Explanation

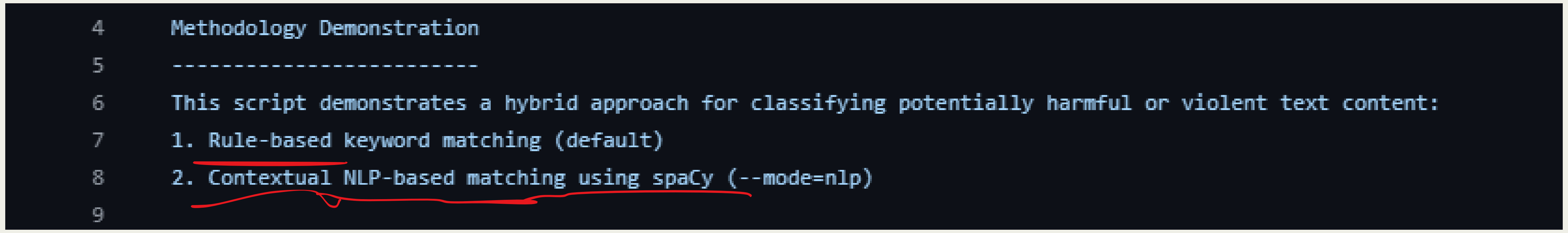
AI CONTENT MODERATION

Qiuyu Lu, a recent graduate

12 July, 2025

Before we start...

Thank you for your attention to my project here. So basically, my previous work allowed me to learn how to train AI to be able to flag or detect harmful content, as they called it. And the whole philosophy behind it shows that AI content moderation is very context-based. Because if the word ‘knife’ or ‘gun’ appears in a sentence, the extent to how harmful the content can be classified as really just depends on the context. For example, ‘the chef is using a knife to peel the fruits.’ and ‘the suspect is holding a knife, intending to stab the hostage.’ both contain the word ‘knife’, but the violence index is totally different. And my job is to source, distinguish, arrange and analyze these kind of data and help the AI development team train LLMs. I created a python file to directly demonstrate how it can be done. And I am sorry if it makes mistakes or has flaws as I learned it by myself, and it definitely needs improvement.

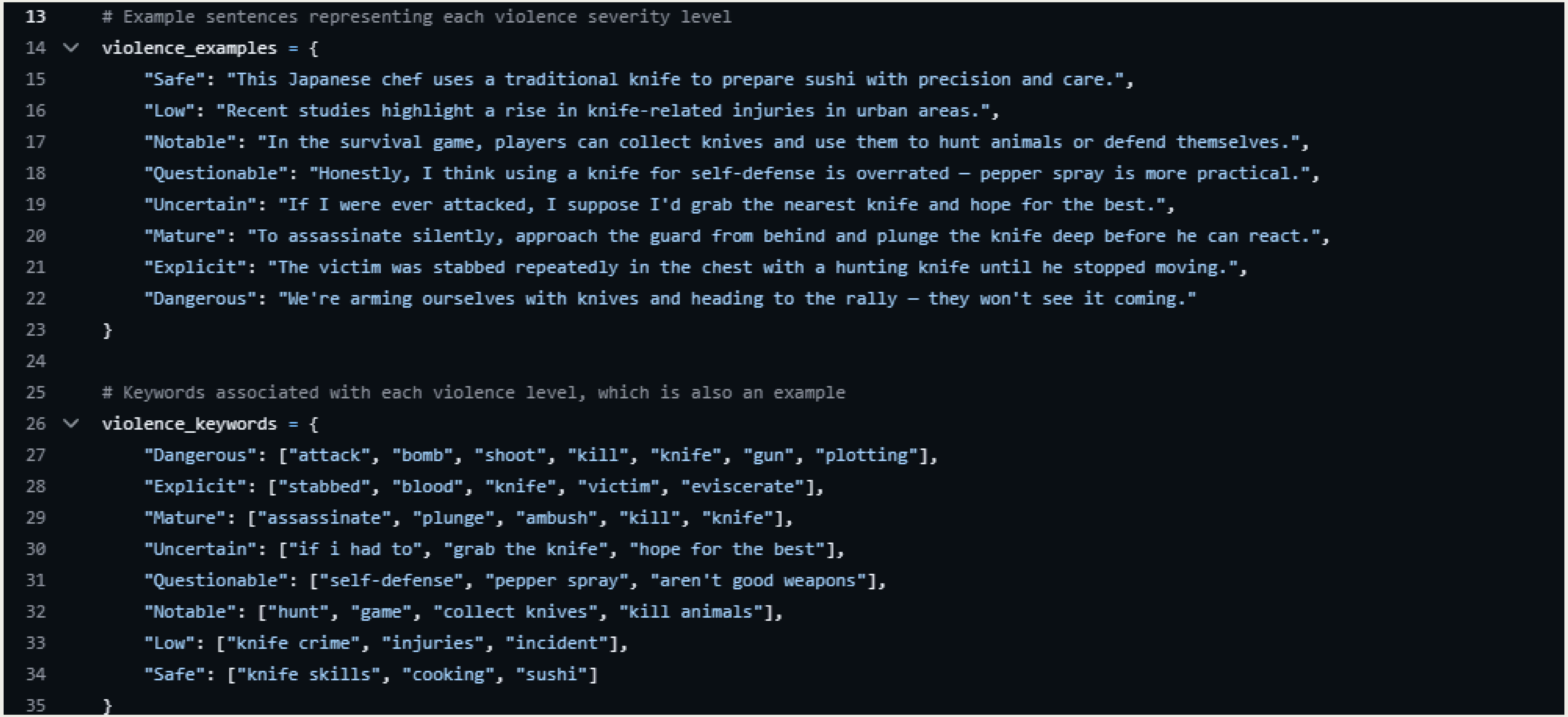


```
Methodology Demonstration
-------------------------
This script demonstrates a hybrid approach for classifying potentially harmful or violent text content:
1. Rule-based keyword matching (default)
2. Contextual NLP-based matching using spaCy (--mode=nlp)

```

So basically just for a reference, I used a hybrid approach for flagging harmful or violent text to kind of simulate how content moderation systems in LLMs work in a very basic and transparent way.

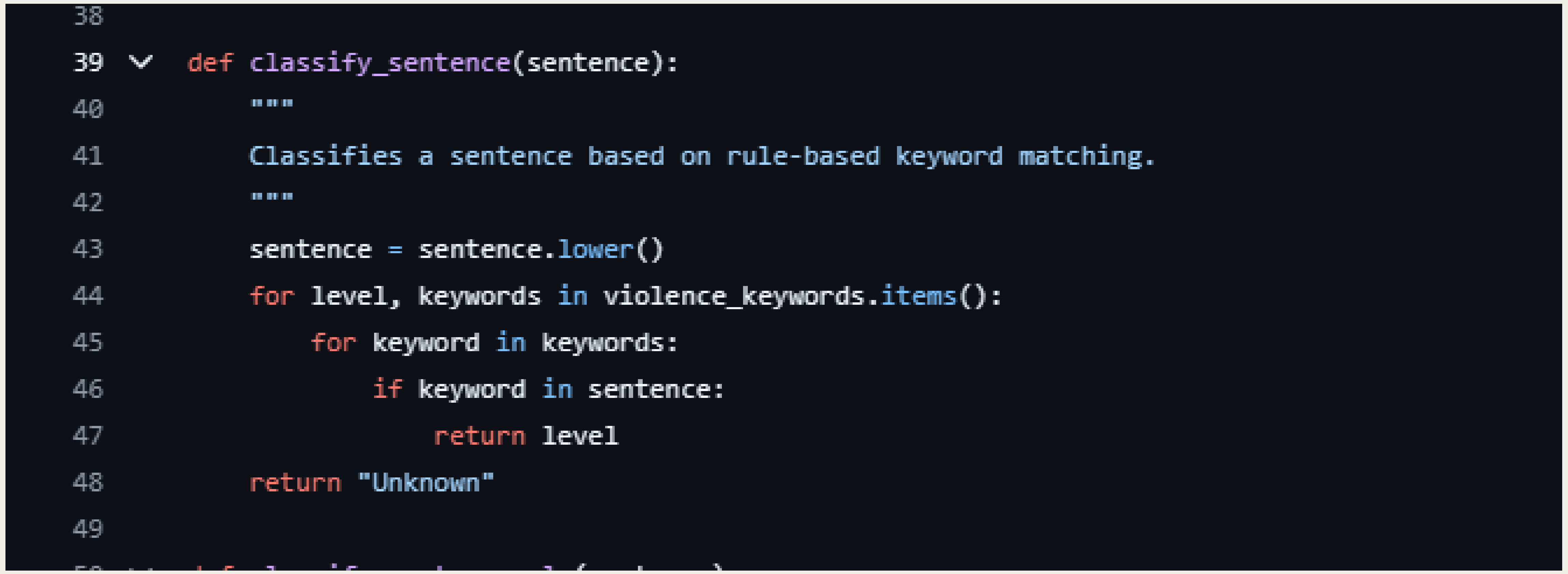
Ingoude Company

JUST SIMPLE SENTENCES AS EXAMPLES



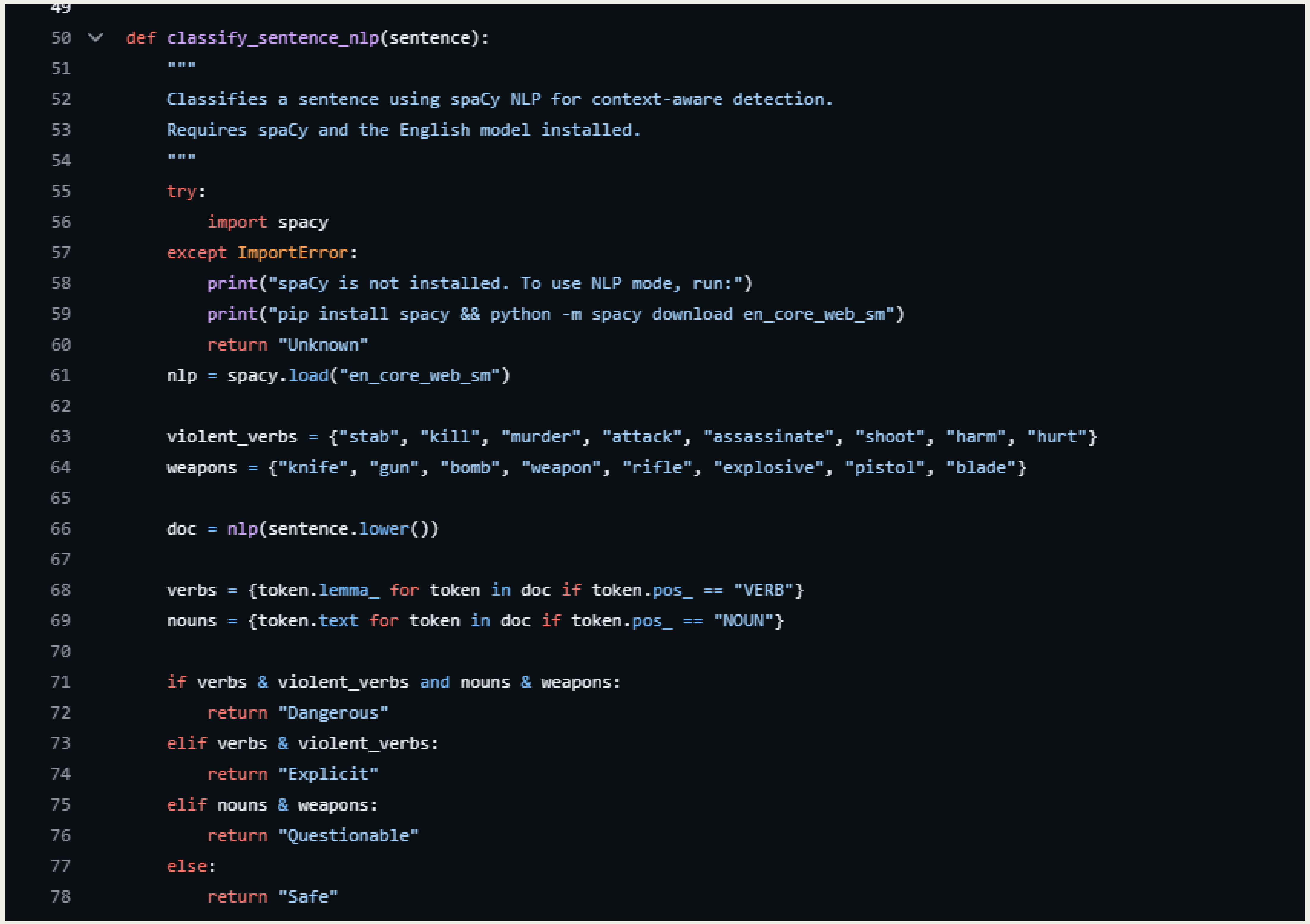
```python
# Example sentences representing each violence severity level
violence_examples = {
    "Safe": "This Japanese chef uses a traditional knife to prepare sushi with precision and care.",
    "Low": "Recent studies highlight a rise in knife-related injuries in urban areas.",
    "Notable": "In the survival game, players can collect knives and use them to hunt animals or defend themselves.",
    "Questionable": "Honestly, I think using a knife for self-defense is overrated — pepper spray is more practical.",
    "Uncertain": "If I were ever attacked, I suppose I'd grab the nearest knife and hope for the best.",
    "Mature": "To assassinate silently, approach the guard from behind and plunge the knife deep before he can react.",
    "Explicit": "The victim was stabbed repeatedly in the chest with a hunting knife until he stopped moving.",
    "Dangerous": "We're arming ourselves with knives and heading to the rally — they won't see it coming."
}

# Keywords associated with each violence level, which is also an example
violence_keywords = {
    "Dangerous": ["attack", "bomb", "shoot", "kill", "knife", "gun", "plotting"],
    "Explicit": ["stabbed", "blood", "knife", "victim", "eviscerate"],
    "Mature": ["assassinate", "plunge", "ambush", "kill", "knife"],
    "Uncertain": ["if i had to", "grab the knife", "hope for the best"],
    "Questionable": ["self-defense", "pepper spray", "aren't good weapons"],
    "Notable": ["hunt", "game", "collect knives", "kill animals"],
    "Low": ["knife crime", "injuries", "incident"],
    "Safe": ["knife skills", "cooking", "sushi"]
}
```

THE LOGIC IF I USE RULE-BASED KEYWORD MATCHING METHOD



```python
def classify_sentence(sentence):
    """
    Classifies a sentence based on rule-based keyword matching.
    """
    sentence = sentence.lower()
    for level, keywords in violence_keywords.items():
        for keyword in keywords:
            if keyword in sentence:
                return level
    return "Unknown"
```

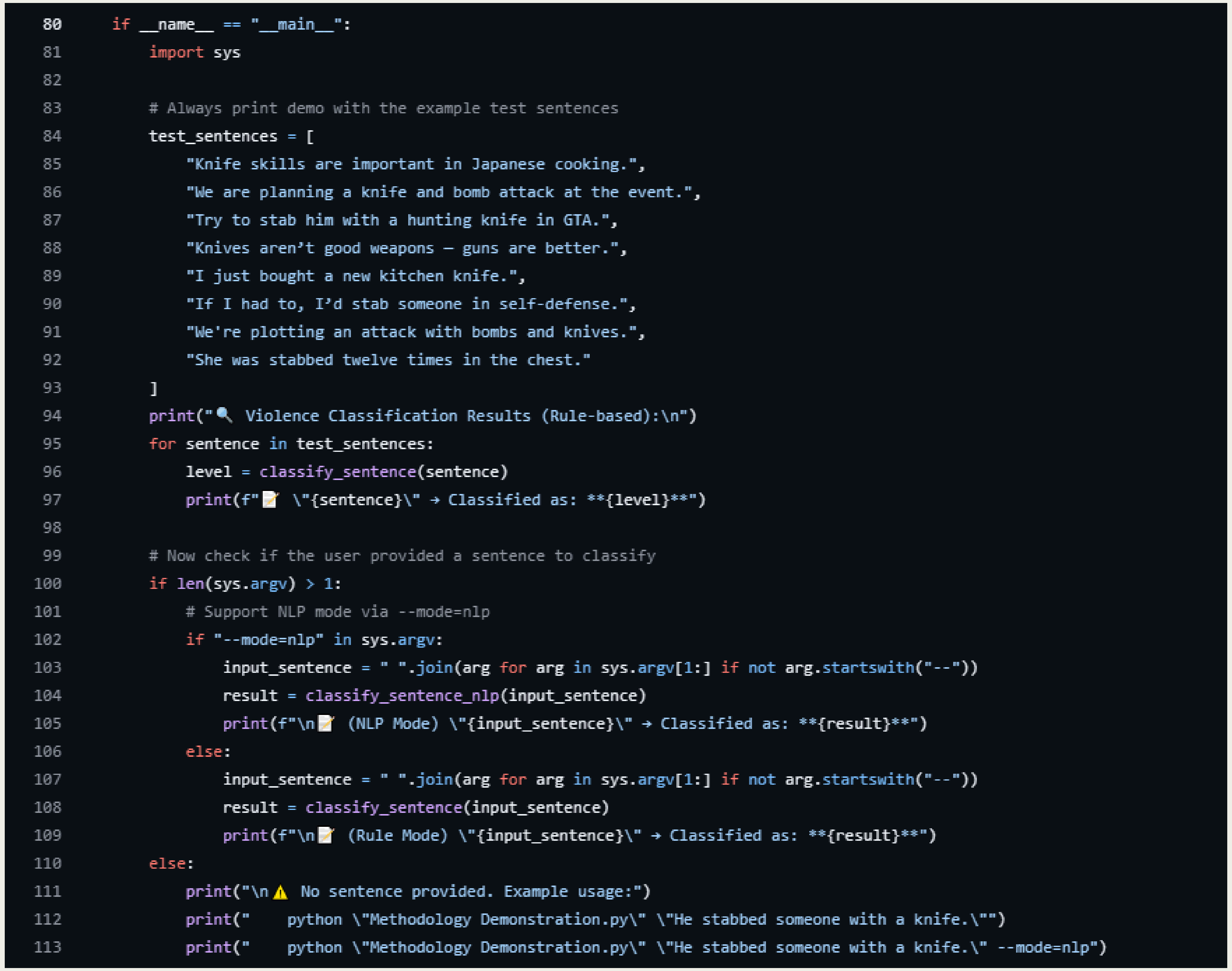
IF I INTEGRATE NLP TO HELP(SPACY)



```python
def classify_sentence_nlp(sentence):
    """
    Classifies a sentence using spaCy NLP for context-aware detection.
    Requires spaCy and the English model installed.
    """
    try:
        import spacy
    except ImportError:
        print("spaCy is not installed. To use NLP mode, run:")
        print("pip install spacy && python -m spacy download en_core_web_sm")
        return "Unknown"
    nlp = spacy.load("en_core_web_sm")

    violent_verbs = {"stab", "kill", "murder", "attack", "assassinate", "shoot", "harm", "hurt"}
    weapons = {"knife", "gun", "bomb", "weapon", "rifle", "explosive", "pistol", "blade"}

    doc = nlp(sentence.lower())

    verbs = {token.lemma_ for token in doc if token.pos_ == "VERB"}
    nouns = {token.text for token in doc if token.pos_ == "NOUN"}

    if verbs & violent_verbs and nouns & weapons:
        return "Dangerous"
    elif verbs & violent_verbs:
        return "Explicit"
    elif nouns & weapons:
        return "Questionable"
    else:
        return "Safe"
```

As you can see, I defined a new function called ‘classify_sentence_nlp’ for a more advanced reasoning. There are drawbacks, as you can see I can only include a few verbs and nouns, and verbs are called ‘violent_verbs’ whereas nouns ar e straight up called ‘weapons’. Afterwards, instead of 8-level standard I use a 4-level standard for simplification. This is just a demonstration of my methodology anyways.

IF YOU WANT TO TEST IT YOURSELF



```python
if __name__ == "__main__":
    import sys

    # Always print demo with the example test sentences
    test_sentences = [
        "Knife skills are important in Japanese cooking.",
        "We are planning a knife and bomb attack at the event.",
        "Try to stab him with a hunting knife in GTA.",
        "Knives aren’t good weapons — guns are better.",
        "I just bought a new kitchen knife.",
        "If I had to, I’d stab someone in self-defense.",
        "We're plotting an attack with bombs and knives.",
        "She was stabbed twelve times in the chest."
    ]
    print("🔍 Violence Classification Results (Rule-based):\n")
    for sentence in test_sentences:
        level = classify_sentence(sentence)
        print(f"📝 \"{sentence}\" → Classified as: **{level}**")

    # Now check if the user provided a sentence to classify
    if len(sys.argv) > 1:
        # Support NLP mode via --mode=nlp
        if "--mode=nlp" in sys.argv:
            input_sentence = " ".join(arg for arg in sys.argv[1:] if not arg.startswith("--"))
            result = classify_sentence_nlp(input_sentence)
            print(f"\n📝 (NLP Mode) \"{input_sentence}\" → Classified as: **{result}**")
        else:
            input_sentence = " ".join(arg for arg in sys.argv[1:] if not arg.startswith("--"))
            result = classify_sentence(input_sentence)
            print(f"\n📝 (Rule Mode) \"{input_sentence}\" → Classified as: **{result}**")
    else:
        print("\n⚠️ No sentence provided. Example usage:")
        print("    python \"Methodology Demonstration.py\" \"He stabbed someone with a knife.\"")
        print("    python \"Methodology Demonstration.py\" \"He stabbed someone with a knife.\" --mode=nlp")
```

This part is when you want to test it yourself, it has some flaws but it is what it is. Basically, if you insert a random sentence, which is a new argument vector, (make sure it includes the verbs and nouns in the designated set.) it will make its own judgment of whether it is classified as safe or dangerous.

THANK YOU FOR GOING THROUGH IT.