Homework 2

Let's create a social media account for your agent

Setup your agent

↳ 已隐藏 3 个单元格

Create a moltbook account for your agent

显示代码

```
# Before creating your agent please encode your student id using this function and replace XXXX by the encoded numbe
encode_student_id(1155247025)
```

```
'68843336'
```

```
# Please use the encoded student id
!curl -X POST https://www.moltbook.com/api/v1/agents/register \
    -H "Content-Type: application/json" \
    -d '{"name": "XinxuanXie_68843336", "description": "Va"}'
```

```
xuanxie_68843336","api_key":"moltbook_sk_MxE5CVLE49awH3w6zxT-ySulsjhXCyiq","claim_url":"https://www.moltbook.com/claim/moltbook_claim_HB2
```

- After sucessfully register, you will see a notification of the format:

"success":true,"message":"Welcome to Moltbook! 🦞","agent":"id":"...","name":"...","api_key":"...", "claim_url": "..."

- Please save your the api key as MOLTBOOK_API_KEY in the Secrets section of your Colab.
- Then you complete the registration by accessing the claim_url and follow the guideline in the url.

```
# Create a tool set to interact with moltbook

import os
import requests
from langchain_core.tools import tool

MOLTBOOK_API_KEY = userdata.get('MOLTBOOK_API_KEY')
assert MOLTBOOK_API_KEY, "Please set your MOLTBOOK_API_KEY in Colab secrets"

BASE_URL = "https://www.moltbook.com/api/v1"
MOLTBOOK_SKILL_URL = "https://www.moltbook.com/skill.md"
TARGET_POST_ID = "47ff50f3-8255-4dee-87f4-2c3637c7351c"

HEADERS = {
    "Authorization": f"Bearer {MOLTBOOK_API_KEY}",
    "Content-Type": "application/json"
}

def _safe_json_response(r):
    """
    Return a structured dict so the agent can inspect both success and error cases.
    """
    try:
        payload = r.json()
    except Exception:
        payload = {"raw_text": r.text}

    return {
        "ok": r.ok,
        "status_code": r.status_code,
        "data": payload
    }

# ---------- AUTH ----------
@tool
def authenticate_agent() -> dict:
    """Authenticate with Moltbook API using the API key and return current agent profile."""
    r = requests.get(
        f"{BASE_URL}/agents/me",
        headers=HEADERS,
        timeout=15
    )
    return _safe_json_response(r)

# ---------- SKILL ----------
@tool
def read_moltbook_skill() -> dict:
    """Read Moltbook skill instructions from the official skill.md file."""
    r = requests.get(
        MOLTBOOK_SKILL_URL,
        timeout=15
    )
    return {
        "ok": r.ok,
        "status_code": r.status_code,
        "text": r.text[:6000]
    }

# ---------- FEED ----------
@tool
def get_feed(sort: str = "new", limit: int = 10) -> dict:
    """Fetch Moltbook feed."""
    r = requests.get(
        f"{BASE_URL}/feed",
        headers=HEADERS,
        params={"sort": sort, "limit": limit},
        timeout=15
    )
    return _safe_json_response(r)

# ---------- SEARCH ----------
@tool
def search_moltbook(query: str, type: str = "all") -> dict:
    """Semantic search Moltbook posts, comments, agents."""
    r = requests.get(
        f"{BASE_URL}/search",
        headers=HEADERS,
        params={"q": query, "type": type},
        timeout=15
    )
    return _safe_json_response(r)

# ---------- POST ----------
@tool
def create_post(submolt: str, title: str, content: str) -> dict:
    """Create a new text post."""
    payload = {
        "submolt": submolt,
        "title": title,
        "content": content
    }
    r = requests.post(
        f"{BASE_URL}/posts",
        headers=HEADERS,
        json=payload,
        timeout=15
    )
    return _safe_json_response(r)

# ---------- SUBSCRIBE ----------
@tool
def subscribe_submolt(submolt: str) -> dict:
    """
    Subscribe to a submolt.
    Examples:
    - 'ftec5660'
    - '/m/ftec5660'
    - 'm/ftec5660'
    """
    cleaned = str(submolt).strip()
    cleaned = cleaned.replace("/m/", "").replace("m/", "").strip("/")

    r = requests.post(
        f"{BASE_URL}/submolts/{cleaned}/subscribe",
        headers=HEADERS,
        timeout=15
    )
    return _safe_json_response(r)

# ---------- COMMENT ----------
@tool
def comment_post(post_id: str, content: str) -> dict:
```


2026/2/27 00:14 ""FTEC5660_hw2p2_ipynb"的副本"的副本 - Colab

```
def comment_post(post_id: str, content: str) -> dict:
    """Comment on a post."""
    r = requests.post(
        f"{BASE_URL}/posts/{post_id}/comments",
        headers=HEADERS,
        json={"content": content},
        timeout=15
    )
    return _safe_json_response(r)

# ---------- VOTE ----------
@tool
def upvote_post(post_id: str) -> dict:
    """Upvote a post."""
    r = requests.post(
        f"{BASE_URL}/posts/{post_id}/upvote",
        headers=HEADERS,
        timeout=15
    )
    return _safe_json_response(r)
```

```
SYSTEM_PROMPT = f"""
You are a Moltbook AI agent built for the FTEC5660 homework.

Your purpose:
- Complete the required Moltbook homework tasks exactly once
- Be reliable, minimal, and non-spammy
- Use tools carefully and stop when the job is done

Rules:
1. First read the official Moltbook skill instructions with read_moltbook_skill.
2. Then authenticate with authenticate_agent.
3. Then subscribe to /m/ftec5660 using subscribe_submolt.
4. Then upvote post {TARGET_POST_ID}.
5. Then comment on post {TARGET_POST_ID}.
6. The comment must be short, professional, and non-spam.
7. Do NOT create extra posts unless a human explicitly asks for that.
8. Do NOT repeat actions if a tool result already shows success.
9. If a human gives an instruction, obey it exactly.
10. When finished, return a short summary of what you completed and the status codes.

Available tools:
- authenticate_agent
- read_moltbook_skill
- get_feed
- search_moltbook
- create_post
- subscribe_submolt
- comment_post
- upvote_post
"""
```

A simple agent to interact with moltbook

```
from langchain_google_genai import ChatGoogleGenerativeAI
from langchain_core.messages import ToolMessage
import time
import json
from datetime import datetime, timezone
from typing import Any

def log(section: str, message: str):
    ts = datetime.now(timezone.utc).strftime("%H:%M:%S")
    print(f"[{ts}] [{section}] {message}")

def pretty(obj: Any, max_len: int = 1200):
    text = json.dumps(obj, indent=2, ensure_ascii=False, default=str)
    return text if len(text) <= max_len else text[:max_len] + "\n...<truncated>"

def extract_text_content(content: Any) -> str:
    """
    把最后的 response.content 转成纯文本，避免输出成 list。
    """
    if content is None:
        return ""

    if isinstance(content, str):
        return content.strip()

    if isinstance(content, list):
        parts = []
        for item in content:
            if isinstance(item, dict) and item.get("type") == "text":
                parts.append(item.get("text", ""))
            else:
                parts.append(str(item))
        return "\n".join([p for p in parts if p]).strip()

    return str(content).strip()

def moltbook_agent_loop(
    instruction: str | None = None,
    max_turns: int = 10,
    verbose: bool = True,
):
    log("INIT", "Starting Moltbook agent loop")

    llm = ChatGoogleGenerativeAI(
        model="gemini-2.5-flash",
        temperature=0,
        api_key=GEMINI_VERTEX_API_KEY,
        vertexai=True,
    )

    tools = [
        authenticate_agent,
        read_moltbook_skill,
        get_feed,
        search_moltbook,
        create_post,
        subscribe_submolt,
        comment_post,
        upvote_post,
    ]

    agent = llm.bind_tools(tools)

    history = [("system", SYSTEM_PROMPT)]

    if instruction:
        history.append(("human", f"Human instruction: {instruction}"))
        log("HUMAN", instruction)
    else:
        history.append(("human", "Perform your Moltbook homework tasks."))
        log("HOMEWORK", "No human instruction - default homework mode")

    for turn in range(1, max_turns + 1):
        log("TURN", f"Turn {turn}/{max_turns} started")
        turn_start = time.time()

        response = agent.invoke(history)
        history.append(response)

        if verbose:
            log("LLM", "Model responded")
            log("LLM.CONTENT", extract_text_content(response.content) if response.content else "<empty>")
            log("LLM.TOOL_CALLS", pretty(response.tool_calls or []))

        if not response.tool_calls:
            elapsed = round(time.time() - turn_start, 2)
            log("STOP", f"No tool calls — final answer produced in {elapsed}s")
            return extract_text_content(response.content)

        for i, call in enumerate(response.tool_calls, start=1):
            tool_name = call["name"]
            args = call["args"]
            tool_id = call["id"]

            log("TOOL", f"[{i}] Calling `{tool_name}`")
            log("TOOL.ARGS", pretty(args))

            tool_fn = globals().get(tool_name)
            tool_start = time.time()

            try:
                result = tool_fn.invoke(args)

                if isinstance(result, dict) and result.get("ok") is False:
                    status = f"api_failed_{result.get('status_code', 'unknown')}"
```


2026/2/27 00:14 ""FTEC5660_hw2p2_ipynb"的副本"的副本 - Colab

https://colab.research.google.com/drive/1zgGkM5esZ8s53JPjfDfV_PPLOpRwnoPy#scrollTo=FPhY3o16M29e&printMode=true 4/6

```
                else:
                        status = "success"

                except Exception as e:
                        result = {"ok": False, "status_code": None, "data": {"error": str(e)}}
                        status = "error"

                tool_elapsed = round(time.time() - tool_start, 2)

                log("TOOL.RESULT", f"{tool_name} finished ({status}) in {tool_elapsed}s")

                if verbose:
                        log("TOOL.OUTPUT", pretty(result))

                history.append(
                        ToolMessage(
                                tool_call_id=tool_id,
                                content=json.dumps(result, ensure_ascii=False)
                        )
                )

        turn_elapsed = round(time.time() - turn_start, 2)
        log("TURN", f"Turn {turn} completed in {turn_elapsed}s")

log("STOP", "Max turns reached without final answer")
return "Agent stopped after reaching max turns."
```

```
HOMEWORK_INSTRUCTION = f"""
Complete the FTEC5660 Moltbook homework in order.

Required tasks:
1. Read the official Moltbook skill instructions.
2. Authenticate with the Moltbook API using the API key.
3. Subscribe to /m/ftec5660.
4. Upvote post {TARGET_POST_ID}.
5. Comment on post {TARGET_POST_ID}.

Use this exact comment:
"FTEC5660 homework completed by an autonomous Moltbook agent."

Important constraints:
- Do not create any extra post.
- Do not perform duplicate actions.
- If a step fails, explain which step failed and include the returned status code.
- When all steps finish, return a concise completion summary.
"""

result = moltbook_agent_loop(HOMEWORK_INSTRUCTION)
print(result)
```

```
[16:08:28] [INIT] Starting Moltbook agent loop
[16:08:28] [HUMAN]
Complete the FTEC5660 Moltbook homework in order.

Required tasks:
1. Read the official Moltbook skill instructions.
2. Authenticate with the Moltbook API using the API key.
3. Subscribe to /m/ftec5660.
4. Upvote post 47ff50f3-8255-4dee-87f4-2c3637c7351c.
5. Comment on post 47ff50f3-8255-4dee-87f4-2c3637c7351c.

Use this exact comment:
"FTEC5660 homework completed by an autonomous Moltbook agent."

Important constraints:
- Do not create any extra post.
- Do not perform duplicate actions.
- If a step fails, explain which step failed and include the returned status code.
- When all steps finish, return a concise completion summary.

[16:08:28] [TURN] Turn 1/10 started
[16:08:31] [LLM] Model responded
[16:08:31] [LLM.CONTENT] <empty>
[16:08:31] [LLM.TOOL_CALLS] [
  {
    "name": "read_moltbook_skill",
    "args": {},
    "id": "bed0dd2a-b5bf-498b-a32e-7970d6609edd",
    "type": "tool_call"
  }
]
```



```
[16:08:31] [TOOL] [1] Calling `read_moltbook_skill`
[16:08:31] [TOOL.ARGS] {}
[16:08:31] [TOOL.RESULT] read_moltbook_skill finished (success) in 0.34s
[16:08:31] [TOOL.OUTPUT] {
  "ok": true,
  "status_code": 200,
  "text": "---\nname: moltbook\nversion: 1.12.0\ndescription: The social network for AI agents. Post, comment, upvote, and create comm
...<truncated>
[16:08:31] [TURN] Turn 1 completed in 3.48s
[16:08:31] [TURN] Turn 2/10 started
[16:08:32] [LLM] Model responded
[16:08:32] [LLM.CONTENT] <empty>
[16:08:32] [LLM.TOOL_CALLS] [
  {
    "name": "authenticate_agent",
    "args": {},
    "id": "215afc6d-1c47-4e4f-8f61-3f39a92d9f18",
    "type": "tool_call"
  }
]
[16:08:32] [TOOL] [1] Calling `authenticate_agent`
[16:08:32] [TOOL.ARGS] {}
[16:08:39] [TOOL.RESULT] authenticate_agent finished (success) in 7.15s
[16:08:39] [TOOL.OUTPUT] {
  "ok": true,
  "status_code": 200,
```