

如何通过ollama部署本地大模型-deepseek-r1

1. Ollama

ollama 官网：

- <https://ollama.com>



Get up and running with large
language models.

Run Llama 3.3, DeepSeek-R1, Phi-4, Mistral,
Gemma 2, and other models, locally.

Download ↓

Available for macOS,
Linux, and Windows

Ollama 是一个开源的大型语言模型运行框架，旨在帮助用户快速在本地部署和运行大模型。它通过简化的命令行界面和 API 接口，使得用户无需复杂的配置即可轻松使用大模型。

核心特点

- **易用性**: Ollama 提供了直观的命令行工具，用户只需一条命令即可启动大模型，例如 `ollama run llama3`。
- **多平台支持**: 支持 macOS、Windows 和 Linux 系统，用户可以通过 Docker 容器或直接安装的方式运行。
- **多模型支持**: Ollama 支持多种流行的开源大模型，如 Llama、Falcon、Code Llama 等，并允许用户通过 Modelfile 导入自定义模型。
- **本地化运行**: 不依赖云端服务，用户可以在本地设备上运行模型，保护数据隐私。
- **GPU/CPU 混合模式**: 支持根据硬件配置选择运行模式，优化性能。

架构与工作流程

Ollama 采用经典的 Client-Server 架构：

- **Client**: 通过命令行与用户交互。
- **Server**: 负责加载和运行模型，核心组件包括 `ollama-http-server`（处理客户端请求）和 `llama.cpp`（作为推理引擎）。
- **存储结构**: 本地存储路径为 `$HOME/.ollama`，包含日志文件、密钥文件和模型文件（如 `blobs` 和 `manifests`）。

使用场景

- 开发者快速测试和部署大模型。
- 企业本地化部署大模型，保护数据隐私。
- 教育与研究领域，用于模型实验和教学。

llama.cpp

llama.cpp 是一个开源的 C++ 库，专注于加载和运行 LLaMA 系列大模型。它通过量化和优化技术，使得大模型能够在资源有限的设备上运行。

核心特点

- **轻量化**：llama.cpp 通过量化技术（如 GGUF 格式）大幅降低模型对硬件资源的需求，使其能够在低配置设备（如树莓派）上运行。
- **跨平台支持**：支持多种硬件平台，包括 ARM 架构设备（如树莓派 Zero）和 x86 架构设备。
- **高性能优化**：通过 C++ 实现的高效推理代码，能够在低资源设备上实现可接受的推理速度。
- **灵活性**：支持从 Hugging Face 等平台下载模型，并转换为 GGUF 格式后运行。

使用场景

- 在资源受限的设备（如树莓派、手机）上运行大模型。
- 开发者进行模型推理优化和实验。
- 教育与研究领域，用于低资源环境下的模型部署。

Ollama 与 llama.cpp 的关系

- **依赖关系**：Ollama 依赖于 llama.cpp 作为其推理引擎，llama.cpp 提供了底层的模型加载和推理功能，而 Ollama 在此基础上提供了更高级的管理和交互功能。
- **功能互补**：llama.cpp 更适合技术开发者进行底层优化和实验，而 Ollama 更适合普通用户快速部署和使用大模型。


2. linux 安装

通过下载 `install.sh` 来下载

```
curl -fsSL https://ollama.com/install.sh | sh
```

脚本中会从 `https://ollama.com/download` 下载安装文件

```
curl --fail --show-error --location --progress-bar -o $TEMP_DIR/ollama "https://ollama.com/download/ollama-1
```



如果下载有问题，做了本地版本的详见下面链接，解压下载

```
# 通过网盘分享的文件：ollama_local.zip  
# 链接：https://pan.baidu.com/s/1l4SqFEdNiFrg48k0AnqLrQ 提取码：hhg8
```

```
sh install.sh
```

```
# 验证
```

```
ollama --version
```

```
# ollama version is 0.3.1
```

```
[root@llm_app]# ll
total 112024
-rw-r--r-- 1 root root 12002 Aug 3 2024 install.sh
-rw-r--r-- 1 root root 585680472 Aug 2 2024 ollama-linux-amd64
-rw-r--r-- 1 root root 566215679 Mar 8 14:17 ollama_local.zip
[root@llm_app]# ollama --version
ollama version is 0.3.1
```

参考文档:

- <https://github.com/ollama/ollama/blob/main/docs/linux.md>
- <https://github.com/ollama/ollama/blob/main/docs/faq.md>

ollama 命令和启动

```
# 启动服务
# 后台执行 nohup ollama serve > serve.log 2>&1 &
ollama serve

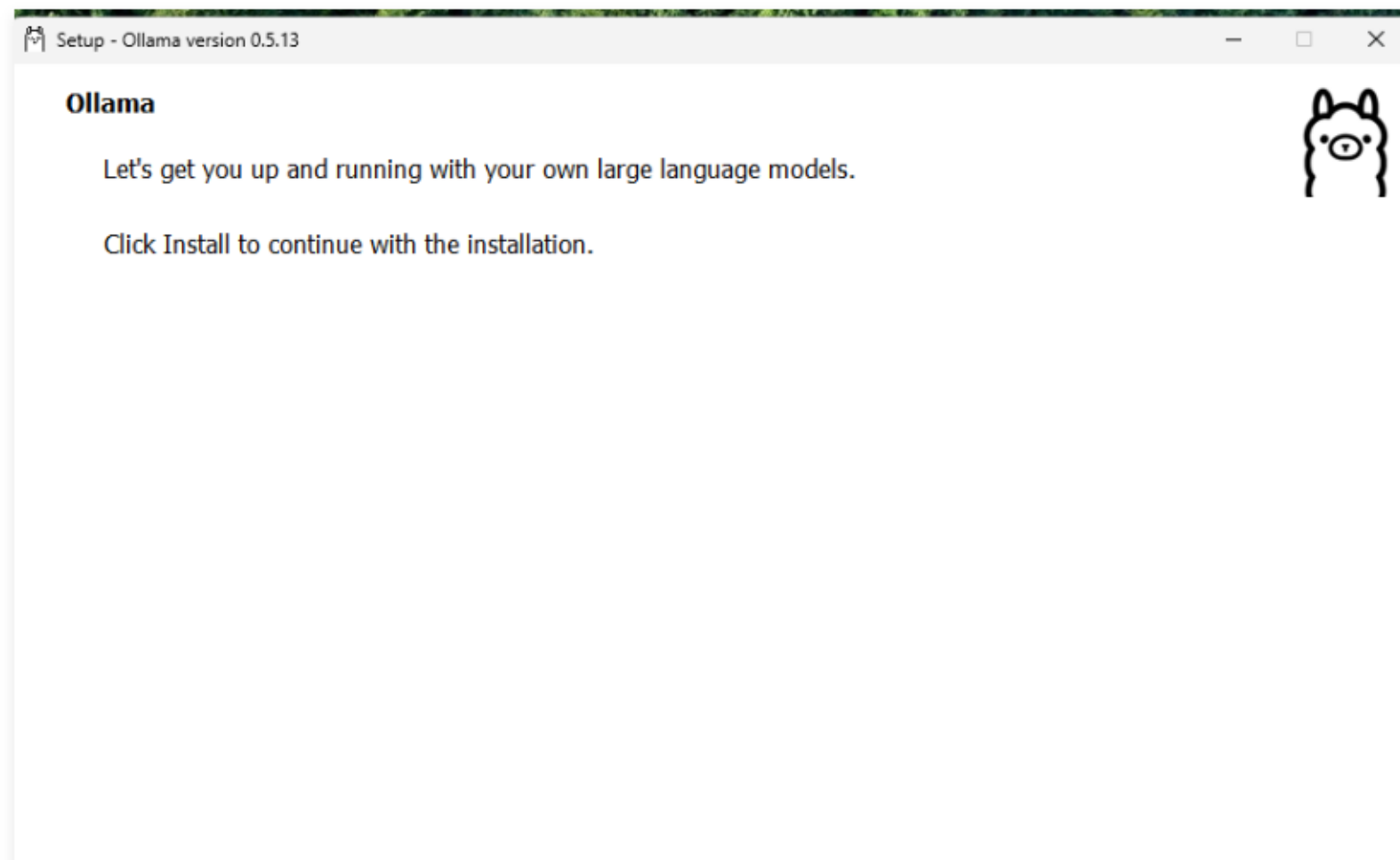
# 下载和启动模型
ollama run deepseek-r1:1.5b

# 模型默认下载路径为: /usr/share/ollama/.ollama/models
```

```
pulling manifest
pulling manifest
pulling manifest
pulling manifest
pulling manifest
```

```
pulling manifest
pulling manifest
pulling manifest
pulling manifest
pulling manifest
pulling manifest
pulling manifest
pulling manifest
pulling manifest
pulling manifest
pulling aabd4dbf0c8... 39%
| 434 MB/1.1 GB 21 MB/s 31s
```

3. windows 安装



Install

Cancel

Setup - Ollama version 0.5.13

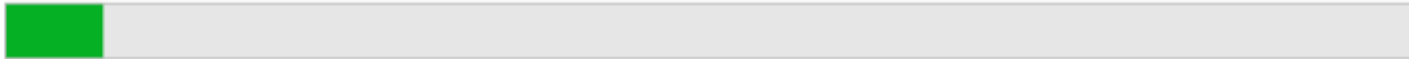
Installing

Please wait while Setup installs Ollama on your computer.



Extracting files...

C:\Users\████████\AppData\Local\Programs\Ollama\lib\ollama\cuda_v11\cublasLt64_11.dll



Cancel

Windows PowerShell

PS C:\Users\>

PS C:\Users\> ollama -v

ollama version is 0.5.13

PS C:\Users\> ollama -h

Large language model runner

Usage:

ollama [flags]

ollama [command]

Available Commands:

serve	Start ollama
create	Create a model from a Modelfile
show	Show information for a model
run	Run a model
stop	Stop a running model
pull	Pull a model from a registry
push	Push a model to a registry
list	List models
ps	List running models
cp	Copy a model
rm	Remove a model
help	Help about any command

Flags:

-h, --help	help for ollama
-v, --version	Show version information

Use "ollama [command] --help" for more information about a command.

PS C:\Users\> |

C:\WINDOWS\System32\Win

Welcome to Ollama!

Run your first model:

ollama run llama3.2


```
PS C:\Windows\System32> ollama run deepseek-r1:1.5b
pulling manifest
pulling aabd4debfc8... 0% | 894 KB/1.1 GB 447 KB/s 41m35s
```

```
>>>
>>>
>>> 我要学习python, 介绍下python的学习路径和学习资料
<think>
嗯, 我现在想系统地学习Python语言。可是我对它还没有什么了解, 所以得先从头开始。首先, 我需要明确Python的基本概念, 了解它的语法和表达式。

我记得Python是面向对象的编程语言, 这可能意味着代码结构会比较复杂。我还听说过模块化设计, 这应该是指在代码中分解较大的功能为多个独立的小单元, 这样可以提高可维护性和扩展性。这对于我的学习来说很重要, 可以帮助我更好地组织自己的知识。

接下来, 基础语法部分。我知道Python的变量和表达式, 但我不太清楚它们之间的关系。可能需要花一些时间去理解数据类型, 比如整数、字符串、布尔值等的区别和用法。然后是运算符, 我记得有加减乘除这些基本的算术操作, 还有条件判断和循环结构。

控制流部分听起来有点复杂, 但我记得Python中的循环可以用for循环和while循环来实现。这应该对我的编程能力有所帮助。此外, 函数的定义和调用也很重要, 我需要学习如何编写函数, 并进行参数化处理。

模块化设计方面, 我还不太清楚它具体指的是什么。可能是指将代码分成几个模块, 每个模块集中解决一个特定的问题。这可能会帮助我更系统地管理自己的编程项目。

然后是数据结构, 我记得列表、元组和字典都是基本的数据类型。列表可以存储多个元素, 元组是有序的列表, 而字典是键值对的映射关系。这些数据结构在处理不同类型的输入和输出时非常有用。

我还需要学习条件语句, 这应该是指根据某种条件执行不同的代码块。比如if-elif-else结构。函数调用部分可能涉及到函数的参数、返回值以及返回后的结果处理。
```

表达式和算术运算符是基础吗？我记得Python中的运算符包括加减乘除、幂次方、取余、双变量操作数等等，这些在进行数值计算时非常重要。

输入输出方面，我需要了解如何从标准输入读取数据，比如使用input()函数。同时也要知道输出结果的方法，例如print()函数和sys.stdin

异常处理部分，除了基本的try-except块，我还想了解如何利用try-except来重新定义函数或模块，这可能有助于更复杂的异常处理逻辑。

总结一下，我需要逐步学习Python的基础概念和高级功能，从语法到结构，从迭代到模块化设计，通过实践练习不同的编程任务。同时，使用文档和技术论坛可以获取更多的信息，解决遇到的问题，并找到学习上的建议。

</think>

要系统地学习Python语言，可以从以下几个方面进行学习：

1. 基础语法

- **变量和表达式**：了解变量的命名规则和基本操作符（+，-，*，/，**，%）。
- **数据类型**：掌握整数、字符串、布尔值和浮点数的区别和用法。

2. 模块化设计

- 学习模块的重要性，包括代码的可维护性。
- 熟悉模块的基本结构，如导入模块（import）、定义模块（def module_name）以及模块之间的调用。

3. 控制流

- **条件语句**：学习if、elif和else语句，用于根据条件执行不同的代码块。
- **循环结构**：掌握for循环和while循环，并了解break和continue语句的使用。

4. 数据结构

- **列表**：了解列表的访问（index）、拼接（append、extend）以及排序（sort）等操作。
- **字典**：学习字典的创建、初始化和查询方式，以及如何处理缺失或无效键的情况。
- **集合**：掌握集合的基本操作，如交集、并集和差集。

5. 输入输出

- 熟练使用input()函数读取标准输入，并将结果打印出来。
- 使用sys.stdin模块快速读取标准输入，提高代码效率。

- 熟悉模块的基本结构，如导入模块（import）、定义模块（def module_name）以及模块之间的调用。

3. 控制流

- **条件语句**：学习if、elif和else语句，用于根据条件执行不同的代码块。
- **循环结构**：掌握for循环和while循环，并了解break和continue语句的使用。

4. 数据结构

- ****列表****: 了解列表的访问 (index)、拼接 (append、extend) 以及排序 (sort) 等操作。
- ****字典****: 学习字典的创建、初始化和查询方式, 以及如何处理缺失或无效键的情况。
- ****集合****: 掌握集合的基本操作, 如交集、并集和差集。

5. 输入输出

- 熟练使用input()函数读取标准输入, 并将结果打印出来。
- 使用sys.stdin模块快速读取标准输入, 提高代码效率。

6. 异常处理

- 学习ValueError, TypeError等异常类型及其捕获方法。
- 在try-except块中定义函数或模块以捕捉并处理异常信息。

7. 迭代与遍历

- ****循环语句****: 学习for和while循环, 并理解复合循环的用法, 如for key in my_dict.items()。
- ****集合操作****: 掌握集合的运算符 (&、|、^) 及其应用。

8. 函数定义与调用

- 熟悉函数的基本结构 (def func_name参数: return 结果), 学习复合函数和嵌套函数。
- 掌握try-except块中return statement的使用, 避免程序崩溃。

通过逐步练习和实践, 将上述知识点运用到实际编程任务中, 将帮助你构建一个扎实的Python基础, 并提升你的编程能力。

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

4. deepseek-r1 蒸馏版本使用

```
>>> 除湿器的原理是什么?
```

```
<think>
```

嗯, 我现在想了解一下除湿器的工作原理。我记得以前听说过除湿器是用于干燥物体的, 但具体是怎么工作的呢? 首先, 它可能涉及到一些基本的物理原理吧。我想, 除湿器可能是通过某种方式让物品干燥, 这可能和温度、气压或者压力有关。

然后, 是不是有一些化学反应发生? 比如, 水蒸气遇到固体物质可能会分解产生二氧化碳, 从而带走水分呢? 或者是用热的方式加热来加快干燥过程? 我记得以前在实验室里看到过类似的东西, 可能就是这样的道理。不过, 具体的原理我不太确定, 得仔细想想。

还有, 除湿器是不是有一些 kind of fans或者涡轮机, 用来推动气体移动? 这样可以让湿气从物体表面吹走, 而不是直接让物体内部干燥。这点挺重要的, 因为如果物体内部的水分被吸收了, 可能不需要除湿器就能保持干燥。所以, 除湿器的作用应该是带走外界湿度, 并且让物体内的水分减少。

接下来, 可能需要用到哪些材料呢? 比如, 除湿器的材料可能是有机纤维或者塑料? 有机纤维可能更耐用一些, 不易发霉或者分解。而塑料可能会随着时间变质, 所以有机纤维可能更适合长期使用。此外, 除湿器内部有没有冷却系统, 确保湿度不会持续过高或过低?

还有, 是不是需要一些调节参数, 比如湿度、温度和压力等? 这些参数应该会影响除湿器的工作效率和效果。湿度太高的话, 除湿器可能工作不足; 湿度太低又干燥不够, 可能需要增加风速或者时间。温度控制也很重要, 保持在适宜的范围内, 避免 overheating或过冷。

另外, 我还听说有些除湿器会使用一些特殊的材料来吸收水分, 比如有机纤维材料和塑料纤维。这种材料可能会在吸水的同时减少干燥速度, 所以可能是一种结合了吸水率和耐久性的材料。而其他类型的材料可能更注重效率, 不需要像有机纤维那样吸水。

</think>

除湿器的原理涉及多个物理、化学和机械原理，主要通过以下方式带走湿度：

1. ****基本原理****:
 - 除湿器利用外部空气中的水分蒸发或干燥，去除物体表面或内部的水分。其作用是带走湿气并减少湿度。
2. ****工作机制****:
 - 使用加热系统（如热风源、蒸汽源）或涡轮机将水蒸气提升到一定高度，迫使湿气吹离物体表面。
 - 有机纤维材料可能在吸水率和耐久性上更优，适合长期使用。
3. ****材料与结构****:
 - 结构通常由机械部件（如风扇、涡轮）和材料组成，如塑料或有机纤维，以确保干燥效率和可靠性。
4. ****调节参数****:
 - 控制湿度、温度和压力，以优化干燥效果。湿度过高可能需要调整风速，而过低则可能导致过多干燥。
5. ****设计与用途****:
 - 可根据物体大小或使用场景设计，如家庭除湿器较大，建筑或工厂使用较小的设备。
 - 考虑智能控制、颜色显示和生物降解材料（稀有金属）以提高效率并减少环境影响。

deepseek-r1是长思考的推理模型

- 思考部分用 `<think></think>` 剩下的部分为大语言模型的回答

如果只有CPU的机器，内存的最低要求：

5. QWQ 推理模型

大规模强化学习（RL）有潜力超越传统的预训练和后训练方法来提升模型性能。近期的研究表明，强化学习可以显著提高模型的推理能力。例如，DeepSeek R1 通过整合冷启动数据和多阶段训练，实现了最先进的性能，使其

能够进行深度思考和复杂推理。这一次，我们探讨了大规模强化学习（RL）对大语言模型的智能的提升作用，同时很高兴推出我们最新的推理模型 QwQ-32B。这是一款拥有 320 亿参数的模型，其性能可与具备 6710 亿参数（其中 370 亿被激活）的 DeepSeek-R1 媲美。这一成果突显了将强化学习应用于经过大规模预训练的强大基础模型的有效性。此外，我们还在推理模型中集成了与 Agent 相关的能力，使其能够在使用工具的同时进行批判性思考，并根据环境反馈调整推理过程。我们希望我们的一点努力能够证明强大的基础模型叠加大规模强化学习也许是一条通往通用人工智能的可行之路。

```
ollama run qwq
```

- <https://qwenlm.github.io/zh/blog/qwq-32b/>
- 如果是CPU推理的机器，模型需要占内存 20GB

```
removing any unused layers
success
>>> 介绍一下你自己
<think>

</think>

你好！我是Qwen，一个由阿里云开发的大语言模型，我的设计目标是能够帮助用户完成各种任务，比如回答问题、创作文字（如写故事、写公文、写邮件、写剧本等）、进行逻辑推理、编程，甚至表达观点和玩游戏。

我具备以下特点：
1. **广泛的知识**：我可以根据训练数据中的信息提供广泛的主题的答案。
2. **多语言支持**：除了中文，我还支持多种其他语言，满足国际化的使用需求。
3. **对话理解能力**：经过多轮迭代和优化，我的对话理解能力得到了显著提高，能够准确理解与用户交互的对话历史，为用户提供更自然、流畅的对话体验。
4. **代码写作能力**：我经过充分的训练，具有多种编程语言的理解和生成能力。

如果你有任何问题或需要帮助，请随时告诉我！
```

QWQ也是长思考的推理模型

- 思考部分用 `<think></think>` 剩下的部分为大语言模型的回答

6. ollama模型本地保存默认路径

```
Linux: /usr/share/ollama/.ollama/models  
Windows: C:\Users\%username%\ollama\models  
macOS: ~/.ollama/models
```

下一节： 实战：使用大语言模型（本地和API、GPU和CPU） -1(17:39)

下一节