Introduction

Are you completely new to recursion? No idea what it means or what we're talking about? After a lot of intimidating comments regarding recursion, you ended up here hoping to learn and solve recursive problems all by yourself. Well, don't fear. You've come to the right place! In this course, we start with the very basics of recursion using Java as our base language, and we build up to relatively complex problems that you'll be able to solve by yourself as you progress through this course. Awesome, right?

What is Recursion?

Recursion is when a *method calls itself again and again until it reaches a specified stopping condition.* In general, the method mentioned above is called a **recursive method**.

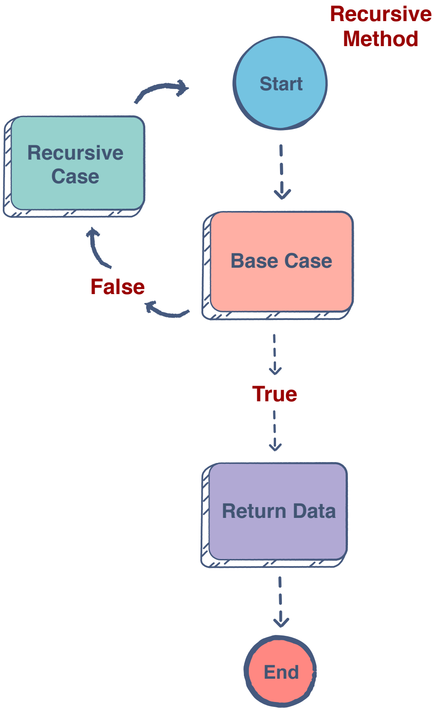
Why Recursion?

- Recursive code is generally shorter to write as compared to an iterative code.
- In general, loops are converted into recursive methods when they are compiled or interpreted.
- A recursive method is most useful for tasks that can be defined in terms of similar subtasks.

Format of a Recursive Method

Each recursive method consists of 2 parts:

1. **Base Case:** The base case is where the call to the method stops, meaning, it does not make any subsequent recursive calls.
2. **Recursive Case:** The recursive case is where the method calls itself again and again until it reaches the base case.



Generic Recursive Algorithm

```
RecursiveMethod() {
  // Base Case
  if (base case condition) {
    return some base case value;
  }
  else {
    // Recursive Case
    return (some work and then a recursive call)
  }
}
```

Now that you're familiar with the term *recursion*, let's move on and discuss how memory is located in different recursive calls.