## 1: Word Tokenization

is used in various natural language processing (NLP) and text analysis tasks. Here's when and where you might use it:

## When to Use Word Tokenization:

1. **Text Analysis**: When you need to analyze or manipulate individual words within a text. For example, in sentiment analysis, you might want to assess the sentiment of each word.
2. **Text Classification**: For tasks such as spam detection or topic categorization, where understanding individual words is crucial for classifying text.

## Where to Use Word Tokenization:

1. **Preprocessing**: Before applying any machine learning model or NLP technique, to prepare the text data for further analysis.
2. **Feature Extraction**: When converting text into features for models, such as creating word frequency vectors or embeddings.
3. **Text Cleaning**: To remove unwanted characters, normalize text, or prepare data for other tokenization methods.

For Word Tokenization, the further types are:

**A: Whitespace Tokenization**: Splits text into words based on spaces.

**Whitespace Tokenization is used when:**

- **Text is well-formatted**: This means the text has clear spaces (like spaces, and tabs) between words.
- **Example**: Imagine a sentence like "This is a sentence." Each word is separated by a space.

**Why Use Whitespace Tokenization?**

- **Simple Structure**: If your text has simple spacing and no complex punctuation, whitespace tokenization can effectively split the text into words just by looking at spaces.
- **When to Avoid**: If the text includes complex structures like abbreviations, contractions, or punctuation that are important for understanding meaning, other methods might be more suitable.

**In Practice:**

- If you have a sentence like "I love programming", whitespace tokenization will split it into ["I", "love", "programming"].

### B: Punctuation-Based Tokenization

**Punctuation-Based Tokenization**: Splits text into tokens where punctuation marks such as periods, commas, exclamation points, question marks, and other symbols serve as delimiters. This method often keeps punctuation marks as separate tokens or uses them to define the boundaries between tokens

## When to Use Punctuation-Based Tokenization:

1. **Sentence Segmentation**: When you need to break down text into sentences. Punctuation marks like periods, exclamation points, and question marks are used to identify sentence boundaries.

## Examples:

- **Sentence Splitting**: Converting "Hello there! How are you doing today?" into ["Hello there!", "How are you doing today?"].

## C: Regex-Based Tokenization

**Regex-based tokenization** means splitting text into tokens using regular expressions (regex). Regular expressions are patterns used to match specific sequences in the text.

## When to Use Regex-Based Tokenization:

1. **Custom Tokenization**: When text does not follow standard structures or contains specific patterns that need to be identified. For instance, tokenizing dates, email addresses, or URLs.
2. **Data Extraction**: To extract specific information from text, such as phone numbers, hashtags, or code snippets, based on predefined patterns.
3. **Text Normalization**: When you need to clean and normalize text by removing or separating specific patterns, such as special characters or formatting inconsistencies.
4. **Complex Text Processing**: In scenarios where standard tokenization methods are insufficient, there is a need for complex pattern matching to correctly identify tokens.

## Examples:

- **Email Extraction**: Using a regex pattern like `[\w\.-]+@[\w\.-]+` to extract email addresses from text.
- **Hashtag Extraction**: Using a regex pattern like `#\w+` to find hashtags in social media posts.
- **URL Extraction**: Using a regex pattern like `https?://(?:[-\w.]|(?:%[\da-fA-F]{2}))+` to extract URLs from a text.

## D: Rule-Based Tokenization

**Rule-based tokenization** is a method of splitting text into tokens based on predefined rules. These rules are often customized to fit the specific characteristics of the text being processed. The rules can be based on linguistic features like spaces, punctuation, capitalization, or specific patterns in the text.

### When to Use Rule-Based Tokenization:

1. **Complex Text Structures**: When the text has complex structures or specific formatting that requires customized rules for tokenization. For example, tokenizing legal documents, code, or scientific text where standard methods might fail.
2. **Domain-Specific Language**: When working with specialized domains like medicine, law, or finance, where standard tokenization methods might not be adequate. Customized rules can be created to handle specific terminology, abbreviations, or symbols.
3. **Multi-Language Texts**: When the text contains multiple languages or scripts, and you need to apply different tokenization rules depending on the language or script used.
4. **Context-Aware Tokenization**: When you need to consider the context to correctly tokenize words or phrases, such as in cases where certain phrases should not be split despite spaces or punctuation.

## E: Dictionary-Based Tokenization

Dictionary-based tokenization is a way of breaking down text into smaller parts (tokens) using a predefined list of words, called a dictionary. The text is scanned, and whenever a word in the text matches one in the dictionary, it is marked as a token.

dictionary = ["apple", "banana", "fruit", "fresh", "rotten"]

text = "I have a fresh apple and a rotten banana."

The tokens recognized would be:

- `"fresh"`
- `"apple"`
- `"rotten"`
- `"banana"`