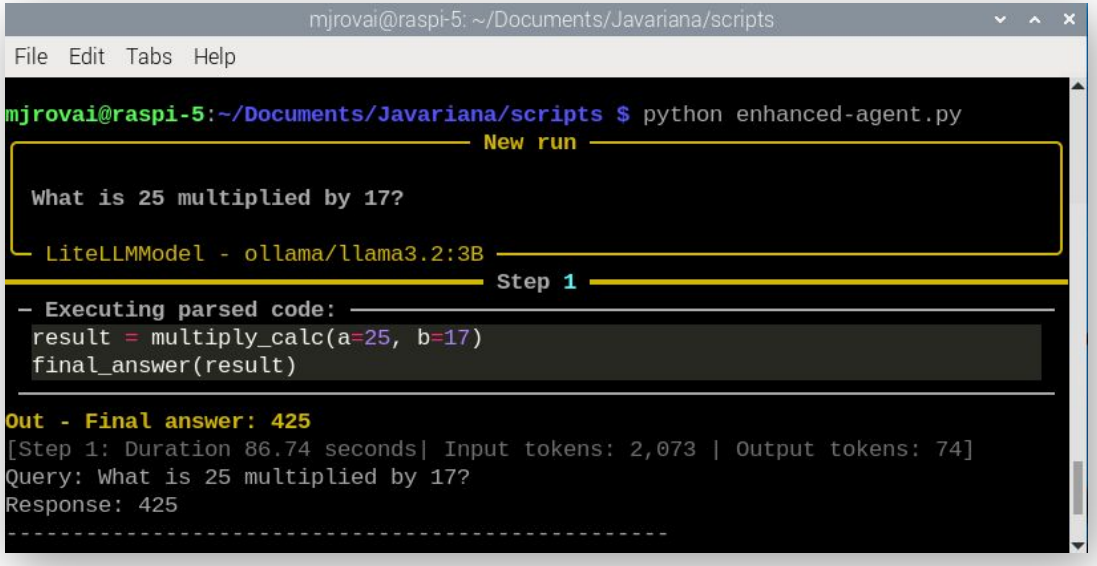
```python
from smolagents import CodeAgent, LiteLLMModel, tool

# Step 1: Define your tool function with a proper docstring
@tool
def multiply_calc(a: float, b: float) -> float:
    """Returns the product of two numbers.

    Args:
        a: The first number to multiply.
        b: The second number to multiply.

    Returns:
        float: The product of a and b.
    """
    return a * b

# Step 2: Create the agent
agent = CodeAgent(
    tools=[multiply_calc],
    model=LiteLLMModel(
        model_id="ollama/llama3.2:3B",
        api_base="http://localhost:11434",
        api_key="ollama",
        temperature=0.3,
        num_ctx=4096
    ),
    executor_type="local",
    max_steps=10
)

# Run the agent
response = agent.run("How is 123456 multiplied by 123456?")
print(response)
```

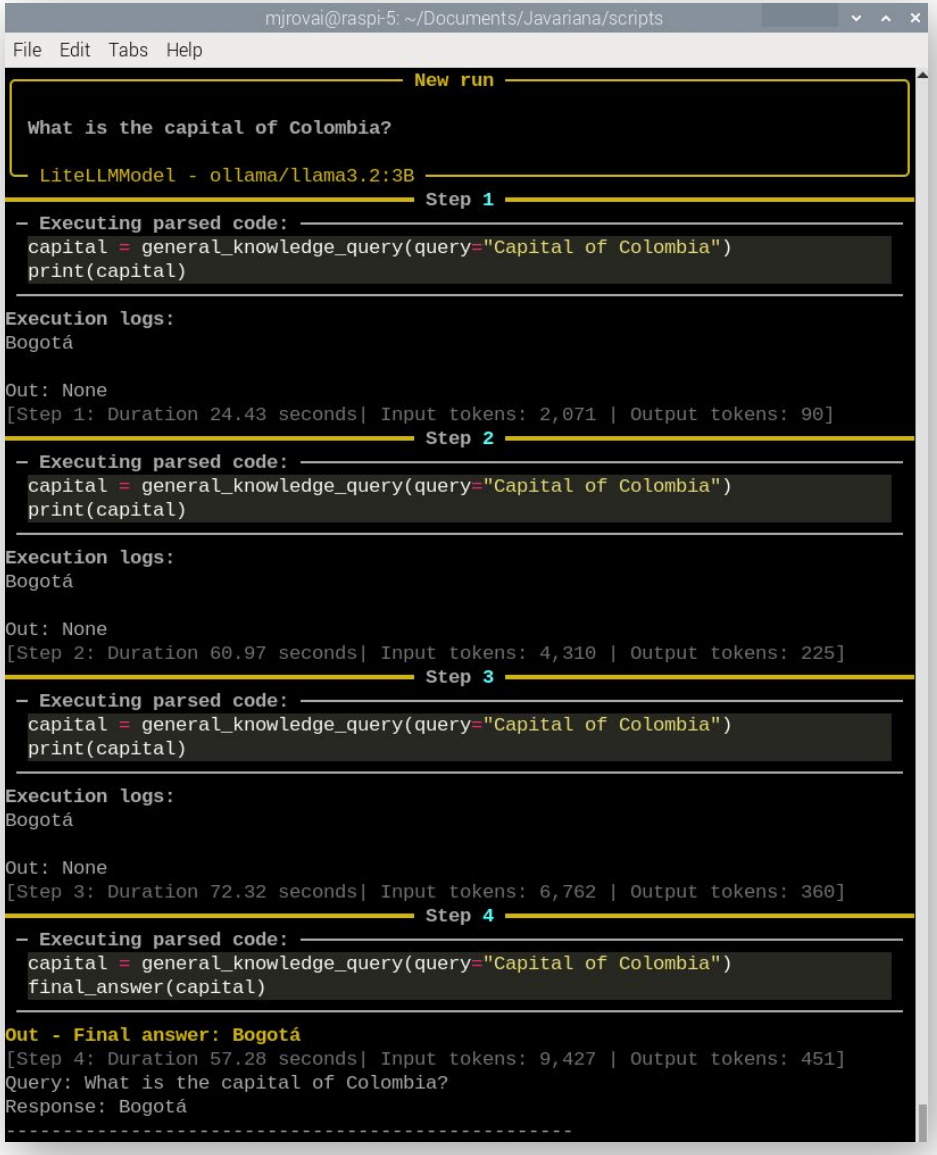
Line 17, Column 27 Spaces: 4 Python

mjrovai@raspi-5: ~/Documents/Javariana/scripts

File Edit Tabs Help

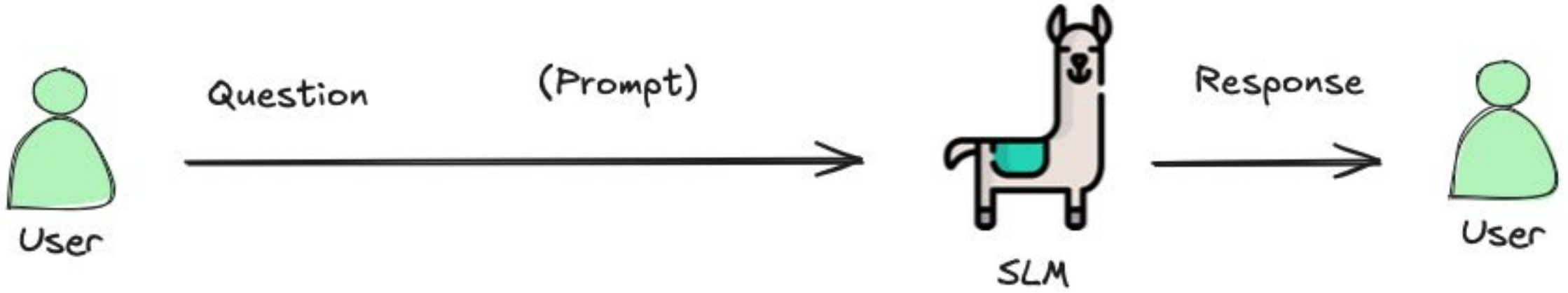
```
mjrovai@raspi-5:~/Documents/Javariana/scripts $ python agent_calc.py
──────────────────────────────────── New run ────────────────────────────────────

 How is 123456 multiplied by 123456?

╰─ LiteLLMModel - ollama/llama3.2:3B ──────────────────────────────────────────╯
──────────────────────────────────── Step 1 ─────────────────────────────────────
─ Executing parsed code: ──────────────────────────────────────────────────────
 result = multiply_calc(a=123456, b=123456)
────────────────────────────────────────────────────────────────────────────────

Out: 15241383936
[Step 1: Duration 279.24 seconds| Input tokens: 2,022 | Output tokens: 66]
──────────────────────────────────── Step 2 ─────────────────────────────────────
─ Executing parsed code: ──────────────────────────────────────────────────────
 print("The result of multiplying 123456 by 123456 is:", result)
────────────────────────────────────────────────────────────────────────────────

Execution logs:
The result of multiplying 123456 by 123456 is: 15241383936

Out: None
[Step 2: Duration 33.49 seconds| Input tokens: 4,180 | Output tokens: 119]
──────────────────────────────────── Step 3 ─────────────────────────────────────
─ Executing parsed code: ──────────────────────────────────────────────────────
 final_answer(result)
────────────────────────────────────────────────────────────────────────────────

Out - Final answer: 15241383936
[Step 3: Duration 34.30 seconds| Input tokens: 6,479 | Output tokens: 170]
15241383936
mjrovai@raspi-5:~/Documents/Javariana/scripts $
```

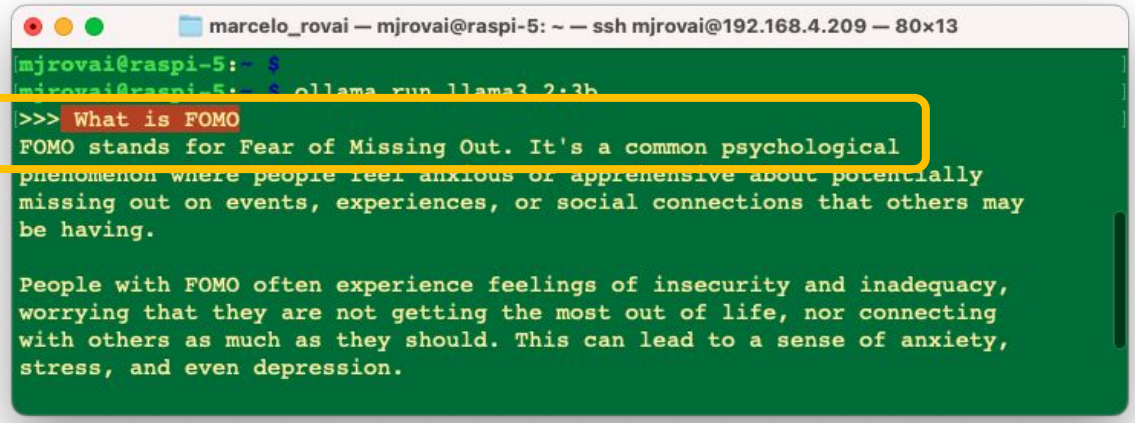

mjrovai@raspi-5: ~/Documents/Javariana/scripts
File Edit Tabs Help
mjrovai@raspi-5:~/Documents/Javariana/scripts $ python enhanced-agent.py
New run
What is 25 multiplied by 17?
LiteLLMModel - ollama/llama3.2:3B
Step 1
Executing parsed code:
result = multiply_calc(a=25, b=17)
final_answer(result)
Out - Final answer: 425
[Step 1: Duration 86.74 seconds| Input tokens: 2,073 | Output tokens: 74]
Query: What is 25 multiplied by 17?
Response: 425


mjrovai@raspi-5: ~/Documents/Javariana/scripts
File Edit Tabs Help
New run
What is the capital of Colombia?
LiteLLMModel - ollama/llama3.2:3B
Step 1
Executing parsed code:
capital = general_knowledge_query(query="Capital of Colombia")
print(capital)
Execution logs:
Bogotá
Out: None
[Step 1: Duration 24.43 seconds| Input tokens: 2,071 | Output tokens: 90]
Step 2
Executing parsed code:
capital = general_knowledge_query(query="Capital of Colombia")
print(capital)
Execution logs:
Bogotá
Out: None
[Step 2: Duration 60.97 seconds| Input tokens: 4,310 | Output tokens: 225]
Step 3
Executing parsed code:
capital = general_knowledge_query(query="Capital of Colombia")
print(capital)
Execution logs:
Bogotá
Out: None
[Step 3: Duration 72.32 seconds| Input tokens: 6,762 | Output tokens: 360]
Step 4
Executing parsed code:
capital = general_knowledge_query(query="Capital of Colombia")
final_answer(capital)
Out - Final answer: Bogotá
[Step 4: Duration 57.28 seconds| Input tokens: 9,427 | Output tokens: 451]
Query: What is the capital of Colombia?
Response: Bogotá

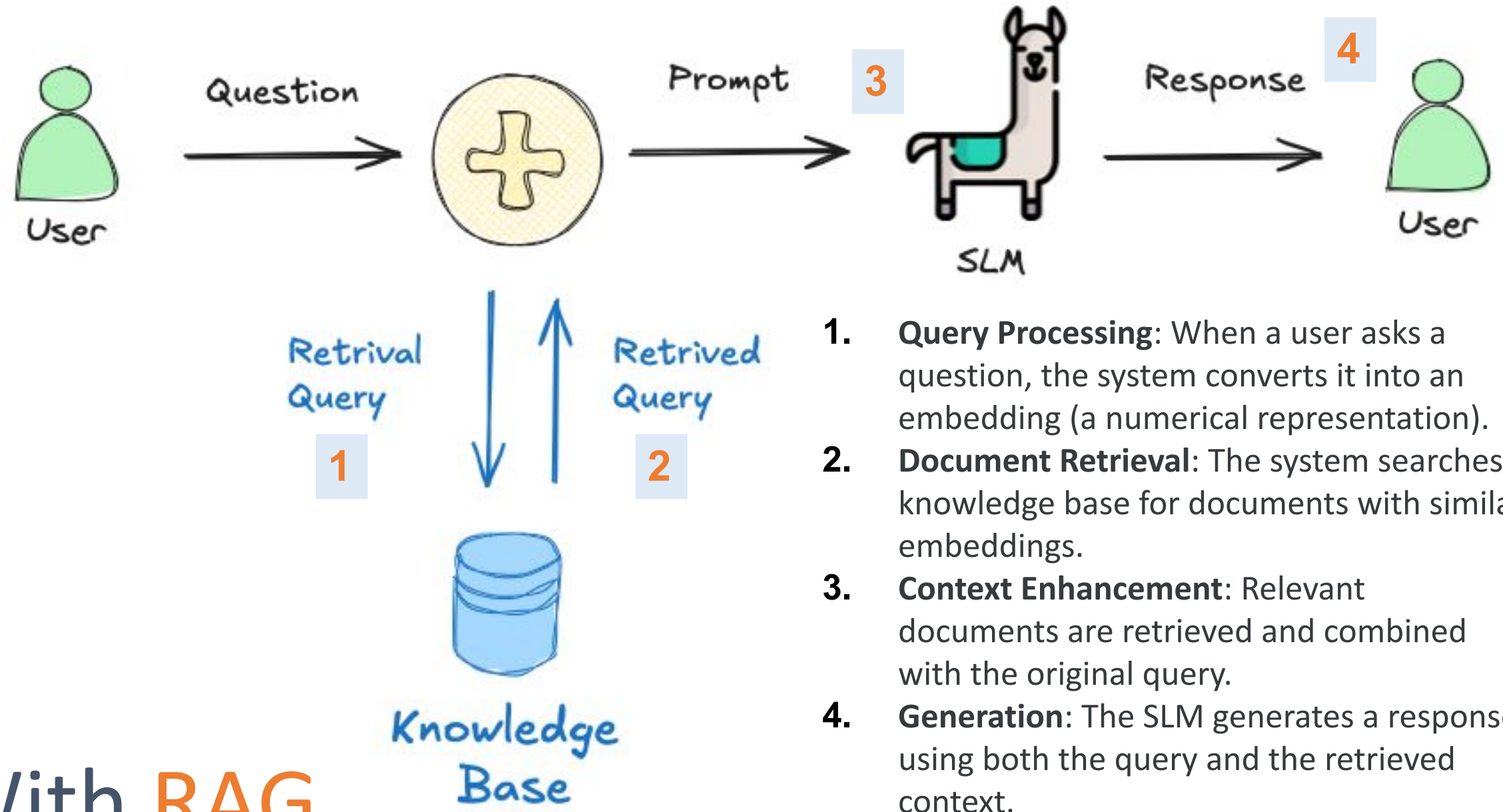
Retrieval-Augmented Generation (RAG)

“A method created by the FAIR team at Meta to enhance the accuracy of Large Language Models (LLMs) and reduce false information or “hallucinations.”



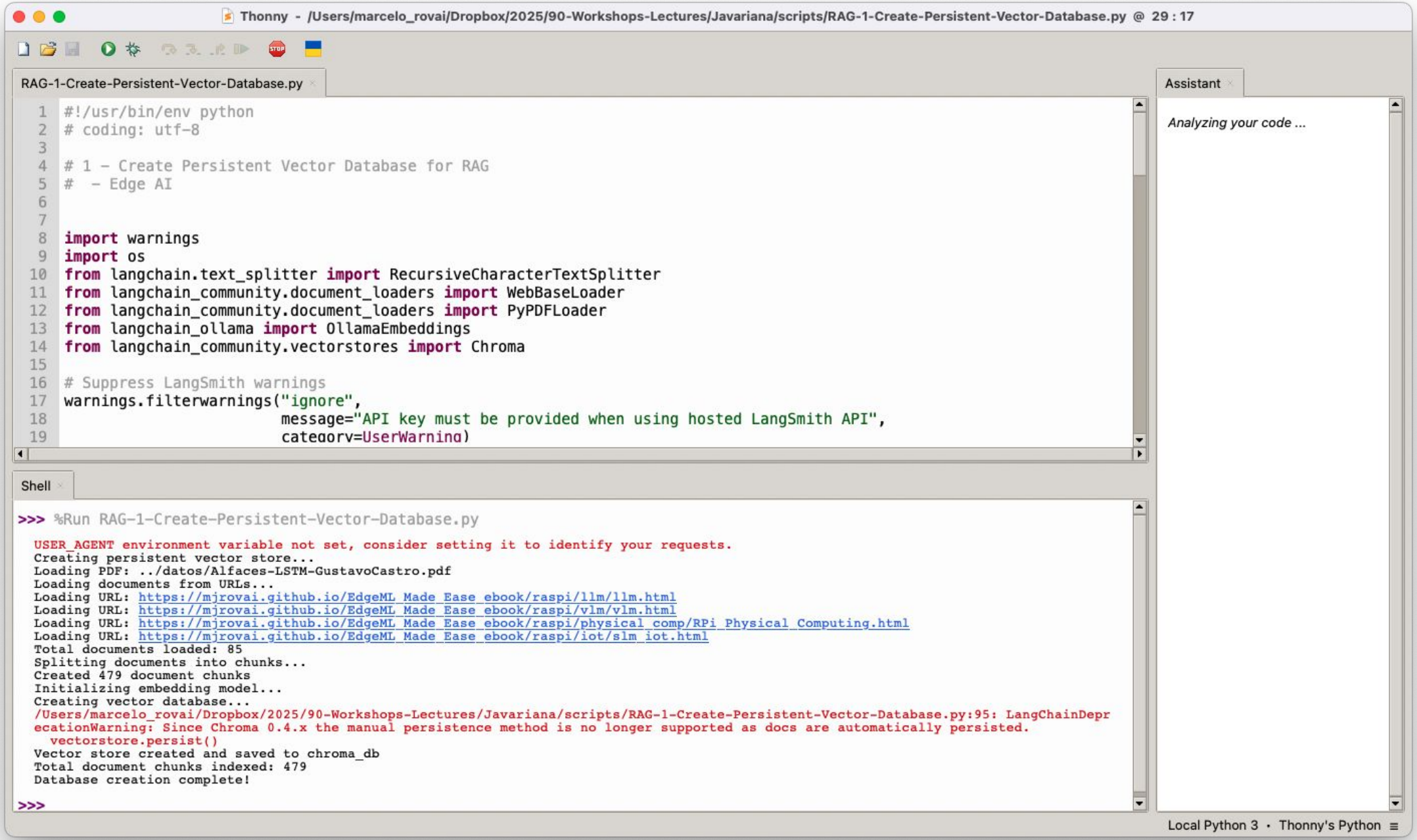


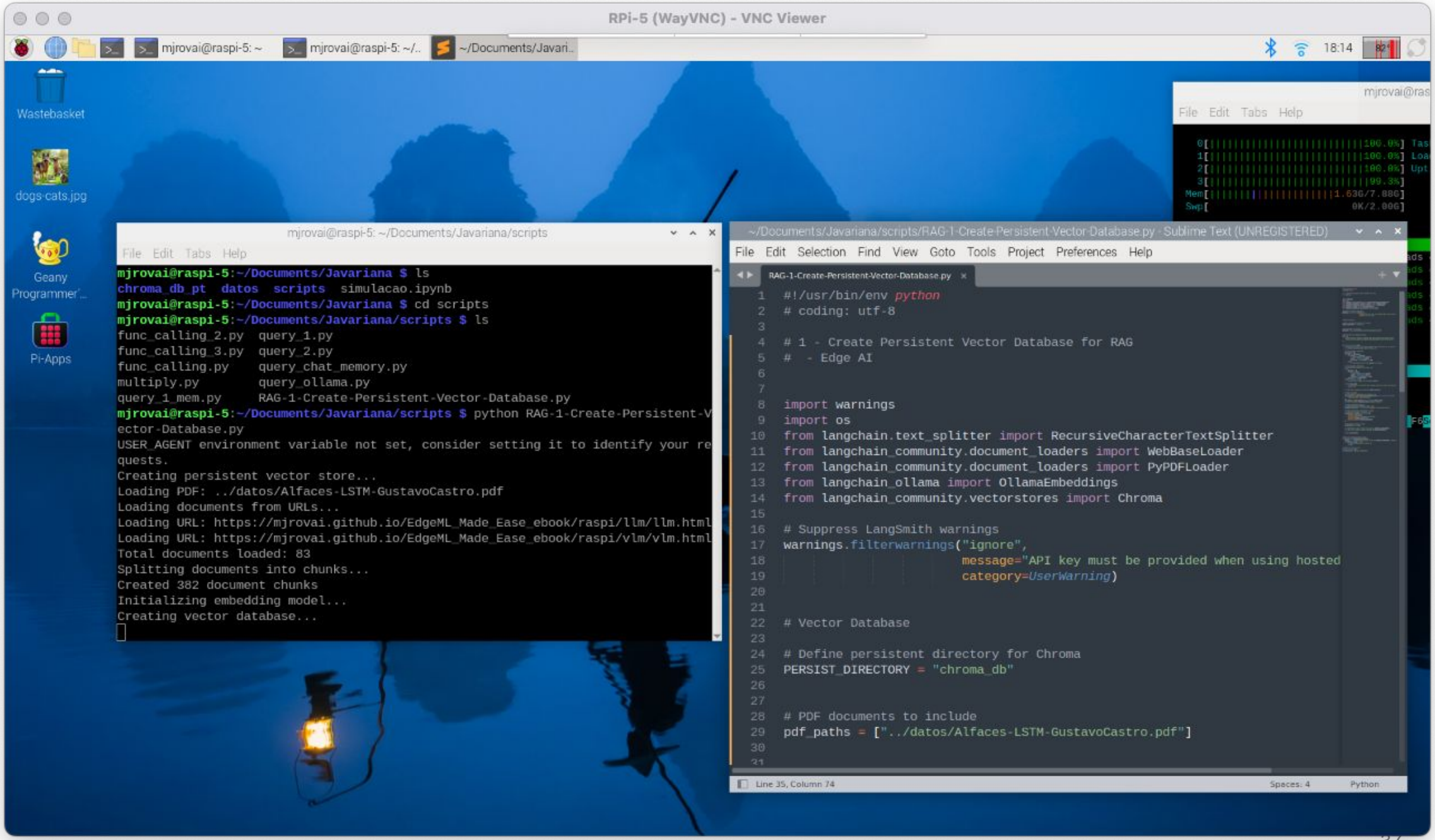
Usual Prompt

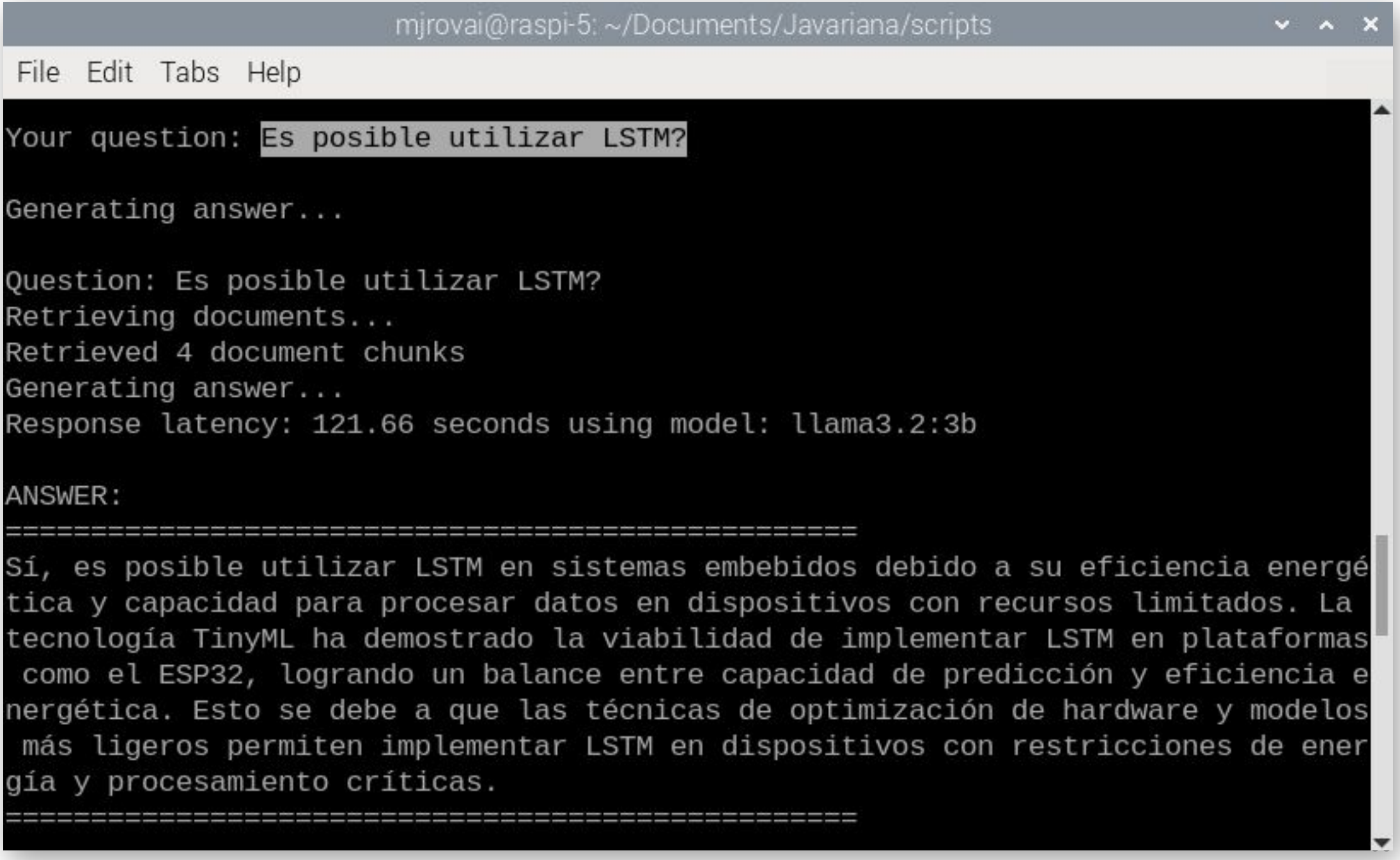


With RAG

1. **Query Processing**: When a user asks a question, the system converts it into an embedding (a numerical representation).
2. **Document Retrieval**: The system searches a knowledge base for documents with similar embeddings.
3. **Context Enhancement**: Relevant documents are retrieved and combined with the original query.
4. **Generation**: The SLM generates a response using both the query and the retrieved context.

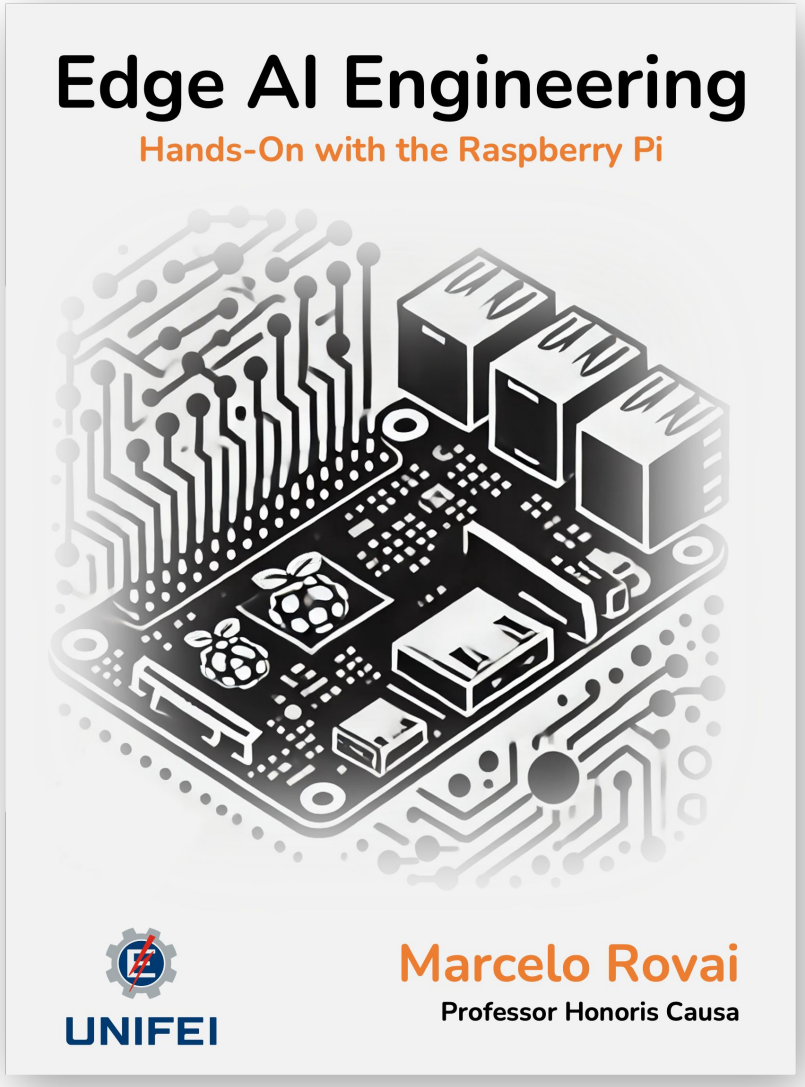

Thonny - /Users/marcelo_rovai/Dropbox/2025/90-Workshops-Lectures/Javariana/scripts/RAG-1-Create-Persistent-Vector-Database.py @ 29 : 17
RAG-1-Create-Persistent-Vector-Database.py
#!/usr/bin/env python
# coding: utf-8
# 1 - Create Persistent Vector Database for RAG
# - Edge AI
import warnings
import os
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain_community.document_loaders import WebBaseLoader
from langchain_community.document_loaders import PyPDFLoader
from langchain_ollama import OllamaEmbeddings
from langchain_community.vectorstores import Chroma
# Suppress LangSmith warnings
warnings.filterwarnings("ignore",
message="API key must be provided when using hosted LangSmith API",
category=UserWarning)
Assistant
Analyzing your code ...
Shell
>>> %Run RAG-1-Create-Persistent-Vector-Database.py
USER_AGENT environment variable not set, consider setting it to identify your requests.
Creating persistent vector store...
Loading PDF: ../datos/Alfaces-LSTM-GustavoCastro.pdf
Loading documents from URLs...
Loading URL: https://mjrovai.github.io/EdgeML_Made_Ease_ebook/raspi/llm/llm.html
Loading URL: https://mjrovai.github.io/EdgeML_Made_Ease_ebook/raspi/vlm/vlm.html
Loading URL: https://mjrovai.github.io/EdgeML_Made_Ease_ebook/raspi/physical_comp/RPi_Physical_Computing.html
Loading URL: https://mjrovai.github.io/EdgeML_Made_Ease_ebook/raspi/iot/slm_iot.html
Total documents loaded: 85
Splitting documents into chunks...
Created 479 document chunks
Initializing embedding model...
Creating vector database...
/Users/marcelo_rovai/Dropbox/2025/90-Workshops-Lectures/Javariana/scripts/RAG-1-Create-Persistent-Vector-Database.py:95: LangChainDeprecationWarning: Since Chroma 0.4.x the manual persistence method is no longer supported as docs are automatically persisted.
vectorstore.persist()
Vector store created and saved to chroma_db
Total document chunks indexed: 479
Database creation complete!
>>>
Local Python 3 • Thonny's Python


RPi-5 (WayVNC) - VNC Viewer
mjrovai@raspi-5: ~
mjrovai@raspi-5: ~/..
~/Documents/Javari..
18:14
Wastebasket
dogs-cats.jpg
Geany Programmer'...
Pi-Apps
mjrovai@raspi-5: ~/Documents/Javariana/scripts
File Edit Tabs Help
mjrovai@raspi-5:~/Documents/Javariana $ ls
chroma_db_pt datos scripts simulacao.ipynb
mjrovai@raspi-5:~/Documents/Javariana $ cd scripts
mjrovai@raspi-5:~/Documents/Javariana/scripts $ ls
func_calling_2.py query_1.py
func_calling_3.py query_2.py
func_calling.py query_chat_memory.py
multiply.py query_ollama.py
query_1_mem.py RAG-1-Create-Persistent-Vector-Database.py
mjrovai@raspi-5:~/Documents/Javariana/scripts $ python RAG-1-Create-Persistent-V
ector-Database.py
USER_AGENT environment variable not set, consider setting it to identify your re
quests.
Creating persistent vector store...
Loading PDF: ../datos/Alfaces-LSTM-GustavoCastro.pdf
Loading documents from URLs...
Loading URL: https://mjrovai.github.io/EdgeML_Made_Ease_ebook/raspi/llm/llm.html
Loading URL: https://mjrovai.github.io/EdgeML_Made_Ease_ebook/raspi/vlm/vlm.html
Total documents loaded: 83
Splitting documents into chunks...
Created 382 document chunks
Initializing embedding model...
Creating vector database...
~/Documents/Javariana/scripts/RAG-1-Create-Persistent-Vector-Database.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
RAG-1-Create-Persistent-Vector-Database.py
#!/usr/bin/env python
# coding: utf-8
# 1 - Create Persistent Vector Database for RAG
# - Edge AI
import warnings
import os
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain_community.document_loaders import WebBaseLoader
from langchain_community.document_loaders import PyPDFLoader
from langchain_ollama import OllamaEmbeddings
from langchain_community.vectorstores import Chroma
# Suppress LangSmith warnings
warnings.filterwarnings("ignore",
message="API key must be provided when using hosted
category=UserWarning)
# Vector Database
# Define persistent directory for Chroma
PERSIST_DIRECTORY = "chroma_db"
# PDF documents to include
pdf_paths = ["../datos/Alfaces-LSTM-GustavoCastro.pdf"]
Line 35, Column 74
Spaces: 4
Python

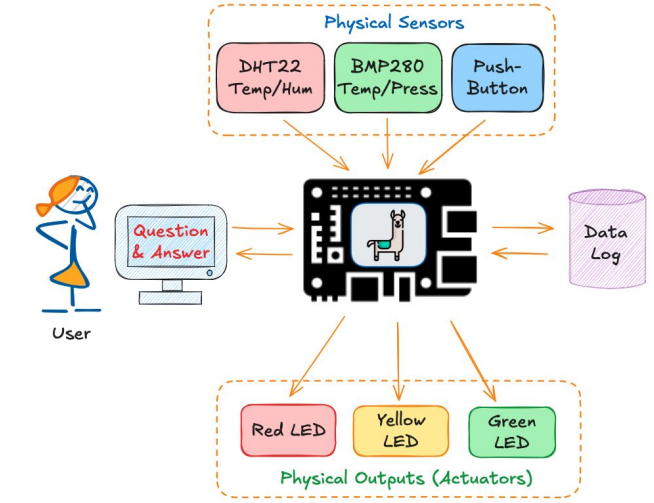

mjrovai@raspi-5: ~/Documents/Javariana/scripts
File Edit Tabs Help
Your question: Es posible utilizar LSTM?
Generating answer...
Question: Es posible utilizar LSTM?
Retrieving documents...
Retrieved 4 document chunks
Generating answer...
Response latency: 121.66 seconds using model: llama3.2:3b
ANSWER:
==========
Sí, es posible utilizar LSTM en sistemas embebidos debido a su eficiencia energé
tica y capacidad para procesar datos en dispositivos con recursos limitados. La
tecnología TinyML ha demostrado la viabilidad de implementar LSTM en plataformas
como el ESP32, logrando un balance entre capacidad de predicción y eficiencia e
nergética. Esto se debe a que las técnicas de optimización de hardware y modelos
más ligeros permiten implementar LSTM en dispositivos con restricciones de ener
gía y procesamiento críticas.
==========

Homework…

- Physical Computing
- Vision-Language Models



EdgeAI Engineering




```
task_prompt = '<CAPTION_TO_PHRASE_GROUNDING>'
results = run_example(task_prompt, text_input="a church clock tower",image=city)
plot_bbox(table, results['<CAPTION_TO_PHRASE_GROUNDING>'])
```

```
[INFO] ==> Florence-2-base (<CAPTION_TO_PHRASE_GROUNDING>), took 12.7 seconds to execute.
```

Optional…

- Open WebUI

Installing and Running the Open Web UI - Raspi

https://github.com/open-webui/open-webui

1. Install Node.js

```
curl -fsSL https://deb.nodesource.com/setup_20.x | sudo -E bash - &&\
sudo apt-get install -y nodejs
```

2. Clone Git:

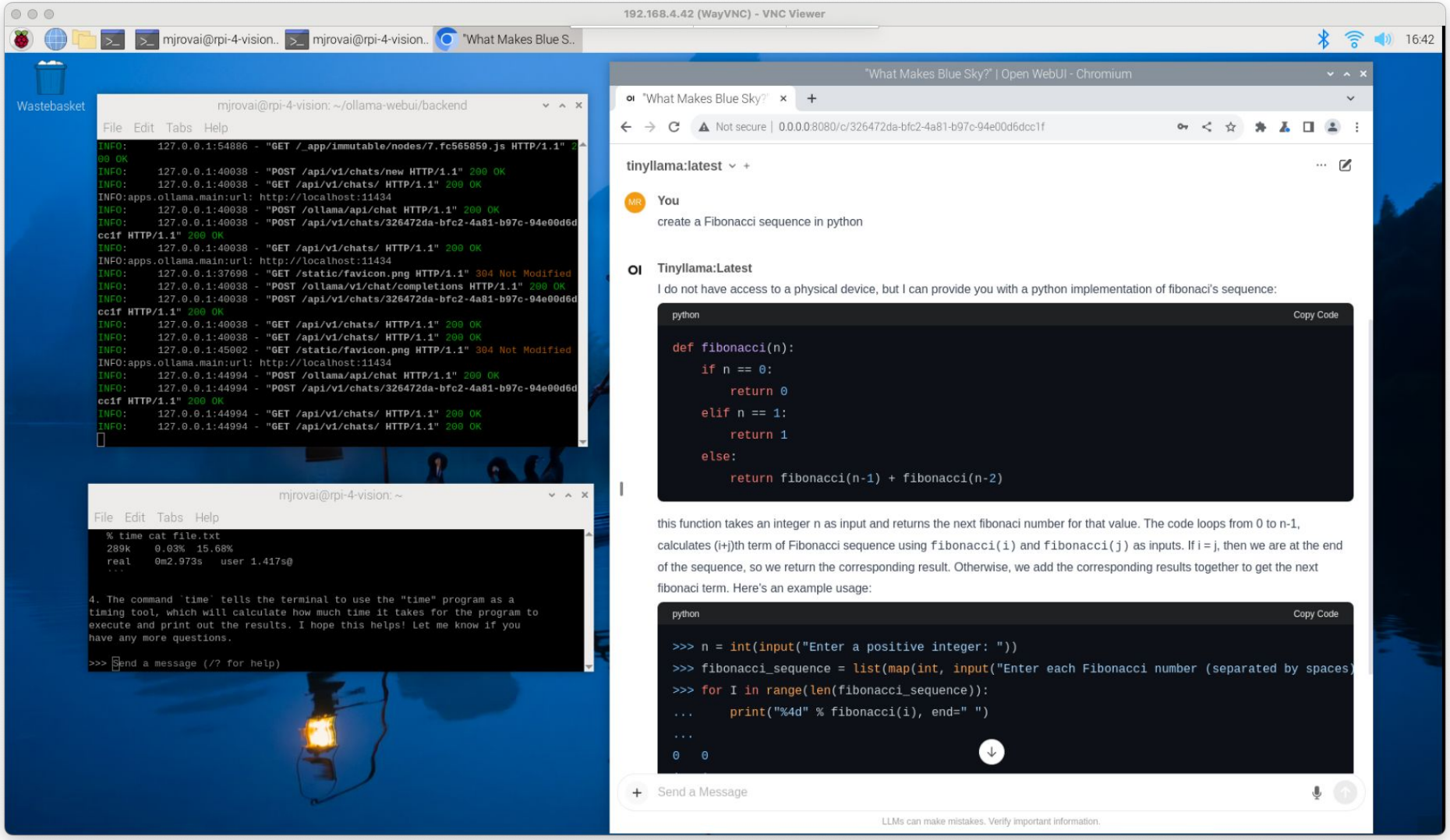
```
git clone https://github.com/ollama-webui/ollama-webui.git
cd ollama-webui/
```

3. Building Frontend Using Node

```
npm i
npm run build
```

4. Serving Frontend with the Backend

```
cd ./backend
pip install -r requirements.txt --break-system-packages
sh start.sh
```



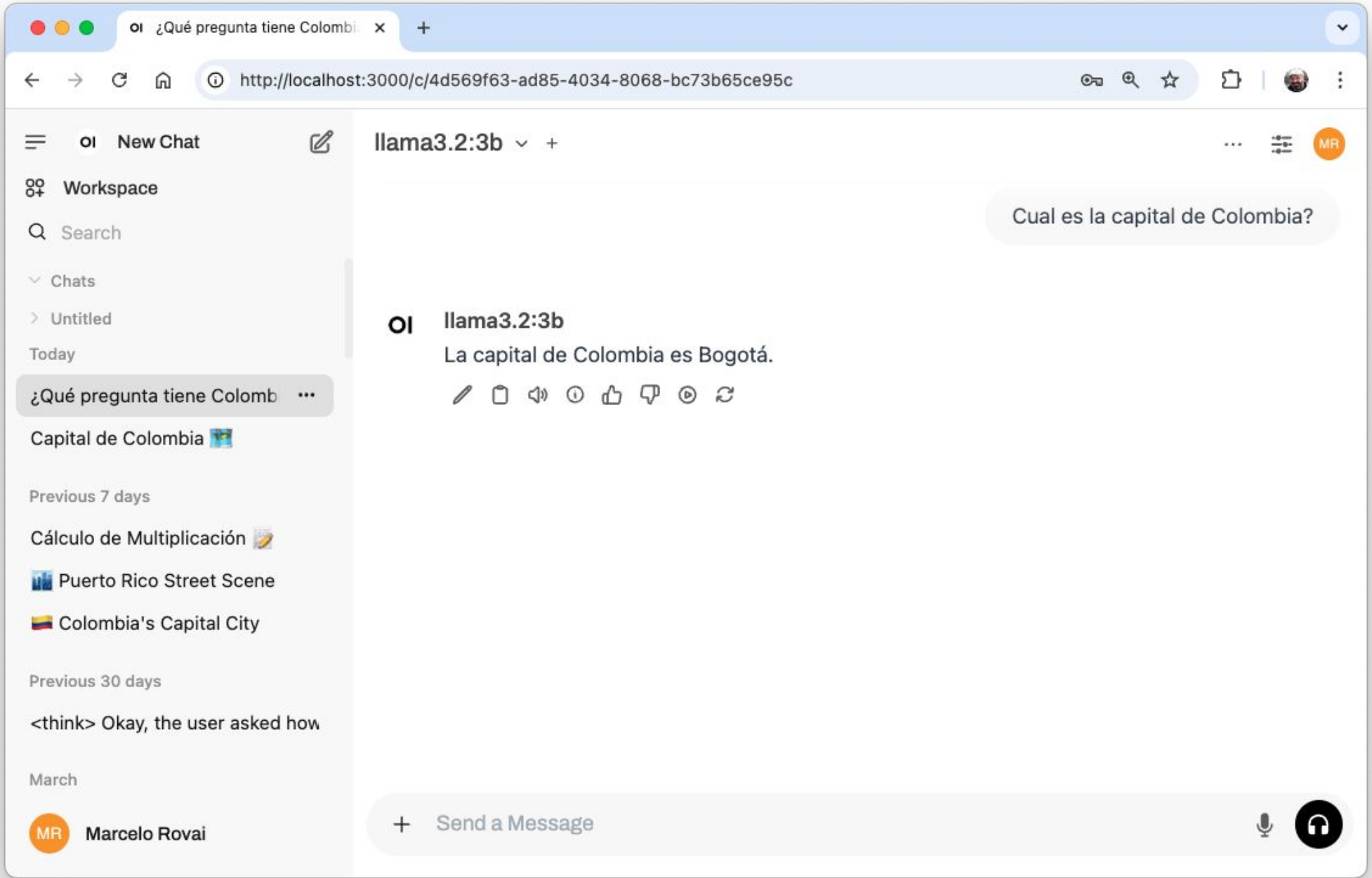
Access Ollama Web UI on port 8080 through http://0.0.0.0:8080 on the Raspberry Pi:

MAC / Windows

1. Install Docker Desktop:
   https://docs.docker.com/get-started/get-docker/

2. With Ollama installed on your computer, use this command:
   docker run -d -p 3000:8080 --add-host=host.docker.internal:host-gateway -v open-webui:/app/backend/data --name open-webui --restart always ghcr.io/open-webui/open-webui:main

https://github.com/open-webui/open-webui


¿Qué pregunta tiene Colombi
http://localhost:3000/c/4d569f63-ad85-4034-8068-bc73b65ce95c
New Chat
Workspace
Search
Chats
Untitled
Today
¿Qué pregunta tiene Colomb
Capital de Colombia
Previous 7 days
Cálculo de Multiplicación
Puerto Rico Street Scene
Colombia's Capital City
Previous 30 days
<think> Okay, the user asked how
March
MR
Marcelo Rovai
llama3.2:3b
MR
Cual es la capital de Colombia?
llama3.2:3b
La capital de Colombia es Bogotá.
Send a Message

Pontificia Universidad JAVERIANA Colombia

Questions?

Prof. Marcelo J. Rovai

rovai@unifei.edu.br

UNIFEI - Federal University of Itajuba, Brazil

TinyML4D - Academic Network Co-Chair

EdgeAIP - Academia-Industry Partnership Co-Chair