# CharacterTextSplitter vs RecursiveCharacterTextSplitter

| 구분 | **CharacterTextSplitter** | **RecursiveCharacterTextSplitter** |
|---|---|---|
| 장점 | 구분자를 기준으로 빠르고 효율적으로 텍스트 분리 가능.<br>단순한 텍스트 구조에 적합.<br>근거: 단일 구분자로 작동하므로 간단한 문단이나 줄바꿈 중심의 텍스트에서 잘 작동. | 다양한 구분자를 계층적으로 적용해 복잡한 텍스트도 세분화 가능.<br>주제별 분리에 적합.<br>근거: 계층적 구분으로 큰 섹션부터 세부 단위까지 점진적으로 분리 가능. |
| 단점 | 구분자 의존성이 강하며, 복잡한 텍스트를 주제별로 나누기 어려움.<br>근거: 구분자가 없는 긴 문단에서는 제대로 분리되지 않음. | 추가 연산 비용 발생. 구분자를 잘못 설정하면 과도한 청크 생성 가능.<br>근거: 작은 구분자까지 적용 시 청크가 불필요하게 많아지고 계산 시간이 증가. |
| 특징 | 단순 구조에서는 잘 작동하지만, 긴 문단에서 구분자가 없으면 청크 크기를 초과.<br>근거: 특정 구분자로만 작동하므로 긴 텍스트에서 예외 상황 발생 가능. | 긴 텍스트에서도 주제별로 나뉘지만, 중복된 텍스트가 반복될 수 있음.<br>근거: 중복 설정(chunk_overlap)으로 문맥 연결성을 높이지만 결과에 불필요한 중복 포함. |

## 구현하면서 느낀 점

1. CharacterTextSplitter

- 단순한 텍스트 구조에서는 빠르고 효율적, 단일 구분자로 처리해서 코드도 간단하지만 구분자가 없으면 텍스트가 과도하게 길어져 청크 크기를 초과하거나 분리되지 않는 문제가 생김.

2. RecursiveCharacterTextSplitter

- 긴 텍스트에서도 계층적 구분자를 사용해 주제별로 나누기 적합하고 다양한 텍스트 구조를 유연하게 처리 가능하지만 연산 비용이 크고 구분자 설정이 잘못되면 효율성이 떨어짐.