

به نام خدا

اجرای مدل های زبان بزرگ در Raspberry Pi

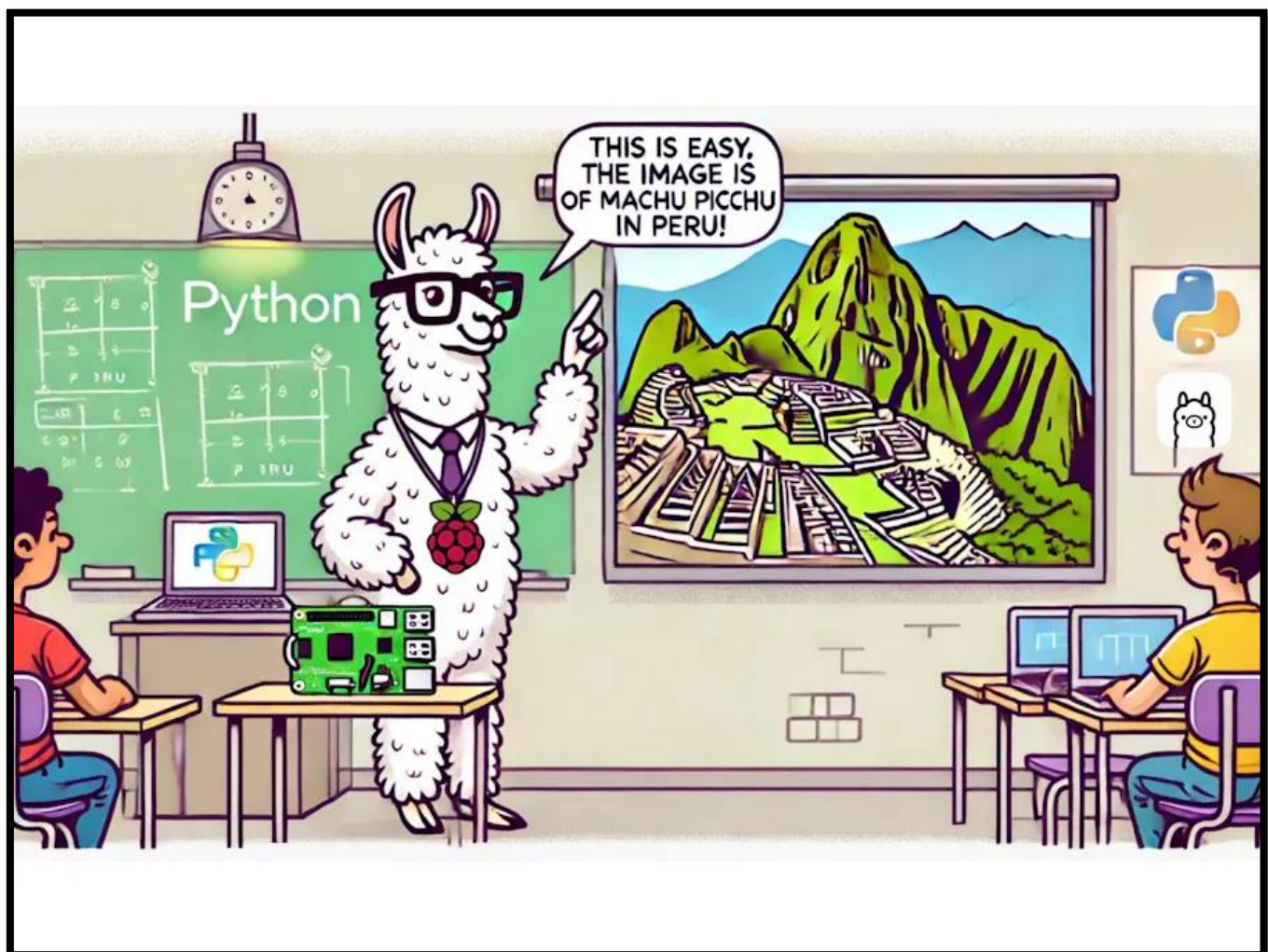
نام و نام خانوادگی: زهرا صفایی

درس: اصول Robotic

استاد: محمد زارع

Raspberry Pi

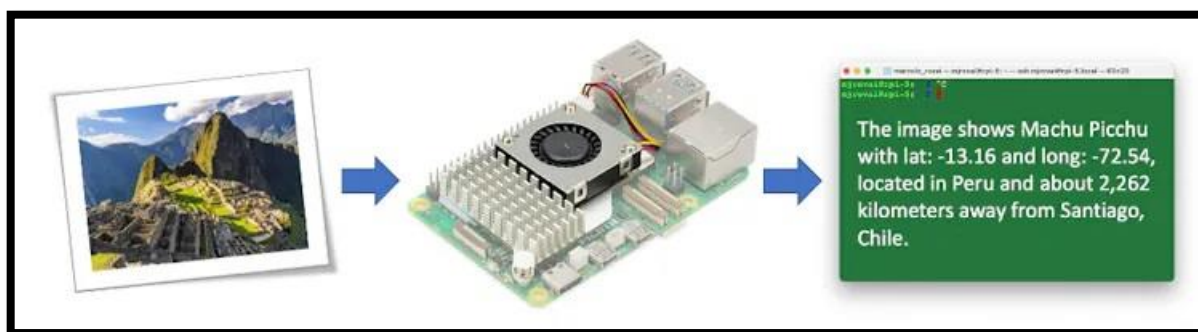
به یک هاب هوش مصنوعی قدرتمند تبدیل کنید با استفاده از Olama و Python، LLM ها را برای تجزیه و تحلیل داده ها و بینش در محل در زمان واقعی اجرا کنید.



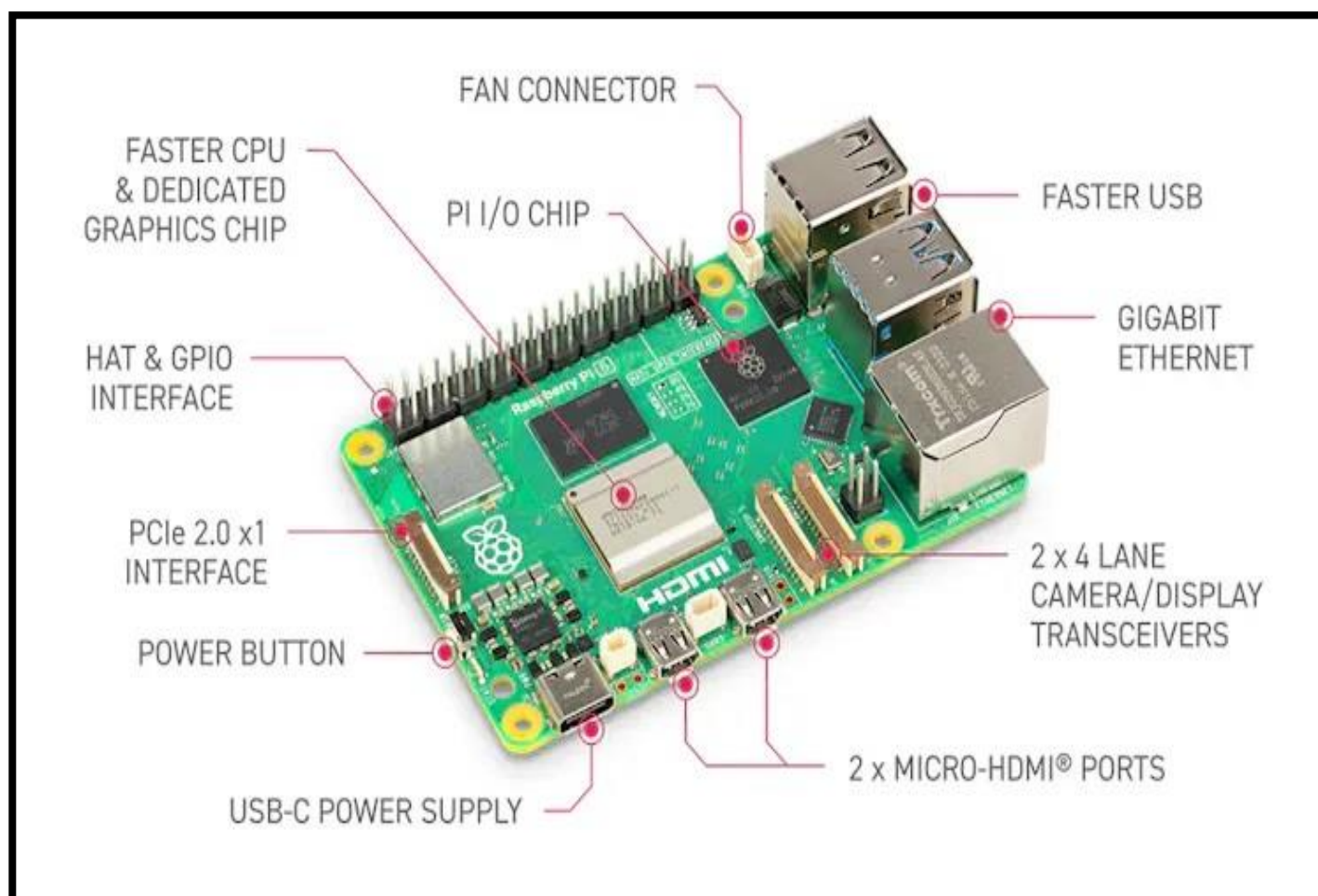
معرفی

در حوزه رشد سریع هوش مصنوعی، محاسبات لبه فرصتی را برای تمرکززدایی از قابلیت‌هایی که به طور سنتی برای سرورهای قدرتمند و متمرکز محفوظ است، ارائه می‌کند. این آموزش ادغام عملی نسخه‌های کوچک مدل‌های زبان سنتی بزرگ (LLM) را در Raspberry Pi ۵ بررسی می‌کند و این دستگاه لبه را به یک هاب هوش مصنوعی تبدیل می‌کند که قادر به پردازش داده‌ها در زمان واقعی و در محل است.

هدف ما کاوش در نصب و استفاده از Ollama است، یک چارچوب منبع باز که به ما امکان می‌دهد LLM ها را به صورت محلی روی دستگاه‌هایمان (که می‌تواند دسکتاپ یا دستگاه‌های لبه‌ای مانند Raspberry Pis یا NVidia Jetsons باشد) اجرا کنیم. Ollama به گونه ای طراحی شده است که کارآمد، مقیاس پذیر و آسان برای استفاده باشد و آن را به گزینه خوبی برای استقرار مدل های هوش مصنوعی مانند Llama، Phi، LLaVa (Multimodal) و TinyLLama تبدیل می کند. ما برخی از این مدل ها را در پروژه‌هایی با استفاده از اکوسیستم پایتون ادغام می‌کنیم و پتانسیل‌های آن‌ها را در سناریوهای دنیای واقعی بررسی می‌کنیم (یا حداقل در این جهت اشاره می‌کنیم).



راه اندازی سخت افزار



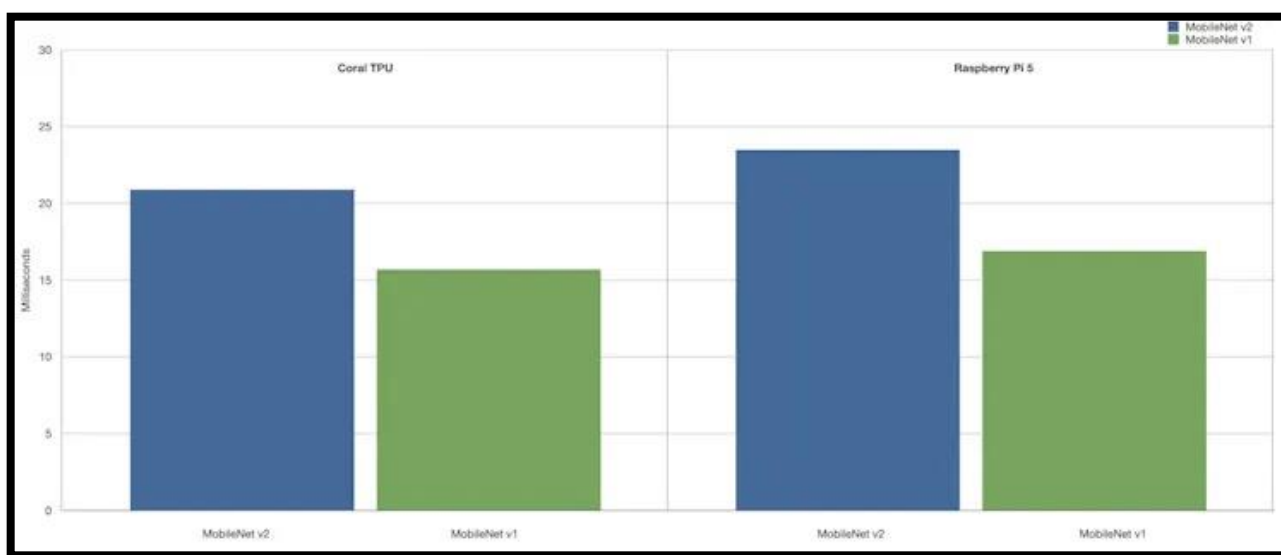
Raspberry Pi یک پلتفرم قوی است که نسخه ۴ و ۵ قبلی را به طور قابل ملاحظه ای ارتقا می دهد.

مجهز به Broadcom BCM ۲۷۱۲، یک پردازنده ۶۴ بیتی Arm Cortex-A ۷۶ چهار هسته ای با فرکانس ۲/۴ گیگاهرتز که دارای پسوند رمزنگاری و قابلیت های حافظه کش پیشرفته است. این پردازنده گرافیکی VideoCore VII، دو خروجی HDMI® ۶۰Kp۴ با HDR و رمزگشای ۶۰Kp۴ HEVC دارد.

گزینه های حافظه شامل ۴ گیگابایت و ۸ گیگابایت) انتخاب ما برای اجرای (LLM از LPDDR ۴ SDRAM X پرسرعت است.

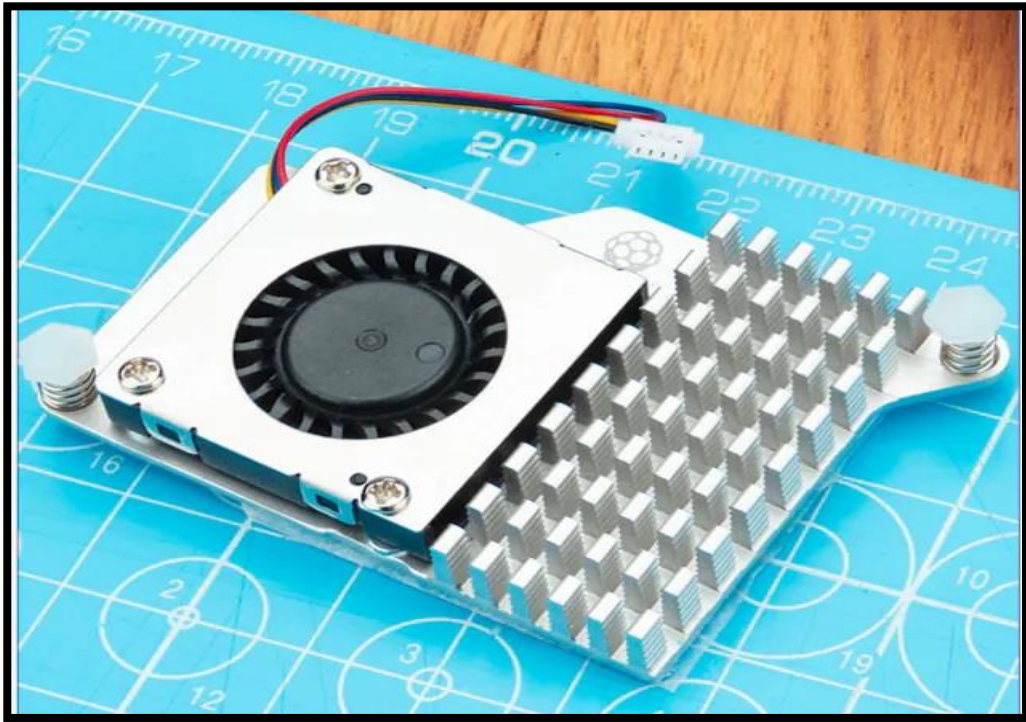
اتصال جامع است و Wi-Fi® ac ۸۰۲/۱۱ دو بانده، بلوتوث ۵/۰ BLE، اتترنت گیگابیتی با پشتیبانی از PoE+ و پورت های USB مختلف را ارائه می دهد. همچنین دارای حافظه قابل ارتقا از طریق اسلات کارت microSD، رابط PCIe ۲/۰ برای تجهیزات جانبی سریع، و ورودی برق ۵V/۵ A DC از طریق USB-C با Power Delivery است. ویژگی های اضافی شامل ساعت بی درنگ، دکمه روشن/خاموش و هدر استاندارد ۴۰ پین است که آن را به یک انتخاب همه کاره برای برنامه های کاربردی دنیای واقعی تبدیل می کند.

به هر حال، همانطور که Alasdair Allan بحث کرد، سخت‌افزار شتاب‌دهنده سفارشی ممکن است دیگر برای برخی از وظایف استنتاج در لبه مورد نیاز نباشد، زیرا استنباط مستقیم روی CPU Raspberry Pi ۵ بدون شتاب GPU - اکنون با عملکرد CoralTPU برابری می‌کند.

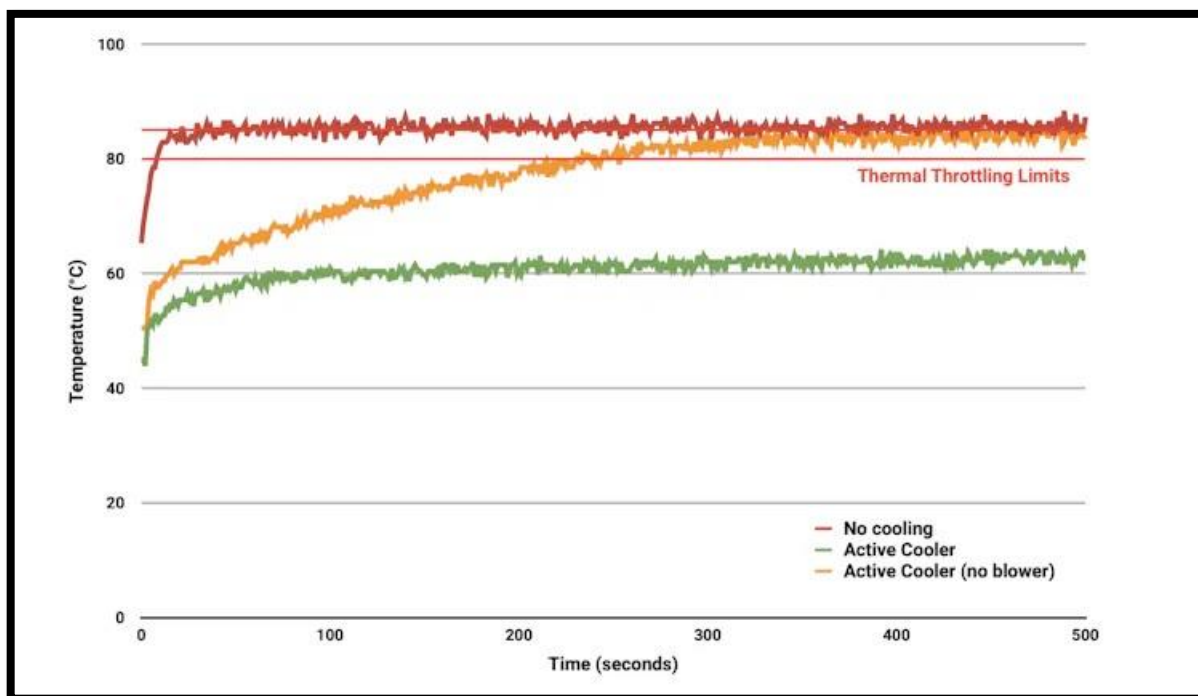


خنک کننده اکتیو رزبری پای

ما باید یک خنک کننده فعال، یک راه حل خنک کننده اختصاصی برای Raspberry Pi ۵ (Rasp) را برای این پروژه نصب کنیم. این دستگاه یک هیت سینک آلومینیومی را با یک فن دمیده با دمای کنترل شده ترکیب می‌کند تا Rasp ۵ را تحت بارهای سنگین مانند اجرای LLM به راحتی کار کند.



کولر اکتیو دارای پدهای حرارتی از قبل برای انتقال حرارت است و مستقیماً روی برد Raspberry Pi با استفاده از پین های فشار فنی نصب می شود. سیستم عامل Raspberry Pi به طور فعال آن را مدیریت می کند: در دمای ۶۰ درجه سانتی گراد، فن دمنده روشن می شود. در دمای ۶۷/۵ درجه سانتیگراد، سرعت فن افزایش می یابد. و در نهایت در دمای ۷۵ درجه سانتیگراد فن به سرعت کامل افزایش می یابد. فن دمنده زمانی که دما به زیر این حد می رسد به طور خودکار به پایین می چرخد.



برای جلوگیری از گرمای بیش از حد، تمام بردهای Raspberry Pi زمانی که دما به ۸۰ درجه می‌رسد شروع به دریاچه گاز می‌کنند و زمانی که به حداکثر دمای ۸۵ درجه سانتی‌گراد برسد، گاز را بیشتر می‌کند. جزئیات بیشتر، اینجا.

سیستم عامل را نصب کنید.

برای استفاده از Raspberry Pi به یک سیستم عامل نیاز داریم. به طور پیش فرض، Raspberry Pi یک سیستم عامل را روی هر کارت SD که در اسلات قرار داده شده است، بررسی می کند، بنابراین باید با استفاده از Raspberry Pi Imager، سیستم عاملی را نصب کنیم.

Raspberry Pi Imager ابزاری برای دانلود و نوشتن تصاویر در macOS، ویندوز و لینوکس است. این شامل بسیاری از تصاویر سیستم عامل محبوب برای Raspberry Pi است. ما همچنین از Imager برای از پیش پیکربندی اعتبارنامه ها و تنظیمات دسترسی از راه دور استفاده خواهیم کرد.

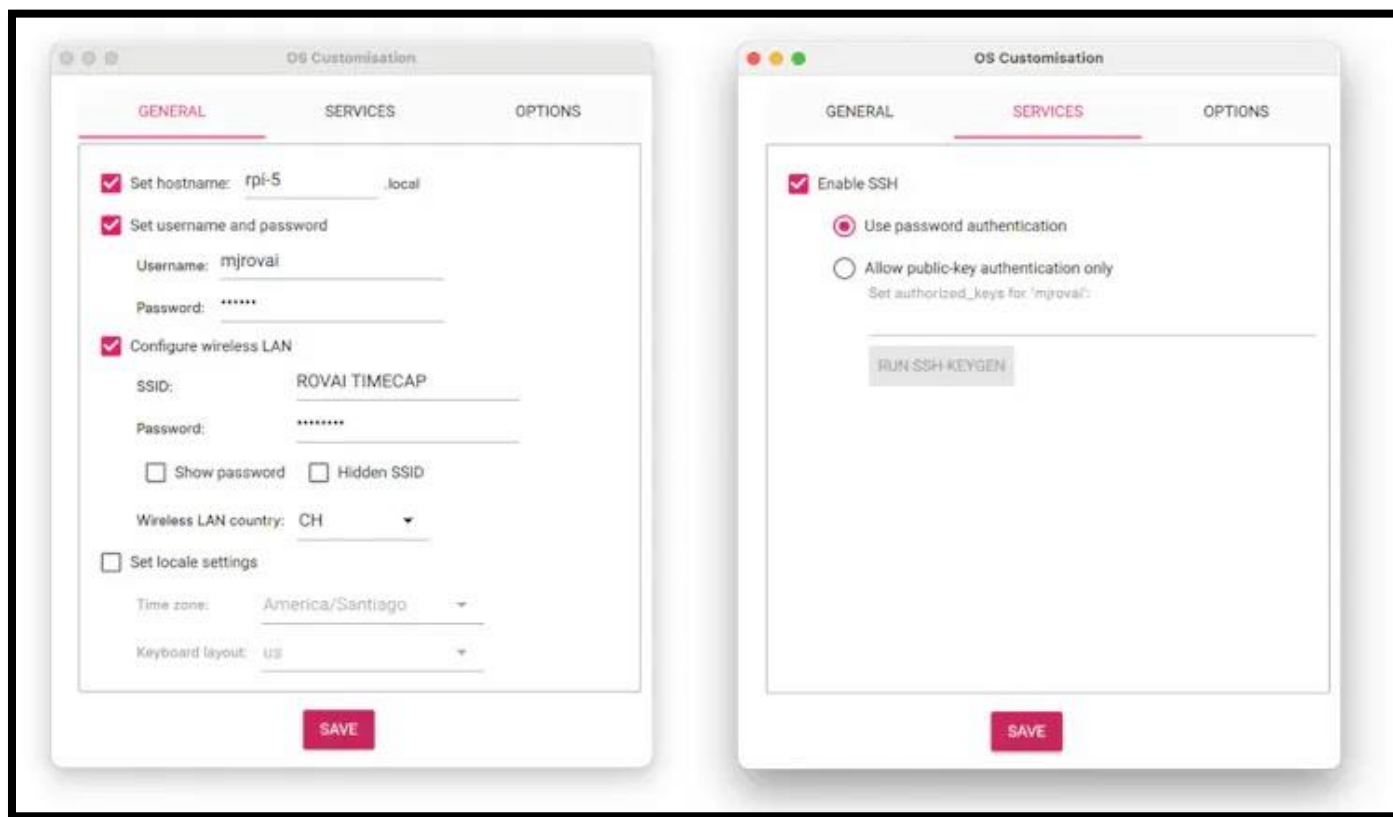
پس از دانلود Imager و نصب آن بر روی کامپیوتر خود، از یک کارت SD جدید و خالی استفاده کنید. دستگاه (RASPBerry PI ۵)، سیستم عامل (RASPBerry PI OS ۶۴-BIT-) و دستگاه ذخیره سازی خود را انتخاب کنید:



برای استفاده از Olama باید از نسخه ۶۴ بیتی استفاده کنیم.

همچنین باید گزینه هایی مانند نام میزبان، نام کاربری، رمز عبور، پیکربندی LAN (در GENERAL TAB) و مهمتر از آن، SSH Enable در SERVICES تب تعریف کنیم.

می‌توانید با استفاده از پروتکل (SSH) Secure Shell از راه دور از رایانه دیگری در همان شبکه به پایانه Raspberry Pi دسترسی پیدا کنید.



پس از رایت کردن سیستم عامل روی کارت SD، آن را در اسلات SD ۵ Raspi نصب کرده و منبع تغذیه ۵ ولت را وصل کنید.

via SSH ۵Interacting with the Rasp

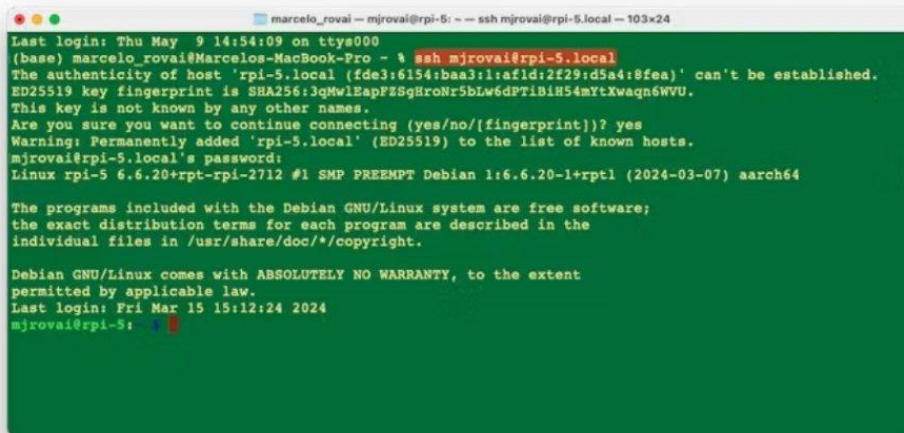
عامل با Rasp ۵ از طریق SSH

ساده ترین راه برای تعامل با Rasp-Zero از طریق SSH ("Headless") است. می‌توانید از ترمینال (MAC/Linux) یا PuTTY (ویندوز) استفاده کنید.

On terminal type:

```
ssh mjrovai@rpi-5.local
```

You should replace mjrovai with your *username* and rpi-5 with the *hostname* chosen during set-u



```
marcelo_rovai ~ mjrovai@rpi-5: ~ -- ssh mjrovai@rpi-5.local -- 103x24
Last login: Thu May  9 14:54:09 on ttya000
(base) marcelo.rovai@Marcelos-MacBook-Pro ~ % ssh mjrovai@rpi-5.local
The authenticity of host 'rpi-5.local (fde3:6154:bba3:1:af1d:2f29:d5a4:8fea)' can't be established.
ED25519 key fingerprint is SHA256:3gMw1EapFSSgHroNr5bLw6dPTiBiH54mYtXwaqn6WVU.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'rpi-5.local' (ED25519) to the list of known hosts.
mjrovai@rpi-5.local's password:
Linux rpi-5 6.6.20+rpt-rpi-2712 #1 SMP PREEMPT Debian 1:6.6.20-1+rpt1 (2024-03-07) aarch64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri Mar 15 15:12:24 2024
mjrovai@rpi-5: ~
```

When you see the prompt:

```
mjrovai@rpi-5:~ $
```

When you see the prompt: \$ ~:~mjrovai@rpi-

این بدان معناست که شما از راه دور با Raspberry ۵ خود در حال تعامل هستید. به روز رسانی منظم سیستم تمرین خوبی است. برای آن، شما باید اجرا کنید:

```
sudo apt-get update
```

Pip ابزاری برای نصب ماژول‌های خارجی پایتون در Raspberry Pi است. با این حال، در نسخه‌های اخیر سیستم عامل فعال نشده است. برای اجازه دادن به آن، باید دستور را (فقط یک بار) اجرا کنید:

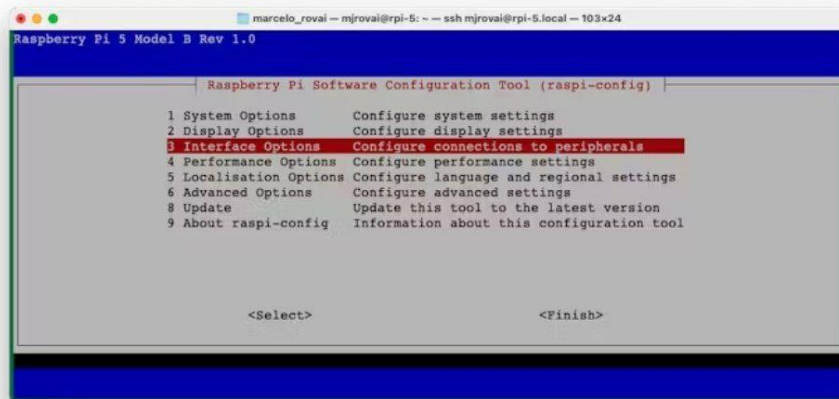
```
/EXTERNALLY-MANAGED³/ sudo rm /usr/lib/python
```

اجرای نسخه دسکتاپ Raspberry Pi از راه دور

را با دستور زیر اجرا کنید Raspberry ۵ پیکربندی

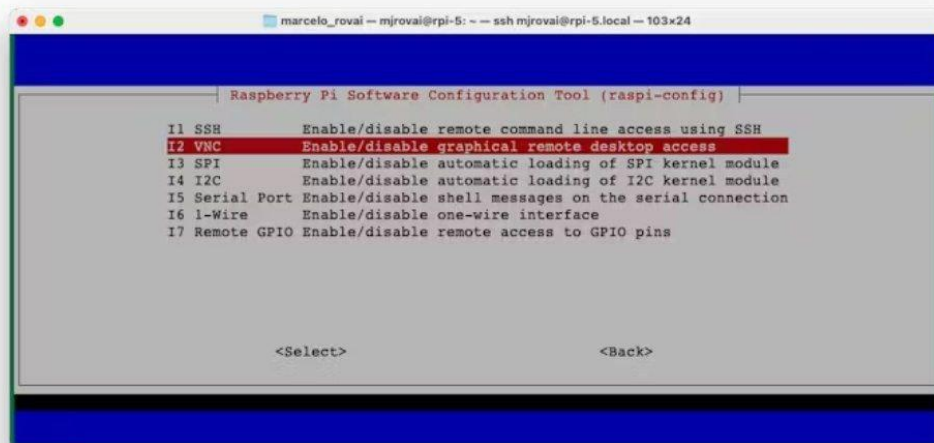
```
sudo raspi-config
```

and select 3 Interface Options:

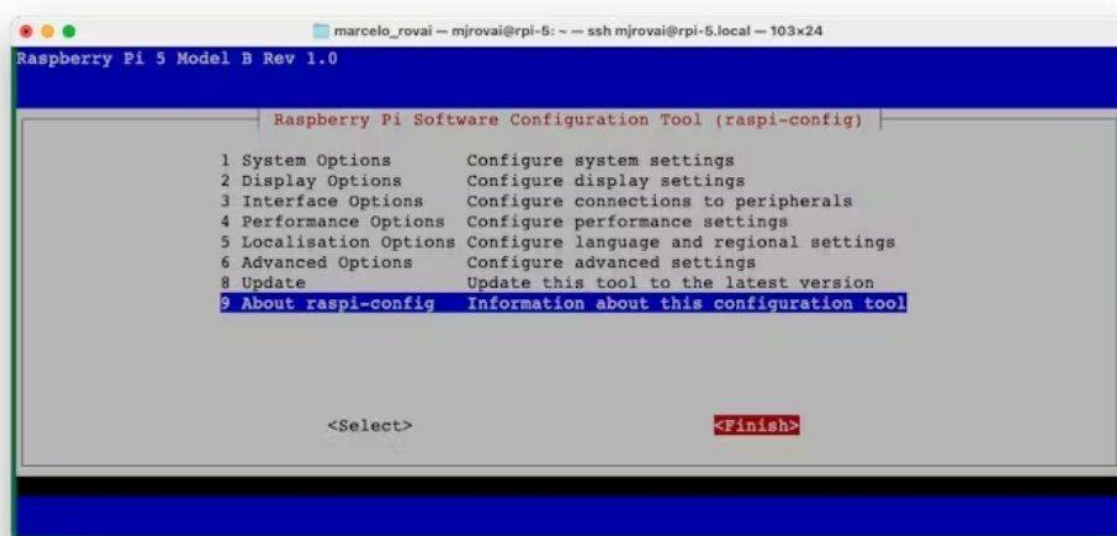


Once there, Select I2 VNC:

Once there, Select I2 VNC:



شما یک پاپ آپ دریافت خواهید کرد که درخواست تایید می کند <YES> . را بعد از <OK> فشار دهید. باید به صفحه اصلی برگردید. با استفاده از کلید تب، <Finish> را انتخاب کنید.

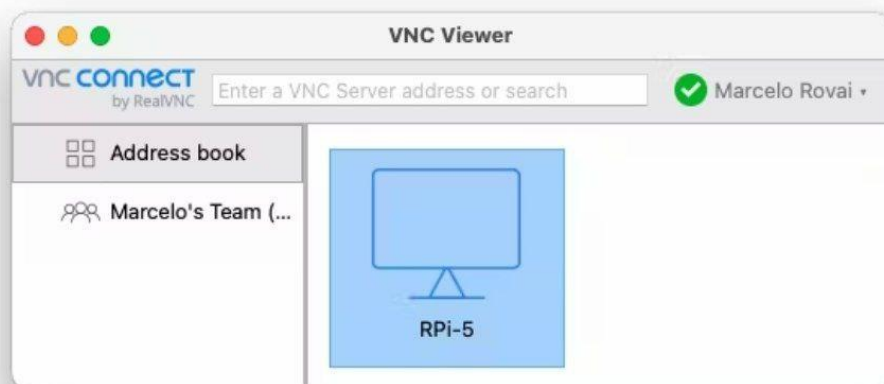


اکنون، باید VNC Viewer را روی رایانه خود نصب کنید. پس از نصب، باید آدرس IP Rasp ۵ را تأیید کنید. به عنوان مثال، در ترمینال، می توانید استفاده کنید.

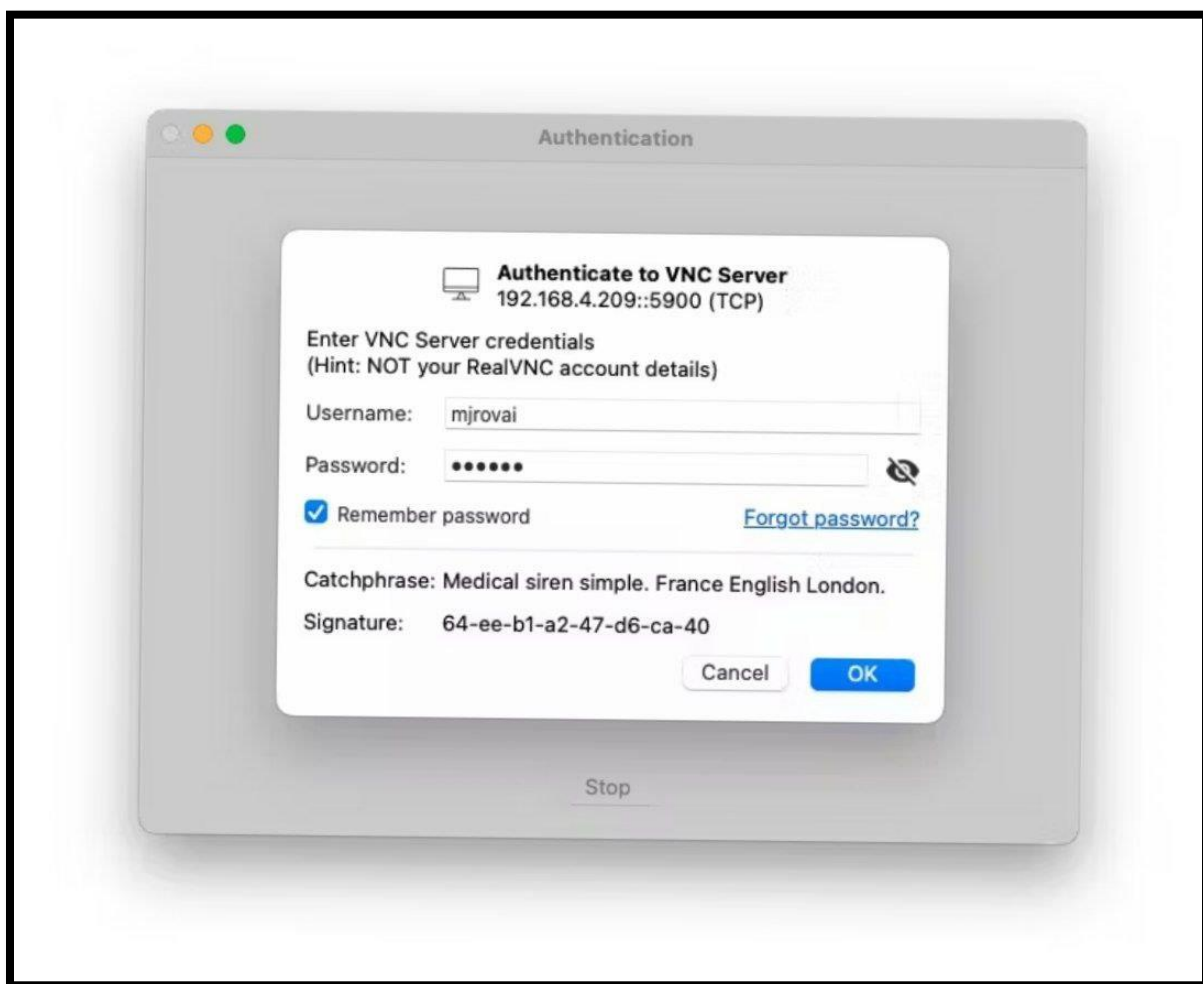

```
marcelo_rovai — mjrovai@rpi-5: ~ — ssh mjrovai@rpi-5.local — 57x5
mjrovai@rpi-5:~$ hostname -I
192.168.4.209 fde3:6154:baa3:1:af1d:2f29:d5a4:8fea
mjrovai@rpi-5:~$
```

Run the VNC Viewer, start a New Connection, and give a name (for example, RPi-5):

VNC Viewer را اجرا کنید، یک اتصال جدید راه اندازی کنید و یک نام بگذارید (به عنوان مثال RPi-5):

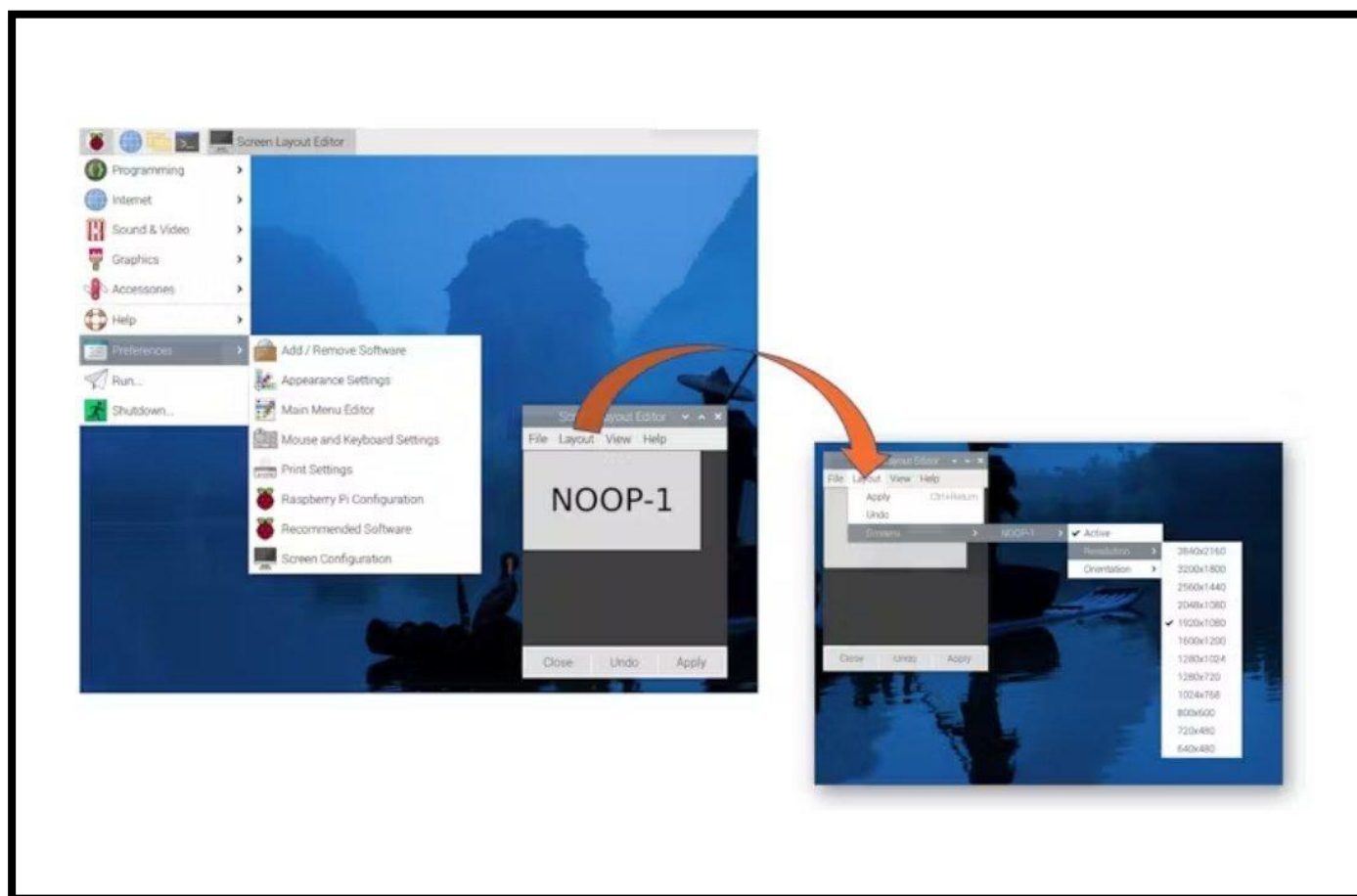


شما باید یک پنجره بازشو دریافت کنید که از شما می‌خواهد اعتبار شما را تأیید کند (نام کاربری و رمز عبور Rasp):



و همین است! Raspberry Pi Desktop باید روی مانیتور کامپیوتر شما ظاهر شود.

به طور پیش فرض، رزولوشن دسکتاپ زیاد نیست، اما می‌توانید با رفتن به منو (نماد Raspberry در گوشه بالا سمت چپ) و انتخاب بهترین تعریف صفحه نمایش برای مانیتور، آن را تغییر دهید:



انتقال فایل ها بین Rasp ۵ و دسکتاپ

انتقال فایل ها بین Rasp ۵ و کامپیوتر اصلی ما می تواند با استفاده از یک درایو قلم یا یک برنامه FTP از طریق شبکه انجام شود. برای مورد آخر، از FileZilla FTP Client استفاده کنیم.

دستورالعمل ها را دنبال کنید و برنامه را برای سیستم عامل دسکتاپ خود نصب کنید.

انتقال فایل ها بین Rasp و دسکتاپ:

انتقال فایل ها بین Rasp و کامپیوتر اصلی ما می تواند با استفاده از یک درایو قلم یا یک برنامه FTP از طریق شبکه انجام شود. برای مورد آخر، از FileZilla FTP Client استفاده کنیم.

دستورالعمل ها را دنبال کنید و برنامه را برای سیستم عامل دسکتاپ خود نصب کنید.

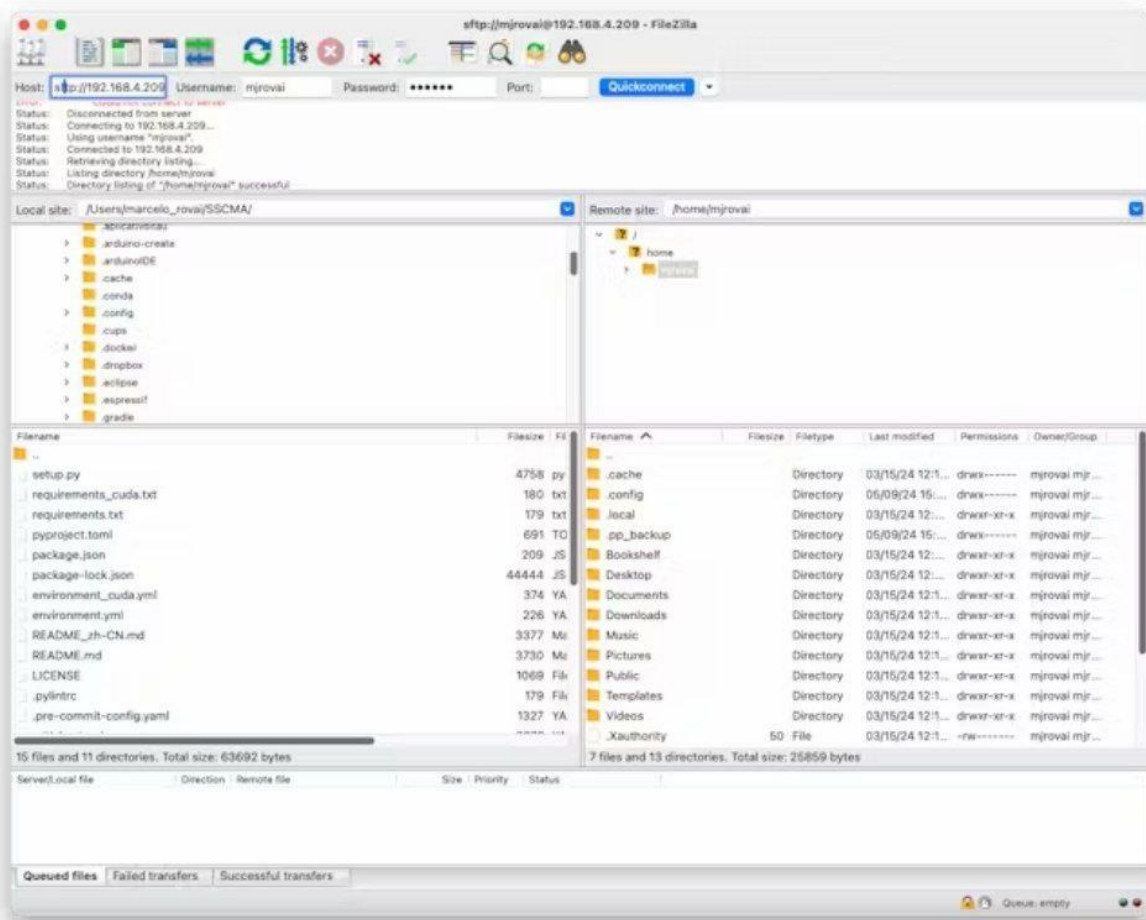
لازم است بدانید که Rasp IP ۵ مانند قبل در استفاده از ترمینال چیست:

نام میزبان ۱-

از این IP به عنوان میزبان در فایل استفاده کنید:

sftp://۱۹۲/۱۶۸/۴/۲۰۹

و نام کاربری و رمز عبور Rasp خود را وارد کنید. با فشار دادن Quickconnect دو پنجره جداگانه باز می شود، یکی برای دسکتاپ شما (راست) و دیگری برای Rasp (چپ).



اولاما

ما در اینجا جزئیاتی در مورد چستی Ollama و نحوه عملکرد آن در زیر کاپوت وارد نمی کنیم. اما باید این ویدیوی کوتاه از مت ویلیامز درباره اولاما را ببینید:

Installing Ollama

```
curl https://ollama.ai/install.sh | sh
```

در نتیجه، یک API در پس‌زمینه در ۱۱۴۳۴/۰/۰/۱۲۷ اجرا می‌شود (در مورد من). از این پس می‌توانید Ollama را از طریق ترمینال اجرا کنید.

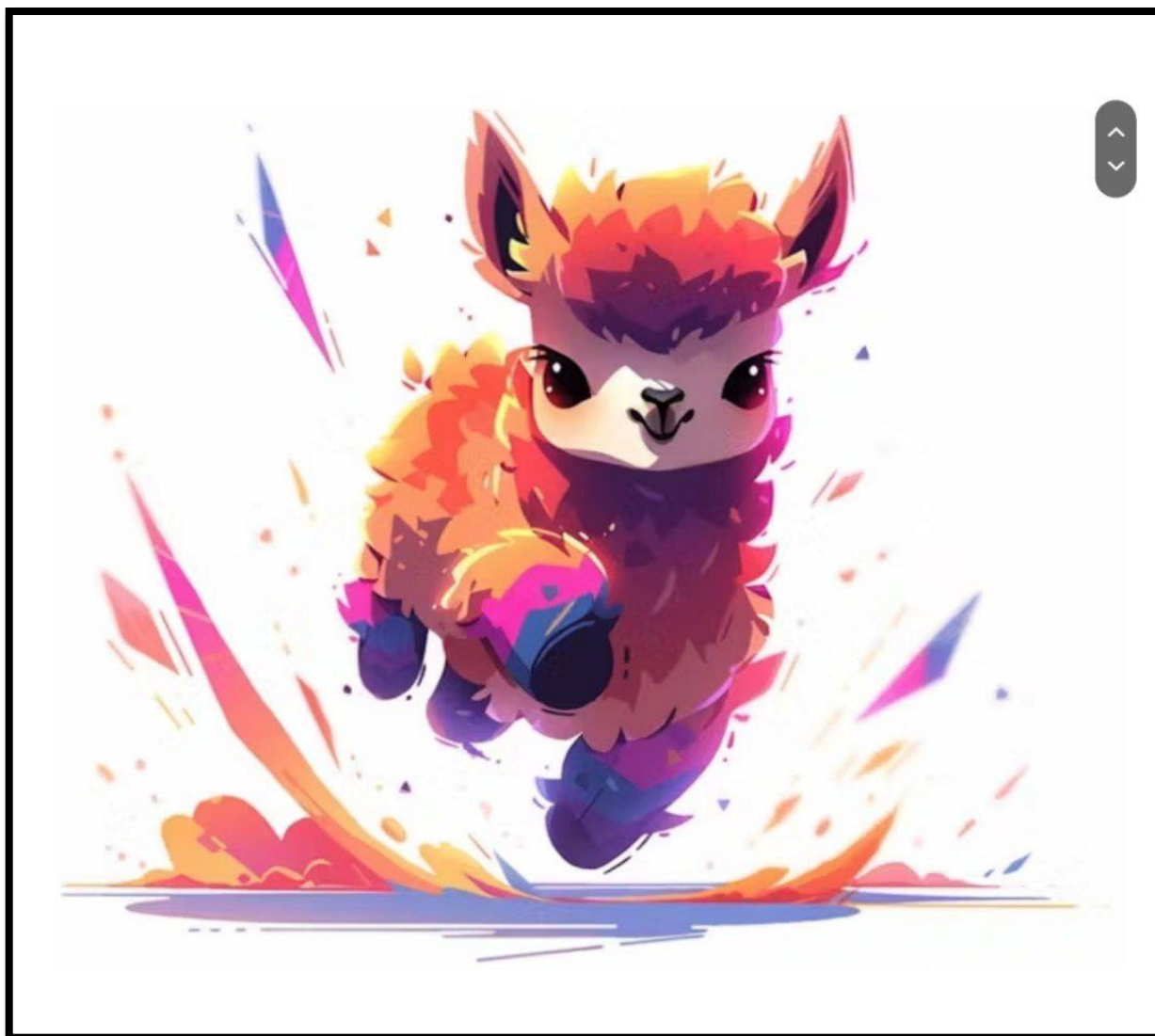
```
mjrovai@rpi-5: ~  
File Edit Tabs Help  
mjrovai@rpi-5:~ $ curl https://ollama.ai/install.sh | sh  
% Total % Received % Xferd Average Speed Time Time Time Current  
 Dload Upload Total Spent Left Speed  
0 0 0 0 0 0 0 0 --:--:-- --:--:-- --:--:-- 0>>  
> Downloading ollama...  
100 10941 0 10941 0 0 29032 0 --:--:-- --:--:-- --:--:-- 29021  
##### 100.0%#  
##### 100.0%  
>>> Installing ollama to /usr/local/bin...  
>>> Creating ollama user...  
>>> 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...  
Created symlink /etc/systemd/system/default.target.wants/ollama.service → /etc/s  
ystemd/system/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.  
mjrovai@rpi-5:~ $
```

متأسفانه، پردازنده گرافیکی Rasp ۵ توسط Ollama برای سرعت بخشیدن به استنتاج استفاده نمی‌شود. ما امیدواریم که این موضوع در آینده تغییر کند و دستگاه‌های جدیدتری چون Raspberry Pi AI KIT قابل استفاده باشند.

با اجرای زیر می‌توان تأیید کرد که Ollama به درستی نصب شده است:

```
ollama list
```

هنگامی که ما هنوز هیچ مدلی را نصب نکردیم، باید یک لیست خالی ببینید. بیایید اولین مدل خود را نصب و اجرا کنیم، TinyLlama، یک مدل ۱/۱B Llama از پیش آموزش دیده با ۳ تریلیون توکن.



TinyLlama یک مدل زبان کوچک اما همچنان قوی است که می‌تواند برای بسیاری از برنامه‌ها، مانند استقرار در دستگاه‌های لبه با حافظه محدود و ظرفیت‌های محاسباتی، برای عملکردهایی مانند ترجمه ماشینی بلادرنگ بدون اتصال به اینترنت مفید باشد (TinyLlama-۱.۱ کوانتیزه شده با ۴ بیت وزن B تنها ۶۳۷ مگابایت را اشغال می‌کند، که امکان تولید گفتگوی بی‌درنگ در بازی‌های ویدیویی و استفاده در اتوماسیون خانگی را فراهم می‌کند.

Install the Model

```
ollama pull tinyllama
```

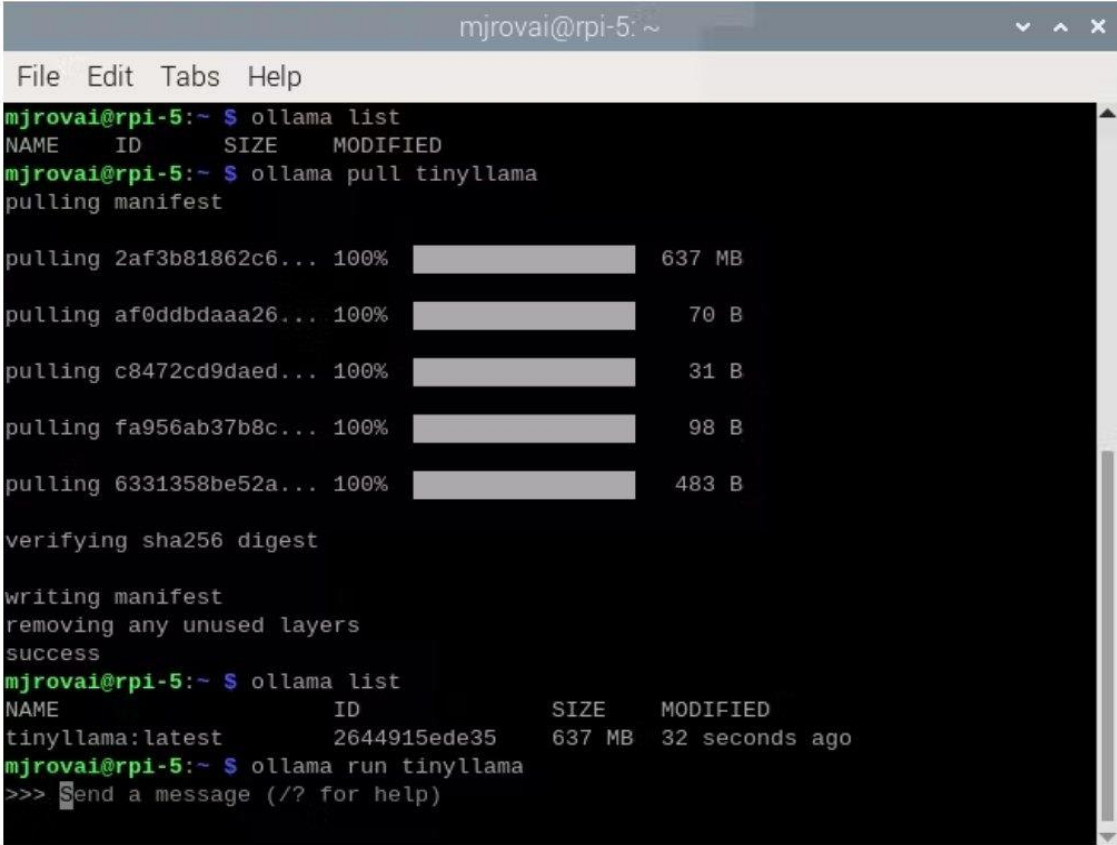
حالا اگر دوباره اجرا کنید:

```
ollama list
```

باید ببینید که یک مدل جدید با حجم ۶۳۶ مگابایت نصب شده است.

```
ollama run tinyllama
```

اکنون، با اجرای مدل با دستور قبلی، باید اعلان اولاما را در دسترس داشته باشید تا بتوانیم یک سؤال را وارد کنیم و با مدل LLM چت کنیم:



```
mjrovai@rpi-5: ~  
File Edit Tabs Help  
mjrovai@rpi-5:~ $ ollama list  
NAME      ID      SIZE      MODIFIED  
mjrovai@rpi-5:~ $ ollama pull tinylama  
pulling manifest  
  
pulling 2af3b81862c6... 100% ██████████ 637 MB  
pulling af0ddbdaaa26... 100% ██████████ 70 B  
pulling c8472cd9daed... 100% ██████████ 31 B  
pulling fa956ab37b8c... 100% ██████████ 98 B  
pulling 6331358be52a... 100% ██████████ 483 B  
  
verifying sha256 digest  
writing manifest  
removing any unused layers  
success  
mjrovai@rpi-5:~ $ ollama list  
NAME      ID      SIZE      MODIFIED  
tinylama:latest 2644915ede35 637 MB 32 seconds ago  
mjrovai@rpi-5:~ $ ollama run tinylama  
>>> Send a message (/? for help)
```

مثال ها:

>>> پایتخت فرانسه کدام است؟

پایتخت فرانسه پاریس است. نام رسمی این شهر «پاریس» و نام سابق آن «شهر نورها» بوده است. با این حال، استفاده از «شهر نورها» به عنوان نام رسمی پاریس در فرانسه یک روش معمول نیست. در عوض، بیشتر با نام اختصاری آن، «پاریس» شناخته می شود.

>>> فاصله بین پاریس و سانتیاگو، شیلی چقدر است؟

فاصله بین پاریس و سانتیاگو، شیلی تقریباً ۷۰۴۲ مایل یا ۱۱۳۲۹ کیلومتر است. حدود ۱۸ ساعت و ۵۶ دقیقه طول می‌کشد تا با ماشین یا هواپیما سفر کنید. فاصله پرواز ممکن است بسته به خط هوایی و عوامل دیگر متفاوت باشد.

>>> طول و عرض جغرافیایی پاریس چقدر است؟

طول و عرض جغرافیایی پاریس به شرح زیر است:

عرض جغرافیایی: ۴۸/۸۵۳ درجه تا ۴۹/۶۲۴ درجه

طول جغرافیایی: -۰/۱۷ درجه تا -۰/۸۳ درجه

از این مختصات می‌توان برای یافتن فاصله بین پاریس و نقاط مختلف سطح زمین مانند شهرها، نقاط دیدنی و مکان‌های دیگر استفاده کرد.

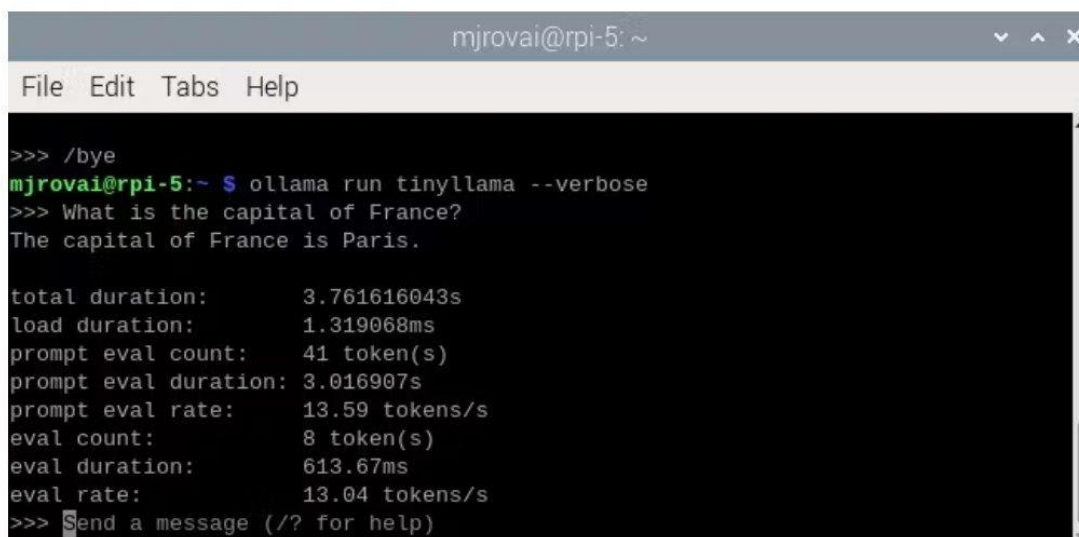
شگفت آور است که چگونه چنین مدل کوچکی به طور کاملاً آفلاین بر روی دستگاه لبه ای مانند Raspberry ۵ کار می کند. با توجه به اینکه ما فقط از CPU استفاده می کنیم، سرعت پاسخگویی بسیار خوب است. هنگام فراخوانی مدل، همچنین می توانیم آماری درباره عملکرد استنتاج مدل با استفاده از `--verbose` به دست آوریم. ابتدا باید استفاده از مدل را متوقف کنیم:

/ >>> بای

و دوباره آن را اجرا کنید:

and run it again:

```
ollama run tinyllama --verbose
```



```
mjrovai@rpi-5: ~  
File Edit Tabs Help  
>>> /bye  
mjrovai@rpi-5:~ $ ollama run tinyllama --verbose  
>>> What is the capital of France?  
The capital of France is Paris.  
  
total duration:      3.761616043s  
load duration:      1.319068ms  
prompt eval count:   41 token(s)  
prompt eval duration: 3.016907s  
prompt eval rate:    13.59 tokens/s  
eval count:          8 token(s)  
eval duration:       613.67ms  
eval rate:           13.04 tokens/s  
>>> Send a message (/? for help)
```

مثلا دوباره پرسیدیم پایتخت فرانسه چیست؟ این بار مدل فقط یک پاسخ کوتاه به ما داد که با قبل متفاوت بود اما همچنان درست بود. همچنین، ما چندین آمار در مورد معیارهای عملکرد و زمان بندی برای مدل زبان (LLM) اجرا شده در Raspberry Pi دریافت کردیم.

هر معیار بینش هایی در مورد نحوه پردازش ورودی ها و تولید خروجی توسط مدل ارائه می کند. در اینجا خلاصه ای از معنای هر معیار آورده شده است:

- مدت زمان کل (۳/۷۶۱۶۱۶۰۴۳s):

این زمان کامل از شروع فرمان تا تکمیل پاسخ است. این شامل بارگذاری مدل، پردازش اعلان ورودی و تولید پاسخ است.

Load Duration (۱/۳۱۹۰۶۸ms) :

این مدت زمان بسیار کوتاه زمان بارگذاری مدل یا اجزای لازم در حافظه را نشان می دهد. با توجه به مختصر بودن آن، این نشان می دهد که مدل از قبل بارگذاری شده بود یا فقط به حداقل تنظیمات نیاز بود.

Prompt Eval Duration (۳/۰۱۶۹۰۷s):

این زمان مدل برای ارزیابی یا پردازش اعلان ورودی را اندازه گیری می کند. بخش عمده ای از کل مدت زمان را به خود اختصاص می دهد، به این معنی که درک پرس و جو و آماده سازی پاسخ، زمان برترین بخش فرآیند است.

Prompt Eval Rate (توکن در ثانیه ۱۳/۵۹):

این نرخ نشان می دهد که مدل با چه سرعتی توکن ها را از اعلان ورودی پردازش می کند. سرعت مدل را از نظر درک زبان طبیعی منعکس می کند.

Eval Count (توکن ۸):

این تعداد توکن در پاسخ مدل است که در این مورد عبارت بود از: "پایتخت فرانسه پاریس است".

مدت زمان ارزشیابی (۶۱۳/۶۷ میلی ثانیه):

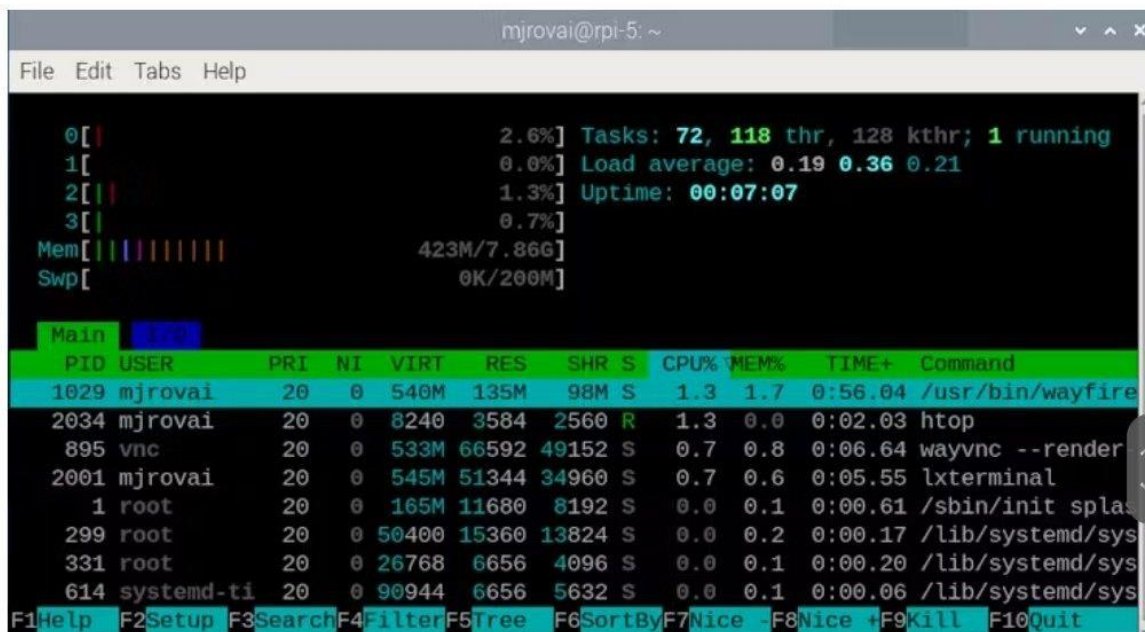
این زمان صرف شده برای تولید خروجی بر اساس ورودی ارزیابی شده است. این بسیار کوتاهتر از ارزیابی سریع است، و نشان می دهد که ایجاد پاسخ نسبت به درک اعلان پیچیده تر یا از نظر محاسباتی فشرده تر است.

این تفکیک دقیق می‌تواند به درک نیازهای محاسباتی و ویژگی‌های عملکرد LLM‌های در حال اجرا مانند TinyLlama در دستگاه‌های لبه‌ای مانند Raspberry Pi ۵ کمک کند. این نشان می‌دهد که اگرچه ارزیابی سریع زمان‌برتر است، تولید واقعی پاسخ‌ها نسبتاً سریع‌تر است. این تجزیه و تحلیل برای بهینه‌سازی عملکرد و تشخیص تنگناهای بالقوه در برنامه‌های کاربردی بلادرنگ بسیار مهم است.

بازرسی منابع محلی

با استفاده از htop، می‌توانید به راحتی منابع در حال اجرا در دستگاهتان را نظارت کنید. با دستور در ترمینال وارد کنید:

htop



PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
1029	mjrovai	20	0	540M	135M	98M	S	1.3	1.7	0:56.04	/usr/bin/wayfire
2034	mjrovai	20	0	8240	3584	2560	R	1.3	0.0	0:02.03	htop
895	vnc	20	0	533M	66592	49152	S	0.7	0.8	0:06.64	wayvnc --render
2001	mjrovai	20	0	545M	51344	34960	S	0.7	0.6	0:05.55	lxterminal
1	root	20	0	165M	11680	8192	S	0.0	0.1	0:00.61	/sbin/init splash
299	root	20	0	50400	15360	13824	S	0.0	0.2	0:00.17	/lib/systemd/sys
331	root	20	0	26768	6656	4096	S	0.0	0.1	0:00.20	/lib/systemd/sys
614	systemd-ti	20	0	90944	6656	5632	S	0.0	0.1	0:00.06	/lib/systemd/sys

پس از راه اندازی مجدد، می بینیم که سیستم با ۴۲۴ مگابایت از ۸ گیگابایت SRAM خود کار می کند و ۴ پردازنده مشترک کار زیادی انجام نمی دهند. بیایید TinyLlama را دوباره اجرا کنیم.

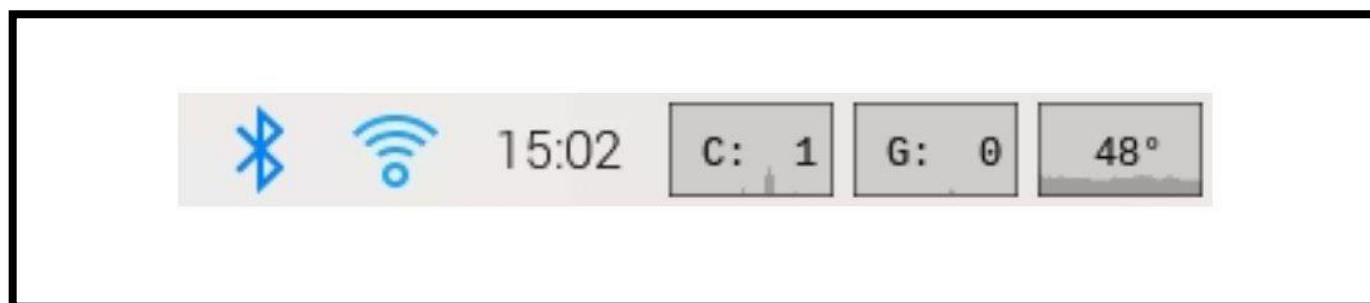
```
ollama run tinyllama --verbose
```

حافظه حدود ۱۰۰ مگابایت افزایش یافته است
بیایید چت کنیم و دوباره به حلقه نگاه کنیم:

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
2154	ollama	20	0	1152M	678M	610M	R	377.5	8.4	0:31.69	/tmp/ollama77710
2263	ollama	20	0	1152M	678M	610M	R	98.2	8.4	0:02.50	/tmp/ollama77710
2265	ollama	20	0	1152M	678M	610M	R	98.5	8.4	0:02.50	/tmp/ollama77710
2264	ollama	20	0	1152M	678M	610M	R	97.9	8.4	0:02.54	/tmp/ollama77710
1029	mjrovai	20	0	543M	137M	100M	S	15.2	1.7	1:15.02	/usr/bin/wayfire
2001	mjrovai	20	0	549M	53584	37200	S	2.6	0.6	0:07.14	lxterminal
895	vnc	20	0	533M	66592	49152	S	2.0	0.8	0:09.93	wayvnc --render-
674	avahi	20	0	7744	3072	2560	S	0.7	0.0	0:00.20	avahi-daemon: ru

حافظه
بیایید

می بینیم که هر ۴ CPU با ۱۰۰٪ ظرفیت خود کار می کنند.
نظارت بر دما نیز مهم است. می توانید دمای نشان داده شده در نوار
وظیفه را داشته باشید:



اگر هیچ کاری انجام نمی دهید، برای یک CPU که با ۱٪ کار می کند،
دما حدود ۴۸ درجه سانتیگراد است. در طول استنتاج، با CPU در
۱۰۰٪، دما به ۶۲ درجه سانتیگراد افزایش یافت

Pulling a bigger model

۳، خانواده‌ای از PHI بیایید یک مدل بزرگ‌تر (هنوز کوچک) به نام
(متوسط) سبک وزن توسط B و ۱۴ (Mini) مدل‌های باز پیشرفته ۳
را اجرا کنیم B طراحی کنیم. بیایید مدل ۳ Microsoft

```
olama run phi3 --verbose
```

توجه داشته باشید که به جای استفاده از دستور llama pull model، از
دستور run مستقیم استفاده می‌کنیم. اتفاقی که می افتد این است که
اولا مدل های کشیده شده را یادداشت می کند و زمانی که PHI ۳
وجود نداشته باشد، قبل از اجرای آن، اولاما آن را می کشد.

بیایید با همان دستوری که قبلا استفاده شده بود وارد شویم:

<<< پایتخت فرانسه کدام است؟

پایتخت فرانسه پاریس است. این شهر نه تنها بزرگترین شهر کشور بلکه مرکز جهانی هنر، مد، غذا و فرهنگ است. جمعیت کلان شهری آن در قرن نوزدهم در محدوده اداری آن ۲،۱۷۵،۶۰۱ (در سطح شهری در سال ۲۰۱۸) است. پاریس در شمال مرکزی فرانسه واقع شده و پرجمعیت ترین شهر این کشور است. مساحت این شهر ۱۰۵ کیلومتر مربع (۴۱ مایل مربع) و جمعیت کل آن در ژانویه ۲۰۱۹ حدود ۲،۲۷۳،۹۶۱ نفر است.

پاسخ کمی "پرمخاطب" بود، با "نرخ معادل" ۳/۵۱ توکن در ثانیه) ۳ تا ۴ برابر کندتر از (TinyLlama ، اما می توانیم اعلان بهتری را مشخص کنیم:

>>> پایتخت فرانسه کدام است؟ یک جواب کوتاه با کمتر از ۱۰ توکن به من بدهید.

پاریس.

در این مورد، نرخ ارزیابی ۴/۹۹ توکن در ثانیه بود، اما زمان کلی تقریباً یکسان بود: ۴۴ ثانیه.

انتخاب مناسب‌ترین اعلان یکی از مهم‌ترین مهارت‌هایی است که می‌توان با LLM‌ها بدون توجه به اندازه آن استفاده کرد.

ما می‌توانیم از چنین مدلی به عنوان یک دستیار مهم استفاده کنیم زیرا سرعت آن مناسب است.

Multimodal Models

مدل‌های چندوجهی، سیستم‌های هوش مصنوعی (AI) هستند که می‌توانند اطلاعات را از منابع متعددی مانند تصاویر، متن، صدا و ویدئو پردازش و درک کنند. در زمینه ما، LLM‌های چندوجهی می‌توانند ورودی‌های مختلفی از جمله متن، تصاویر و صدا را به عنوان درخواست پردازش کنند و آن درخواست‌ها را به خروجی‌های مختلف تبدیل کنند، نه فقط به نوع منبع.

ما در اینجا با LLaVA-Phi-3 mini کار خواهیم کرد، یک مدل LLaVA با تنظیم دقیق از Phi-3 Mini ۴. k دارای معیارهای عملکرد قوی است که با مدل اصلی LLaVA (Large Language and Vision Assistant) همتراز است.

mini³LLaVA-Phi- یک مدل چندوجهی بزرگ آموزش دیده سرتاسر است که برای درک و تولید محتوا بر اساس ورودی های بصری (تصاویر) و دستورالعمل های متنی طراحی شده است. این قابلیت های یک رمزگذار بصری و یک مدل زبان را برای پردازش و پاسخ به ورودی های چندوجهی ترکیب می کند.

بیایید مدل را نصب کنیم:

```
ollama run llava-phi3 --verbose
```

بیایید با یک ورودی متن شروع کنیم:

>>> شما یک دستیار هوش مصنوعی مفید هستید. پایتخت فرانسه کجاست؟

پایتخت فرانسه پاریس است.

پاسخ ۶/۴ ثانیه طول کشید، با نرخ ارزیابی ۴/۵ توکن در ثانیه! بد نیست!

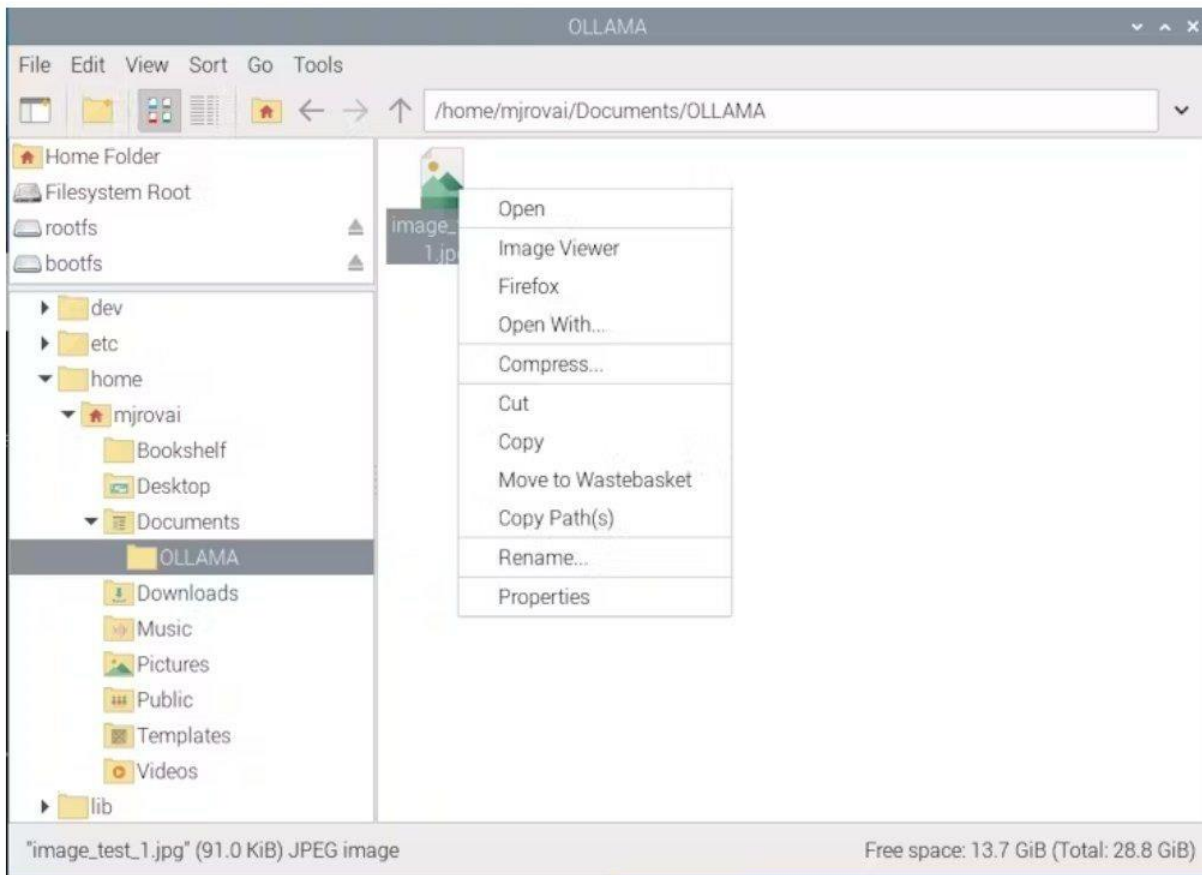
اما به ما اطلاع دهید که با یک تصویر به عنوان ورودی وارد شویم. برای آن، بیایید یک دایرکتوری برای کار ایجاد کنیم:

```
cd Documents/  
mkdir OLLAMA  
cd OLLAMA
```

برای مثال، بیایید یک تصویر را از اینترنت بارگیری کنیم
(ویکی‌پدیا: پاریس، فرانسه):

یک نسخه VGA با نام _image_test.jpg در فهرست راهنمای
OLLAMA دانلود شد. باید مسیر کامل تصویر را کپی کنیم: کپی
مسیر(های):





بیایید با این دستور وارد شویم:

>>> تصویر را توضیح دهید

./home/mjrovai/Documents/OLLAMA/image_test_1.jpg

نتیجه عالی بود، اما تأخیر کلی مهم بود. تقریباً ۴ دقیقه برای انجام استنباط.

```
File Edit Tabs Help
mjrovai@rpi-5:~ $ cd Documents/
mjrovai@rpi-5:~/Documents $ mkdir OLLAMA
mjrovai@rpi-5:~/Documents $ cd OLLAMA
mjrovai@rpi-5:~/Documents/OLLAMA $ ls
image_test_1.jpg
mjrovai@rpi-5:~/Documents/OLLAMA $ ollama run llava-phi3 --verbose
>>> Describe the image /home/mjrovai/Documents/OLLAMA/image_test_1.jpg
Added image '/home/mjrovai/Documents/OLLAMA/image_test_1.jpg'
The image captures a breathtaking view of Paris, France. Dominating the left
side of the frame is the iconic Eiffel Tower, its iron lattice work standing
tall against the skyline. On the right, the Seine River meanders through the
cityscape, with several boats dotting its surface.
Bridges span across the river, connecting different parts of the city.
Buildings of various sizes and shapes fill up the background, their rooftops
creating a patchwork quilt-like pattern against the skyline. The image is taken
from a high vantage point, offering a panoramic view that beautifully
encapsulates the essence of Paris.
The colors in the photo are predominantly blue and green, with hints of gray
and brown adding to the urban charm. A few clouds lazily drift across the sky,
completing this picturesque scene.
total duration:      3m43.298862387s
load duration:       2.523379ms
prompt eval count:    1 token(s)
prompt eval duration: 2m32.574345s
prompt eval rate:     0.01 tokens/s
eval count:           195 token(s)
eval duration:        1m10.577168s
eval rate:            2.76 tokens/s
>>> Send a message (/? for help)
```

بیایید سعی کنیم مدل را فریب دهیم. تصویری از برج ایفل در
لاس وگاس، نوادا، ایالات متحده را دانلود کنید و نام آن
را image_test_۲.jpg بگذارید:



بیایید با این دستور وارد شویم:

>>> تصویر را توضیح دهید

.jpg۲/home/mjrovai/Documents/OLLAMA/image_test_

توضیحات تصویر خوب است اما متاسفانه مدل را فریب دادیم.

```
mjrovai@rpi-5: ~/Documents/OLLAMA
File Edit Tabs Help
mjrovai@rpi-5:~ $ cd Documents/
mjrovai@rpi-5:~/Documents $ mkdir OLLAMA
mjrovai@rpi-5:~/Documents $ cd OLLAMA
mjrovai@rpi-5:~/Documents/OLLAMA $ ls
image_test_1.jpg
mjrovai@rpi-5:~/Documents/OLLAMA $ ollama run llava-phi3 --verbose
>>> DDescribe the image /home/mjrovai/Documents/OLLAMA/image_test_1.jpg
Added image '/home/mjrovai/Documents/OLLAMA/image_test_1.jpg'
The image captures a breathtaking view of Paris, France. Dominating the left
side of the frame is the iconic Eiffel Tower, its iron lattice work standing
tall against the skyline. On the right, the Seine River meanders through the
cityscape, with several boats dotting its surface.

Bridges span across the river, connecting different parts of the city.
Buildings of various sizes and shapes fill up the background, their rooftops
creating a patchwork quilt-like pattern against the skyline. The image is taken
from a high vantage point, offering a panoramic view that beautifully
encapsulates the essence of Paris.

The colors in the photo are predominantly blue and green, with hints of gray
and brown adding to the urban charm. A few clouds lazily drift across the sky,
completing this picturesque scene.

total duration:      3m43.298862387s
load duration:      2.523379ms
prompt eval count:   1 token(s)
prompt eval duration: 2m32.574345s
prompt eval rate:    0.01 tokens/s
eval count:          195 token(s)
eval duration:       1m10.577168s
eval rate:           2.76 tokens/s
>>> Send a message (/? for help)
```

این احتمال وجود دارد که آب به علاوه برج باعث شده مدل آن را پاریس توصیف کند. بیاید تصویر دیگری را برای همان مکان (لاس وگاس) اما با استفاده از زاویه دیگر و بدون آب امتحان کنیم:



عالی! مدل تصویر را به درستی درک کرد.

```

mjrovai@rpi-5: ~
File Edit Tabs Help

0[|||||||||||||||||||||||||100.0%] Tasks: 80, 166 thr, 127 kthr; 4 running
1[|||||||||||||||||||||||||100.0%] Load average: 3.78 2.13 1.90
2[|||||||||||||||||||||||||100.0%] Uptime: 02:57:01
3[|||||||||||||||||||||||||100.0%]
Mem[|||||||||||||||||||||2.69G/7.86G]
Swp[|||||||||||||||||0K/200M]

Main 373
PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
21665 ollama 20 0 5069M 4461M 2216M R 376.9 55.4 6:21.46 /tmp/ollama77710
21710 ollama 20 0 5069M 4461M 2216M R 99.5 55.4 1:15.00 /tmp/ollama77710
21711 ollama 20 0 5069M 4461M 2216M R 98.8 55.4 1:15.18 /tmp/ollama77710
21709 ollama 20 0 5069M 4461M 2216M R 84.3 55.4 1:14.91 /tmp/ollama77710
1029 mjrovai 20 0 910M 146M 105M S 15.2 1.8 12:19.25 /usr/bin/wayfire
2001 mjrovai 20 0 551M 56464 38032 S 3.3 0.7 1:31.49 lxterminal
895 vnc 20 0 533M 70176 49152 S 2.0 0.9 1:50.14 wayvnc --render-
1089 vnc 20 0 533M 70176 49152 S 0.7 0.9 0:10.16 wayvnc --render-
F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice F8Nice +F9Kill F10Quit

```


کتابخانه اولاما پایتون

تا کنون، ما قابلیت چت LLM ها را با استفاده از خط فرمان روی ترمینال بررسی کرده ایم. با این حال، ما می خواهیم آن مدل ها را در پروژه های خود ادغام کنیم، بنابراین به نظر می رسد پایتون مسیر درستی است. خبر خوب این است که اولاما چنین کتابخانه ای دارد.

کتابخانه Ollama Python تعامل با مدل های پیشرفته LLM را ساده می کند و پاسخ ها و قابلیت های پیچیده تری را امکان پذیر می کند، علاوه بر این، ساده ترین راه را برای ادغام پروژه های Python ۳/۸ + با Ollama فراهم می کند.

برای درک بهتر نحوه ایجاد برنامه ها با استفاده از Ollama با پایتون، ویدیوهای مت ویلیامز را اکیداً مانند ویدیوی زیر توصیه می کنم:



نصب و راه اندازی:

```
pip install ollama
```

برای ایجاد یک اسکریپت پایتون به یک ویرایشگر متن یا یک IDE نیاز داریم. Raspberry Desktop همیشه به طور پیش فرض نصب شده است (به وسیله [Menu][Programming])، Thonny و Geany قابل دسترسی است. اگر IDE دیگری مانند Visual Studio Code ترجیح می دهید، می توانید آن را از [Menu][Preferences][Recommended Software] دانلود کنید.

وقتی پنجره باز شد، به [برنامه نویسی] بروید، گزینه دلخواه خود را انتخاب کنید و [اعمال] را فشار دهید.

بیایید با یک اسکریپت بسیار ساده وارد شویم:

واردات اولاما

MODEL = 'llava-phi۳'

PROMPT = 'پایتخت فرانسه کدام است؟'

res = ollama.generate(model=MODEL, prompt=PROMPT)

چاپ(res)

و آن را ذخیره کنید، برای مثال، test_ollama.py. ما می توانیم از IDE برای اجرای آن استفاده کنیم یا آن را مستقیماً در ترمینال انجام دهیم:

پایتون test_ollama.py

در نتیجه، ما پاسخ مدل را در قالب JSON خواهیم داشت:

```
{
  'model': 'llava-phi3',
  'created_at': '2024-06-13T15:50:13.49687474
4Z',
  'response': ' The capital of France is Pari
s. It is also the largest city in the country a
nd serves as a major center for culture, politi
cs, and economics. The French government, inclu
ding both the executive branch led by the Presi
dent and the legislative branch with the Nation
al Assembly and Senate, are located in Paris. T
he official residence of the president is at th
e Élysée Palace, while the French Parliament
meets in the Palais Bourbon.',
  'done': True,
  'done_reason': 'stop',
  'context': [32010, 1724, 338, 278, 7483, 31
0, 3444, 29973, 32007, 32001, 450, 7483, 310, 3
444, 338, 3681, 29889, 739, 338, 884, 278, 1015
0, 4272, 297, 278, 4234, 322, 19700, 408, 263,
4655, 4818, 363, 9257, 29892, 22661, 29892, 32
```

همانطور که می بینیم، چندین قطعه اطلاعات تولید می شود، مانند:

"context": شناسه‌های نشانه‌ای که ورودی و زمینه مورد استفاده مدل را نشان می‌دهند. نشانه‌ها نمایش‌های عددی متنی هستند که برای پردازش توسط مدل زبان استفاده می‌شوند. 'response': متن خروجی اصلی که توسط مدل در پاسخ به درخواست ما تولید می‌شود.

معیارهای عملکرد:

total_duration: کل زمان صرف شده برای عملیات بر حسب نانوثانیه. در این حالت، تقریباً ۲۳/۶۳ ثانیه. **load_duration**: زمان صرف شده برای بارگذاری مدل یا اجزا در نانوثانیه. حدود ۱/۱۲ **milliseconds.prompt_eval_duration**: زمان صرف شده برای ارزیابی درخواست در نانوثانیه. حدود ۰/۲۴ ثانیه. **eval_count**: تعداد توکن های ارزیابی شده در طول تولید. در اینجا، ۸۹ **tokens.eval_duration**: زمان صرف شده برای مدل برای تولید پاسخ در نانوثانیه. تقریباً ۲۳/۲۵ ثانیه.

اما آنچه ما می خواهیم "پاسخ" ساده و شاید برای تجزیه و تحلیل، کل مدت زمان استنتاج است، بنابراین بیاید کد را تغییر دهیم تا آن را از JSON استخراج کنیم:

```
import ollama

MODEL = 'llava-phi3'
PROMPT = 'What is the capital of France?'

res = ollama.generate(model=MODEL, prompt=PROMPT)
print(f"\n{res['response']}")
print(f"\n [INFO] Total Duration: {(res['total_duration']/1e9):.2f} seconds")
```

حالا ما گرفتیم:

پایتخت فرانسه پاریس است. این شهر همچنین بزرگترین شهر کشور است و به عنوان یک مرکز مهم فرهنگی، اقتصادی و سیاسی عمل می کند. پاریس به خاطر تاریخ غنی، معماری زیبا و بناهای مشهور جهانی مانند برج ایفل، کلیسای نوتردام و موزه لوور شهرت دارد. جمعیت این شهر حدود ۲/۱۴ میلیون نفر است و ۸/۶ میلیون نفر دیگر در کلانشهر اطراف آن هستند. پاریس سالانه میلیون ها گردشگر را به خود جذب می کند تا موزه های معروف، گالری های هنری، رستوران ها و صحنه های زندگی شبانه پر جنب و جوش را کشف کنند.

[INFO] مدت زمان کل: ۳۵/۳۴ ثانیه

استفاده از Olama.chat()

راه دیگر برای دریافت پاسخ ما استفاده از olama.chat():

```
import ollama
```

```
MODEL = 'llava-phi3'
```

```
PROMPT = 'What is the capital of France?'
```

```
res = ollama.generate(model=MODEL, prompt=PROMPT)
```

```
print(f"\n{res['response']}")
```

```
print(f"\n [INFO] Total Duration: {(res['total_  
duration']/1e9):.2f} seconds")
```

```
mjrovai@rpi-5:~/Documents/OLLAMA $ python describe_image.py
```

The image captures a breathtaking view of Paris, France. From a high vantage point, the city unfolds below with its iconic landmarks and bustling streets. Dominating the skyline is the Eiffel Tower, an iron lattice structure painted in a dark hue that stands tall against the blue sky speckled with clouds.

The Seine River meanders through the heart of the city, reflecting the sky above. A bridge arches over its tranquil waters, connecting two parts of the city. Numerous boats dot the river's surface, adding a dynamic element to this serene waterway.

Paris is densely packed with buildings that showcase a mix of architectural styles. Their varying heights and colors create a visually interesting skyline that speaks volumes about the city's rich history and vibrant present. The street below is alive with activity, lined with trees that add a touch of green to the urban landscape.

In summary, this image presents a comprehensive view of Paris, encapsulating its iconic landmarks like the Eiffel Tower, its winding rivers filled with boats, and the bustling streets teeming with life against a backdrop of diverse architecture under a sky dotted with clouds.

```
[INFO] Total Duration: 236.58 seconds
```

```
mjrovai@rpi-5:~/Documents/OLLAMA $
```

در کد بالا، ما دو پرس و جو را اجرا می کنیم و اعلان دوم نتیجه اولی را در نظر می گیرد.

در اینجا نحوه پاسخ مدل آمده است:

دریافت توضیحات تصویر:

همانطور که از مدل LLaVa-Phi-۳ با خط فرمان برای تجزیه و تحلیل یک تصویر استفاده کرده ایم، همین کار را می توان در اینجا با

تا اینجا، می توانیم ببینیم که با پاسخ مدل ("پاسخ") به یک متغیر، می توانیم به راحتی با آن کار کنیم و آن را در پروژه های دنیای واقعی ادغام کنیم. یا نه؟؟

این در تئوری کار می کند، اما یک مشکل بزرگ این است که مدل همیشه به همان درخواست متفاوت پاسخ می دهد. بیا ببینیم آنچه ما دوست داریم، به عنوان پاسخ مدل در آخرین نمونه ها، فقط نام پایتخت یک کشور است، نه چیزی بیشتر. می توانیم از فراخوانی تابع Ollama استفاده کنیم که کاملاً با OpenAI API سازگار است.

اما دقیقاً «تعداد فراخوانی تابع» چیست؟

در هوش مصنوعی مدرن، فراخوانی تابع با مدل‌های زبان بزرگ (LLM) به این مدل‌ها اجازه می‌دهد تا اقداماتی فراتر از تولید متن انجام دهند. با ادغام با توابع خارجی یا API ها، LLM ها می‌توانند به داده‌های بلادرنگ دسترسی داشته باشند، وظایف را خودکار کنند و با سیستم‌های مختلف تعامل داشته باشند.

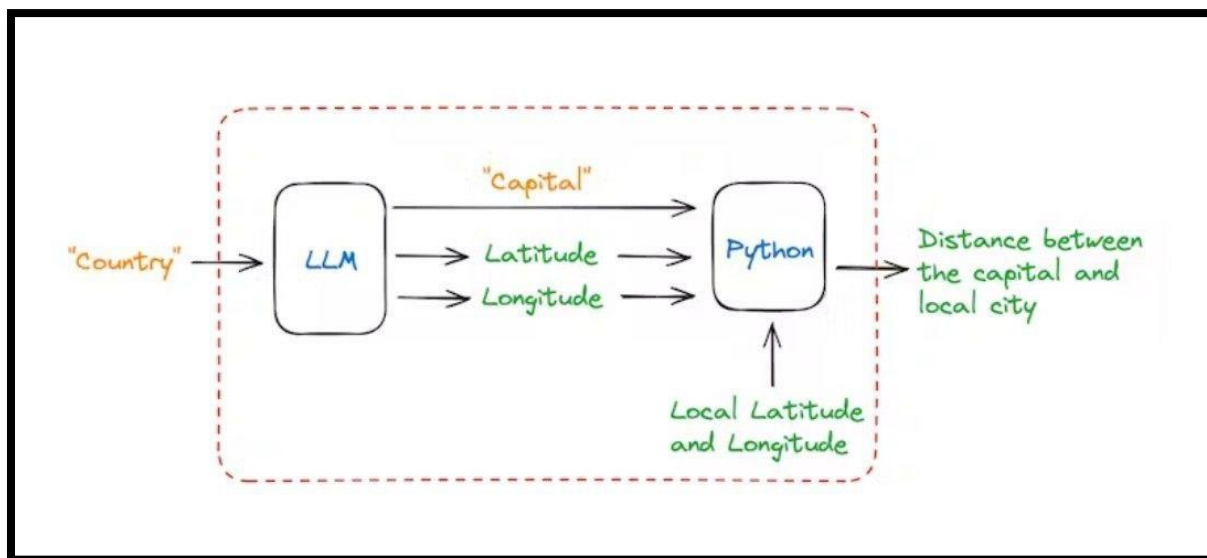
به عنوان مثال، یک LLM به جای پاسخ دادن صرف به یک پرسش در مورد آب و هوا، می‌تواند یک API آب و هوا را برای واکنشی شرایط فعلی و ارائه اطلاعات دقیق و به روز فراخوانی کند. این قابلیت ارتباط و دقت پاسخ‌های مدل را افزایش می‌دهد و آن را به ابزاری قدرتمند برای هدایت جریان‌های کاری و خودکارسازی فرآیندها تبدیل می‌کند و آن را به یک شرکت‌کننده فعال در برنامه‌های کاربردی دنیای واقعی تبدیل می‌کند.

برای جزئیات بیشتر، لطفاً این ویدیوی عالی ساخته شده توسط زندان ماروین را ببینید:



بیایید یک پروژه ایجاد کنیم.

ما می‌خواهیم برنامه‌ای ایجاد کنیم که کاربر نام یک کشور را وارد کند و به عنوان خروجی، فاصله کیلومتر از پایتخت چنین کشوری و مکان برنامه را دریافت کند (برای سادگی، از موقعیت مکانی خود استفاده خواهیم کرد، سانتیاگو، شیلی، به عنوان نقطه محلی).



هنگامی که کاربر نام کشوری را وارد می کند، مدل نام پایتخت آن (به عنوان رشته) و طول و عرض جغرافیایی آن شهر (به صورت شناور) را برمی گرداند. با استفاده از آن مختصات، می توانیم از یک کتابخانه ساده پایتون (haversine) برای محاسبه فاصله بین آن ۲ نقطه استفاده کنیم.

ایده این پروژه نشان دادن ترکیبی از تعامل مدل زبان (IA)، مدیریت داده های ساخت یافته با Pydantic، و محاسبات جغرافیایی با استفاده از فرمول هاورسین (محاسبات سنتی) است.

ابتدا اجازه دهید چند کتابخانه را نصب کنیم. علاوه بر Haversine، اصلی ترین کتابخانه OpenAI Python است که دسترسی راحت به OpenAI REST API را از هر برنامه Python ۳/۷+ فراهم می کند. یکی دیگر، Pydantic (و instructor)، یک کتابخانه مدیریت تنظیمات و اعتبارسنجی داده قدرتمند است که توسط Python مهندسی شده است تا استحکام و قابلیت اطمینان پایگاه کد شما را افزایش دهد. به طور خلاصه، Pydantic کمک خواهد کرد تا اطمینان حاصل شود که پاسخ مدل ما همیشه سازگار خواهد بود.

1. Importing Libraries

```
import sys
from haversine import haversine
from openai import OpenAI
from pydantic import BaseModel, Field
import instructor
```

sys: دسترسی به پارامترها و عملکردهای خاص سیستم را فراهم می کند. برای دریافت آرگومان های خط فرمان استفاده می شود. haversine: تابعی از کتابخانه Haversine که فاصله بین دو نقطه جغرافیایی را با استفاده از فرمول Haversine محاسبه می کند. OpenAI: ماژولی برای تعامل با OpenAI API (اگرچه همراه با تنظیمات محلی استفاده می شود. ، اولاما). اینجا همه چیز آفلاین است. pydantic: اعتبارسنجی داده ها و مدیریت تنظیمات را با استفاده از حاشیه نویسی از نوع Python ارائه می دهد. برای تعریف ساختار پاسخ مورد انتظار data.instructor استفاده می شود: یک ماژول برای وصله کردن کلاینت OpenAI برای کار در یک حالت خاص (احتمالاً مربوط به مدیریت داده های ساخت یافته) استفاده می شود.

پایان