The logic behind the binary search. It is great to learn something new.

When we want to check if an element is in a list, like check if Will Smith is in our employee database by matching his SSN with traversal. But if we have a huge database, and his SSN is at very end, it will be a waste of time and the server resource. Binary search is a good way to solve this problem in sorted arrays: it is a classic algorithm in computer science like below.

list	11	20	35	47	***58***	60	73	89	99	105
index	0	1	2	3	***4***	5	6	7	8	9

We have a sorted random list with index below accordingly. Define lb (lower bound), hb(higher bound), and mid(middle) to represent the lowest, highest, and middle indexes of the list, we get:

```
lb = 0                  # list[lb] = list[0] = 11
hb = len(list) – 1      # list[hb] = list[9] = 105
mid = (lb+hb) // 2      # take round down for even numbers of elements:  list[mid] = 58
target = int(input('The targeted search to see if it is in list: '))        # assume = 99
```

Compare target with list[mid]: 99 > (list[4] = 58), then we can work on the mid to hb part only, with new lb of lb = mid + 1 = 5, and the new mid will be (lb+hb) // 2 =7

list	11	20	35	47	58	60	73	***89***	99	105
index	0	1	2	3	4	5	6	***7***	8	9

Compare target with list[mid]: 99 > (list[7] = 89) again, then we can work on the mid to hb part only, with new lb of lb = mid + 1 = 8, and the new mid will be (lb+hb) // 2 =8

list	11	20	35	47	58	60	73	89	***99***	105
index	0	1	2	3	4	5	6	7	***8***	9

Compare target with list[mid]: 99 == 99, print target is in the list. If no same element found in list, print target not in list.

Codes for traversal and binary search in py document.