Vorteile der Rekursion gegenüber Schleifen

✅ Vorteile der Rekursion gegenüber Schleifen

1. Eleganter Code für rekursive Probleme (Bäume, Graphen)
🔷 Rekursion für die Fakultät:

```
 int fakultaet(int n) {
   if (n <= 1) return 1;
   return n * fakultaet(n - 1);
 }

int fakultaetLoop(int n) {
int result = 1;
for (int i = 1; i <= n; i++) {
  result *= i;
}
return result;
}
```

2. Teilen und Herrschen (Divide & Conquer) 🔷 Rekursion für Binärsuche:

```
int binarySearch(List<int> arr, int low, int high, int key) {
  if (low > high) return -1;
  int mid = (low + high) ~/ 2;
  if (arr[mid] == key) return mid;
  if (arr[mid] > key) return binarySearch(arr, low, mid - 1, key);
  return binarySearch(arr, mid + 1, high, key);
}
```

3. Einfacher in UML-Diagrammen und Pseudocode darstellbar

Rekursive Algorithmen sind oft leichter zu visualisieren und im Pseudocode zu beschreiben, weil sie die Wiederholung durch den rekursiven Funktionsaufruf direkt darstellen. Beispiel in UML: Ein rekursiver Algorithmus zeigt sich klar als Zyklus von Funktionsaufrufen in einem Aktivitätsdiagramm.

⛔ Nachteile der Rekursion gegenüber Schleifen:

1. Hoher Speicherverbrauch → Jeder rekursive Aufruf belegt Speicher im Call Stack.

2. Langsam bei zu vielen Aufrufen → Mehr Funktionsaufrufe sind langsamer als Schleifen.

Fazit

Rekursion: Gut für rekursive Probleme, Divide & Conquer, und einfachere Darstellung in UML/Pseudocode.
Schleifen: Besser bei einfachem Durchlaufen und wenn Leistung und Speicher wichtig sind.