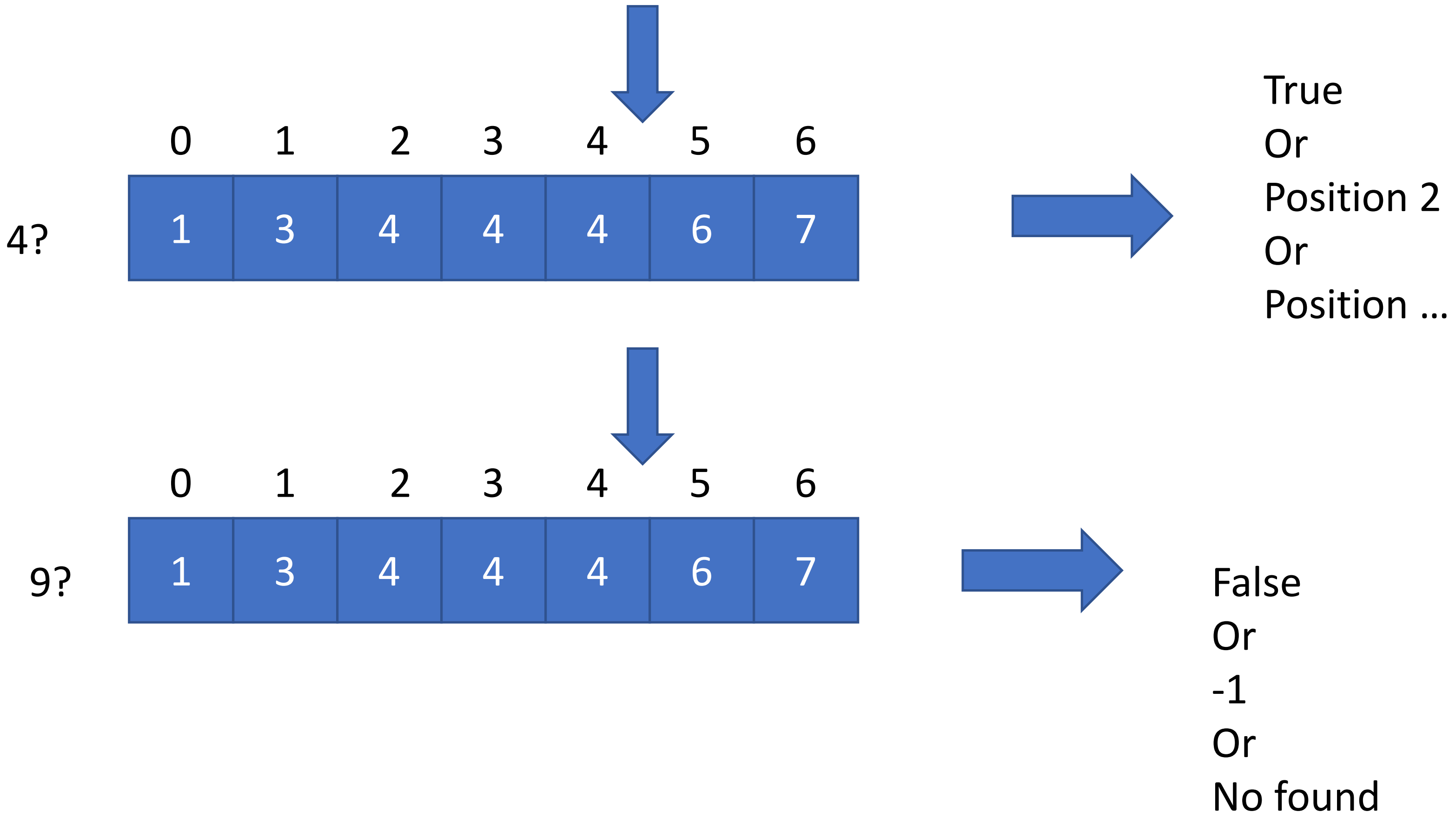
# On Searching

support

# Searching



4?
0 1 2 3 4 5 6
1 3 4 4 4 6 7
True
Or
Position 2
Or
Position …
9?
0 1 2 3 4 5 6
1 3 4 4 4 6 7
False
Or
-1
Or
No found

## You already used

- On list, …
- X in lst
  - True if x is in Lst
- Lst.index(x)
  - Look for value position in lst

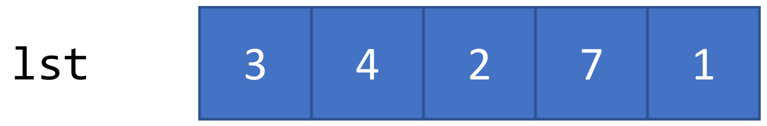
## How it is done? ( one way at least)

```
def myindex( lst, x) :
  for i in range( len(lst)
  if x==lst[i] :
    return i
  return None
```

## How it is done? ( one way at least)

```
def myindex( lst, x) :
  for i in range( len(lst)
  if x==lst[i] :
    return i
  return None
```

```
myindex( lst, 2)
```

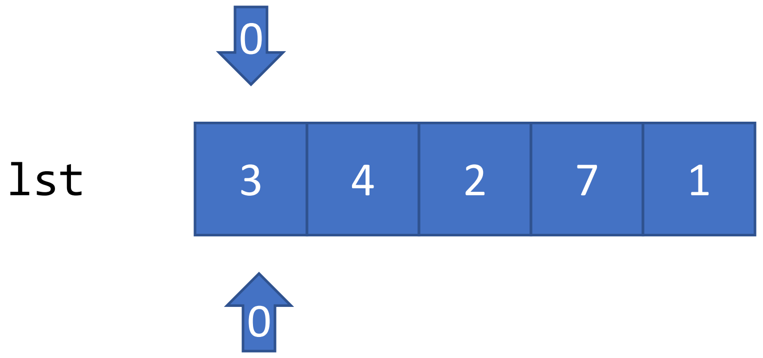


```
myindex( lst, 9)
```

## How it is done? ( one way at least)

```
def myindex( lst, x) :
  for i in range( len(lst)
  if x==lst[i] :
    return i
  return None
```

```
myindex( lst, 2)
```

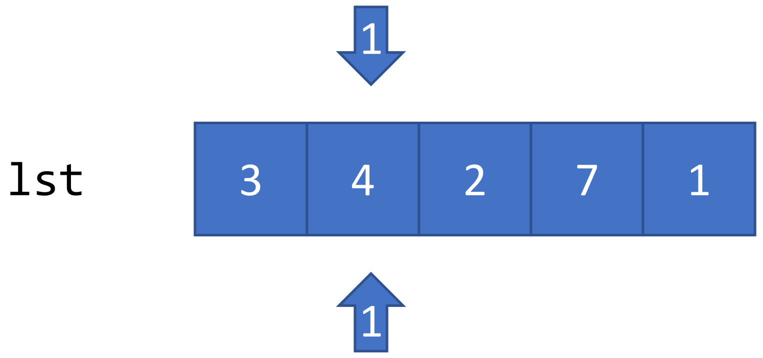


```
myindex( lst, 9)
```

## How it is done? ( one way at least)

```
def myindex( lst, x) :
  for i in range( len(lst)
  if x==lst[i] :
    return i
  return None
```

```
myindex( lst, 2)
```



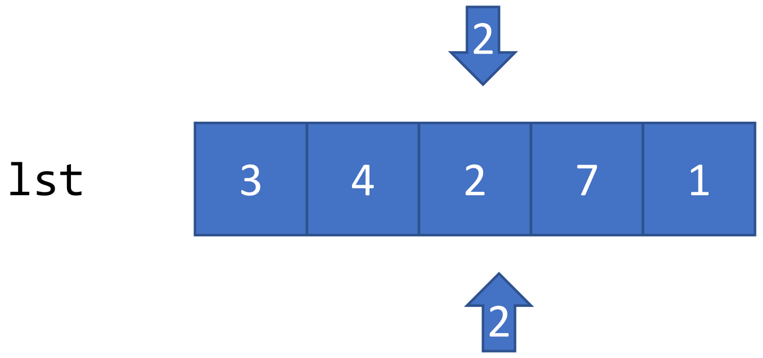
```
myindex( lst, 9)
```

# How it is done? ( one way at least)

```
def myindex( lst, x) :
  for i in range( len(lst)
  if x==lst[i] :
    return i
  return None
```

Found 2 –
return 2

```
 myindex( lst, 2)
```

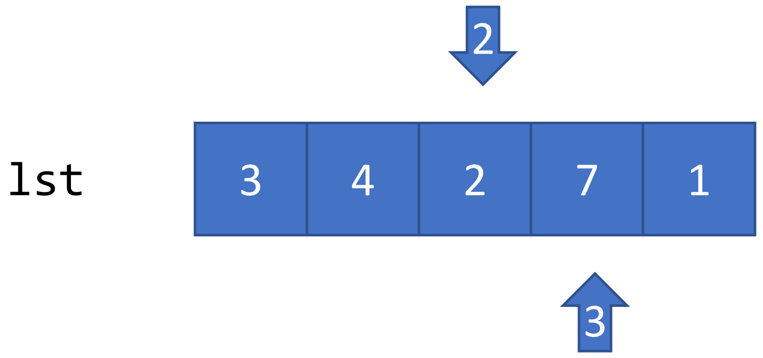


```
 myindex( lst, 9)
```

## How it is done? ( one way at least)

```
def myindex( lst, x) :
  for i in range( len(lst)
  if x==lst[i] :
    return i
  return None
```

```
    myindex( lst, 2)
```



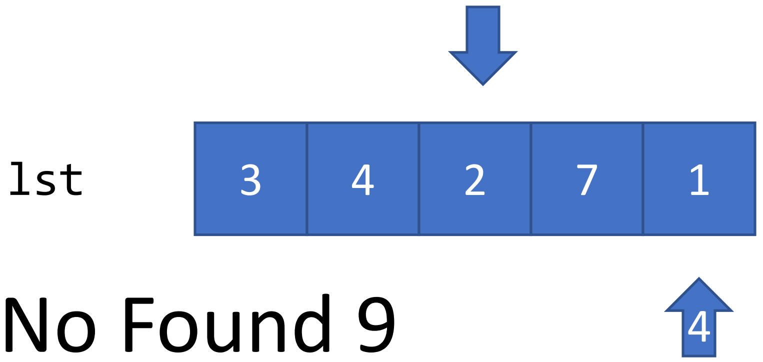
```
    myindex( lst, 9)
```

## How it is done? ( one way at least)

```
def myindex( lst, x) :
  for i in range( len(lst)
  if x==lst[i] :
    return i
  return None
```

```
 myindex( lst, 2)
```

```
 myindex( lst, 9)
```

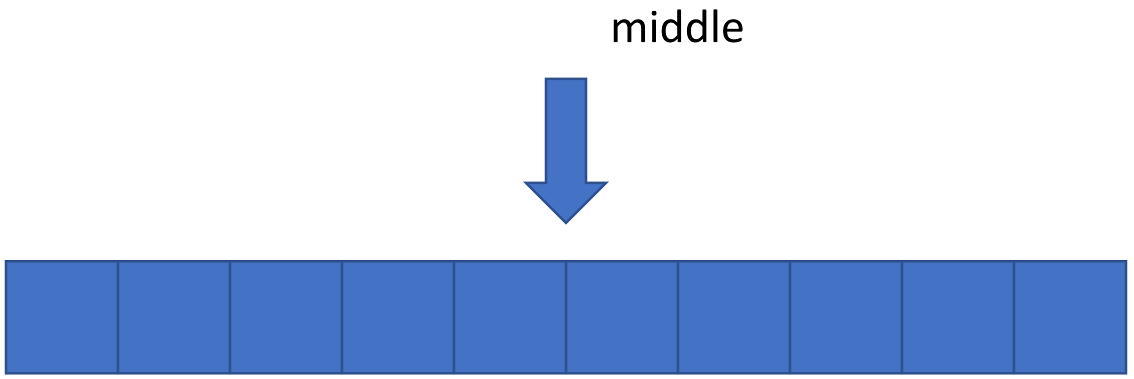


No Found 9

return None

# Binary search

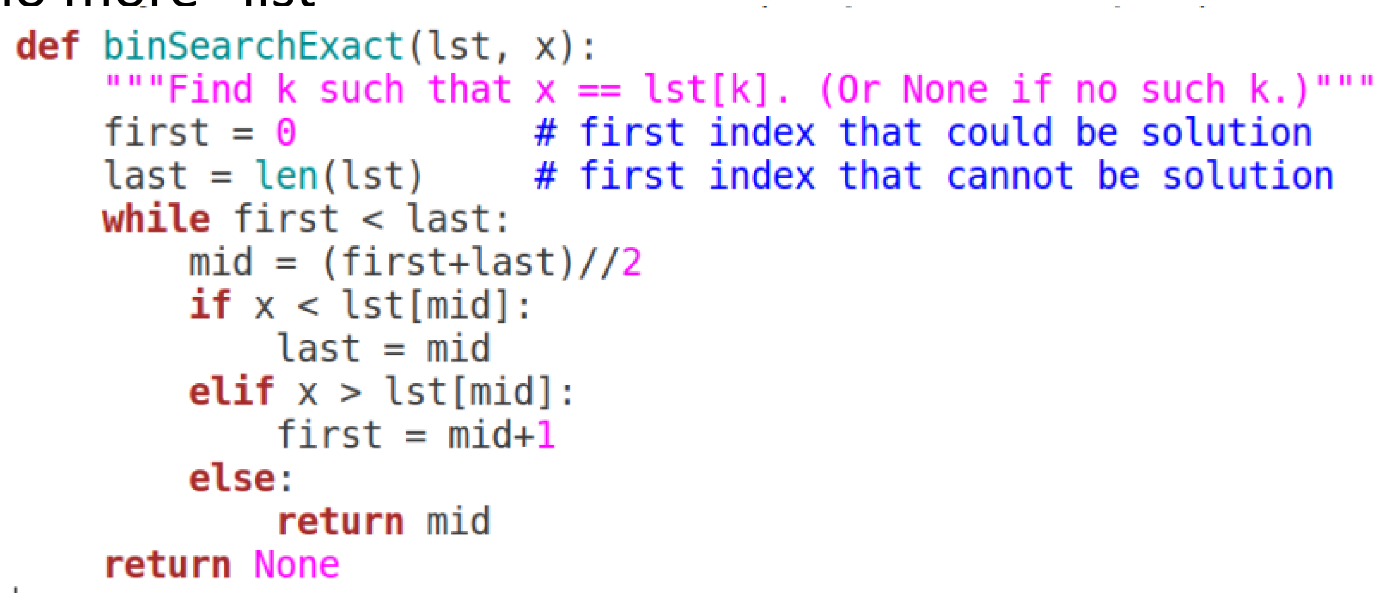
- Get to the middle
- Turn left or right
- Do it until you find what you are looking for
- Or until no more "list"



Left (>) or right (<)?

## Binary search

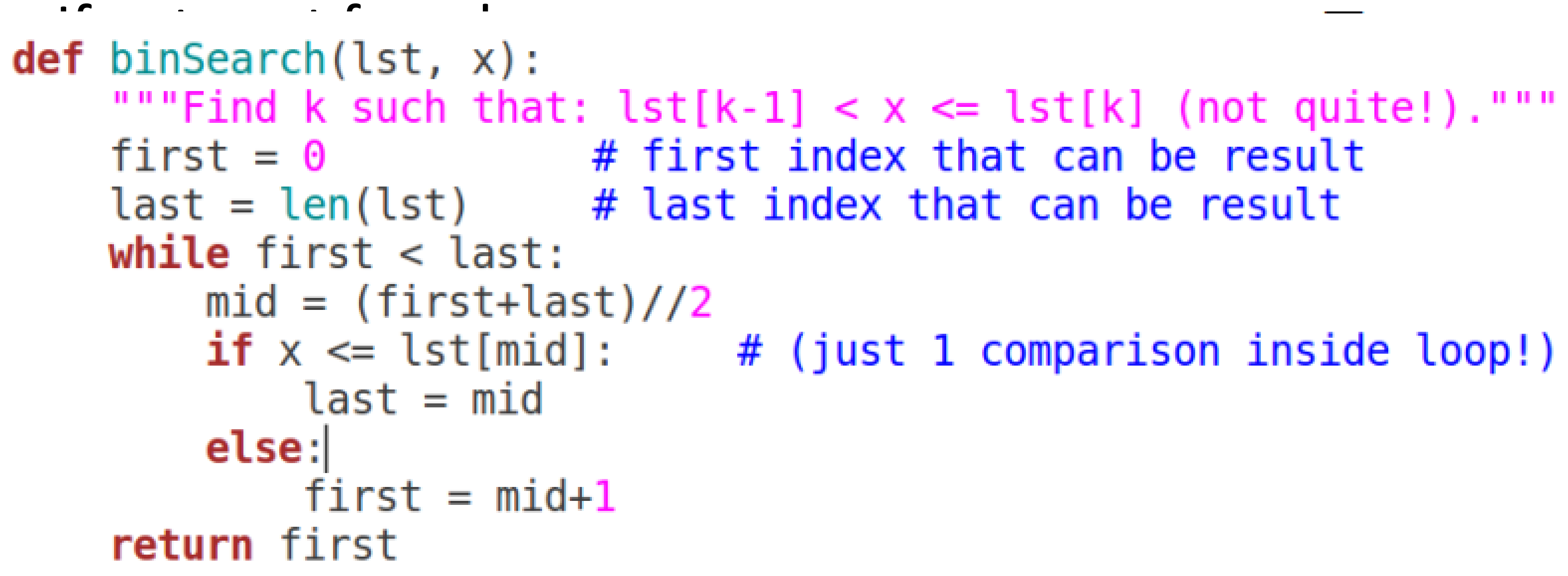
- Get to the middle
- Turn left or right
- Do it until you find what you are looking for
- Or until no more “list”



```python
def binSearchExact(lst, x):
    """Find k such that x == lst[k]. (Or None if no such k.)"""
    first = 0              # first index that could be solution
    last = len(lst)        # first index that cannot be solution
    while first < last:
        mid = (first+last)//2
        if x < lst[mid]:
            last = mid
        elif x > lst[mid]:
            first = mid+1
        else:
            return mid
    return None
```

# Binary search v2

- Look for position, see if found it in the end
- Always get position
- If value in there you found it
    - (k<len(lst) and x == lst[k], then we know x was found.)



```
def binSearch(lst, x):
    """Find k such that: lst[k-1] < x <= lst[k] (not quite!)."""
    first = 0               # first index that can be result
    last = len(lst)         # last index that can be result
    while first < last:
        mid = (first+last)//2
        if x <= lst[mid]:       # (just 1 comparison inside loop!)
            last = mid
        else:
            first = mid+1
    return first
```

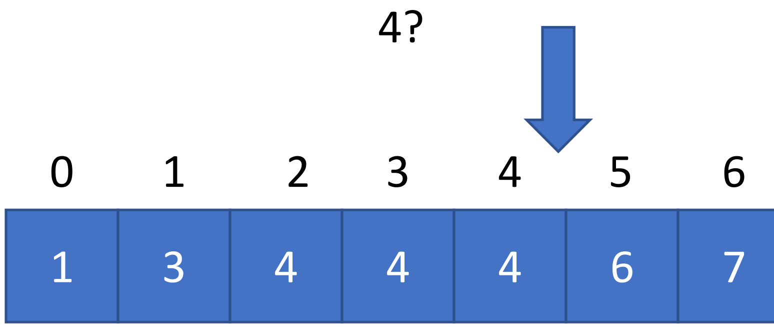
## bisect

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | 3 | 4 | 4 | 4 | 6 | 7 |

```
li = [1, 3, 4, 4, 4, 6, 7]

# using bisect() to find index to insert new element
# returns 5 ( right most possible index )
print ("The rightmost index to insert, so list remains sorted is  : ", end="")
print (bisect.bisect(li, 4))

# using bisect_left() to find index to insert new element
# returns 2 ( left most possible index )
print ("The leftmost index to insert, so list remains sorted is  : ", end="")
print (bisect.bisect_left(li, 4))

# using bisect_right() to find index to insert new element
# returns 4 ( right most possible index )
print ("The rightmost index to insert, so list remains sorted is  : ", end="")
print (bisect.bisect_right(li, 4, 0, 4))
```

https://www.geeksforgeeks.org/bisect-algorithm-functions-in-python/

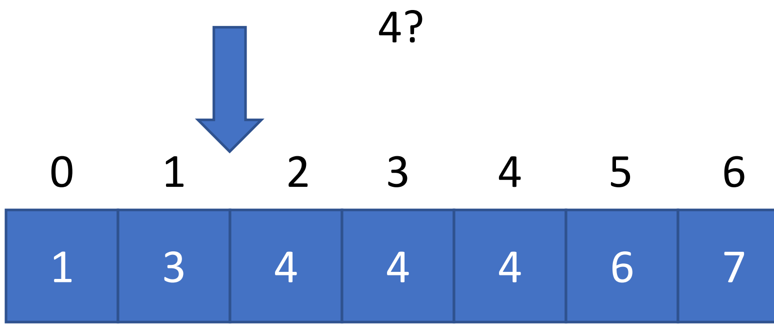
# bisect



```
li = [1, 3, 4, 4, 4, 6, 7]

# using bisect() to find index to insert new element
# returns 5 ( right most possible index )
print ("The rightmost index to insert, so list remains sorted is  : ", end="")
print (bisect.bisect(li, 4))

# using bisect_left() to find index to insert new element
# returns 2 ( left most possible index )
print ("The leftmost index to insert, so list remains sorted is  : ", end="")
print (bisect.bisect_left(li, 4))

# using bisect_right() to find index to insert new element
# returns 4 ( right most possible index )
print ("The rightmost index to insert, so list remains sorted is  : ", end="")
print (bisect.bisect_right(li, 4, 0, 4))
```

https://www.geeksforgeeks.org/bisect-algorithm-functions-in-python/

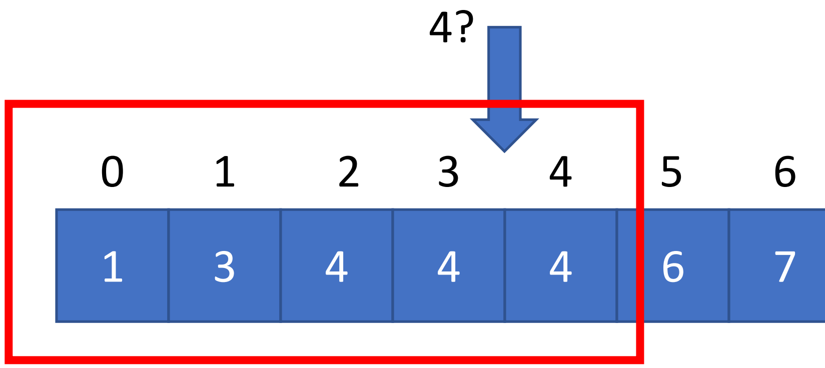
# bisect



```
li = [1, 3, 4, 4, 4, 6, 7]

# using bisect() to find index to insert new element
# returns 5 ( right most possible index )
print ("The rightmost index to insert, so list remains sorted is  : ", end="")
print (bisect.bisect(li, 4))

# using bisect_left() to find index to insert new element
# returns 2 ( left most possible index )
print ("The leftmost index to insert, so list remains sorted is  : ", end="")
print (bisect.bisect_left(li, 4))

# using bisect_right() to find index to insert new element
# returns 4 ( right most possible index )
print ("The rightmost index to insert, so list remains sorted is  : ", end="")
print (bisect.bisect_right(li, 4, 0, 4))
```

https://www.geeksforgeeks.org/bisect-algorithm-functions-in-python/

## bisect



```
li = [1, 3, 4, 4, 4, 6, 7]

# using bisect() to find index to insert new element
# returns 5 ( right most possible index )
print ("The rightmost index to insert, so list remains sorted is  : ", end="")
print (bisect.bisect(li, 4))

# using bisect_left() to find index to insert new element
# returns 2 ( left most possible index )
print ("The leftmost index to insert, so list remains sorted is  : ", end="")
print (bisect.bisect_left(li, 4))

# using bisect_right() to find index to insert new element
# returns 4 ( right most possible index )
print ("The rightmost index to insert, so list remains sorted is  : ", end="")
print (bisect.bisect_right(li, 4, 0, 4))
```

https://www.geeksforgeeks.org/bisect-algorithm-functions-in-python/

The END