Searching in Python

This presentation explores the fundamental concepts of searching in Python, emphasizing the trade-offs between linear and binary search algorithms. We'll also delve into exception handling and assertion techniques, crucial aspects of robust code development.

Searching in Python refers to the process of finding a specific element in a list, dictionary, set, or other data structures. The goal is to determine whether the element exists and, if so, find its position or retrieve its value.

Linear Search: Sequential Approach

How It Works

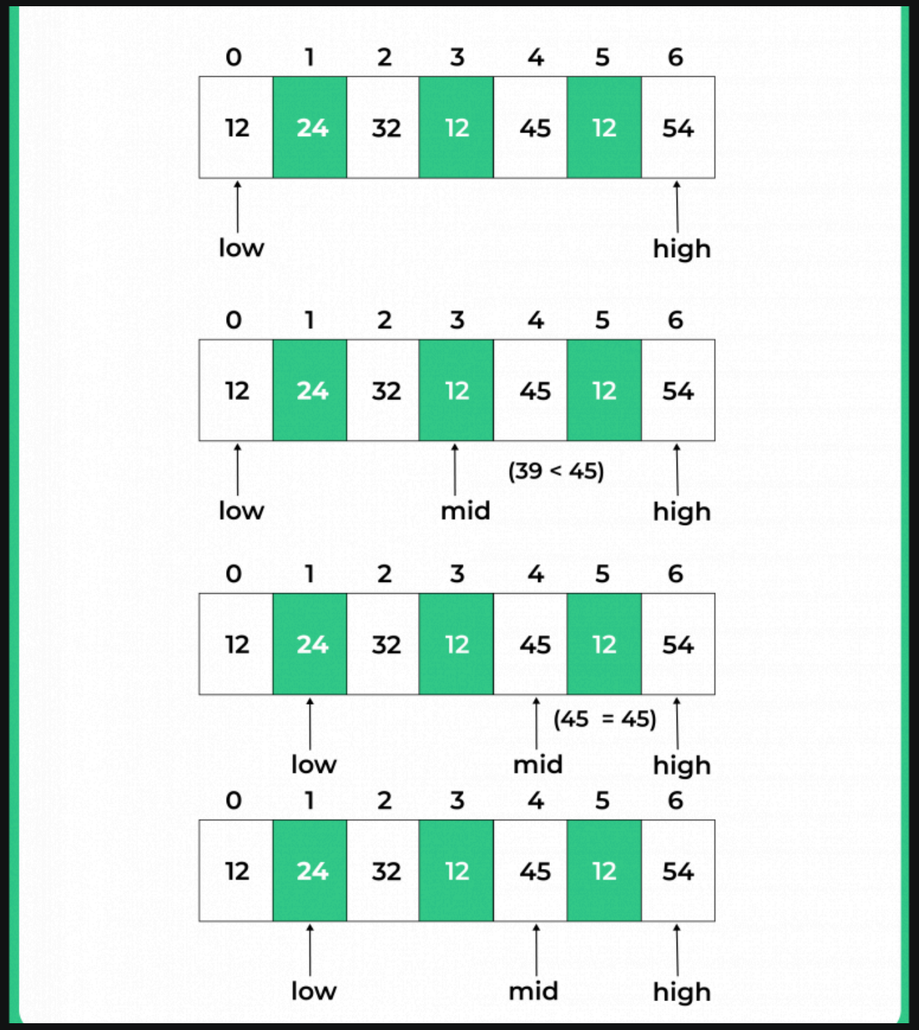
Linear search iterates through each element in a list, comparing it to the target value. If a match is found, the index is returned.

Advantages

Simple to implement, suitable for small datasets or unsorted lists.

Disadvantages

Inefficient for large datasets, requiring a traversal of every element.



Binary Search: Divide and Conquer

1 **Divide**

The search space is repeatedly halved by comparing the target value with the middle element.

2 **Conquer**

If the target is greater than the middle element, the search continues in the right half; otherwise, in the left half.

3 **Repeat**

This process continues until the target is found or the search space is exhausted.

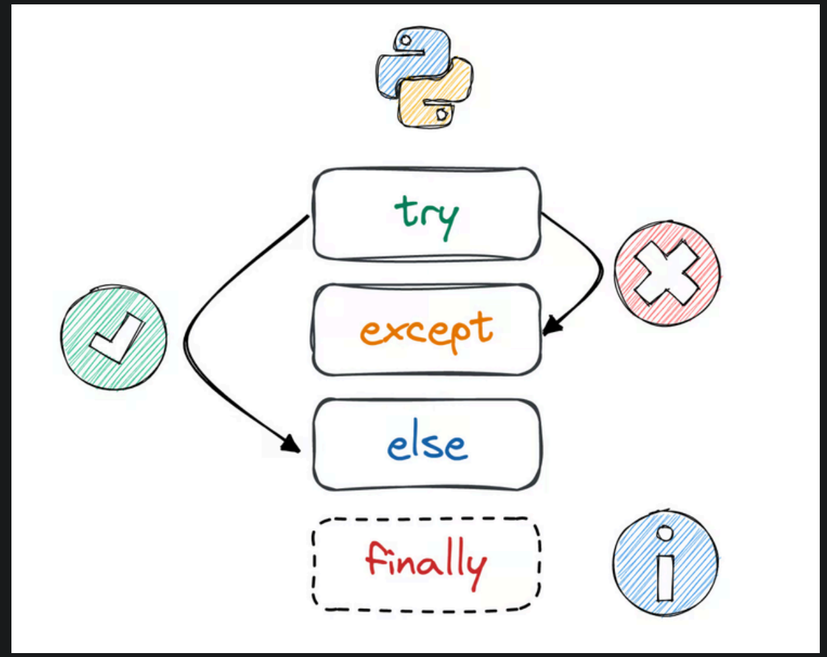
Exception Handling: try, except, else, finally

try
The code that might raise an exception is placed within the try block.

except
Handles specific exception types to prevent program termination. The code within the except block is executed if the exception is raised in the try block.

else
Executes if no exception is raised in the try block.

finally
Code within the finally block executes regardless of whether an exception is raised or not. Often used for cleanup operations.



Assertion and Raise: Possible Combinations and Errors

Assertion

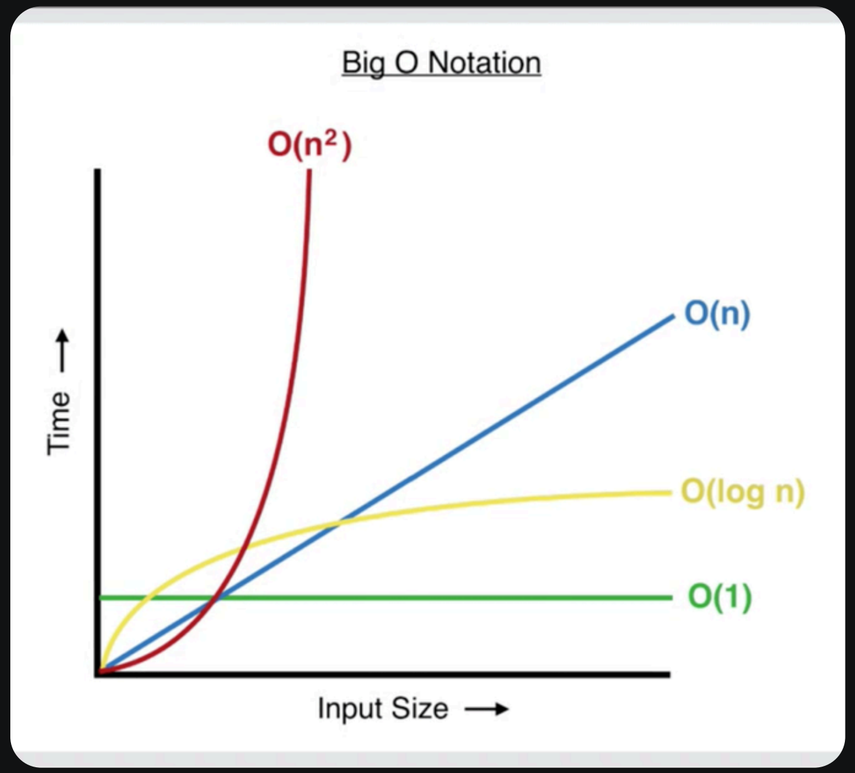
Used to check conditions at runtime. If the condition is false, an AssertionError is raised.

Raise

Explicitly raises an exception, allowing for controlled error handling. You can create custom exception classes as well.

Combinations

Assertions can be used to check for errors before performing operations that might raise exceptions.



Time Complexity of Search Algorithms

1

Linear Search

Time complexity is O(n), meaning the search time increases linearly with the size of the data.

2

Binary Search

Time complexity is O(log n), indicating that the search time grows logarithmically with the data size.

Choosing the Right Search Algorithm

Data Size

For small datasets, linear search is often sufficient due to its simplicity.

Data Sorted

If the data is sorted, binary search offers significant performance advantages for large datasets.

Time Complexity

Binary search's logarithmic time complexity makes it the preferred choice for efficient searches in large, sorted datasets.