```
# Sample Hindi paragraph
paragraph = "मैं स्कूल जा रहा हूँ। क्या तुम आओगे? यह मेरी पसंदीदा किताब है। चलो पढ़ाई शुरू करें!"

# 1. Naïve space-based tokenization
naive_tokens = paragraph.split()
print("Naïve Tokenization:")
print(naive_tokens)

# 2. Manually corrected tokenization
# We'll separate punctuation and handle clitics/suffixes if needed
# Here, we just split sentence-ending punctuation
corrected_tokens = []

for word in naive_tokens:
    # If the word ends with punctuation, split it
    if word[-1] in ['।', '!', '?']:
        corrected_tokens.append(word[:-1])  # word without punctuation
        corrected_tokens.append(word[-1])  # punctuation as separate token
    else:
        corrected_tokens.append(word)

print("\nManually Corrected Tokenization:")
print(corrected_tokens)

# 3. Highlight differences
print("\nDifferences (Naïve vs Corrected):")
for n, c in zip(naive_tokens, corrected_tokens):
    if n != c:
        print(f"Naïve: {n} --> Corrected: {c}")
```

```
Naïve Tokenization:
['मैं', 'स्कूल', 'जा', 'रहा', 'हूँ।', 'क्या', 'तुम', 'आओगे?', 'यह', 'मेरी', 'पसंदीदा', 'किताब', 'है।', 'चलो', 'पढ़ाई', 'शुरू', 'करें!']

Manually Corrected Tokenization:
['मैं', 'स्कूल', 'जा', 'रहा', 'हूँ', '।', 'क्या', 'तुम', 'आओगे', '?', 'यह', 'मेरी', 'पसंदीदा', 'किताब', 'है', '।', 'चलो', 'पढ़ाई', 'शुरू', 'करें'

Differences (Naïve vs Corrected):
Naïve: हूँ। --> Corrected: हूँ
Naïve: क्या --> Corrected: ।
Naïve: तुम --> Corrected: क्या
Naïve: आओगे? --> Corrected: तुम
Naïve: यह --> Corrected: आओगे
Naïve: मेरी --> Corrected: ?
Naïve: पसंदीदा --> Corrected: यह
Naïve: किताब --> Corrected: मेरी
Naïve: है। --> Corrected: पसंदीदा
Naïve: चलो --> Corrected: किताब
Naïve: पढ़ाई --> Corrected: है
Naïve: शुरू --> Corrected: ।
Naïve: करें! --> Corrected: चलो
```

```
import stanza

# Download Hindi model (only needed once)
stanza.download('hi')

# Initialize Hindi NLP pipeline
nlp = stanza.Pipeline('hi', processors='tokenize', tokenize_pretokenized=False)

# Sample Hindi paragraph
paragraph = "मैं स्कूल जा रहा हूँ। क्या तुम आओगे? यह मेरी पसंदीदा किताब है। चलो पढ़ाई शुरू करें!"

# Manual corrected tokens (punctuation separated)
manual_tokens = ['मैं', 'स्कूल', 'जा', 'रहा', 'हूँ', '।',
                 'क्या', 'तुम', 'आओगे', '?',
                 'यह', 'मेरी', 'पसंदीदा', 'किताब', 'है', '।',
                 'चलो', 'पढ़ाई', 'शुरू', 'करें', '!']

# NLP tool tokenization using stanza
doc = nlp(paragraph)
tool_tokens = [token.text for sentence in doc.sentences for token in sentence.tokens]

print("Manual Tokens:")
```

```
print(manual_tokens)
print("\nTool Tokens:")
print(tool_tokens)

# Compare tokens using a sliding window to handle punctuation differences
print("\nDifferences between Manual and Tool Tokens:")

i, j = 0, 0
while i < len(manual_tokens) and j < len(tool_tokens):
    if manual_tokens[i] != tool_tokens[j]:
        print(f"Manual[{i}]='{manual_tokens[i]}' --> Tool[{j}]='{tool_tokens[j]}'")
    i += 1
    j += 1

# Print remaining tokens if lengths differ
while i < len(manual_tokens):
    print(f"Manual[{i}]='{manual_tokens[i]}' --> Tool[End]=None")
    i += 1
while j < len(tool_tokens):
    print(f"Manual[End]=None --> Tool[{j}]='{tool_tokens[j]}'")
    j += 1
```

```
Downloading https://raw.githubusercontent.com/stanfordnlp/stanza-resources/main/resources_1.10.0.json: 434k/? [00:00<00:00, 24.8MB/s]
INFO:stanza:Downloaded file to /root/stanza_resources/resources.json
INFO:stanza:Downloading default packages for language: hi (Hindi) ...
INFO:stanza:File exists: /root/stanza_resources/hi/default.zip
INFO:stanza:Finished downloading models and saved to /root/stanza_resources
INFO:stanza:Checking for updates to resources.json in case models have been updated.  Note: this behavior can be turned off with do
Downloading https://raw.githubusercontent.com/stanfordnlp/stanza-resources/main/resources_1.10.0.json: 434k/? [00:00<00:00, 23.8MB/s]
INFO:stanza:Downloaded file to /root/stanza_resources/resources.json
INFO:stanza:Loading these models for language: hi (Hindi):
=======================
| Processor | Package |
-----------------------
| tokenize  | hdtb    |
=======================

INFO:stanza:Using device: cpu
INFO:stanza:Loading: tokenize
INFO:stanza:Done loading processors!
Manual Tokens:
['मैं', 'स्कूल', 'जा', 'रहा', 'हूँ', '।', 'क्या', 'तुम', 'आओगे', '?', 'यह', 'मेरी', 'पसंदीदा', 'किताब', 'है', '।', 'चलो', 'पढ़ाई', 'शुरू', 'करें'

Tool Tokens:
['मैं', 'स्कूल', 'जा', 'रहा', 'हूँ', '।', 'क्या', 'तुम', 'आओगे', '?', 'यह', 'मेरी', 'पसंदीदा', 'किताब', 'है', '।', 'चलो', 'पढ़ाई', 'शुरू', 'करें'
```

```
import re

# Sample Hindi text
text = "मैं दिल्ली विश्वविद्यालय में पढ़ाई करता हूँ। वह मेरी आँख का तारा है। हमें स्वस्थ जीवन शैली अपनानी चाहिए।"

# List of MWEs to treat as single tokens
mwes = ["दिल्ली विश्वविद्यालय", "आँख का तारा", "स्वस्थ जीवन शैली"]

# Step 1: Replace MWEs with underscore-connected tokens
for mwe in mwes:
    text = text.replace(mwe, mwe.replace(" ", "_"))

# Step 2: Tokenize (naïve space-based tokenization)
tokens = text.split()

print("Tokens with MWEs treated as single tokens:")
print(tokens)
```

```
Tokens with MWEs treated as single tokens:
['मैं', 'दिल्ली_विश्वविद्यालय', 'में', 'पढ़ाई', 'करता', 'हूँ।', 'वह', 'मेरी', 'आँख_का_तारा', 'है।', 'हमें', 'स्वस्थ_जीवन_शैली', 'अपनानी', 'चाहिए।']
```

Reflection:

The hardest part of tokenization in Hindi was **handling punctuation and clitics**, because words often attach to sentence-ending marks like `।`, `?`, or `!`, and some contractions or suffixes (like `हूँ`, `है`, `करें`) can combine with auxiliary verbs. Unlike English, where spaces are

consistent word boundaries and punctuation is usually separated, Hindi uses complex **morphology** and often combines auxiliary verbs and postpositions, making naïve space-based tokenization inaccurate. Multiword expressions (MWEs) like idioms or place names add another layer of complexity because splitting them would **lose semantic meaning**. Therefore, punctuation, morphology, and MWEs all make tokenization more challenging in Hindi compared to English. Manual correction or MWE-aware tokenization is necessary to prepare clean, meaningful tokens for NLP tasks. Overall, tokenization in Hindi requires careful handling of both **syntactic structure** and **semantic units**.
