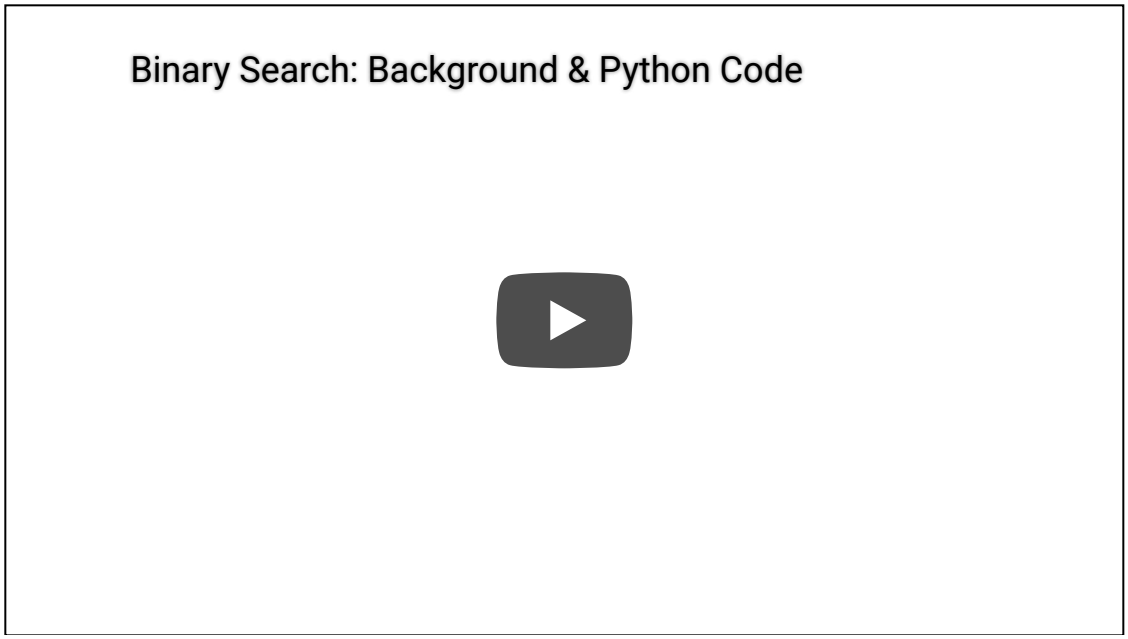
Binary Search



Let's look at another example of a function utilizing **recursion (https://maryville.instructure.com/courses/43640/pages/overview-of-recursion)** .

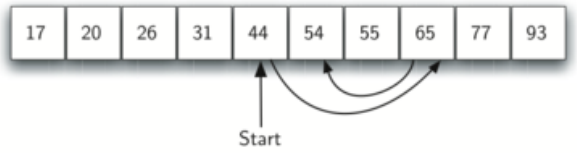
Unsorted Searching

Unsorted searches loop every element until either finding the target or exhausting the data set. This is a **sequential search**. Think of it in terms of looking through a phonebook. A sequential search would have us start at "Aaron Aaron" and have us search through every name in order until we reach the name that we are searching for (let's say we are looking for Gary Gygax). That's a lot of searching and will take a large amount of time.

Locating a Target in a Sorted Sequence

Programming a **binary search** is used to efficiently locate a target value within a sorted sequence. By utilizing a binary search, we can start at the midpoint looking for an element and if that isn't it, we can eliminate all the other targets behind it. For our phonebook example, it would be like opening the phonebook in the E's. We know that nothing before that E section can possibly fit our search because we are looking for Gary Gygax. No E's there.

Numerically, this looks like this:



We start our search looking for 54 at the number 44. The algorithm then recursively runs again to 65 and then back to 54. It is much quicker than if we would have started at the beginning of the list with 17. This is because our search is **sorted** and **indexable**.

Source: Problem Solving and Algorithms in Python (http://interactivepython.org/runestone/static/pythonds/index.html#) from Bradley Miller on www.interactivepython.org (http://interactivepython.org/runestone/static/pythonds/index.html#).