Problem Domain

- Write a function called BinarySearch which takes in 2 parameters: a sorted array and the search key. Without utilizing any of the built-in methods available to your language, return the index of the array's element that is equal to the value of the search key, or -1 if the element is not in the array.

Input — A LIST, AN KEY

Output — Return index of List where Key matches

Test Cases

1. Given — [1,2,3,4,5] key = 3, return index(2)
2. Given — [1,2,2,2,3,4,5] key= 2, return index(1,2,3)
3. Given — [3,4,5] key =2, return -1

Big O

Big O time O(n) because we are only iterating through list one time
Big O Space O(1) because we will store multiply indexes for the same key (worst case scenario)

Visualization

```
[1, 2,3,4,5] key = 3
 0  1 2 3 4  are my indexes
key is at index 2
```

Code

```
def binary_search(arr, search_key):
    left = 0
    right = len(arr) - 1
    while left <= right:
        mid = (left + right) // 2
        if arr[mid] == search_key:
            return mid
        elif arr[mid] < search_key:
            left = mid + 1
        else:
            right = mid - 1
    return -1

arr = [1, 3, 5, 7, 9, 11]
search_key = 9
index = binary_search(arr, search_key)
print(index)
```

Algorithm

-Write a function that takes in a list and a search key as parameters.
-Iterate through the list
-capture index of key
-return index
else
-return -1 if list does not contain key

Step Through

[1,	2,	3,	4]	key = 3
0	1	2	3	indexes
return index(2)				