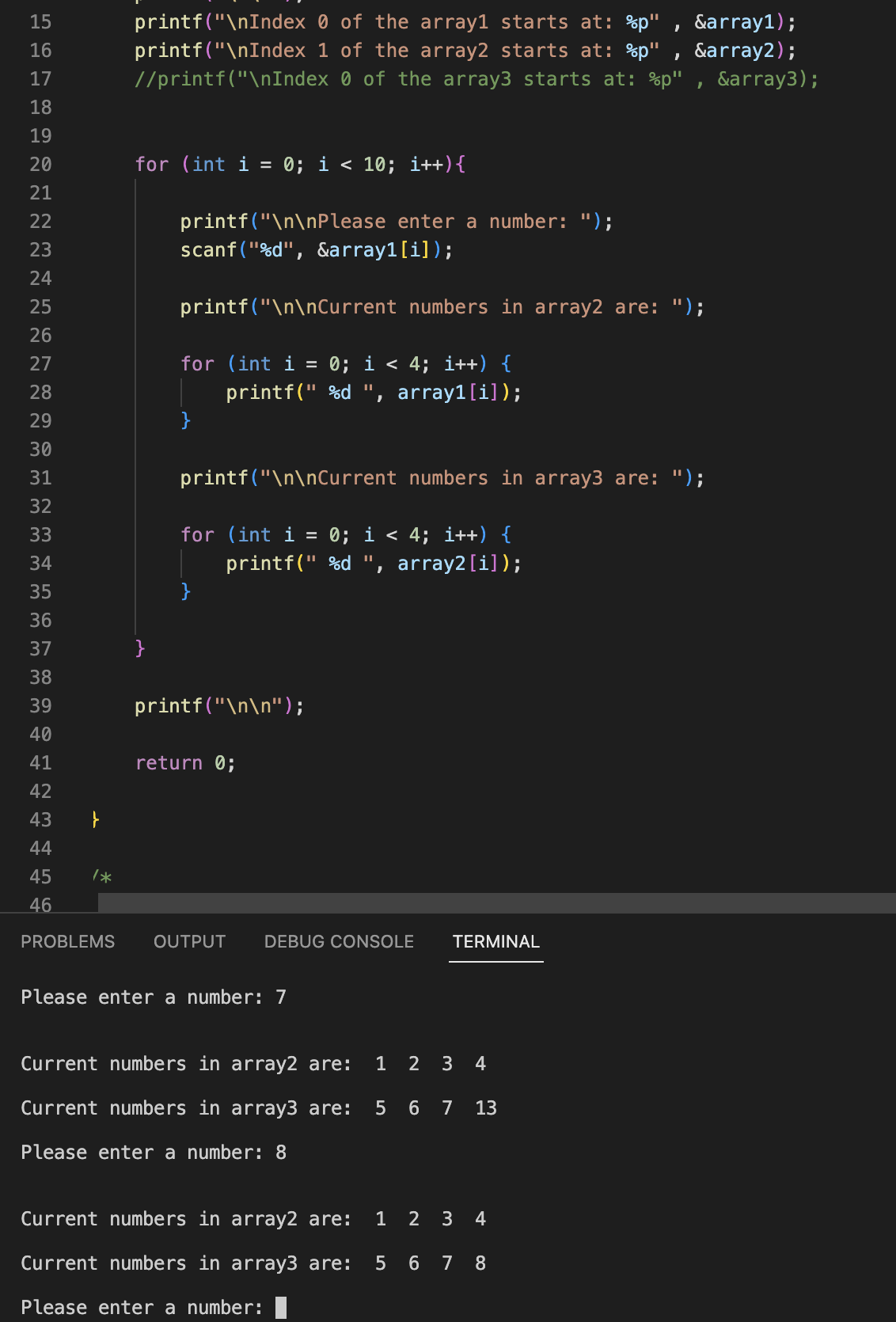
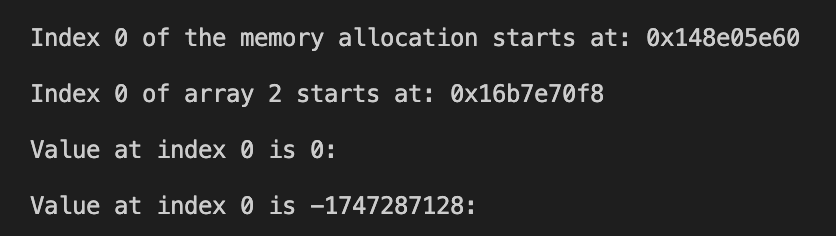
1. Usestaticarray



The user is given two arrays, both have 4 integers inside, making 16-byte padding spacing between the two arrays. When the user enters a number input, it’ll fill the first array that’s empty, this array will be the first 16-byte in the memory. Once the user over-extends and goes out of bounds, the array has taken up all 16-byte and the input has nowhere to go except for the next “available” spacing in the second array. It continues to the second array because that’s what’s next within the memory. Doing this will mess up the array order and potentially create bugs or unwanted outputs in the code. To prevent this, we’ll need to create a new array and copy or move the existing numbers in the old array to a new, expanded array.

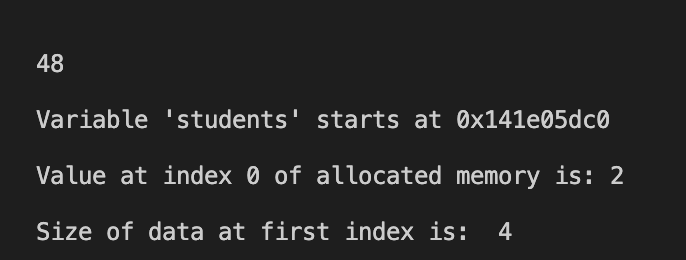
1. Mallocprinter



1. Dynamically allocating

My approach to using a while loop for continuous user input involves maintaining a count variable to track the number of values entered. This helps transfer all user-provided numbers into a new array when needed. To allow the user to exit the loop, I chose to use a negative number as a way to break out of the loop. While this approach may not be ideal for every project, it’s what I decided to use for this one. This method works by adjusting the array size based on user input. If the user exceeds outside of the array bounds, I double its size, copy the existing elements into the new array, and continue inputs without losing data. If the expanded array has unused space, the remaining spaces will contain zeros as they are uninitialized memory slots.

1. Simple Dynamic



1. Python Lists are Objects Not Arrays

An object is an instance of a class that includes both the data (attributes) and behavior (methods). It follows the principles of object-oriented programming like encapsulation, inheritance, polymorphism, and abstraction. Python’s list embodies the principles of OOP because it’s an object that’s included in the built-in “List” class.

1. Linked Lists

A linked list dynamically allocates memory for each node allowing for efficient insertion and deletion without worrying about a fixed size. This provides flexible allocation, reduces fragmentation, and enables efficient insertion and deletion. Unlike arrays, which require a fixed size, a linked list dynamically allocates memory for each node, allowing it to grow and shrink as needed. Linked lists also eliminate the need for shifting elements within the list for insertion and deletion as they enable O(1) operations by updating a few pointers. Addressing these topics, linked lists provide a more flexible and efficient approach to managing memory dynamically.