

Building a Private Coding Copilot with a Local LLM on Mac M1 Max

A walkthrough of setting up a local, offline AI assistant for coding using Ollama and DeepSeek-Coder

Introduction: I wanted a fast, private, and local coding assistant that didn't rely on cloud APIs or send data to external servers. With the rise of open-source LLMs and tools like Ollama, I realized I could run a full-featured code copilot directly on my Mac M1 Max. This document walks through the setup, tools used, model choices, and how I'm using this private AI assistant in my coding workflow. This project was developed with the assistance of AI tools to enhance research, analysis, and code efficiency.

System & Tools:

Hardware and Tools Used

Device: MacBook Pro (M1 Max chip)

Operating System: macOS Sonoma (or list your current version)

Local LLM Platform: [Ollama](#)

Optional Tools: Terminal, DeepSeek-Coder 6.7B (quantized for M1 compatibility)

Installation Steps:

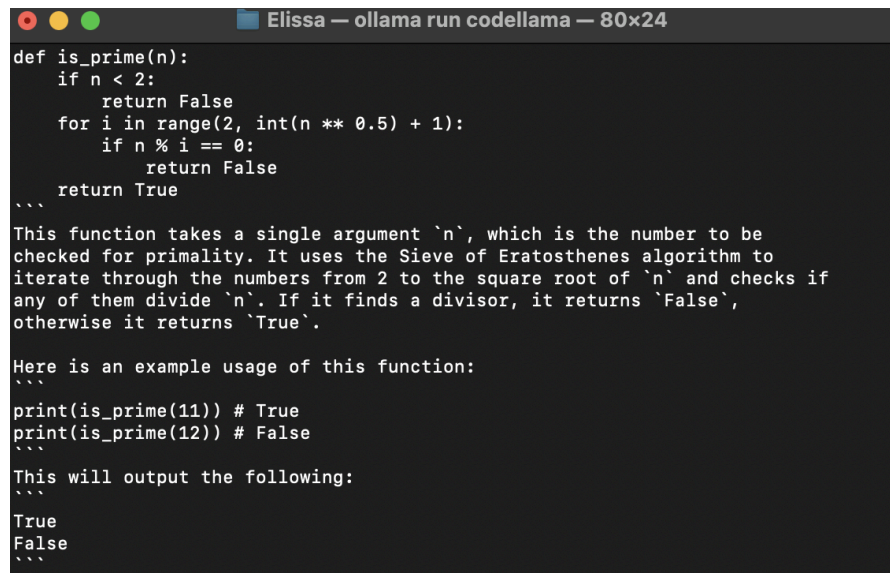
Install Ollama for macOS

Open in bash: `ollama run codellama`

Install DeepSeek: `ollama run deepseek-coder`

Test: Write a Python function that checks if a number is prime:

Output:



```
Elissa — ollama run codellama — 80x24
def is_prime(n):
    if n < 2:
        return False
    for i in range(2, int(n ** 0.5) + 1):
        if n % i == 0:
            return False
    return True
'''
This function takes a single argument 'n', which is the number to be
checked for primality. It uses the Sieve of Eratosthenes algorithm to
iterate through the numbers from 2 to the square root of 'n' and checks if
any of them divide 'n'. If it finds a divisor, it returns 'False',
otherwise it returns 'True'.

Here is an example usage of this function:
'''
print(is_prime(11)) # True
print(is_prime(12)) # False
'''
This will output the following:
'''
True
False
'''
```

Test: Can you help me debug this error in my code?

def calculate_total(items):

return sum(item.price for item in items)

```
Elissa — ollama run codellama — 140x50

a few issues with this code:

1. The 'sum' function expects an iterable as its argument, but 'item.price
for item in items' is not an iterable. It's a generator expression, which
is a type of expression that returns a generator object, rather than a
list or other iterable. To fix this, you can convert the generator
expression to a list using the 'list()' function:
'''
def calculate_total(items):
    return sum(list(item.price for item in items))
'''

2. The 'sum' function will only work if all of the elements in the
iterable are numbers or can be converted to numbers. In this case, you may
have items that do not have a price attribute, or where the price
attribute is not a number. You can handle this by using a conditional
statement to check for missing prices and returning 'None' if there is no
price:
'''
def calculate_total(items):
    total = 0
    for item in items:
        if hasattr(item, 'price') and isinstance(item.price, (int,
float)):
            total += item.price
        else:
            return None
    return total
'''

This will return 'None' if any of the items do not have a price or if the
price is not a number. You can also choose to throw an error instead of
returning 'None'.

3. Another issue with this code is that it does not handle negative
prices. If you want to allow for negative prices, you should modify the
conditional statement to check for both positive and negative numbers:
'''
def calculate_total(items):
    total = 0
    for item in items:
        if hasattr(item, 'price') and isinstance(item.price, (int, float))
and item.price >= 0:
            total += item.price
        else:
            return None
    return total
'''

This will allow for negative prices and handle them correctly by returning
'None'. You can also choose to throw an error instead of returning 'None'.
```

Test: Translate this Python code into JavaScript:

for i in range(10):

print(i)

```
Elissa — ollama run codellama — 75x16

... Translate this Python code into JavaScript:
... for i in range(10):
...     print(i)

Here is the equivalent JavaScript code for the above Python code:
'''
for (let i = 0; i < 10; i++) {
    console.log(i);
}
'''

Note that in JavaScript, the 'range' function is not necessary, as you can
use a simple loop to iterate over a range of numbers.
Additionally, the 'print' statement is used instead of the 'console.log' fu
nction in JavaScript to print output to the console.

>>> end a message (/? for help)
```

Building a Private AI Chatbot Trained on Internal Business Policies

Creating a secure, local LLM assistant using Ollama, LangChain, and simulated company documentation

Introduction: I wanted to demonstrate how a large language model (LLM) could be customized to assist with internal business workflows — specifically, answering questions about company policies and procedures in a secure, private environment.

Using open-source tools like Ollama and LlamaIndex, I built a fully local AI chatbot that runs entirely offline, without relying on cloud APIs or sending sensitive data over the internet.

This document walks through the tools used, how the model was connected to business policy data, and how the chatbot is able to retrieve and answer questions based on simulated company documentation — mimicking a real-world enterprise use case.

Simulated company policy:

<https://doc.clickup.com/9013904302/d/h/8cmagxe-53/ae1ae01503eecf7>

System & Tools:

Device: MacBook Pro (M1 Max chip)

Operating System: macOS Sonoma

LLM Platform: [Ollama](#) (local model runner)

LLM Used: Code LLaMA (switchable to DeepSeek-Coder or Mistral)

Embedding Model: Mistral (for local document embedding)

RAG Framework: [LlamaIndex](#) (for retrieval-augmented generation)

Document Type: Simulated internal HR and IT policy document (sample employee handbook)

Languages/Tools: Python 3.11, Terminal, venv for virtual environments

Installation:

Upload policy using bash:

```
cd ~/Documents
mkdir -p solara_chatbot/docs
cd solara_chatbot/docs
nano solara_employee_handbook.txt
```

Set up virtually environment and update packages:

```
cd ~/Documents/solara_chatbot
python3 -m venv venv
source venv/bin/activate
pip install langchain llama-index pypdf ollama
ollama pull mistral
```

Create python script:

```
touch chatbot.py
```

```
nano chatbot.py
```

Link to script used:

https://github.com/Eesterlein/Building-Local-LLMs-for-Private-Workflows/blob/main/solara_policy_chatbot.py

Run chatbot:

```
python chatbot.py
```

Test:

What's the maximum PTO carryover?

Output:

```
solara_chatbot — Python chatbot.py — 93x17
(venv) Elissa@Christophers-MacBook-Pro solara_chatbot % python3 chatbot.py

♦ SolaraTech Internal Chatbot ♦
Ask a question about the employee policy handbook. Type 'exit' to quit.

You: What's the maximum PTO carryover?

🧠 Bot: The maximum PTO carryover is 15 days of unused PTO that can be carried over to the next calendar year. Any unused time beyond that limit will be forfeited unless otherwise agreed in writing.

You:
🧠 Bot: Up to 15 days of unused PTO may be carried over to the next calendar year. Any unused time beyond that limit will be forfeited unless otherwise agreed in writing.

You: █
```

Test: What should I do if I lose a company laptop?

Output:

```
solara_chatbot — Python chatbot.py — 89x16

You: What should I do if I lose a company laptop?

🧠 Bot: The maximum PTO carryover is up to 15 days of unused time beyond that limit will be forfeited unless otherwise agreed in writing.

You:
🧠 Bot: If you lose a company laptop, you should report it to IT immediately. You can do this by submitting a lost device report through the SolaraTrack system, or by contacting your department head or HR directly. It's important to note that any lost equipment must be reported within 2 hours of discovery, regardless of location or time of day. Additionally, you should take steps to securely dispose of the laptop and avoid using public Wi-Fi without VPN enabled.

You: █
```


Test: How do I report a safety issue at the office?

Output:

```
solara_chatbot — Python chatbot.py — 95x29
You: How do I report a safety issue at the office?

🧠 Bot: The company's policy on reporting safety issues in the workplace is outlined in section 2 of the handbook. All employees are expected to adhere to posted safety protocols, which include appropriate workstation ergonomics, restricted access zones (RAZ), and incident logging in the Health & Safety portal (under Maintenance > Reports).

If a safety incident occurs, it is important to follow the established emergency procedures. Each floor is equipped with an Emergency Evacuation Map, which provides clear directions for evacuating the building. Drills occur quarterly and participation is mandatory. Employees should know the designated safety marshals in their departments.

In addition, employees are expected to stay home or work remotely if symptoms are mild. A written notice must be submitted to HR through the SolaraSelf system. If absent for more than 3 consecutive workdays, a return-to-work clearance may be required.

It is important to note that all safety protocols and emergency procedures are in place to ensure a safe and healthy work environment. Employees should take any concerns or incidents seriously and report them immediately to their supervisor or HR.

You:
🧠 Bot: To report a safety issue at the office, you can follow the emergency procedures outlined in section 2 of the employee handbook. Each floor is equipped with an Emergency Evacuation Map, and drills occur quarterly. Employees should know the designated safety marshals in their departments. If you are experiencing a safety issue that requires immediate attention, please contact HR immediately through the SolaraSelf system.

You: █
```

Project Summary:

This project demonstrates how a local LLM can be used to support internal business operations, such as HR or IT support, without relying on cloud APIs. By integrating open-source tools like ollama and LlamaIndex, I built a secure chatbot that can respond to questions about company policy documents using retrieval-augmented generation (RAG). This solution showcases how businesses can retain full data privacy while deploying powerful AI tools.