☼Hi, welcome to Algorithm Implementation: Searching in Python.

Searching involves finding some piece of data within a large collection. A *linear search* starts at the beginning and looks at consecutive elements until the target is found. A *binary search* splits the data into two halves each time an element in the set is examined and so it is faster but it depends on the data being sorted.

We will discuss how to code linear search and binary search in Python.

☼After this lesson, you should be able to:

- Use index method in Python
- Explain the importance of coding searching algorithms in Python
- Code linear search in Python
- Code binary search in Python
- Identify other search algorithms written in Python
- Apply your knowledge and understanding of searching algorithms to your problem solving in Python

☼ In this lesson, we will cover a few topics. Let's go through them in details one by one.

☼Let's recall the application: Is Pizza hut in **North Spine Plaza?**

☼In Linear Search, we sequentially iterate over the given list and check if the element we are looking for is equal to the one in the list. As we know, The keyword in precedes the sequence.

In this program, The variable item is a variable associated with the for loop that is assigned the value of an element in the sequence. if the element is not our target

The variable item is assigned a different element during each pass of the for loop.

Eventually, item will have been assigned to each element in the sequence or until the target.

The two keywords for in are used to implement the basic Linear Search in Python.

☼ Python also has built-in operations on a list that will do this.

The index method is used for searching lists to find the index of the first occurrence of a value.

The **index** method does a linear search and stops at the first matching item. If no matching item is found, it raises a **ValueError** exception.

☼Now we use index() method to solve the same problem. Check if Pizza hut is in North Spine Plaza.

Pizza hut is the fifth item, and its index is 4. So the return value of first index method is 4. This method returns index of the found object.

Otherwise, it raises an exception indicating that the value is not found. Python is case-sensitive. So, Second index function raises an exception.

☼ We already know that Linear search is simple and easy to implement, it can be used when elements in the list are **not** sorted. But it is inefficient compared to Binary Search for sorted list.

This slide shows the flow chart of binary search.

Instead of searching the list in the sequence, a **binary search** will start by examining the middle item. If that item is the one we are searching for, we are done. If it is not the correct item, we can use the sorted nature of the list to eliminate half of the remaining items. If the search key is greater than the middle item, we know that the entire lower half of the list as well as the middle item can be eliminated from further consideration. The item, if it is in the list, must be in the upper half.
We can then repeat the process with the upper half. Start at the middle item and compare it against the search key. Again, we either find it or split the list in half, therefore eliminating another large part of our possible search space.

☼Let's revisit the Hi-Low" Number Guessing Game to review the working schemes of linear search and binary search.

☼ A reason to examine searching algorithms is that someone had to implement the operations for the Python system itself, and they

had to know how. Did he do a good job? Are the built-in operations as fast as ones that a programmer could code for himself?

Program in this slide gives the iterative version of the binary search algorithm.

The function binarysearch () is implemented with the while loop in which the operations inside are executed until the search key is found or the end of the list is reached without success. The operations perform the finding of the mid-point of the list, comparing the value stored at the mid-point element with the search key, and searching the one half of the array if the search key is not found.

☼As we have learnt, "Recursion" naturally supports the divide-and-conquer strategy. Binary search algorithm can also be implemented using a recursive approach. The function first checks if it comes to end of list, if yes, returns false.

Or else, locates the middle point of the list and compares middle item with search key. If they match, the search key is found, return true.

If not, we continue the process to search one half of the list by invoking the function itself. If the search key is less than the middle element of the list, then the first half of the list will be searched. Otherwise, the second half of the list will be searched. This process is repeated until it comes to **the end of the list or** the search key is found.

☼Quick summary

Searching is the act of determining whether some specific data item appears in a list.

The **index** method does a linear search, and stops at the first matching item. If no matching item is found, it raises a **ValueError** exception.

Linear search is a simple and easy way to implement searching technique but it is inefficient compared to Binary Search for sorted list.

Binary search algorithm can be implemented either using an iterative or recursive approach.

I hope you enjoy this lecture.