```
# Colab cell: install packages
!pip install -qU sentence-transformers faiss-cpu fastapi "uvicorn[standard]" nest-asyncio pyngrok requests sympy python-multipart pydantic
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 87.7/87.7 kB 6.1 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 31.4/31.4 MB 61.5 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 108.4/108.4 kB 10.2 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 64.7/64.7 kB 5.9 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6.3/6.3 MB 113.9 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 462.4/462.4 kB 36.2 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.1/2.1 MB 69.6 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 517.7/517.7 kB 38.2 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.4/4.4 MB 114.9 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 456.8/456.8 kB 36.2 MB/s eta 0:00:00
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the following dependency conflicts.
google-colab 1.0.0 requires requests==2.32.4, but you have requests 2.32.5 which is incompatible.
gradio 5.49.1 requires pydantic<2.12,>=2.0, but you have pydantic 2.12.3 which is incompatible.
```

```
# Colab cell
import os, json, uuid, time, typing
from typing import List, Dict, Any, Optional
from sentence_transformers import SentenceTransformer
import numpy as np
import faiss
from fastapi import FastAPI, Request, HTTPException
from fastapi.middleware.cors import CORSMiddleware
from pydantic import BaseModel
import uvicorn
import nest_asyncio
from pyngrok import ngrok
import requests
import sympy as sp

# Config - replace keys below with your own
OPENAI_API_KEY = os.environ.get("OPENAI_API_KEY", "")   # optional
MCP_SEARCH_API_KEY = os.environ.get("MCP_SEARCH_API_KEY", "")  # optional

# Embedding model
EMBED_MODEL_NAME = "sentence-transformers/all-mpnet-base-v2"
embed_model = SentenceTransformer(EMBED_MODEL_NAME)

# FAISS / KB storage (in-memory)
DIM = embed_model.get_sentence_embedding_dimension()
index = faiss.IndexFlatL2(DIM)
kb_docs: Dict[str, Dict] = {}  # id -> doc (doc includes question, solution, metadata)
vectors = []  # list of vectors in same order as IDs stored in ids_list
ids_list = []
```

```
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as secret in your Google Colab and restart your session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
modules.json: 100%  349/349 [00:00<00:00, 28.8kB/s]
config_sentence_transformers.json: 100%  116/116 [00:00<00:00, 7.64kB/s]
README.md:  11.6k/? [00:00<00:00, 1.00MB/s]
sentence_bert_config.json: 100%  53.0/53.0 [00:00<00:00, 5.66kB/s]
config.json: 100%  571/571 [00:00<00:00, 50.4kB/s]
model.safetensors: 100%  438M/438M [00:02<00:00, 250MB/s]
tokenizer_config.json: 100%  363/363 [00:00<00:00, 34.3kB/s]
vocab.txt:  232k/? [00:00<00:00, 4.09MB/s]
tokenizer.json:  466k/? [00:00<00:00, 11.2MB/s]
special_tokens_map.json: 100%  239/239 [00:00<00:00, 20.6kB/s]
```

Helper functions: embed, add to KB, search KB

```
# Colab cell
def embed_text(text: str) -> np.ndarray:
    vec = embed_model.encode(text, convert_to_numpy=True)
    return vec / (np.linalg.norm(vec) + 1e-12)

def kb_add(doc: Dict[str, Any]) -> str:
    """
    doc = {
        "question": "<LaTeX or plain text>",
        "solution": "<step-by-step in LaTeX/plain>",
        "summary": "...",
        "tags": ["calculus"],
        "difficulty": 3,
        "source": "internal"
    }
    """
    doc_id = doc.get("id") or str(uuid.uuid4())
    doc["id"] = doc_id
    kb_docs[doc_id] = doc
    vec = embed_text(doc["question"])
    index.add(np.array([vec], dtype='float32'))
    ids_list.append(doc_id)
    return doc_id

def kb_search(question: str, top_k:int=5):
    qv = embed_text(question).astype('float32')
    if index.ntotal == 0:
        return []
    D, I = index.search(np.array([qv]), top_k)
    results = []
    for dist, i in zip(D[0], I[0]):
        if i < 0:
```

```
            continue
        doc_id = ids_list[i]
        score = float(1/(1+dist))  # convert L2 to pseudo-sim
        results.append({"id": doc_id, "score": score, "doc": kb_docs[doc_id]})
    return results
```

```
# Colab cell: seed sample KB
sample_docs = [
    {
        "id":"KB1",
        "question":"Solve 2x^2 - 5x + 2 = 0",
        "solution":"1) Use quadratic formula x = [5 ± sqrt(25-16)]/4 => sqrt = 3, so x = (5 ± 3)/4 => x=2 or x=1/2 (check).",
        "summary":"Quadratic equation solved using quadratic formula.",
        "tags":["algebra","quadratic"],
        "difficulty":1,
        "source":"seed"
    },
    {
        "id":"KB2",
        "question":"Evaluate integral from 0 to 1 of x^2 dx",
        "solution":"Antiderivative of x^2 is x^3/3. Evaluate from 0 to 1 => 1/3.",
        "summary":"Basic definite integral using power rule.",
        "tags":["calculus","integration"],
        "difficulty":1,
        "source":"seed"
    },
    {
        "id":"KB3",
        "question":"State and show a simple proof of Fermat's little theorem for prime p: a^p ≡ a (mod p)",
        "solution":"One can use binomial expansion (a+1)^p ≡ a^p +1 (mod p) and induction, or group theory over Z_p^*. See standard proofs.",
        "summary":"Fermat's little theorem sketch proof.",
        "tags":["number_theory","theorem"],
        "difficulty":3,
        "source":"seed"
    }
]

for d in sample_docs:
    kb_add(d)

print("KB seeded. current KB size:", len(kb_docs))
```

```
KB seeded. current KB size: 3
```

Simple mock-MCP web search function

```
# Colab cell
def mock_mcp_search(question: str, top_k=3):
    """
    Mock MCP web search: returns structured passages.
```

```
    Replace with actual MCP/Serper/Tavily calls.
    """
    # In production: call your MCP server or Serper and return MCP-style structured dicts
    # Here: we return an empty list if we consider not found
    web_passages = []
    # Example: for integration testing, we include one synthetic passage for Fourier question
    if "Fourier transform of a rectangular pulse" in question:
        web_passages.append({
            "id":"web1",
            "title":"Fourier transform of rectangular pulse",
            "url":"https://example.edu/fourier/rect",
            "snippet":"The Fourier transform of a rectangular pulse of width T is T sinc(f T).",
            "text":"If rect(t/T) then FT is T * sinc(pi f T) (conventions vary).",
            "score":0.9
        })
    return web_passages
```

LLM call wrapper (placeholder)

```
# Colab cell
def call_llm_generate(question: str, context_docs: List[Dict], use_kb: bool):
    """
    This is a placeholder. Replace with real LLM call (OpenAI/Anthropic).
    The function must return a dict with keys: steps (list), final_answer (str), citations (list), confidence (float)
    """
    # Build a simple deterministic answer using KB if present
    if use_kb and context_docs:
        # Use top KB doc
        doc = context_docs[0]["doc"]
        return {
            "steps":[{"step":i+1,"desc":s.strip(),"citation":doc["id"]} for i,s in enumerate(doc["solution"].split(".")) if s.strip()],
            "final_answer": doc["solution"].split(".")[-2].strip() if "." in doc["solution"] else doc["solution"],
            "citations":[doc["id"]],
            "confidence":0.95,
            "used_kb": True
        }
    # If no KB, but mock web context
    if context_docs:
        doc = context_docs[0]
        return {
            "steps":[{"step":1,"desc":doc["text"],"citation":doc["url"]}],
            "final_answer":"See derivation in "+doc["url"],
            "citations":[doc["url"]],
            "confidence":0.75,
            "used_kb": False
        }
    # If nothing: attempt naive derivation (very low confidence)
    return {
        "steps":[{"step":1,"desc":"I couldn't find exact resource; attempt: try reasoning from basics.", "citation":None}],
        "final_answer":"No confident solution (human review needed).",
        "citations":[],
        "confidence":0.2,
```

```
        "used_kb": False
    }
```

```
# Colab cell
def route_and_solve(question: str, kb_threshold=0.6):
    """
    1) sanitize question (simple),
    2) search KB,
    3) if match above threshold -> use KB,
    4) else call MCP/web,
    5) call LLM with context,
    6) return final structured response
    """
    # 0. Basic sanitization - remove undesirable content (very simple)
    q = question.strip()
    # 1. KB search
    kb_hits = kb_search(q, top_k=3)
    if kb_hits and kb_hits[0]["score"] >= kb_threshold:
        # Use KB
        llm_out = call_llm_generate(q, kb_hits, use_kb=True)
        llm_out["source"] = "KB"
        llm_out["kb_hits"] = kb_hits
        return llm_out
    # 2. MCP / web
    web_hits = mock_mcp_search(q, top_k=3)
    if web_hits:
        llm_out = call_llm_generate(q, web_hits, use_kb=False)
        llm_out["source"] = "WEB"
        llm_out["web_hits"] = web_hits
        return llm_out
    # 3. Nothing found: attempt derivation (LLM low-confidence)
    llm_out = call_llm_generate(q, [], use_kb=False)
    llm_out["source"] = "DERIVATION"
    return llm_out
```

```
# Colab cell
def is_equivalent_symbolic(ans1: str, ans2: str) -> bool:
    """
    Try to interpret numeric or symbolic equivalence using sympy.
    ans1, ans2: strings like '1/3', 'x=2', '2' etc.
    """
    try:
        # Clean
        a1 = ans1.replace("=", "").strip()
        a2 = ans2.replace("=", "").strip()
        # Try numeric
        v1 = sp.sympify(a1)
        v2 = sp.sympify(a2)
        return sp.simplify(v1 - v2) == 0
    except Exception as e:
        # fallback simple string match
        return a1.strip().lower() == a2.strip().lower()
```

```
# Colab cell
def benchmark_on_dataset(dataset: List[Dict[str,str]]):
    """
    dataset: list of {id, question, gold_answer}
    returns basic metrics
    """
    results = []
    for item in dataset:
        out = route_and_solve(item["question"])
        final = out.get("final_answer", "")
        correct = is_equivalent_symbolic(final, item["gold_answer"])
        results.append({
            "id": item["id"],
            "question": item["question"],
            "final_answer": final,
            "gold": item["gold_answer"],
            "correct": correct,
            "confidence": out.get("confidence"),
            "source": out.get("source")
        })
    # summary
    accuracy = sum(r["correct"] for r in results)/len(results) if results else 0
    return {"accuracy":accuracy, "n":len(results), "detail":results}

# small synthetic "JEE Bench" subset for testing
jee_sample = [
    {"id":"J1","question":"Evaluate integral from 0 to 1 of x^2 dx","gold_answer":"1/3"},
    {"id":"J2","question":"Solve 2x^2 - 5x + 2 = 0","gold_answer":"x=2 or x=1/2"},
    {"id":"J3","question":"Fourier transform of a rectangular pulse","gold_answer":"T * sinc(...)"}  # fuzzy
]

bench_res = benchmark_on_dataset(jee_sample)
bench_res
```

```
{'accuracy': 0.0,
 'n': 3,
```

```
 'detail': [{'id': 'J1',
   'question': 'Evaluate integral from 0 to 1 of x^2 dx',
   'final_answer': 'Evaluate from 0 to 1 => 1/3',
   'gold': '1/3',
   'correct': False,
   'confidence': 0.95,
   'source': 'KB'},
  {'id': 'J2',
   'question': 'Solve 2x^2 - 5x + 2 = 0',
   'final_answer': '1) Use quadratic formula x = [5 ± sqrt(25-16)]/4 => sqrt = 3, so x = (5 ± 3)/4 => x=2 or x=1/2 (check)',
   'gold': 'x=2 or x=1/2',
   'correct': False,
   'confidence': 0.95,
   'source': 'KB'},
  {'id': 'J3',
   'question': 'Fourier transform of a rectangular pulse',
   'final_answer': 'See derivation in https://example.edu/fourier/rect',
   'gold': 'T * sinc(...)',
   'correct': False,
   'confidence': 0.75,
   'source': 'WEB'}]}
```

```
# Colab cell
# We'll run a tiny FastAPI app inside Colab. Use ngrok to expose if needed.

app = FastAPI(title="Math Agent API")

# CORS for testing
app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

# Input guardrail - middleware
@app.middleware("http")
async def input_guardrail(request: Request, call_next):
    # Only simple JSON POST for /solve expected
    try:
        if request.url.path == "/solve" and request.method == "POST":
            body = await request.json()
            q = body.get("question", "")
            if not q or len(q.strip()) == 0:
                raise HTTPException(status_code=400, detail="Empty question. Please provide a math question.")
            # scope filter: basic heuristic (allow common math words and symbols)
            forbidden_keywords = ["lawyer","medical","diagnosis","copyright","piracy"]
            if any(k in q.lower() for k in forbidden_keywords):
                raise HTTPException(status_code=400, detail="Out of scope. This assistant handles math education only.")
            # PII simple strip (very naive): remove emails and phone-like tokens
            import re
            q = re.sub(r'\S+@\S+', '[email_removed]', q)
            q = re.sub(r'\+?\d[\d\-\s]{7,}\d', '[phone_removed]', q)
            # mutate request body
            request._body = json.dumps({"question": q})
    except HTTPException as e:
```

```
            raise e
        except Exception as e:
            # allow other routes
            pass
        response = await call_next(request)
        # Output guardrail: ensure JSON schema for /solve
        if request.url.path == "/solve" and response.status_code == 200:
            # we assume endpoints return dicts; you might validate here
            pass
        return response

    # Pydantic models
    class SolveRequest(BaseModel):
        question: str

    class FeedbackRequest(BaseModel):
        question_id: Optional[str]
        original_question: str
        correction: str
        user: Optional[str] = "anonymous"

    # Routes
    @app.post("/solve")
    async def solve_endpoint(req: SolveRequest):
        q = req.question
        out = route_and_solve(q)
        # Basic output guardrail: enforce schema fields
        schema_keys = {"steps","final_answer","citations","confidence","used_kb","source"}
        for k in ["steps","final_answer","citations","confidence","used_kb","source"]:
            if k not in out:
                # normalize to safe defaults
                out.setdefault("steps", [{"step":1,"desc":"No steps available","citation":None}])
                out.setdefault("final_answer", "")
                out.setdefault("citations", [])
                out.setdefault("confidence", 0.0)
                out.setdefault("used_kb", False)
                out.setdefault("source", out.get("source","unknown"))
                break
        return out

    @app.post("/feedback")
    async def feedback_endpoint(req: FeedbackRequest):
        """
        Stores user correction and updates KB (creates new KB doc and ingests embedding)
        """
        # Create new KB entry from correction
        doc = {
            "question": req.original_question,
            "solution": req.correction,
            "summary": "User correction",
            "tags": ["user_feedback"],
            "difficulty": 3,
            "source": f"user:{req.user}"
        }
        new_id = kb_add(doc)
        return {"status":"ok","new_kb_id":new_id}
```

```
# run API via ngrok
```

```
from fastapi import FastAPI
from pyngrok import ngrok
import nest_asyncio
import uvicorn

# ✅ Add your ngrok token here
NGROK_TOKEN = "YOUR_REAL_NGROK_TOKEN"

# Apply async fix
nest_asyncio.apply()

# ✅ Set ngrok auth token
ngrok.set_auth_token(NGROK_TOKEN)

# Create FastAPI app
app = FastAPI()

@app.get("/")
def root():
    return {"status": "✅ API is running with Ngrok!"}

# ✅ Start ngrok tunnel
public_url = ngrok.connect(8000)
print("🚀 Public URL:", public_url)

# Run the server
uvicorn.run(app, port=8000)
```

```
ERROR:pyngrok.process.ngrok:t=2025-10-30T10:07:33+0000 lvl=eror msg="failed to reconnect session" obj=tunnels.session err="authentication failed: The authtoken you specified does not look
---------------------------------------------------------------------------
PyngrokNgrokError                         Traceback (most recent call last)
/tmp/ipython-input-3635929803.py in <cell line: 0>()
     21
     22 # ✅ Start ngrok tunnel
---> 23 public_url = ngrok.connect(8000)
     24 print("🚀 Public URL:", public_url)
     25

                                    3 frames
/usr/local/lib/python3.12/dist-packages/pyngrok/process.py in _start_process(pyngrok_config)
    445
    446         if ngrok_process.startup_error is not None:
--> 447             raise PyngrokNgrokError(f"The ngrok process errored on start: {ngrok_process.startup_error}.",
    448                                     ngrok_process.logs,
    449                                     ngrok_process.startup_error)

PyngrokNgrokError: The ngrok process errored on start: authentication failed: The authtoken you specified does not look like a proper ngrok authtoken.\nYour authtoken:
YOUR_REAL_NGROK_TOKEN\nInstructions to install your authtoken are on your ngrok dashboard:\nhttps://dashboard.ngrok.com/get-started/your-authtoken\r\n\r\nERR_NGROK_105\r\n.
```

```
import nest_asyncio
nest_asyncio.apply()

port = 8000
public_url = ngrok.connect(port).public_url
print("Public URL:", public_url)
```

```
ERROR:pyngrok.process.ngrok:t=2025-10-30T10:04:26+0000 lvl=eror msg="failed to reconnect session" obj=tunnels.session err="authentication failed: The authtoken you specified does not look
ERROR:pyngrok.process.ngrok:t=2025-10-30T10:04:26+0000 lvl=eror msg="session closing" obj=tunnels.session err="authentication failed: The authtoken you specified does not look like a prope
---------------------------------------------------------------------------
PyngrokNgrokError                         Traceback (most recent call last)
/tmp/ipython-input-2415930136.py in <cell line: 0>()
      3
      4 port = 8000
----> 5 public_url = ngrok.connect(port).public_url
      6 print("Public URL:", public_url)

                                 3 frames
/usr/local/lib/python3.12/dist-packages/pyngrok/process.py in _start_process(pyngrok_config)
    445
    446         if ngrok_process.startup_error is not None:
--> 447             raise PyngrokNgrokError(f"The ngrok process errored on start: {ngrok_process.startup_error}.",
    448                                     ngrok_process.logs,
    449                                     ngrok_process.startup_error)

PyngrokNgrokError: The ngrok process errored on start: authentication failed: The authtoken you specified does not look like a proper ngrok authtoken.\nYour authtoken:
YOUR_NGROK_AUTHTOKEN_HERE\nInstructions to install your authtoken are on your ngrok dashboard:\nhttps://dashboard.ngrok.com/get-started/your-authtoken\r\n\r\nERR_NGROK_105\r\n.
```

```
# Colab cell: run FastAPI with ngrok in the same notebook
nest_asyncio.apply()
port = 8000
public_url = ngrok.connect(port).public_url
print("Public URL:", public_url)

# Run uvicorn in background
import threading, subprocess, sys, os, time
def run_uvicorn():
    uvicorn.run(app, host="0.0.0.0", port=port)

t = threading.Thread(target=run_uvicorn, daemon=True)
t.start()
time.sleep(1)
print("FastAPI running at", public_url)
```

```
ERROR:pyngrok.process.ngrok:t=2025-10-30T10:04:11+0000 lvl=eror msg="failed to reconnect session" obj=tunnels.session err="authentication failed: The authtoken you specified does not look
ERROR:pyngrok.process.ngrok:t=2025-10-30T10:04:11+0000 lvl=eror msg="session closing" obj=tunnels.session err="authentication failed: The authtoken you specified does not look like a prope
ERROR:pyngrok.process.ngrok:t=2025-10-30T10:04:11+0000 lvl=eror msg="terminating with error" obj=app err="authentication failed: The authtoken you specified does not look like a proper ngr
---------------------------------------------------------------------------
PyngrokNgrokError                         Traceback (most recent call last)
/tmp/ipython-input-760747090.py in <cell line: 0>()
      2 nest_asyncio.apply()
      3 port = 8000
----> 4 public_url = ngrok.connect(port).public_url
      5 print("Public URL:", public_url)
      6

 3 frames
/usr/local/lib/python3.12/dist-packages/pyngrok/process.py in _start_process(pyngrok_config)
    445
    446         if ngrok_process.startup_error is not None:
--> 447             raise PyngrokNgrokError(f"The ngrok process errored on start: {ngrok_process.startup_error}.",
    448                                     ngrok_process.logs,
    449                                     ngrok_process.startup_error)

PyngrokNgrokError: The ngrok process errored on start: authentication failed: The authtoken you specified does not look like a proper ngrok authtoken.\nYour authtoken:
YOUR_NGROK_AUTHTOKEN_HERE\nInstructions to install your authtoken are on your ngrok dashboard:\nhttps://dashboard.ngrok.com/get-started/your-authtoken\r\n\r\nERR_NGROK_105\r\n.
```

```
# Colab cell: test calls to the running API
base = public_url
# 1) KB question
r1 = requests.post(base + "/solve", json={"question":"Evaluate integral from 0 to 1 of x^2 dx"})
print("KB question response:", r1.json())

# 2) KB question 2
r2 = requests.post(base + "/solve", json={"question":"Solve 2x^2 - 5x + 2 = 0"})
print("KB question 2 response:", r2.json())

# 3) Web-only (mock MCP)
r3 = requests.post(base + "/solve", json={"question":"Fourier transform of a rectangular pulse"})
print("Web question response:", r3.json())

# 4) Submit feedback to add a corrected solution
fb = requests.post(base + "/feedback", json={"original_question":"Solve 3x-6=0","correction":"3x-6=0 => x=2","user":"tester"})
print("Feedback response:", fb.json())
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/tmp/ipython-input-374399250.py in <cell line: 0>()
      1 # Colab cell: test calls to the running API
----> 2 base = public_url
      3 # 1) KB question
      4 r1 = requests.post(base + "/solve", json={"question":"Evaluate integral from 0 to 1 of x^2 dx"})
      5 print("KB question response:", r1.json())

NameError: name 'public_url' is not defined
```

New

0. Install required packages (run once)

```
#@title 0. Install required packages (run once)
# Run in Colab cell
!pip install -q sentence-transformers faiss-cpu transformers "duckduckgo-search" uvicorn fastapi nest-asyncio pyngrok pydantic python-multipart aiofiles joblib
# Optional: for nicer notebooks
!pip install -q gradio
```

```
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 31.4/31.4 MB 61.1 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.3/3.3 MB 103.8 MB/s eta 0:00:00
```

1. Imports and config

```
#@title 1. Imports and config
import os
import json
import time
import math
import uuid
from typing import List, Dict, Any, Optional
from dataclasses import dataclass, asdict

# Model selection env variables
OPENAI_API_KEY = os.environ.get("OPENAI_API_KEY", None)  # optional
USE_OPENAI = False if OPENAI_API_KEY is None else True

# Similarity threshold for KB match (tuneable)
SIMILARITY_THRESHOLD = 0.70

# For embedding model:
EMBED_MODEL_NAME = "paraphrase-MiniLM-L6-v2"  # small & fast

# For generation fallback model:
GEN_MODEL = "google/flan-t5-small"  # good small text2text; optional GPU speed-up

print("USE_OPENAI:", USE_OPENAI)
```

```
USE_OPENAI: False
```

2. Embedding & FAISS utilities

```
#@title 2. Embedding & FAISS utilities
from sentence_transformers import SentenceTransformer
import numpy as np
import faiss
from joblib import dump, load

# Load embedding model (this will download to Colab cache)
```

```
embed_model = SentenceTransformer(EMBED_MODEL_NAME)

@dataclass
class KBItem:
    id: str
    question: str
    answer: str
    metadata: Dict[str, Any]

class InMemoryFAISS:
    def __init__(self, dim: int):
        self.dim = dim
        self.index = faiss.IndexFlatIP(dim)  # inner product for cosine if vectors normalized
        self.id_map: List[str] = []
        self.vectors = None

    def add(self, vectors: np.ndarray, ids: List[str]):
        # vectors must be normalized
        faiss.normalize_L2(vectors)
        if self.vectors is None:
            self.vectors = vectors.copy()
            self.index.add(vectors)
            self.id_map = ids.copy()
        else:
            self.vectors = np.vstack([self.vectors, vectors])
            self.index.add(vectors)
            self.id_map.extend(ids)

    def search(self, query_vec: np.ndarray, top_k=5):
        faiss.normalize_L2(query_vec)
        D, I = self.index.search(query_vec, top_k)
        # D: similarity scores (inner product), I: row indices
        return D, I

# wrapper class to manage KB items + index
class KnowledgeBase:
    def __init__(self, embed_model):
        self.embed_model = embed_model
        self.dim = embed_model.get_sentence_embedding_dimension()
        self.store = InMemoryFAISS(self.dim)
        self.items: Dict[str, KBItem] = {}

    def add_items(self, items: List[KBItem]):
        texts = [it.question for it in items]
        embeddings = self.embed_model.encode(texts, convert_to_numpy=True)
        # normalize
        # (FAISS wrapper will normalize on add/search)
        ids = [it.id for it in items]
        self.store.add(embeddings, ids)
        for it in items:
            self.items[it.id] = it

    def query(self, text: str, top_k=3):
        emb = self.embed_model.encode([text], convert_to_numpy=True)
        D, I = self.store.search(emb, top_k)
        results = []
```

```
        for dist_row, idx_row in zip(D, I):
            for score, idx in zip(dist_row, idx_row):
                if idx == -1: continue
                kid = self.store.id_map[idx]
                results.append({"id": kid, "score": float(score), "item": self.items[kid]})
        return results

# init empty KB
KB = KnowledgeBase(embed_model)
print("Embedding dim:", KB.dim)
```

```
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as secret in your Google Colab and restart your session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
modules.json: 100%  229/229 [00:00<00:00, 6.38kB/s]
config_sentence_transformers.json: 100%  122/122 [00:00<00:00, 4.12kB/s]
README.md: 3.51k/? [00:00<00:00, 113kB/s]
sentence_bert_config.json: 100%  53.0/53.0 [00:00<00:00, 1.69kB/s]
config.json: 100%  629/629 [00:00<00:00, 14.3kB/s]
model.safetensors: 100%  90.9M/90.9M [00:01<00:00, 81.3MB/s]
tokenizer_config.json: 100%  314/314 [00:00<00:00, 10.4kB/s]
vocab.txt: 232k/? [00:00<00:00, 11.7MB/s]
tokenizer.json: 466k/? [00:00<00:00, 29.7MB/s]
special_tokens_map.json: 100%  112/112 [00:00<00:00, 13.2kB/s]
config.json: 100%  190/190 [00:00<00:00, 25.3kB/s]
Embedding dim: 384
```

3. Seed KB (sample problems)

```
#@title 3. Seed KB (sample problems)
seed = [
    KBItem(
        id=str(uuid.uuid4()),
        question="Find the derivative of f(x) = x^3 + 2x^2 - 5x + 7.",
        answer=("Differentiate term-wise: f'(x) = 3x^2 + 4x - 5. "
                "Step1: d/dx x^3 = 3x^2. Step2: d/dx 2x^2 = 4x. Step3: d/dx -5x = -5."),
        metadata={"source":"seed","topic":"calculus","difficulty":"easy"}
    ),
    KBItem(
        id=str(uuid.uuid4()),
        question="Solve the integral ∫(2x) dx.",
        answer=("∫2x dx = x^2 + C. Steps: Factor 2, integrate x -> x^2/2 then multiply by 2 -> x^2 + C."),
        metadata={"source":"seed","topic":"calculus","difficulty":"easy"}
    ),
```

```
    KBItem(
        id=str(uuid.uuid4()),
        question="If A and B are independent events with P(A)=0.3 and P(B)=0.5, find P(AUB).",
        answer=("For independent events, P(AUB) = P(A)+P(B)-P(A)P(B) = 0.3+0.5-0.15 = 0.65."),
        metadata={"source":"seed","topic":"probability","difficulty":"easy"}
    ),
]

KB.add_items(seed)
print("Seeded KB with", len(seed), "items.")
```

4. LLM generation utilities + MCP (Model Context Protocol) prompt wrapper

```
#@title 4. LLM generation utilities + MCP (Model Context Protocol) prompt wrapper

# We'll use HF flan-t5-small as default LLM (no API key needed).
from transformers import AutoTokenizer, AutoModelForSeq2SeqLM, pipeline

def load_flan_model():
    tok = AutoTokenizer.from_pretrained(GEN_MODEL)
    model = AutoModelForSeq2SeqLM.from_pretrained(GEN_MODEL)
    gen = pipeline("text2text-generation", model=model, tokenizer=tok, device=0 if __import__('torch').cuda.is_available() else -1)
    return gen

# lazy load
GEN_PIPELINE = None
def get_gen_pipeline():
    global GEN_PIPELINE
    if GEN_PIPELINE is None:
        GEN_PIPELINE = load_flan_model()
    return GEN_PIPELINE

def build_mcp_prompt(query: str, retrieved: List[Dict], instructions: str = "", constraints: str = "") -> str:
    """
    Build a structured prompt following a simple MCP pattern.
    """
    header = "MCP-BEGIN\nModel Context Protocol v1.0\n"
    meta = f"Query: {query}\n"
    meta += f"Timestamp: {time.asctime()}\n"
    if instructions:
        meta += f"Model Instructions: {instructions}\n"
    if constraints:
        meta += f"Constraints: {constraints}\n"
    meta += "Retrieved Documents:\n"
    if not retrieved:
        meta += " - None\n"
    else:
        for i, r in enumerate(retrieved,1):
            meta += f" - Doc {i} | score={r.get('score'):.4f} | source={r['item'].metadata.get('source','unknown')}\n   Q: {r['item'].question}\n   A: {r['item'].answer}\n"
    meta += "\nTask: Provide a clear, step-by-step mathematical solution to the Query above. Keep it educational for a student. Include provenance: whether KB or Web and confidence (0-1)
    return header + meta

def generate_solution_via_llm(prompt: str, max_length: int = 512) -> str:
    """
```

```
    Uses the generation pipeline. If OPENAI is enabled and you prefer OpenAI,
    you can put that branch here (not implemented by default).
    """
    pipe = get_gen_pipeline()
    out = pipe(prompt, max_length=max_length, do_sample=False)
    text = out[0]['generated_text']
    return text
```

5. Guardrails: input and output validation (AI Gateway)

```
#@title 5. Guardrails: input and output validation (AI Gateway)
import re

def input_guardrail(user_text: str) -> (bool, str):
    """
    Return (allowed, reason). Reject if text contains disallowed topics or requests.
    We allow: math, equations, problem statements, educational context.
    Disallow: medical diagnosis, legal/political campaigning, personal data extraction, etc.
    """
    blocked_keywords = [
        # high-level examples
        "medical", "diagnos", "suicide", "self-harm", "bomb", "explosive", "attack", "credit card",
        "password", "social security", "ssn", "visa", "passport"
    ]
    low = user_text.lower()
    for kw in blocked_keywords:
        if kw in low:
            return False, f"Input rejected: contains blocked keyword '{kw}'"
    # check for presence of at least some math-like tokens (heuristic)
    math_tokens = re.search(r"[0-9\+\-\*\/\=\^]|integral|derivative|limit|sqrt|log|sin|cos|tan|matrix|determinant", low)
    if not math_tokens:
        # still allow if user adds "math" intent - but prefer math-only
        return False, "Input rejected: please provide a math question or explicitly request mathematical help."
    return True, "OK"

def output_guardrail(solution_text: str) -> (bool, str):
    """
    Checks that output stays educational and not harmful. Also enforce step-by-step format.
    """
    # don't allow personal data leaks (naive)
    if re.search(r"\b\d{3}-\d{2}-\d{4}\b", solution_text):
        return False, "Output rejected: contains protected personal data"
    # ensure we have multiple steps
    steps = re.findall(r"Step\b|\d+\.\s", solution_text, flags=re.IGNORECASE)
    if len(steps) < 1:
        # not strict — allow but flag
        return True, "Warning: solution might not be formatted as step-by-step"
    # ensure no non-educational disclaimers
    if any(w in solution_text.lower() for w in ["illegal", "hack", "exploit", "bomb"]):
        return False, "Output rejected: contains disallowed content"
    return True, "OK"
```

```
!pip install duckduckgo_search --quiet

from duckduckgo_search import DDGS

def web_search_extract(query: str, max_results: int = 3):
    """
    DuckDuckGo Web Search fallback — works with latest DDGS API.
    Returns search title + snippet text.
    """
    results = []

    with DDGS() as ddg:
        for r in ddg.text(query, max_results=max_results):
            results.append({
                "title": r.get("title", ""),
                "href": r.get("href", ""),
                "snippet": r.get("body", "")
            })

    return results


# ✅ Test it
query = "Latest updates on Google's Gemini AI release"
results = web_search_extract(query)

for i, r in enumerate(results, 1):
    print(f"\nResult {i}:")
    print("Title:", r["title"])
    print("URL:", r["href"])
    print("Snippet:", r["snippet"])
```

```
/tmp/ipython-input-101026720.py:12: RuntimeWarning: This package (`duckduckgo_search`) has been renamed to `ddgs`! Use `pip install ddgs` instead.
  with DDGS() as ddg:
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use timez
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use timez
  return datetime.utcnow().replace(tzinfo=utc)
```

7. Routing agent: main solve() logic

```
#@title 7. Routing agent: main solve() logic
def solve_query(user_query: str, top_k:int=3, sim_threshold:float=SIMILARITY_THRESHOLD):
    # 1. Input guardrail
    allowed, reason = input_guardrail(user_query)
    if not allowed:
        return {"status":"blocked","reason": reason}
    # 2. Query KB
    kb_hits = KB.query(user_query, top_k=top_k)
    # Note: FAISS returns inner product on normalized vectors, which approximates cosine similarity in [0,1]
    if kb_hits and kb_hits[0]['score'] >= sim_threshold:
        # Use KB
        chosen = kb_hits[0]
        provenance = {"type":"kb", "matched_question": chosen['item'].question, "score": chosen['score']}
```

```
        # Build MCP prompt
        prompt = build_mcp_prompt(user_query, [chosen], instructions="Use KB answer as primary provenance. Simplify for a student and show steps.")
        sol = generate_solution_via_llm(prompt)
        ok, out_reason = output_guardrail(sol)
        return {"status":"ok","provenance":provenance,"solution":sol,"guardrail_ok":ok,"guardrail_reason":out_reason}
    else:
        # Not found in KB => Web search
        web_docs = web_search_extract(user_query, max_results=4)
        # if empty web_docs -> refuse to hallucinate
        if not web_docs:
            return {"status":"no_web_evidence","reason":"No trustworthy web results found; human review required."}
        # Convert web_docs into KBItem-like objects for MCP context
        retrieved = []
        for w in web_docs:
            fake_kb_item = KBItem(id=str(uuid.uuid4()), question=w['title'] or user_query, answer=w['snippet'], metadata={"source":w['url']})
            retrieved.append({"item": fake_kb_item, "score": 0.0})
        prompt = build_mcp_prompt(user_query, retrieved, instructions="Use web excerpts as evidence. Do not invent. If uncertain, say so and request human review.")
        sol = generate_solution_via_llm(prompt)
        # Determine a simple confidence heuristic: presence of numeric result and "Confidence" line in model output.
        ok, out_reason = output_guardrail(sol)
        provenance = {"type":"web", "sources":[d['url'] for d in web_docs]}
        return {"status":"ok","provenance":provenance,"solution":sol,"guardrail_ok":ok,"guardrail_reason":out_reason}
```

```
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use timez
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use timez
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use timez
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use timez
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use timez
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use timez
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use timez
  return datetime.utcnow().replace(tzinfo=utc)
```

8. FastAPI app: endpoints for solve & feedback

```
#@title 8. FastAPI app: endpoints for solve & feedback
# Run in Colab: use nest_asyncio + pyngrok or ngrok for public URL if needed.
from fastapi import FastAPI, HTTPException
from pydantic import BaseModel
import nest_asyncio
from pyngrok import ngrok
import threading
import uvicorn

app = FastAPI()

class SolveRequest(BaseModel):
    query: str

class FeedbackRequest(BaseModel):
```

```
    query: str
    solution_text: str
    verdict: str  # 'correct'/'incorrect'/'partial'
    reviewer_notes: Optional[str] = None

# in-memory feedback store
FEEDBACK_STORE = []

@app.post("/solve")
async def api_solve(req: SolveRequest):
    res = solve_query(req.query)
    return res

@app.post("/feedback")
async def api_feedback(req: FeedbackRequest):
    # store feedback
    rec = {"id":str(uuid.uuid4()), "query":req.query, "solution":req.solution_text, "verdict":req.verdict, "notes":req.reviewer_notes, "ts":time.time()}
    FEEDBACK_STORE.append(rec)
    # If verdict == correct, add to KB
    if req.verdict == "correct":
        kb_item = KBItem(id=str(uuid.uuid4()), question=req.query, answer=req.solution_text, metadata={"source":"human_feedback"})
        KB.add_items([kb_item])
    return {"status":"ok","stored":rec}

@app.get("/health")
async def health():
    return {"status":"ok","kb_size":len(KB.items)}

# start the server (call this cell to run)
def run_server(port=8000):
    nest_asyncio.apply()
    # expose via ngrok for demo
    public_url = ngrok.connect(port).public_url
    print("ngrok public url:", public_url)
    uvicorn.run(app, host="0.0.0.0", port=port)

# To run in Colab: uncomment the next lines
# import threading
# t = threading.Thread(target=run_server, kwargs={"port":8000}, daemon=True)
# t.start()
# print("Server started in background. Call /solve to test.")
```

```
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
```

```
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
```

11. Sample queries

```
#@title 11. Sample queries
kb_queries = [
  "Find the derivative of f(x) = x^3 + 2x^2 - 5x + 7.",
  "Solve the integral ∫(2x) dx.",
  "If A and B are independent events with P(A)=0.3 and P(B)=0.5, find P(AUB)."
]

web_queries = [
  "What is the closed form for the sum 1^3 + 2^3 + ... + n^3?",
  "Show step-by-step how to find the eigenvalues of the matrix [[2,1],[1,2]].",
  "Prove that derivative of sin(x) is cos(x) using first principles."
]
```

```
print("KB queries (should hit KB):")
for q in kb_queries:
    print(q)

print("\nWeb fallback queries (likely not in KB):")
for q in web_queries:
    print(q)
```

```
KB queries (should hit KB):
Find the derivative of f(x) = x^3 + 2x^2 - 5x + 7.
Solve the integral ∫(2x) dx.
If A and B are independent events with P(A)=0.3 and P(B)=0.5, find P(A∪B).

Web fallback queries (likely not in KB):
What is the closed form for the sum 1^3 + 2^3 + ... + n^3?
Show step-by-step how to find the eigenvalues of the matrix [[2,1],[1,2]].
Prove that derivative of sin(x) is cos(x) using first principles.
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use timez
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use timez
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use timez
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use timez
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use timez
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use timez
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use timez
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use timez
  return datetime.utcnow().replace(tzinfo=utc)
```

12. Quick test (run in Colab cell)

```
#@title 12. Quick test (run in Colab cell)
test_q = "Find the derivative of f(x) = x^3 + 2x^2 - 5x + 7."
print("Query:", test_q)
res = solve_query(test_q)
print(json.dumps(res, indent=2))
```

```
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
```

```
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
Query: Find the derivative of f(x) = x^3 + 2x^2 - 5x + 7.
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
```

13. JEE Bench skeleton: how to run benchmarking

```
#@title 13. JEE Bench skeleton: how to run benchmarking
# Expectation: you will provide a JEE Bench dataset (CSV/JSON with columns: id, question, answer)
# This skeleton demonstrates how to iterate test set and compute simple accuracy via human-labeled answers
import csv
def evaluate_on_dataset(dataset_path, sample_limit=100):
    # load dataset: minimal expected format
    with open(dataset_path, 'r', encoding='utf8') as f:
        data = json.load(f)
    results = []
    for i, ex in enumerate(data):
        if i >= sample_limit: break
        q = ex['question']
        ground = ex.get('answer')
```

```
        out = solve_query(q)
        # naive exact-match or manual review required — here we store output & provenance for human review
        results.append({"id":ex.get('id',i),"query":q,"generated":out.get('solution'),"provenance":out.get('provenance')})
    # Save results
    with open('benchmark_results.json','w',encoding='utf8') as fo:
        json.dump(results, fo, indent=2)
    return results

# Example usage: prepare JEE bench JSON (not included). Call evaluate_on_dataset('/content/jeebench.json')
```

```
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use timez
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use timez
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use timez
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use timez
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use timez
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use timez
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use timez
  return datetime.utcnow().replace(tzinfo=utc)
```

14. Export a minimal PDF proposal (basic)

```
#@title 14. Export a minimal PDF proposal (basic)
# This writes a basic report summarizing architecture to a PDF using reportlab
!pip install -q reportlab
from reportlab.lib.pagesizes import letter
from reportlab.pdfgen import canvas

def write_proposal_pdf(filename="MathAgent_Proposal.pdf"):
    c = canvas.Canvas(filename, pagesize=letter)
    text = c.beginText(40, 700)
    text.setFont("Helvetica", 11)
    text.textLine("Math Routing Agent - Final Proposal")
    text.textLine("")
    text.textLine("1. Input & Output Guardrails: Implemented via input_guardrail() and output_guardrail()")
    text.textLine("2. Knowledge Base: FAISS (in-memory) with paraphrase-MiniLM-L6-v2 embeddings")
    text.textLine("3. MCP: build_mcp_prompt() creates Model Context Protocol wrapper passed to LLM.")
    text.textLine("4. Web Search: duckduckgo_search used for fallback search.")
    text.textLine("5. Human-in-loop: /feedback endpoint appends validated answers to KB.")
    text.textLine("")
    text.textLine("Sample KB questions included; sample web queries provided. See notebook for full details and code.")
    c.drawText(text)
    c.showPage()
    c.save()
    return filename

pdf_file = write_proposal_pdf()
print("Generated PDF:", pdf_file)
```

```
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
```

Start coding or generate with AI.

Streaming output truncated to the last 5000 lines.

```
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use tim
```