### Programming Guide for Dynamic Array

#### Overview of Dynamic Array

A Dynamic Array is a data structure that provides a way to store elements in a contiguous block of memory. Unlike a static array, it can change size dynamically to accommodate the addition or removal of elements.

#### Key Operations

* **Add:** Insert an item at the end of the array.
* **Get:** Retrieve an item at a specified index.
* **Remove:** Delete an item from a specified index.
* **Resize:** Increase or decrease the array’s capacity as needed.

#### Pseudo Code for Dynamic Array

CLASS DynamicArray:  
 FUNCTION \_\_init\_\_():  
 Initialize an array with a predefined capacity  
 Set count to 0  
  
 FUNCTION add(item):  
 IF count equals capacity:  
 Resize the array to double its size  
 Add item to the array at the current count index  
 Increment the count  
  
 FUNCTION get(index):  
 IF index is within the bounds of the array:  
 RETURN the item at the index  
 ELSE:  
 Raise an IndexError  
  
 FUNCTION remove(index):  
 IF index is within the bounds of the array:  
 FOR each element starting from index to end:  
 Shift elements to the left  
 Decrement the count  
 IF count is much smaller than capacity:  
 Resize the array to half its size  
 ELSE:  
 Raise an IndexError  
  
 FUNCTION \_resize(new\_capacity):  
 Create a new array with new\_capacity  
 Copy elements from the old array to the new one  
 Update the array reference and capacity

#### Implementation Tips

* Initialize the array with a fixed initial capacity.
* Implement a private \_resize method that creates a new array with the desired capacity and copies elements from the old array to the new one.
* In the add method, if the array is full (i.e., count equals capacity), call \_resize to double the size of the array before adding the new element.
* In the remove method, after removing an element, if the number of elements is much less than the capacity (e.g., a quarter of the capacity), reduce the size of the array by half to save space.
* Always check for index out-of-bounds in the get and remove methods.

#### Applications

* **Resizable storage:** Dynamic arrays are used in languages like Python, Java, and C++ (as vectors) to provide resizable array-like storage.
* **Building blocks:** They are foundational for higher-level data structures like strings, stacks, queues, and arraylists.
* **Space efficiency:** By resizing, dynamic arrays use memory efficiently and offer flexibility for various applications that require array-like storage.