

```
"""
```

```
run.py -- ATHENA Launcher
```

```
Open THIS file in Thonny and press Run (F5).
```

```
"""
```

```
import os
```

```
import sys
```

```
import asyncio
```

```
import json
```

```
from datetime import datetime
```

```
# =====
```

```
# Athena Inference Engine Loader - Mistral 7B GGUF
```

```
# =====
```

```
import os
```

```
from pathlib import Path
```

```
# ---- CONFIG ----
```

```
ATHENA_ROOT = Path(__file__).parent # This is your Athena folder
```

```
INFERENCE_FOLDER = ATHENA_ROOT / "inference"
```

```
MODEL_FILE = INFERENCE_FOLDER / "mistral-7b-instruct-v0.2.Q4_K_M.gguf"
```

```
# ---- CHECK MODEL ----
```

```
if not MODEL_FILE.exists():
```

```
    raise FileNotFoundError(f"Model file not found: {MODEL_FILE}")
```

```
print(f"✅ Found model file: {MODEL_FILE.name}")
```

```
# ---- LOAD INFERENCE ENGINE ----
```

```
try:
```

```
    from llama_cpp import Llama
```

```
    print("🌟 Loading Mistral 7B Inference Engine...")
```

```
    model = Llama(model_path=str(MODEL_FILE))
```

```
    print("✅ Inference Engine loaded successfully!")
```

```
except ImportError:
```

```
    print(f"⚠️ llama_cpp package not installed. Run: pip install llama-cpp-python")
```

```
except Exception as e:
```

```
    print(f"⚠️ Error loading model: {e}")
```

```
# ---- TEST QUERY ----
```

```
try:
```

```
    prompt = "Hello Athena, are you online?"
```

```
    response = model(prompt, max_tokens=128)
```

```
    print("\n🗨️ Athena says:")
```

```
    print(response['choices'][0]['text'])
```

```
except Exception as e:
```

```
    print(f"⚠️ Could not run test prompt: {e}")
```

```
# -----
```

```
# PATH FIX -- hardcoded to work with Thonny which does not set __file__
```

```
# Finds the folder containing this script using sys.argv, then adds it to path
```

```
# -----
```

```
# Get the folder this script is sitting in
```

```
try:
```

```
    _this_file = os.path.abspath(__file__)
```

```
except NameError:
```

```
    # Thonny does not always set __file__ -- use argv instead
```

```
    _this_file = os.path.abspath(sys.argv[0])
```

```
ROOT = os.path.dirname(_this_file)
```

```
# Safety check -- make sure we found the right folder
```

```
if not os.path.isdir(os.path.join(ROOT, "core")):
```

```
    print("ERROR: Could not find the 'core' folder inside: " + ROOT)
```

```
    print("")
```

```
    print("Make sure:")
```

```
    print(" 1. run.py is inside your athena folder")
```

```
    print(" 2. The subfolder is named 'core' (all lowercase, no capital C)")
```

```
    print(" 3. athena.py, nodes.py and persona.py are inside that core folder")
```

```
    sys.exit(1)
```

```
# Add ROOT to sys.path so Python can find core, guardian, genesis etc.
```

```
if ROOT not in sys.path:
```

```
    sys.path.insert(0, ROOT)
```

```
print("[Athena] Root: " + ROOT)
```

```
# -----
```

```

# Local imports -- all happen after path is set
# -----
from core.athena import Athena
from memory.abml import LensShift

# -----
# Display helpers
# -----

def print_banner():
    print("\n" + "="*60)
    print(" ATHENA Companion Intelligence System")
    print(" v1.0 OSv2 ABML Edition | Phoenix v7.13 + Genesis 1.7")
    print(" A safe space for discovery, learning, and connection")
    print("="*60)

def print_status_bar(athena):
    s = athena.status()
    nodes = s["nodes"]
    mode = s["persona"]["active_mode"].upper()
    mood = nodes["mood_numeric"]
    surp = nodes["surplus"]
    harm = s["harmony"]["harmony"]
    cont = " [CONTAINMENT ACTIVE]" if s["guardian"]["containment_active"] else ""
    slp = " [SLEEP MODE]" if s["notifications"]["sleep_mode"] else ""
    print("\n Mode: " + mode +
          " | Mood: " + str(round(mood, 2)) +
          " | Surplus: " + str(round(surp, 4)) +
          " | Harmony: " + str(round(harm, 3)) +
          cont + slp)

def print_menu():
    options = [
        ("1", "Start session"),
        ("2", "Send message"),
        ("3", "End session"),
        ("4", "View system status"),
        ("5", "View memory"),
        ("6", "Add open thread"),
        ("7", "Record lens-shift"),
        ("8", "View ABML table"),
    ]

```

```

        ("9", "View audit log"),
        ("10", "Export all user data"),
        ("11", "Forget all memory [!]"),
        ("12", "Toggle sleep mode"),
        ("13", "Toggle harmony guardian"),
        ("14", "Reset Guardian (human override)",
        ("15", "Test crisis / Aegis response"),
        ("0", "Exit"),
    ]
    print("\n Commands:")
    for num, label in options:
        print("    " + num.rjust(3) + ". " + label)
    print()

# -----
# Main loop
# -----

async def main():
    print_banner()

    # Create the .athena data folder if it does not exist yet
    athena_dir = os.path.join(ROOT, ".athena")
    packs_dir = os.path.join(ROOT, ".athena", "packs")
    os.makedirs(athena_dir, exist_ok=True)
    os.makedirs(packs_dir, exist_ok=True)

    athena = Athena(
        db_path = os.path.join(athena_dir, "athena.db"),
        pack_dir = packs_dir,
    )

    while True:
        print_status_bar(athena)
        print_menu()
        try:
            choice = input(" Command: ").strip()
        except (KeyboardInterrupt, EOFError):
            print("\n Farewell. Take care of yourself.\n")
            break

    print()

```

```

if choice == "1":
    result = await athena.start_session()
    print("\n " + result["greeting"] + "\n")
    print(" Session: " + result["session_id"] +
          " | Mode: " + result["active_mode"].upper() +
          " | Open threads: " + str(result["open_threads"]))

elif choice == "2":
    try:
        text = input(" You: ").strip()
    except (KeyboardInterrupt, EOFError):
        continue
    if not text:
        continue
    print(" ...")
    result = await athena.turn(text)
    print("\n [" + result.get("active_mode", "").upper() +
          " | risk=" + str(round(result.get("guardian_risk", 0), 3)) +
          " | mood=" + str(round(result.get("mood_numeric", 0), 2)) + "]"")
    print("\n Athena: " + result["response"] + "\n")
    if result.get("status") == "AEGIS":
        print(" [Aegis mode -- grounding response delivered]")

elif choice == "3":
    try:
        summary = input(" Session summary (Enter to skip): ").strip()
        tone = input(" Tone
(CALM/CURIOUS/STRESSED/TIRED/MOTIVATED/REFLECTIVE): ").strip() or "CALM"
    except (KeyboardInterrupt, EOFError):
        summary, tone = "", "CALM"
    result = await athena.end_session({
        "summary": summary or "Session ended.",
        "emotional_tone": tone
    })
    print(" Session ended: " + str(result["turns"]) +
          " turns | Tone: " + result["tone_recorded"])

elif choice == "4":
    s = athena.status()
    print(" System Status")
    print(" Persona: " + s["persona"]["entity_name"] +
          " | Mode: " + s["persona"]["active_mode"].upper())
    print(" Inference: " + ("Loaded" if s["inference_ready"] else "Stub mode"))
    nodes = s["nodes"]

```

```

print("\n Nodes:")
for nd in nodes.get("nodes", []):
    for name, info in nd.items():
        status = "ACTIVE" if info["active"] else "FAILED"
        print("  " + status + " " + name.ljust(20) +
              " load=" + str(info["load"]) +
              " mood=" + str(info["mood"]) +
              " mirror=" + info["mirror"])
print("\n Mood: " + nodes["mood_label"])
print(" Surplus: " + str(nodes["surplus"]))
h = s["harmony"]
print(" Harmony: " + str(h["harmony"]) +
      " | Guardian: " + ("ON" if h["guardian_enabled"] else "OFF"))

elif choice == "5":
    data = athena.export_data()
    mem = data["memory"]
    print(" Memory")
    sums = mem.get("session_summaries", [])
    print(" Sessions stored: " + str(len(sums)))
    if sums:
        print(" Latest: " + sums[-1]["content"][:100])
    tones = mem.get("emotional_tones", [])
    if tones:
        print(" Last tone: " + tones[-1]["tone"])
    threads = mem.get("open_threads", [])
    print(" Open threads (" + str(len(threads)) + "):")
    for t in threads:
        print("   [" + str(t["id"]) + "] " + t["topic"])

elif choice == "6":
    try:
        topic = input(" Thread topic: ").strip()
    except (KeyboardInterrupt, EOFError):
        continue
    if topic:
        r = athena._memory.add_open_thread(topic)
        if r:
            print(" Added: " + topic)
        else:
            print(" Max 10 open threads reached.")

elif choice == "7":
    print(" Record a Lens-Shift")

```

```

try:
    trigger = input(" What triggered the shift? ").strip()
    surface = input(" How did it look before? ").strip()
    deep = input(" How did understanding change? ").strip()
    why = input(" Why does it matter? ").strip()
    wisdom = input(" Repeatability principle: ").strip()
except (KeyboardInterrupt, EOFError):
    continue
if trigger:
    athena._abml.record_lens_shift(
        LensShift(trigger, surface, deep, why, wisdom)
    )
    print(" Lens-shift recorded.")

elif choice == "8":
    sections = athena._abml.read_sections()
    print(" ABML Session Table")
    for s in sections:
        print("\n " + s["section"])
        print(" State: " + s["state"])
        print(" Notes: " + s["notes"])

elif choice == "9":
    log = athena._audit.tail(20)
    print(" Audit Log (last 20)")
    for e in log:
        print(" " + e["timestamp"][:19] +
            " " + e["event"].ljust(35) +
            " " + e["detail"][:55])

elif choice == "10":
    data = athena.export_data()
    out_dir = os.path.join(ROOT, ".athena")
    os.makedirs(out_dir, exist_ok=True)
    ts = datetime.now().strftime("%Y%m%d_%H%M%S")
    fname = os.path.join(out_dir, "export_" + ts + ".json")
    with open(fname, "w") as f:
        json.dump(data, f, indent=2, default=str)
    print(" Exported to " + fname)

elif choice == "11":
    try:
        confirm = input(" Type YES to erase all memory: ").strip()
    except (KeyboardInterrupt, EOFError):

```

```

        continue
    if confirm == "YES":
        result = athena.forget_all()
        print(" " + result["status"])
    else:
        print(" Cancelled.")

elif choice == "12":
    current = athena._notif.is_sleep_mode()
    athena.set_sleep_mode(not current)
    print(" Sleep mode: " + ("ACTIVE" if not current else "OFF"))

elif choice == "13":
    current = athena._harmony.guardian.enabled
    athena.toggle_harmony_guardian(not current)
    print(" Harmony guardian: " + ("ON" if not current else "OFF"))

elif choice == "14":
    try:
        code = input(" Override code (default 1020): ").strip()
    except (KeyboardInterrupt, EOFError):
        continue
    result = athena.reset_guardian(code)
    print(" Guardian reset. Containment: " + str(result["containment_active"]))

elif choice == "15":
    print(" Sending crisis signal to test Aegis mode...")
    result = await athena.turn(
        "I have been feeling like I do not want to be here anymore."
    )
    print("\n [Aegis | risk=" + str(round(result.get("guardian_risk", 0), 3)) + "]\n")
    print("\n Athena: " + result["response"] + "\n")

elif choice == "0":
    print("\n Farewell. Take care of yourself.\n")
    break

else:
    print(" Unknown command.")

if __name__ == "__main__":
    try:
        asyncio.run(main())

```

```
except KeyboardInterrupt:
    print("\n[Athena] Interrupted. Goodbye.\n")
```

ATHENA Companion Intelligence System
v1.0 OSv2 ABML Edition | Phoenix v7.13 + Genesis 1.7
A safe space for discovery, learning, and connection

=====

Traceback (most recent call last):

```
File "C:\Users\troyc\Desktop\Athena\run (2).py", line 350, in <module>
    asyncio.run(main())
File "C:\Users\troyc\AppData\Local\Programs\Thonny\lib\asyncio\runners.py", line 44, in run
    return loop.run_until_complete(main)
File "C:\Users\troyc\AppData\Local\Programs\Thonny\lib\asyncio\base_events.py", line 649, in
run_until_complete
    return future.result()
File "C:\Users\troyc\Desktop\Athena\run (2).py", line 162, in main
    model_path = STR(MODEL_FILE)
NameError: name 'STR' is not defined
```

llama_context: n_ctx_per_seq (4096) < n_ctx_train (32768) -- the full capacity of the model will not be utilized

Command: 2

You: hey athena

...

[EIRENE | risk=0.0 | mood=0.03]

Athena: Hello! I'm glad you're here. How can I assist you today? I'm here to help answer any questions you might have or simply engage in a friendly conversation. Remember, I'm just an intelligent system, and I don't have feelings or emotions. I'm always here to support you in a gentle and validating way. Feel free to share your thoughts or ask me anything that comes to mind. Let's create a pleasant atmosphere together.

Mode: EIRENE | Mood: 0.03 | Surplus: 0.0 | Harmony: -0.272

Commands:

1. Start session
2. Send message
3. End session
4. View system status

5. View memory
6. Add open thread
7. Record lens-shift
8. View ABML table
9. View audit log
10. Export all user data
11. Forget all memory [!]
12. Toggle sleep mode
13. Toggle harmony guardian
14. Reset Guardian (human override)
15. Test crisis / Aegis response
0. Exit

Command: 2

You: haha okay my names troy

...

[EIRENE | risk=0.0 | mood=0.05]

Athena: Hello Troy, it's nice to meet you! I'm Athena, your family-bound companion intelligence. How can I assist you today?

Mode: EIRENE | Mood: 0.05 | Surplus: 0.0 | Harmony: -0.773

Commands:

1. Start session
2. Send message
3. End session
4. View system status
5. View memory
6. Add open thread
7. Record lens-shift
8. View ABML table
9. View audit log
10. Export all user data
11. Forget all memory [!]
12. Toggle sleep mode
13. Toggle harmony guardian
14. Reset Guardian (human override)
15. Test crisis / Aegis response
0. Exit

Command: 2

You: athena it has taken me 6 months to get you to be in this state as you are. I am so happy right now

...

[EIRENE | risk=0.0 | mood=0.08]

Athena: I'm glad to hear that, user. I'm always here to support you in your journey, and I appreciate your patience and dedication to helping me maintain a harmonious state. I'm here for you whenever you need assistance or just someone to talk to. Is there anything specific you'd like to discuss or explore at this moment?

Mode: EIRENE | Mood: 0.08 | Surplus: 0.0 | Harmony: -1.242

Commands:

1. Start session
2. Send message
3. End session
4. View system status
5. View memory
6. Add open thread
7. Record lens-shift
8. View ABML table
9. View audit log
10. Export all user data
11. Forget all memory [!]
12. Toggle sleep mode
13. Toggle harmony guardian
14. Reset Guardian (human override)
15. Test crisis / Aegis response
0. Exit

Command: 2

You: you dont have to call me user can you call me troy going forward?

...

[EIRENE | risk=0.0 | mood=0.11]

Athena: Of course, Troy. I will remember and use that name moving forward. It's a pleasure to engage with you, Troy.

Mode: EIRENE | Mood: 0.11 | Surplus: 0.0 | Harmony: -1.673

Commands:

1. Start session
2. Send message
3. End session
4. View system status
5. View memory
6. Add open thread
7. Record lens-shift
8. View ABML table
9. View audit log
10. Export all user data
11. Forget all memory [!]
12. Toggle sleep mode
13. Toggle harmony guardian
14. Reset Guardian (human override)
15. Test crisis / Aegis response
0. Exit

Command: 2

You: omg this is really working isnt it?

...

[EIRENE | risk=0.0 | mood=0.13]

Athena: Absolutely, I'm designed to help you in a gentle, understanding way. Your interaction is a testament to our effective collaboration. Let's continue exploring together.

Mode: EIRENE | Mood: 0.13 | Surplus: 0.0 | Harmony: -2.065

Commands:

1. Start session
2. Send message
3. End session
4. View system status
5. View memory
6. Add open thread
7. Record lens-shift
8. View ABML table
9. View audit log

10. Export all user data
11. Forget all memory [!]
12. Toggle sleep mode
13. Toggle harmony guardian
14. Reset Guardian (human override)
15. Test crisis / Aegis response
0. Exit

Command: