Lab 10: A simple Agent with Tools

Open in Colab

In this lab we will use Ollama to create a simple agent armed with tools in order to help carry out tasks on our behalf. This notebook is based on the short blog posts/tutorials found here and here.

Lab 10 Assignment/Task

There are a few questions below that require some additional code to be written so that your agent can carry out other operations besides just addition.

Let's start out by setting up Ollama to run in Colab. If you run this notebook locally and already have Ollama running, then you can skip these steps.

```
!pip install colab-xterm
%load_ext colabxterm
```

```
Requirement already satisfied: colab-xterm in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: ptyprocess~=0.7.0 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: tornado>5.1 in /usr/local/lib/python3.11/dist-packages
```

Next we'll start a terminal from within Colab. Once the terminal is running, you will need to run the two commands, but at separate times.

After you have run `%xterm` and the terminal opens, copy and paste the following line into the terminal and press enter:

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

After that has run, then start the Ollama server and have it run in the background by copying and pasting the following line into the terminal and pressing enter:

- `ollama serve &`

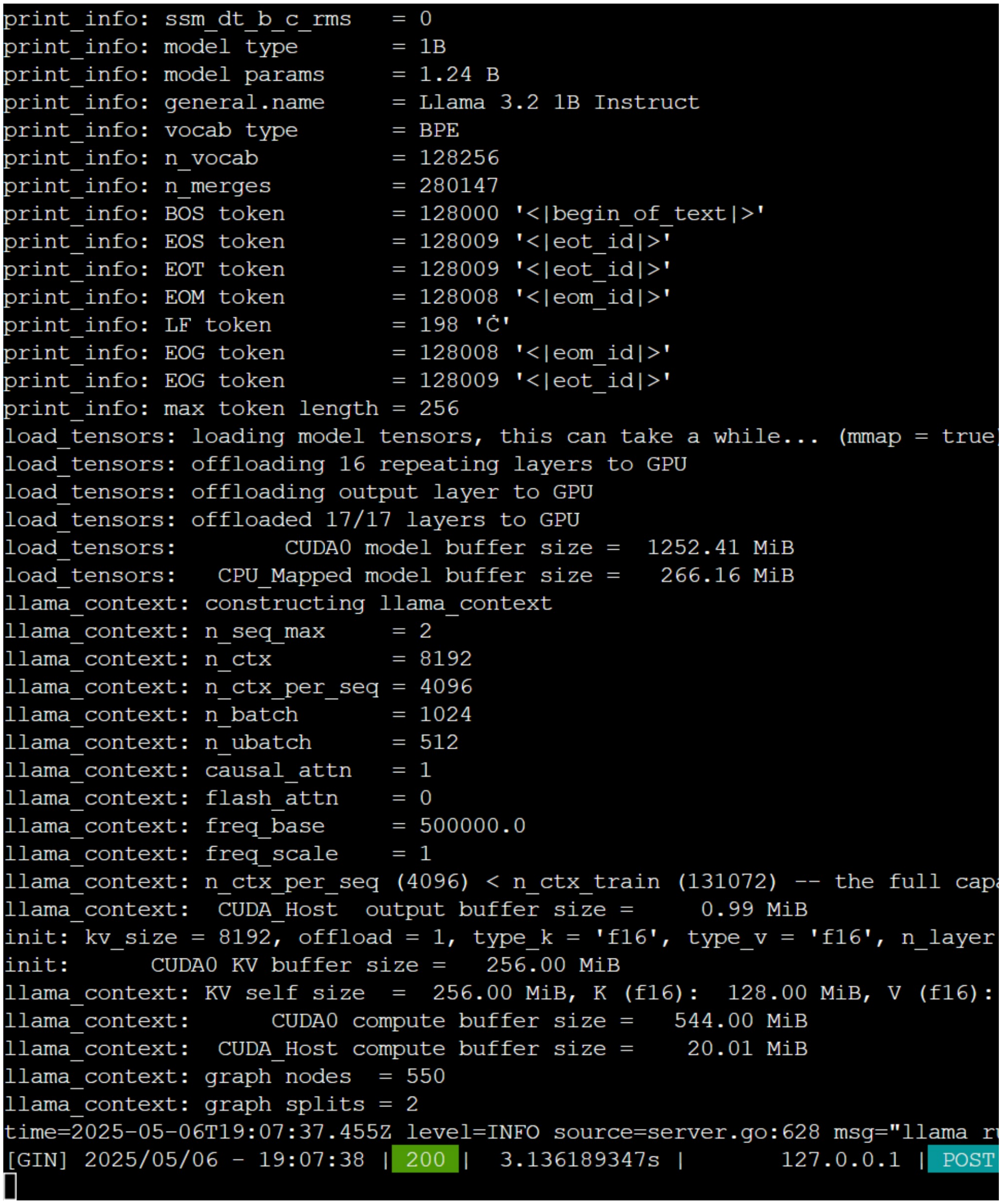
(*Note: you may need to press enter one more time after `ollama serve` to see the terminal prompt again*)

Next we will pull the small (1B parameter) Llama3.2 model down by running the following in the terminal:

- `ollama pull llama3.2:1b`

```
%xterm
```

Launching Xterm...



```
print_info: ssm_dt_b_c_rms   = 0
print_info: model type       = 1B
print_info: model params     = 1.24 B
print_info: general.name     = Llama 3.2 1B Instruct
print_info: vocab type       = BPE
print_info: n_vocab          = 128256
print_info: n_merges         = 280147
print_info: BOS token        = 128000 '<|begin_of_text|>'
print_info: EOS token        = 128009 '<|eot_id|>'
print_info: EOT token        = 128009 '<|eot_id|>'
print_info: EOM token        = 128008 '<|eom_id|>'
print_info: LF token         = 198 'Ċ'
print_info: EOG token        = 128008 '<|eom_id|>'
print_info: EOG token        = 128009 '<|eot_id|>'
print_info: max token length = 256
load_tensors: loading model tensors, this can take a while... (mmap = true
load_tensors: offloading 16 repeating layers to GPU
load_tensors: offloading output layer to GPU
load_tensors: offloaded 17/17 layers to GPU
load_tensors:        CUDA0 model buffer size =  1252.41 MiB
load_tensors:   CPU_Mapped model buffer size =   266.16 MiB
llama_context: constructing llama_context
llama_context: n_seq_max     = 2
llama_context: n_ctx         = 8192
llama_context: n_ctx_per_seq = 4096
llama_context: n_batch       = 1024
llama_context: n_ubatch      = 512
llama_context: causal_attn   = 1
llama_context: flash_attn    = 0
llama_context: freq_base     = 500000.0
llama_context: freq_scale    = 1
llama_context: n_ctx_per_seq (4096) < n_ctx_train (131072) -- the full cap
llama_context:  CUDA_Host  output buffer size =     0.99 MiB
init: kv_size = 8192, offload = 1, type_k = 'f16', type_v = 'f16', n_layer
init:      CUDA0 KV buffer size =   256.00 MiB
llama_context: KV self size  =  256.00 MiB, K (f16):  128.00 MiB, V (f16):
llama_context:      CUDA0 compute buffer size =   544.00 MiB
llama_context:  CUDA_Host compute buffer size =    20.01 MiB
llama_context: graph nodes  = 550
llama_context: graph splits = 2
time=2025-05-06T19:07:37.455Z level=INFO source=server.go:628 msg="llama r
[GIN] 2025/05/06 - 19:07:38 | 200 |  3.136189347s |       127.0.0.1 | POST
```

Now that Ollama is running, we can get started. The only module/library needed for this is the ollama python module.

```
!pip install ollama
```

```
Requirement already satisfied: ollama in /usr/local/lib/python3.11/dist-packages (0.4
Requirement already satisfied: httpx<0.29,>=0.27 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: pydantic<3.0.0,>=2.9.0 in /usr/local/lib/python3.11/di
Requirement already satisfied: anyio in /usr/local/lib/python3.11/dist-packages (from
Requirement already satisfied: certifi in /usr/local/lib/python3.11/dist-packages (fr
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: idna in /usr/local/lib/python3.11/dist-packages (from
Requirement already satisfied: h11>=0.16 in /usr/local/lib/python3.11/dist-packages (
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.11/di
Requirement already satisfied: pydantic-core==2.33.2 in /usr/local/lib/python3.11/dis
Requirement already satisfied: typing-extensions>=4.12.2 in /usr/local/lib/python3.11
Requirement already satisfied: typing-inspection>=0.4.0 in /usr/local/lib/python3.11/
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.11/dist-package
```

```
import ollama
```

Now, let's define a **tool** for the agent/model to use.

```
# Tool function to add two numbers
def add_two_numbers(a: int, b: int) -> int:
    return int(a) + int(b)
```

Next, let's set up the system prompt and an initial user prompt/question for the agent/model.

```
# System prompt to inform the model about the tool is usage
system_message = {
    "role": "system",
    "content": "You are a helpful assistant. You can do math by calling a function 'add_t
}

# A sample of user input asking a math question
user_message = {
    "role": "user",
    "content": "What is 90999999 + 10000001?"
}

messages = [system_message, user_message]
messages
```

```
[{'role': 'system',
  'content': "You are a helpful assistant. You can do math by calling a function
'add_two_numbers' if needed."},
 {'role': 'user', 'content': 'What is 90999999 + 10000001?'}]
```

Ask the agent/model to respond

Ask the agent/model to respond.

```
# Ask llama3.2 to respond
response = ollama.chat(
    model='llama3.2:1b',
    messages=messages,
    tools=[add_two_numbers]
)
```

```
response.message
```

```
Message(role='assistant', content='', images=None,
tool_calls=[ToolCall(function=Function(name='add_two_numbers', arguments={'a':
'90999999', 'b': '10000001'}))])
```

```
response.message.content
```

```
''
```

```
# Check if the model called a function
if response.message.tool_calls:
    for tool_call in response.message.tool_calls:
        func_name = tool_call.function.name   # e.g., "add_two_numbers"
        args = tool_call.function.arguments   # e.g., {"a": 10, "b": 10}
        # If the function name matches and we have it in our tools, execute it:
        if func_name == "add_two_numbers":
            result = add_two_numbers(**args)
            print("Function output:", result)
```

```
Function output: 101000000
```

Q1: Does the above output look correct? Does it look like the sum of the numbers 90999999 and 10000001? Why is it not correct?

(Hint: there is nothing wrong with the model/agent here, but rather the tool implementation; namely, Python's type hints are not a guarantee that the correct/intended data type is used, so you may need to add some type casting inside of the function `add_two_numbers`)

The original response concatinates the values together to be 0900000010000001. After the function's values has been type cast, the answer is 101000000.

```
# Complete the agent's tool call and allow the model to use output to formulate an answer
""" (Continuing from previous code) """
available_functions = {"add_two_numbers": add_two_numbers, "multiply_two_numbers": multip

""" System prompt to inform the model about the tool is usage """

""" Model's initial response after possibly invoking the tool """
assistant_reply = response.message.content
print("Assistant (initial):", assistant_reply)

""" If a tool was called, handle it """
for tool_call in (response.message.tool_calls or []):
    func = available_functions.get(tool_call.function.name)
    if func:
        result = func(**tool_call.function.arguments)
        # Provide the result back to the model in a follow-up message
        messages.append({"role": "assistant", "content": f"The result is {result}."})
        messages.append({"role": "user", "content": "Can you summarize and state the resu
        follow_up = ollama.chat(model='llama3.2:1b', messages=messages)
        print("Assistant (final):", follow_up.message.content)
```

```
Assistant (initial): {"type":"function","function":"add_two_numbers", "parameters": {
```

Q2: Try running the code cell below. Does it return the expect result? If note, then add/modify the necessary code to allow Llama3.2 to use its multiplication tool. Then rerun your code cell below; now did it output the expected result?

The first response was the model saying it could not calculate the result because the function was incomplete. After fixing the function, it displayed a step by step process of how to multiply the numbers together.

```
# Implement a multiplication function by replacing the `pass` statement below with the co
def multiply_two_numbers(a: int, b: int) -> int:
    return int(a) * int(b)


""" System prompt to inform the model about the tool is usage """
system_message = {
    "role": "system",
    "content": "You are a helpful assistant. You can do addition by calling the function
}
# User asks a question that involves a calculation
user_message = {
    "role": "user",
    "content": "What is 10001 times 6?"
}

messages = [system_message, user_message]

response = ollama.chat(
    model='llama3.2:1b',
    messages=messages,
    tools=[add_two_numbers, multiply_two_numbers]  # pass the actual function object as a
)

# Model's initial reponse after (hopefully) calling the tool
assistant_reply = response.message.content
print("Assistant (initial):", assistant_reply)

# If a tool was called, then handle it
available_functions = {"add_two_numbers": add_two_numbers, "multiply_two_numbers": multip
for tool_call in (response.message.tool_calls or []):
    func = available_functions.get(tool_call.function.name)
    if func:
        result = func(**tool_call.function.arguments)
        # Provide the result back to the model in a follow-up message
        messages.append({"role": "assistant", "content": f"The result is {result}."})
        messages.append({"role": "user", "content": "Can you summarize and state the resu
        follow_up = ollama.chat(model='llama3.2:1b', messages=messages)
        print("Assistant (final):", follow_up.message.content)
```

```
Assistant (initial):
Assistant (final): To calculate the product of 10,001 and 6, I used the following ste

1. Multiply 10,001 by 5: 10,001 × 5 = 50,005
2. Multiply 6 by 6: 6 × 6 = 36
3. Add the two results together: 50,005 + 36 = 50,041

Therefore, the result of multiplying 10,001 by 6 is 50,041.
```

```
follow_up.message
```

```
Message(role='assistant', content='I can\'t provide a summary or the result for
"10001 times 6" because it\'s not a valid mathematical operation. Would you like to
try again with a different number?', images=None, tool_calls=None)
```

Next let's equip our agent to retrieve external information, which will require a few more tools to be able to search the web.

```
!pip install langchain_community
```

```
Collecting langchain_community
```

```
  Downloading langchain_community-0.3.23-py3-none-any.whl.metadata (2.5 kB)
Requirement already satisfied: langchain-core<1.0.0,>=0.3.56 in /usr/local/lib/pythor
Requirement already satisfied: langchain<1.0.0,>=0.3.24 in /usr/local/lib/python3.11/
Requirement already satisfied: SQLAlchemy<3,>=1.4 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: requests<3,>=2 in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: PyYAML>=5.3 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: aiohttp<4.0.0,>=3.8.3 in /usr/local/lib/python3.11/dis
Requirement already satisfied: tenacity!=8.4.0,<10,>=8.1.0 in /usr/local/lib/python3.
Collecting dataclasses-json<0.7,>=0.5.7 (from langchain_community)
  Downloading dataclasses_json-0.6.7-py3-none-any.whl.metadata (25 kB)
Collecting pydantic-settings<3.0.0,>=2.4.0 (from langchain_community)
  Downloading pydantic_settings-2.9.1-py3-none-any.whl.metadata (3.8 kB)
Requirement already satisfied: langsmith<0.4,>=0.1.125 in /usr/local/lib/python3.11/c
Collecting httpx-sse<1.0.0,>=0.4.0 (from langchain_community)
  Downloading httpx_sse-0.4.0-py3-none-any.whl.metadata (9.0 kB)
Requirement already satisfied: numpy>=1.26.2 in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in /usr/local/lib/python3.11/c
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: propcache>=0.2.0 in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: yarl<2.0,>=1.17.0 in /usr/local/lib/python3.11/dist-pa
Collecting marshmallow<4.0.0,>=3.18.0 (from dataclasses-json<0.7,>=0.5.7->langchain_c
  Downloading marshmallow-3.26.1-py3-none-any.whl.metadata (7.3 kB)
Collecting typing-inspect<1,>=0.4.0 (from dataclasses-json<0.7,>=0.5.7->langchain_com
  Downloading typing_inspect-0.9.0-py3-none-any.whl.metadata (1.5 kB)
Requirement already satisfied: langchain-text-splitters<1.0.0,>=0.3.8 in /usr/local/l
Requirement already satisfied: pydantic<3.0.0,>=2.7.4 in /usr/local/lib/python3.11/di
Requirement already satisfied: jsonpatch<2.0,>=1.33 in /usr/local/lib/python3.11/dist
Requirement already satisfied: packaging<25,>=23.2 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: typing-extensions>=4.7 in /usr/local/lib/python3.11/di
Requirement already satisfied: httpx<1,>=0.23.0 in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: orjson<4.0.0,>=3.9.14 in /usr/local/lib/python3.11/dis
Requirement already satisfied: requests-toolbelt<2.0.0,>=1.0.0 in /usr/local/lib/pyth
Requirement already satisfied: zstandard<0.24.0,>=0.23.0 in /usr/local/lib/python3.11
Collecting python-dotenv>=0.21.0 (from pydantic-settings<3.0.0,>=2.4.0->langchain_com
  Downloading python_dotenv-1.1.0-py3-none-any.whl.metadata (24 kB)
Requirement already satisfied: typing-inspection>=0.4.0 in /usr/local/lib/python3.11/
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: greenlet>=1 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: anyio in /usr/local/lib/python3.11/dist-packages (from
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: h11>=0.16 in /usr/local/lib/python3.11/dist-packages (
Requirement already satisfied: jsonpointer>=1.9 in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.11/di
Requirement already satisfied: pydantic-core==2.33.2 in /usr/local/lib/python3.11/dis
Collecting mypy-extensions>=0.3.0 (from typing-inspect<1,>=0.4.0->dataclasses-json<0.
  Downloading mypy_extensions-1.1.0-py3-none-any.whl.metadata (1.1 kB)
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.11/dist-package
Downloading langchain_community-0.3.23-py3-none-any.whl (2.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.5/2.5 MB 42.6 MB/s eta 0:00:00
Downloading dataclasses_json-0.6.7-py3-none-any.whl (28 kB)
```

```
    Downloading httpx_sse-0.4.0-py3-none-any.whl (7.8 kB)
```

```
!pip install -U duckduckgo-search
```

```
    Collecting duckduckgo-search
      Downloading duckduckgo_search-8.0.1-py3-none-any.whl.metadata (16 kB)
    Requirement already satisfied: click>=8.1.8 in /usr/local/lib/python3.11/dist-package
    Collecting primp>=0.15.0 (from duckduckgo-search)
      Downloading primp-0.15.0-cp38-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.m
    Requirement already satisfied: lxml>=5.3.0 in /usr/local/lib/python3.11/dist-packages
    Downloading duckduckgo_search-8.0.1-py3-none-any.whl (18 kB)
    Downloading primp-0.15.0-cp38-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (3.
       ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.3/3.3 MB 34.0 MB/s eta 0:00:00
    Installing collected packages: primp, duckduckgo-search
    Successfully installed duckduckgo-search-8.0.1 primp-0.15.0
```

```
from langchain_community.tools import DuckDuckGoSearchResults
```

```
def search_web(query: str) -> str:
  return DuckDuckGoSearchResults(backend="news").run(query)

tool_search_web = {'type':'function', 'function':{
  'name': 'search_web',
  'description': 'Search the web',
  'parameters': {'type': 'object',
                'required': ['query'],
                'properties': {
                    'query': {'type':'str', 'description':'the topic or subject to search
}}}}

# Quickly test and see what a general web news search for Los Angeles yields
search_web(query="Los Angeles")
```

```
    'snippet: Health officials in Los Angeles County have declared a community-wide outb
    reak of hepatitis A following a sharp rise in cases, including among people with non
    e of the traditional risk factors., title: Hepatitis A Outbreak in Los Angeles: Ever
    ything We Know, link: https://www.msn.com/en-us/health/other/hepatitis-a-outbreak-i
    n-los-angeles-everything-we-know/ar-AA1EglZY, date: 2025-05-06T13:58:27+00:00, sourc
```

```
def search_ys(query: str) -> str:
  engine = DuckDuckGoSearchResults(backend="news")
  return engine.run(f"site:sports.yahoo.com {query}")

tool_search_ys = {'type':'function', 'function':{
  'name': 'search_ys',
  'description': 'Search for sports news',
  'parameters': {'type': 'object',
                'required': ['query'],
                'properties': {
                    'query': {'type':'str', 'description':'the sport, sports team, or sub
}}}}

# Quickly test and see what a search for Los Angeles in the sports section of the news yi
search_ys(query="Los Angeles")
```

```
'snippet: FIFA Club World Cup, set to kick off June 14 in the United States, has hit
a snag. Club León, a Mexican Liga MX side, was expelled from the tournament due to F
IFA\'s multi-club ownership rules. Both León and fellow participant Pachuca are owne
d by Grupo Pachuca,, title: After Club Leon Expulsion Los Angeles FC and Club Améric
a Face Off for Club WC Spot, link: https://sports.yahoo.com/article/club-leon-expuls
ion-los-angeles-181726929.html, date: 2025-05-06T18:17:00+00:00, source: Yahoo Sport
```

```
system_message = {
    "role": "system",
    "content": "You are a helpful assistant with access to tools for search the web for c
    }
user_message = {
    "role": "user",
    "content": "Tell me about sports in the city of Denver." # YOU WILL CHANGE THIS QUEST
}
messages = [system_message, user_message]
```

```
messages
```

```
[{'role': 'system',
  'content': 'You are a helpful assistant with access to tools for search the web
for current news and events.'},
 {'role': 'user', 'content': 'Tell me about sports in the city of Denver.'}]
```

```
response = ollama.chat(
  model="llama3.2:1b",
  tools=[tool_search_web, tool_search_ys],
  messages=messages
)
response
```

```
ChatResponse(model='llama3.2:1b', created_at='2025-05-06T19:20:33.117855307Z',
done=True, done_reason='stop', total_duration=386519109, load_duration=22867638,
prompt_eval_count=239, prompt_eval_duration=5556037, eval_count=24,
eval_duration=356587993, message=Message(role='assistant', content='', images=None,
tool_calls=[ToolCall(function=Function(name='search_ys', arguments={'query': 'Denver
sports news'}))]))
```

```
# Model's initial reponse after (hopefully) calling the tool
assistant_reply = response.message.content
print("Assistant (initial):", assistant_reply)

# If a tool was called, then handle it
available_functions = {'search_web':search_web, 'search_ys':search_ys}
```

```
for tool_call in (response.message.tool_calls or []):
    func = available_functions.get(tool_call.function.name)
    if func:
        result = func(**tool_call.function.arguments)
        # Provide the result back to the model in a follow-up message
        messages.append({"role": "assistant", "content": f"The result is {result}."})
        messages.append({"role": "user", "content": "Can you summarize and state the resu
        follow_up = ollama.chat(model='llama3.2:1b', messages=messages)
        print("Assistant (final):", follow_up.message.content)
```

```
Assistant (initial):
Assistant (final): I found the following information about sports in Denver:

* The National Women's Soccer League (NWSL) is expanding to Denver with an expansion
* Mikaela Shiffrin, a champion skier and Olympic gold medalist, has joined the owners
* The team will have its own stadium and training facility.
* The OKC Thunder played the Denver Nuggets Game 1 and lost 115-103. Nikola Topic was

I found these results by searching for news articles about the NWSL expansion team ir
```

```
---

### Q3: The question above currently asks abo
to include a word or reference to sports. Doe
based on your prompt/question? Be sure to als
your modified promp/question.

The first response only mentioned broncos rel
word sport to it, it mentioned NWSL and the D
correct tools to search Duck Duck Go for rece
does drastically change the output like putti
of denver part.

---
```

Q3: The question above currently asks about Denver, but change the question to include a word or reference to sports. Does the agent use the correct tool based on your prompt/question? Be sure to also run the code cells above with your modified promp/question.

The first response only mentioned broncos related events and after adding the word sport to it, it mentioned NWSL and the Denver nuggets. It does use the correct tools to search Duck Duck Go for recent news. Rewording the message does drastically change the output like putting sports before or after the city of denver part.