1. Number of Words in Paragraph

Concept: Count all words using simple split.
Syntax:

```
paragraph = "I love NLP. NLP is interesting. I love learning it."


# Split by space

words = paragraph.split()

num_words = len(words)

print("Number of words:", num_words)
```

---

2. Number of Sentences

Concept: Split paragraph using sentence delimiters . ! ?.
Syntax:

```
import re


sentences = re.split(r'[.!?]', paragraph)

# Remove empty strings

sentences = [s.strip() for s in sentences if s.strip()]

num_sentences = len(sentences)

print("Number of sentences:", num_sentences)
```

---

3. Word Repetition Count

Concept: Count how many times each word appears.

```
from collections import Counter


words_lower = [w.lower() for w in words]  # lowercase for uniformity

word_freq = Counter(words_lower)

print("Word frequencies:", word_freq)
```

```
# Example: check repetition of 'nlp'
print("NLP repetition:", word_freq.get('nlp', 0))
```

---

4. Sentence Repetition Count

```
sentences_lower = [s.lower() for s in sentences]
sentence_freq = Counter(sentences_lower)
print("Repeated sentences:")
for sent, count in sentence_freq.items():
    if count > 1:
        print(sent, "-> appears", count, "times")
```

---

5. Tense of Sentence / Paragraph

Concept: Simple rule-based tense detection using keywords and verbs (without tokenization libraries).

```
def detect_tense(sentence):
    sentence_lower = sentence.lower()
    words = sentence_lower.split()

    # Future tense
    if "will" in words or "shall" in words or "going to" in sentence_lower:
        return "Future"

    # Past tense (basic indicator)
    past_words = ["was", "did", "went", "had", "played", "saw"]  # expand as needed
    if any(w in words for w in past_words):
        return "Past"

    # Present tense
```

```
    present_words = ["is", "am", "are", "play", "go", "eat"]  # expand as needed
    if any(w in words for w in present_words):
        return "Present"

    return "Unknown"

for i, sent in enumerate(sentences, 1):
    print(f"Sentence {i} tense:", detect_tense(sent))
```

---

6. Sentence Sentiment (Positive / Negative / Neutral)

Concept: Check for presence of positive/negative words (simple approach).

```
positive_words = ["good", "great", "happy", "love", "excellent", "awesome"]
negative_words = ["bad", "sad", "hate", "terrible", "poor", "angry"]

def sentiment(sentence):
    words = sentence.lower().split()
    pos_count = sum(1 for w in words if w in positive_words)
    neg_count = sum(1 for w in words if w in negative_words)

    if pos_count > neg_count:
        return "Positive"
    elif neg_count > pos_count:
        return "Negative"
    else:
        return "Neutral"

for i, sent in enumerate(sentences, 1):
    print(f"Sentence {i} sentiment:", sentiment(sent))
```

---

7. Approach / Tone / Impression of Sentence

Concept: Detect casual, formal, aggressive, or emotional based on keywords.

```
casual = ["bro", "dude", "lol", "hey", "cool"]
aggressive = ["stupid", "idiot", "hate", "worst"]
formal = ["kindly", "regarding", "sincerely", "request", "therefore"]
emotional = ["happy", "sad", "angry", "love", "upset", "excited"]

def tone(sentence):
    words = sentence.lower().split()
    if any(w in words for w in aggressive):
        return "Aggressive"
    if any(w in words for w in formal):
        return "Formal"
    if any(w in words for w in emotional):
        return "Emotional"
    if any(w in words for w in casual):
        return "Casual"
    return "Neutral"

for i, sent in enumerate(sentences, 1):
    print(f"Sentence {i} tone:", tone(sent))
```

---

8. Feeling / Expressiveness (Emotion)

Concept: Detect emotion expressed in sentence (happiness, sadness, anger, fear, surprise, disgust).

```
happiness = ["happy", "glad", "excited", "joy", "awesome"]
sadness = ["sad", "upset", "hurt", "crying", "broken"]
```

```
anger = ["angry", "mad", "furious", "hate"]
fear = ["afraid", "scared", "worried", "terrified"]
surprise = ["wow", "unbelievable", "amazed", "shocked"]
disgust = ["disgusting", "gross", "horrible"]

def emotion(sentence):
    words = sentence.lower().split()
    if any(w in words for w in happiness):
        return "Happiness"
    if any(w in words for w in sadness):
        return "Sadness"
    if any(w in words for w in anger):
        return "Anger"
    if any(w in words for w in fear):
        return "Fear"
    if any(w in words for w in surprise):
        return "Surprise"
    if any(w in words for w in disgust):
        return "Disgust"
    return "Neutral"

for i, sent in enumerate(sentences, 1):
    print(f"Sentence {i} emotion:", emotion(sent))
```

✅ **Summary Table of Tasks & Approach**

Task	Method	Python / Syntax
Word Count	split paragraph	len(paragraph.split())

Task	Method	Python / Syntax
Sentence Count	regex split [.!?]	re.split()
Word Repetition	Counter on lowercased words	Counter(words_lower)
Sentence Repetition	Counter on sentences	Counter(sentences_lower)
Tense Detection	keyword/verb matching	detect_tense()
Sentiment	positive/negative word list	sentiment()
Tone / Impression	casual/formal/aggressive/emotional words	tone()
Emotion / Expressiveness	happiness/sadness/anger/fear/disgust/surprise	emotion()

POS Tag	Meaning	Example	Typical Use in NLP
NN	Noun, singular	cat, car	keyword extraction, NER
NNS	Noun, plural	cats, cars	keyword extraction
NNP	Proper noun, singular	John, India	NER, information extraction
NNPS	Proper noun, plural	Americans, Mondays	NER
PRP	Pronoun	he, she, it	coreference resolution
VB	Verb, base form	eat, go	sentiment, tense detection
VBD	Verb, past tense	went, played	past tense detection
VBG	Verb, gerund/present participle	running, eating	present continuous detection
VBN	Verb, past participle	eaten, gone	past perfect / passive detection
VBP	Verb, present, non-3rd person singular	eat, play	present tense detection
VBZ	Verb, present, 3rd person singular	eats, runs	present tense detection
JJ	Adjective	good, beautiful	sentiment / emotion detection
RB	Adverb	quickly, very	sentiment, emotion, emphasis
MD	Modal verb	will, shall, can	future tense detection
IN	Preposition / Subordinating Conjunction	in, on, because	dependency parsing
DT	Determiner	the, a, an	grammar structure
CC	Coordinating conjunction	and, but, or	sentence segmentation

POS Tag	Meaning	Example	Typical Use in NLP
UH	Interjection	wow, oh, hey	emotion / tone detection
TO	“to”	to (infinitive)	dependency parsing, verb phrases
EX	Existential there	there	grammar understanding