1. Sentence-based Chunking

- Splits text into **sentences**, then groups them until the chunk reaches your target size (e.g., 200 characters).
- Ensures chunks **don't cut sentences in the middle**.

Example:

```
Product: Dell Laptop. Brand: Dell. Model: XPS 13.
```

- → All those stay together if they fit within the size.
- ✅ Good for **keeping natural language intact**.
- ❌ Chunks might vary a lot in size (short vs. long sentences).

---

2. Token-based Chunking

- Splits text by **number of tokens** (words/subwords, depending on tokenizer).

Example (target = 50 tokens):

```
 [First 50 tokens] → Chunk 1
[Next 40 tokens + 10 overlapping tokens] → Chunk 2
```

-
- ✅ Good for **consistency** → chunks are uniform in length, so embeddings cost and memory usage are predictable.
- ❌ Might **cut off mid-sentence** or split related info unnaturally (e.g., "Price: 1000" in one chunk, "Availability: In stock" in the next).

---

3. Recursive/Semantic-based Chunking

- Splits text using **a hierarchy of separators** (`\n\n`, `. `, `, `, spaces).
- It tries to **preserve semantic units** first, then fallback to smaller splits if too long.
- Example:
  - First, try to split by paragraphs.
  - If still too long → split by sentences.
  - If still too long → split by tokens.
- ✅ Best for **context preservation**.
  You won't usually lose meaning because it respects natural structure as much as possible.
- ❌ More **variable chunk sizes** than token-based.

🔹 Do They Extract Exact Portions Every Time?

- **Sentence-based** → deterministic, as long as your text & parameters don't change. Always splits at sentence boundaries.
- **Token-based** → deterministic, always splits at the same token positions.
- **Recursive/semantic** → deterministic too, but the **splits depend on text structure** (newlines, punctuation). If text formatting changes, chunking can shift.

So yes, they are **consistent/deterministic**, but they **don't always split in the exact same locations across strategies** → because each has a different "rule of splitting."

🔹 Key Differences (Summary Table)

Strategy	Splitting Rule	Pros	Cons	Best For
Sentence	Sentences + size limit	Preserves grammar	Uneven sizes	QA, summarization
Token	Fixed tokens + overlap	Uniform, predictable cost	Cuts mid-sentence	Search, embeddings
Recursive	Hierarchy of separators	Keeps context & meaning	Irregular sizes	Semantic search, RAG