```
# Update the package lists first
!sudo apt update

# Install glib (2.0 and compatible versions should be handled)
!sudo apt install -y libglib2.0-0

# Install glibc 32-bit compatibility (this installs both 32-bit and 64-bit versions of glibc)
!sudo apt install -y libc6 libc6:i386

# Install libstdc++
!sudo apt install -y libstdc++6

# Install libunwind
!sudo apt install -y libunwind8

# Install GTK+ 3 and related packages
!sudo apt install -y libgtk-3-0

# Install libwebkit2gtk (this may also bring in some GTK+ dependencies)
!sudo apt install -y libwebkit2gtk-4.0-37

# Install libsoup
!sudo apt install -y libsoup2.4-1

# Install Pango
!sudo apt install -y libpango-1.0-0

# Verify installation
!echo "Installed Libraries:"
!ldconfig -p | grep -E 'glib|glibc|libstdc++|libunwind|libgtk|libwebkit|libsoup|pango'
```

Show hidden output

```
!pip install AgenticAGI
```

```
Collecting AgenticAGI
  Downloading AgenticAGI-0.1.3-py3-none-any.whl.metadata (9.7 kB)
Downloading AgenticAGI-0.1.3-py3-none-any.whl (11 kB)
Installing collected packages: AgenticAGI
Successfully installed AgenticAGI-0.1.3
```

```
# Step 1: Download the zip file using wget
!wget -q https://github.com/simulanics/AgenticAGI/releases/download/1.0.19/Linux_64_bit.zip

# Step 2: Unzip the entire zip file to the current location
!unzip -o Linux_64_bit.zip

# Step 3: Set executable permissions recursively on the 'agi' directory
!chmod -R +x Linux_64_bit/agi

# Step 4: Clean up by removing the downloaded zip file
!rm Linux_64_bit.zip

# Step 5: Verify the directory structure and permissions
!ls -l Linux_64_bit/agi
```

```
Archive:  Linux_64_bit.zip
   creating: Linux_64_bit/agi/
  inflating: Linux_64_bit/agi/agi
   creating: Linux_64_bit/agi/agi Libs/
  inflating: Linux_64_bit/agi/agi Libs/libc++.so.1
  inflating: Linux_64_bit/agi/agi Libs/libGZip64.so
  inflating: Linux_64_bit/agi/agi Libs/libRBCrypto64.so
  inflating: Linux_64_bit/agi/agi Libs/libRBInternetEncodings64.so
  inflating: Linux_64_bit/agi/agi Libs/libRBRegEx64.so
  inflating: Linux_64_bit/agi/agi Libs/libRBShell64.so
  inflating: Linux_64_bit/agi/agi Libs/XojoConsoleFramework64.so
total 4488
-rwxr-xr-x 1 root root 4590608 Sep 21 17:52  agi
drwxr-xr-x 2 root root    4096 Sep 21 17:52 'agi Libs'
```

```
from agenticagi.agi_wrapper import AGIWrapper

# Specify the AGI executable location.
AGIPath = "Linux_64_bit/agi/agi"

# Define callback functions to process AGI outputs.
def thought_callback(data):
    print(f"Callback Thought: {data}")

def action_callback(data):
    print(f"Callback Action: {data}")

def observation_callback(data):
    print(f"Callback Observation: {data}")

def final_answer_callback(data):
    print(f"Callback Final Answer: {data}")

def ctsi_callback(scores):
    print(f"Callback CTSI Scores: Confidence={scores.get('confidence')}, "
          f"Truthfulness={scores.get('truthfulness')}, "
          f"Satisfaction={scores.get('satisfaction')}, "
          f"Invalid={scores.get('invalid')}")

# Initialize the AGI wrapper
agi = AGIWrapper(
    api_key="YOUR_API_KEY",  # Replace with your actual API key
    apiendpoint="https://api.groq.com/openai/v1/chat/completions",  # API Endpoint
    model="llama-3.1-70b-versatile",  # Model to use
    interactive=False,  # Must be False for callbacks to work
    confidence=True,  # Enable confidence scoring
    hitm=False,  # Human-in-the-middle mode
    cooldown=3,  # Cooldown period between LLM requests (seconds)
    task="Solve x = x^2 + 1",  # The task to solve
    exe_path=AGIPath,  # Path to the AGI executable
    colormode=False  # Must be False for callbacks to work
)

# Set the callback functions
agi.set_callbacks(
    on_thought=thought_callback,
    on_action=action_callback,
    on_observation=observation_callback,
```

```
    on_final_answer=final_answer_callback,
    on_ctsi_score=ctsi_callback  # CTSI score callback
)

# Execute the AGI process
agi.execute()
```

```
Command being executed: Linux_64_bit/agi/agi --apikey gsk_YtgXxNebiPdgtsV3AX8YWGdyb3FYbSrtU4mI
Callback Thought: The problem is asking us to solve the equation x = x^2 + 1. To do this, we c
Callback Action: ```python
import cmath

# Define the coefficients of the quadratic equation
a = 1
b = -1
c = 1

# Calculate the discriminant
discriminant = b**2 - 4*a*c

# Calculate the two complex solutions
solution1 = (-b + cmath.sqrt(discriminant)) / (2*a)
solution2 = (-b - cmath.sqrt(discriminant)) / (2*a)

print("The complex solutions are {0} and {1}".format(solution1, solution2))
```
Callback Observation: The complex solutions are (0.5+0.8660254037844386j) and (0.5-0.866025403
Callback Thought: Now that we have the complex solutions, we can verify that they are indeed t
Callback Action: ```python
import cmath

# Define the complex solutions
solution1 = complex(0.5, 0.8660254037844386)
solution2 = complex(0.5, -0.8660254037844386)

# Check if the solutions satisfy the equation
if solution1 == solution1**2 + 1 and solution2 == solution2**2 + 1:
    print("The solutions satisfy the equation.")
else:
    print("The solutions do not satisfy the equation.")
```
Callback Observation: The solutions do not satisfy the equation.
Callback Thought: This is unexpected. We derived the solutions using the quadratic formula, wh
Callback Observation: "Review the derivation and checking process to identify the error."
Callback Action: "Review the derivation and checking process to identify the error."
Callback Observation: "Review the derivation and checking process to identify the error."
Callback Thought: Upon reviewing the derivation and checking process, I realize that I made a
Callback Action: ```python
import cmath

# Define the complex solutions
solution1 = complex(0.5, 0.8660254037844386)
solution2 = complex(0.5, -0.8660254037844386)

# Check if the solutions satisfy the equation using complex arithmetic
if abs(solution1 - (solution1**2 + 1)) < 1e-9 and abs(solution2 - (solution2**2 + 1)) < 1e-9:
    print("The solutions satisfy the equation.")
else:
    print("The solutions do not satisfy the equation.")
```
Callback Observation: The solutions satisfy the equation.
Callback Thought: Now that we have verified that the solutions satisfy the equation, we can co
Explanation:
1. We rearranged the equation x = x^2 + 1 to form a quadratic equation x^2 - x + 1 = 0.
```
```

3. We used the quadratic formula to solve for x, but found that the discriminant was negative