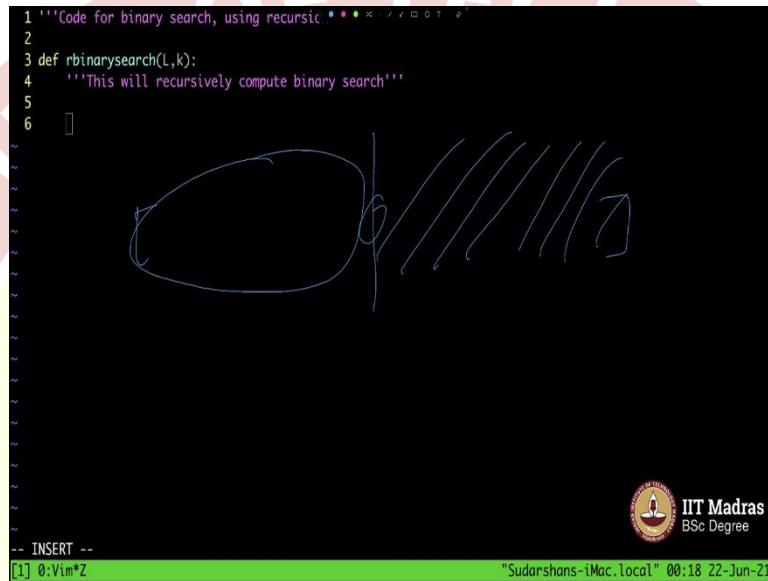




**IIT Madras**  
ONLINE DEGREE

**Programming in Python**  
**Professor Sudarshan Iyengar**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Ropar**  
**Mr. Omkar Joshi**  
**Course Instructor**  
**Online Degree Programme**  
**Indian Institute of Technology, Madras**  
**Binary Search Recursion Way**

(Refer Slide Time: 00:16)

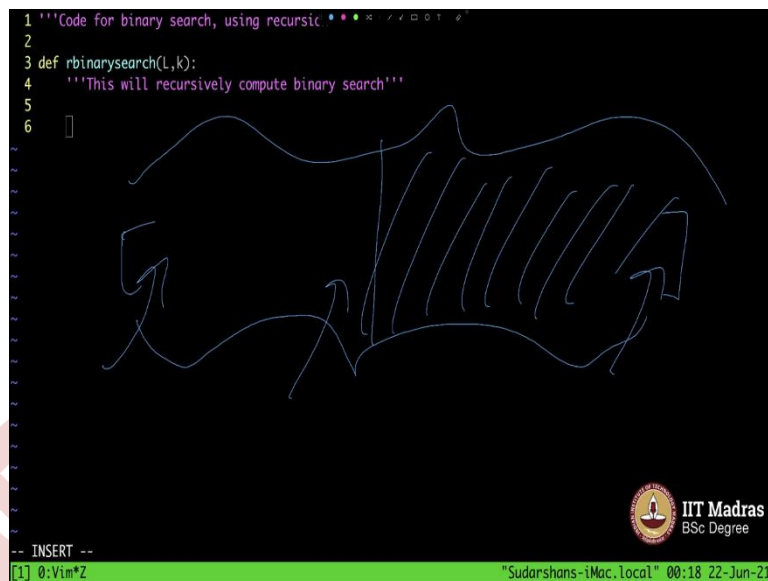


```
1 '''Code for binary search, using recursion'''
2
3 def rbinarysearch(L,k):
4     '''This will recursively compute binary search'''
5
6 ]
```

Try writing a piece of code, code for binary search, but let us use recursion, using recursion. How do I go about this? Let me think. Firstly, let me zoom this, increase the font so that it is clear to you people. What should I do? I should define my recursive binary search. I will take L and k. What should this do? This will recursively help us recursively compute binary search. So, how to go about it, let me just think. Again, the same old list. I am going to have it, remove one part and retain the region of interest, depending upon what the middle is.

(Refer Slide Time: 01:20)

```
1 '''Code for binary search, using recursic
2
3 def rbinarysearch(l,k):
4     '''This will recursively compute binary search'''
5
6     []
```



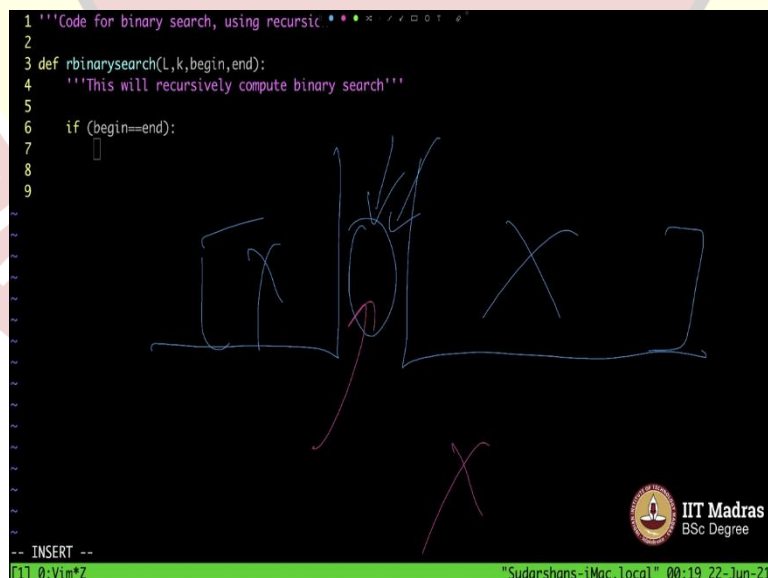
IIT Madras  
BSc Degree

-- INSERT --  
[1] 0:Vim\*Z "Sudarshans-iMac.local" 00:18 22-Jun-21

So, recursively speaking, all I will do is let me think. So, I am as blank as you people are, I am trying to build it from top down. I know, by top down, I mean, I have a big picture. I should slice it down into smaller details. So, what I will do is, binary search of an element  $k$  on this is same as binary search of element  $k$  on one half of it after discarding this. So, this was  $begin$   $end$ , and  $begin$   $end$  becomes this. The list should not be replicated, the same list goes in. But then I think I should include  $begin$  and  $end$  here is what my mind says now.

(Refer Slide Time: 01:59)

```
1 '''Code for binary search, using recursic
2
3 def rbinarysearch(l,k,begin,end):
4     '''This will recursively compute binary search'''
5
6     if (begin==end):
7         []
8
9
```



IIT Madras  
BSc Degree

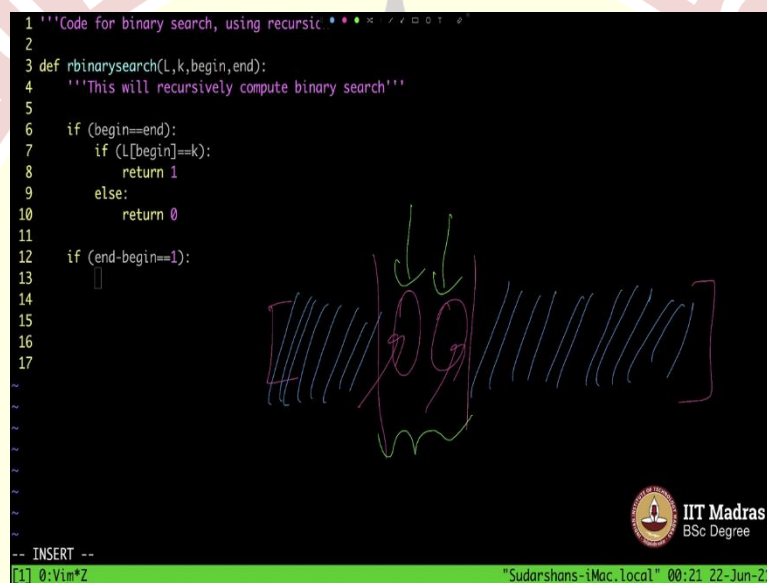
-- INSERT --  
[1] 0:Vim\*Z "Sudarshans-iMac.local" 00:19 22-Jun-21

If your mind does not say it, you do not include it, but in my mind says  $begin$  and  $end$  the region of  $hitter$  should be here. To begin with  $begin$  and  $end$  will actually be 0 and length of  $L$

minus 1, but with time it becomes smaller and smaller. So, what do I do? I should first consider the trivial cases.

What are the trivial cases? That is very easy. When if begin is equal to end, the region of interest will simply be nothing, there is only some begin is the same place end is the same place. If in case, which means these are all discarded. If in case element k is there, it must be here or nowhere. I am being a little fast, because you should be real fast now, because of we have spent a lot of time discussing the trivial codes so far, which today is trivial for you people, but all this while it was not. So, from now onwards I will go a little fast.

(Refer Slide Time: 03:04)



```
1 '''Code for binary search, using recursic
2
3 def rbinarysearch(l,k,begin,end):
4     '''This will recursively compute binary search'''
5
6     if (begin==end):
7         if (L[begin]==k):
8             return 1
9         else:
10            return 0
11
12     if (end-begin==1):
13
14
15
16
17
```

So, all I am trying to say here is if begin is equal to end and it is simply check if L of begin is equal to k. If it is true, then return 1 otherwise I return 0. That is it, case over. This is when begin is equal to end. Begin and end is the same then this happens. What if it is not true? If begin, if end minus begin is equal to 1. If end minus begin is equal to 0 that is what this means and it is equal end minus begin is 0 this happens.

When end minus begin is equal to 1, what exactly does this mean. This simply means that we have begun here and end here continuous terms. Our region of interest is this, and we have removed the other regions of interest. Let me just change that ink here. This is not a region of interest, you are only concentrating here. So, then you should find whether k is here or not, k is here or not. And boom, you are done.

(Refer Slide Time: 04:29)

```
1 '''Code for binary search, using recursion'''
2
3 def rbinarysearch(L,k,begin,end):
4     '''This will recursively compute binary search'''
5
6     if (begin==end):
7         if (L[begin]==k):
8             return 1
9         else:
10            return 0
11
12     if (end-begin==1):
13         if (L[begin]==k) or (L[end]==k):
14             return 1
15         else:
16             return 0
17
18     if (end-begin>1):
19         mid=(begin+end)//2
20
21
22
23
24
25
"rbinarysearch.py" 25L, 416B written
[1] 0:Vim*Z "Sudarshans-iMac.local" 00:22 22-Jun-21
```


So, I am, as you can see, I am thinking very slowly. This might be confusing for you people, as this keep popping. Keeps popping error that is because if your code is incomplete, it keeps showing syntax errors. Anyways, let us ignore it. So, because our code is not complete. So let us go ahead. If end minus begin is equal to 1, then I should check for if L of begin is equal to k, then it is true or if L of end is equal to k, because there are only two regions there. My region of interest itself is only begin and end, only two points. In between there is nothing. Why there is nothing it is because we have end minus begin is equal to 1. Alright.

So, if this is true, then I will return 1 else, I will return 0. It means that region of interest you do not have k. What if this is also not true? If end minus begin is greater than 1, is equal to 1 is taken care of, is equal to this thing is also taken care of. I mean, end minus begin is equal to 0 is also taken care of, is equal to 1 is taken care of. What if it is two or more? That is what I am taking care of here. That is pretty easy.

Let us see what happens. If it is more than this than I do. What do I do, I compute mid you see mid is end plus begin plus end divided by 2. Correct. So, no comments anywhere, so let me just like comments here.

(Refer Slide Time: 06:04)

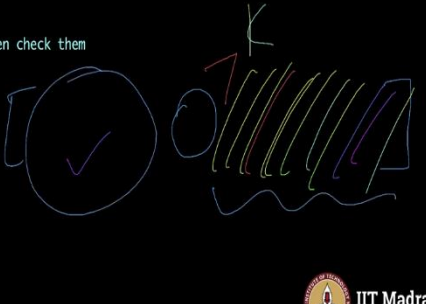
```
1 '''Code for binary search, using recursion'''
2
3 def rbinarysearch(L,k,begin,end):
4     '''This will recursively compute binary search'''
5     #if begin and end are the same, then we need to
6     #just check L[begin]
7     if (begin==end):
8         if (L[begin]==k):
9             return 1
10        else:
11            return 0
12    #if begin and end are consecutive, then check them
13    #individually.
14    if (end-begin==1):
15        if (L[begin]==k) or (L[end]==k):
16            return 1
17        else:
18            return 0
19    #if end-begin>1
20    if (end-begin>1):
21        mid=(begin+end)//2
22
23
24
25
-- INSERT --
[1] 0:Vim*Z "Sudarshans-iMac.local" 00:23 22-Jun-21
```



If, begin and end are the same, then we need to just check L of begin, that is what we are doing here. If begin and end are consecutive, consecutive means one next to the other then check them individually, that is it. But what if end minus begin is greater than 1. By this all I mean is, what if you have begin here, end here and you have at least one element in between these two things and this becomes your region of interest then these two cases are not true. We may have to search through it you see.

(Refer Slide Time: 07:06)

```
1 '''Code for binary search, using recursion'''
2
3 def rbinarysearch(L,k,begin,end):
4     '''This will recursively compute binary search'''
5     #if begin and end are the same, then we need to
6     #just check L[begin]
7     if (begin==end):
8         if (L[begin]==k):
9             return 1
10        else:
11            return 0
12    #if begin and end are consecutive, then check them
13    #individually.
14    if (end-begin==1):
15        if (L[begin]==k) or (L[end]==k):
16            return 1
17        else:
18            return 0
19    #if end-begin>1
20    if (end-begin>1):
21        #compute the middle element
22        mid=(begin+end)//2
23        if (L[mid]>k):
24
25
-- INSERT --
[1] 0:Vim*Z "Sudarshans-iMac.local" 00:25 22-Jun-21
```



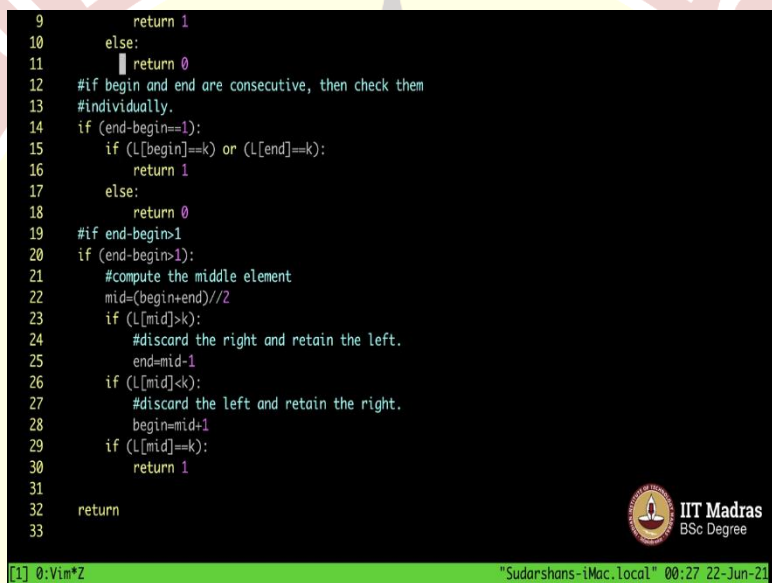
Remember in recursion, we have the trivial cases and then the not the not so trivial case, this is not so trivial case. Middle equals begin plus end by 2 by this what do I mean by this,



compute the middle element. L of mid will be the middle element. If L of mid is greater than k, what does it mean? Just like you all even I am blank at times.

By this, I mean, you may not be blank, I did not mean to belittle you. We generally think on the fly. We go slowly and we may not know everything as we are coding. So, what is this? K if L of mid is greater than k, which means middle is let us say this and L of mid whatever is here is this is greater than k, this is greater means what, k should be in this zone, and this part should be discarded. So, let me write that down, discard the right, retain the left, discard the right, retain the left. Discard the right, retain the left.

(Refer Slide Time: 08:23)



```
9      return 1
10     else:
11         return 0
12     #if begin and end are consecutive, then check them
13     #individually.
14     if (end-begin==1):
15         if (L[begin]==k) or (L[end]==k):
16             return 1
17     else:
18         return 0
19     #if end-begin>1
20     if (end-begin>1):
21         #compute the middle element
22         mid=(begin+end)//2
23         if (L[mid]>k):
24             #discard the right and retain the left.
25             end=mid-1
26         if (L[mid]<k):
27             #discard the left and retain the right.
28             begin=mid+1
29         if (L[mid]==k):
30             return 1
31
32     return
33
```

[1] 0:Vim\*Z "Sudarshans-iMac.local" 00:27 22-Jun-21

Discard the right and retain the left. That is what this means, if L of mid is greater than k. How do I do that? That is very easy. I simply say how do I discard the right, retain the left begin remains the same, your end becomes mid minus 1, that is how you retain the left and discard the right we have discussed this.

And what if this is not true? If L of mid is less than k, then discard the left and retain the right. Do not worry about this. So far, so good. So, this takes care of the situation where L of mid is greater than k, L of mid is less than k, but what if L of mid is equal to k, then hip hip hurray. What I do here, I should write this. How do I discard the left and retain the right? I discard the left by saying my begin is equal to mid plus 1. Correct?

That is only obvious from our from our previous case I mean, from our previous program, we have been seeing this is all in our mind right now. Anyways, so I will just tell you what is

happening here. Whenever I save it, it says there is an error here because the code is incomplete.

This is how I have configured it, and I say L close, and this closes, but do not worry in your editor, this may not be a problem. I should stop saving this as often as I do. So, L of mid is equal to k means what? I am done, return 1. And back home we are done it is over. If L of mid is equal to k, it is less than k then you retain the right if it is greater than k you retain the left. Good. So, what do we do next? This is not a while loop, this is just an if loop. All I do here is I return the function right now it is the function.

(Refer Slide Time: 10:47)



```
4 '''This will recursively compute binary search'''
5 #if begin and end are the same, then we need to
6 #just check L[begin]
7 if (begin==end):
8     if (L[begin]==k):
9         return 1
10    else:
11        return 0
12 #if begin and end are consecutive, then check them
13 #individually.
14 if (end-begin==1):
15     if (L[begin]==k) or (L[end]==k):
16         return 1
17    else:
18        return 0
19 #if end-begin>1
20 if (end-begin>1):
21     #compute the middle element
22     mid=(begin+end)//2
23     if (L[mid]>k):
24         #discard the right and retain the left.
25         end=mid-1
26     if (L[mid]<k):
27         #discard the left and retain the right.
28         begin=mid+1
29     if (L[mid]==k):
30         return 1
31 if (end-begin<0):
32     return 0
33
34 return rbinarysearch(L,k,begin,end)
35
```

srshyengar@Sudarshans-iMac pod % ipython  
Python 3.9.5 (default, May 4 2021, 03:36:27)  
Type 'copyright', 'credits' or 'license' for more information  
IPython 7.24.1 -- An enhanced Interactive Python. Type '?' for help.

In [1]: import rbinarysearch  
In [2]: rbinarysearch.rbinarysearch([1,7,10,16,100,100,100,100],7,0,6  
...: )  
Out[2]: 1  
In [3]: rbinarysearch.rbinarysearch([1,7,10,16,100,100,100,100],-1,0,  
...: 6)  
Out[3]: 0  
In [4]: L=list(range(1000\*1000\*100))  
In [5]: rbinarysearch.rbinarysearch(L,1000000,0,len(L)-1)  
Out[5]: 1  
In [6]: rbinarysearch.rbinarysearch(L,1000000,0,len(L)-1)  
Out[6]: 1  
In [7]: rbinarysearch.rbinarysearch(L,-1,0,len(L)-1)  
Out[7]: 0  
In [8]:

IIT Madras  
BSc Degree

"Sudarshans-iMac.local" 00:32 22-Jun-21

I should return the function r binary search L, k begin and end, my begin and industry right now. Return r binary search L and k with my begin and end, these two are now different. Now, I save and there is no error. So, let me just think through it. Am I right?



(Refer Slide Time: 11:08)

```
6  #just check L[begin]
7  if (begin==end):
8      if (L[begin]==k):
9          return 1
10     else:
11         return 0
12  #if begin and end are consecutive, then check them
13  #individually.
14  if (end-begin==1):
15      if (L[begin]==k) or (L[end]==k):
16          return 1
17      else:
18          return 0
19  #if end-begin>1
20  if (end-begin>1):
21      #compute the middle element
22      mid=(begin+end)//2
23      if (L[mid]>k):
24          #discard the right and retain the left.
25          end=mid-1
26      if (L[mid]<k):
27          #discard the left and retain the right.
28          begin=mid+1
29      if (L[mid]==k):
30          return 1
```

IIT Madras  
BSc Degree

0:Vim\*Z "Sudarshans-iMac.local" 00:28 22-Jun-21

What is the situation, what is the case? When end minus begin is negative? can that happen? It can as well happen. When you end up say end equals mid minus 1 you are not decrementing mid, end may become less than begin also at times we do not know. So, maybe we can write another condition here.

(Refer Slide Time: 11:29)

```
10     else:
11         return 0
12  #if begin and end are consecutive, then check them
13  #individually.
14  if (end-begin==1):
15      if (L[begin]==k) or (L[end]==k):
16          return 1
17      else:
18          return 0
19  #if end-begin>1
20  if (end-begin>1):
21      #compute the middle element
22      mid=(begin+end)//2
23      if (L[mid]>k):
24          #discard the right and retain the left.
25          end=mid-1
26      if (L[mid]<k):
27          #discard the left and retain the right.
28          begin=mid+1
29      if (L[mid]==k):
30          return 1
31  if (end-begin
32
33  return rbinarysearch(L,k,begin,end)
34
```


IIT Madras  
BSc Degree

0:Vim\*Z "Sudarshans-iMac.local" 00:28 22-Jun-21

Say if end minus begin is less than 0.

(Refer Slide Time: 11:39)

```
5  #if begin and end are the same, then we need to
6  #just check L[begin]
7  if (begin==end):
8      if (L[begin]==k):
9          return 1
10     else:
11         return 0
12 #if begin and end are consecutive, then check them
13 #individually.
14 if (end-begin==1):
15     if (L[begin]==k) or (L[end]==k):
16         return 1
17     else:
18         return 0
19 #if end-begin>1
20 if (end-begin>1):
21     #compute the middle element
22     mid=(begin+end)//2
23     if (L[mid]>k):
24         #discard the right and retain the left.
25         end=mid-1
26     if (L[mid]<k):
27         #discard the left and retain the right.
28         begin=mid+1
29     if (L[mid]==k):
```


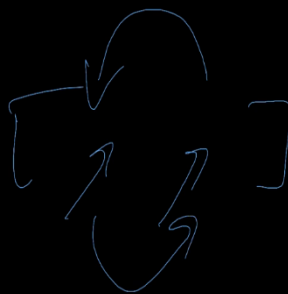


[1] 0:Vim\*Z "Sudarshans-iMac.local" 00:28 22-Jun-21

We have greater than 1 is taken care of is equal to 1 is taken care of this is equal to 0 basically. End minus begin is equal to 0 is this, as I keep saying, but end minus begin is equal to 0 is weird way to write it, so I write it as begin is equal to end.

(Refer Slide Time: 11:51)

```
10     else:
11         return 0
12 #if begin and end are consecutive, then check them
13 #individually.
14 if (end-begin==1):
15     if (L[begin]==k) or (L[end]==k):
16         return 1
17     else:
18         return 0
19 #if end-begin>1
20 if (end-begin>1):
21     #compute the middle element
22     mid=(begin+end)//2
23     if (L[mid]>k):
24         #discard the right and retain the left.
25         end=mid-1
26     if (L[mid]<k):
27         #discard the left and retain the right.
28         begin=mid+1
29     if (L[mid]==k):
30         return 1
31 if (end-begin<0):
32     return 0
33
34 return rbinarysearch(l,k,begin,end)
-- INSERT --
```



[1] 0:Vim\*Z "Sudarshans-iMac.local" 00:28 22-Jun-21

So, if end minus begin is negative then no, you have not found that is what it means end point is begin as negative means what? What, what, what, what does it mean? End begin should be here end should be here. When end becomes comes decide and begin comes this side it means that there is there is no region of interest at all.

(Refer Slide Time: 12:14)

```
4 '''This will recursively compute binary search'''
5 #if begin and end are the same, then we need to
6 #just check L[begin]
7 if (begin==end):
8     if (L[begin]==k):
9         return 1
10    else:
11        return 0
12 #if begin and end are consecutive, then check them
13 #individually.
14 if (end-begin==1):
15     if (L[begin]==k) or (L[end]==k):
16         return 1
17    else:
18        return 0
19 #if end-begin>1
20 if (end-begin>1):
21     #compute the middle element
22     mid=(begin+end)//2
23     if (L[mid]>k):
24         #discard the right and retain the left.
25         end=mid-1
26     if (L[mid]<k):
27         #discard the left and retain the right.
28         begin=mid+1
29     if (L[mid]==k):
30         return 1
31     if (end-begin<0):
32         return 0
33
34 return rbinarysearch(L,k,begin,end)
35
```

```
srsiyengar@Sudarshans-iMac pod % ipython
Python 3.9.5 (default, May  4 2021, 03:36:27)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.24.1 -- An enhanced Interactive Python. Type '?' for help.

In [1]: import rbinarysearch

In [2]: rbinarysearch.rbinarysearch([1,7,10,16,100,108,1008],7,0,6
...: )
Out[2]: 1

In [3]: rbinarysearch.rbinarysearch([1,7,10,16,100,108,1008],-1,0,
...: )
Out[3]: 0

In [4]: L=list(range(1000*1000*100))

In [5]: rbinarysearch.rbinarysearch(L,1000000,0,len(L)-1)
Out[5]: 1

In [6]: rbinarysearch.rbinarysearch(L,1000000,0,len(L)-1)
Out[6]: 1

In [7]: rbinarysearch.rbinarysearch(L,-1,0,len(L)-1)
Out[7]: 0

In [8]:
```

IIT Madras  
BSc Degree

[1] 0:Vin\* "Sudarshans-iMac.local" 00:32 22-Jun-21

So, what I do is I should return 0. I hope, this works. I have no clue. Let us see now. This again, the entire code I have typed it on the fly, very possibility there will be some errors, but let us see. Let me reduce the font size just so that you can see both sides. import rbinarysearch, good, rbinarysearch and rbinarysearch put some list 1 comma, 7 comma 10 comma 16 comma 100, comma 108 comma 1008 so on, and such for the element, let us say I will search for 7 here.

Begin should be 0 obviously, end should be the length of this list, which is 1, 2, 3, 4, 5, 6, 7 which is 6 len of that minus 1. Good. It gives me the answer. My code seems to be right. But what if I type something that is not here? It is not there. Wonderful. My code is working really well. Good.

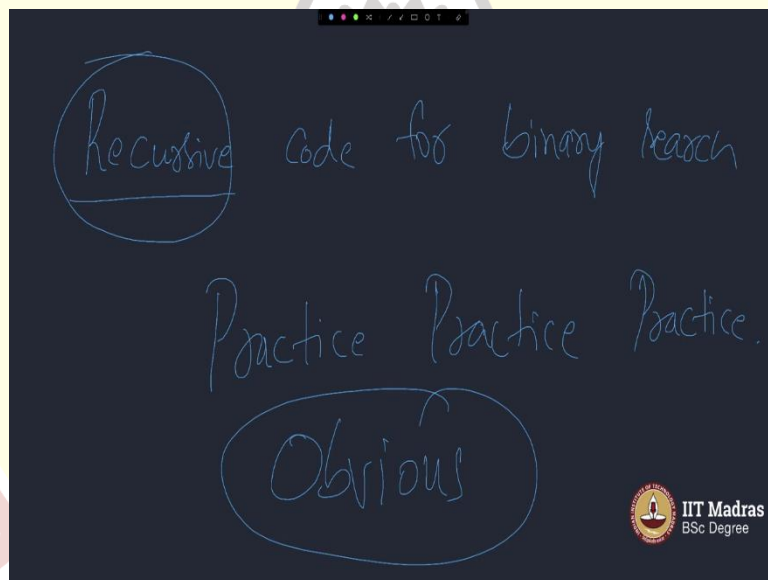
Now, let us do the same old my list is equal to list of what is that range of 100 million. If you are wondering why, I write it like this, it is easy on the mind. 1000 times 1000 is million times 100 is 1 million, we all have our own style of doing a few things. This is my style. You will find it very silly though.

So, that computes it. You can even do it for a billion if you remember, last time we did it for a billion it took its own time. It took months to stop, so I would not type for 1 billion I will try it 100 million. So, I am now going to search for rbinarysearch on my list L, I want to find the number let us say 1 million, which is there in the list by the way, starting from 0 and ending my end is this, why, because we saw in the previous video. I will not get there I mean, it is very easy.

This is begin and end. You start from region of interest 0 to the entire list basically, then boom. It is very, very fast. You see, very quickly it gave me the answer. But is it better than the obvious search? So, let us put something that is not there. Obvious search will take eternity. You saw that it takes a few seconds but beautiful, it very quickly says, no it is not present, as you can see. Alright, wonderful. So, we used recursion to solve the problem as you saw, it is actually easy on the mind.

If you still think, even I am stumbling, programming involves programmer to stumble. Nobody confidently writes a piece of code. I think 10 years from now, if I keep coding the same program for binary search recursively, I will still stumble a little that is how programming happens. You just think logically, I mean, that is not so straightforward. I mean, it is okay to stumble here and there and slowly build logic. Good.

(Refer Slide Time: 15:52)



So, what have we seen, we have seen a recursive program for recursive program for one second, let me just open it. So, we just saw a recursive code for binary search. We also saw a non-recursive code. And with the help of that we built a recursive code for binary search. We use recursion a very powerful technique to illustrate, it can actually be easy on the mind when you use recursion.

As and always, please practice, practice and practice. Try writing this code all by yourself. First time when I write it, it may sound very difficult, but trust me, all that we did was very, very obvious. It is as simple as referring to a dictionary. How is it referring to the dictionary, let me illustrate.

(Refer Slide Time: 16:50)



How is this referring to the dictionary? It is because in a dictionary, if there are 1,000 pages, the problem of finding a word in 1,000 pages is same as the problem of finding in the first 500 page after you have realized that you have discarded the other 1,000 pages.

The problem now boils down to you finding the word here. And then then again, you just halve it and discard one half depends on where the word is and this will become your region of interest. And then again, that gets halved and so on and so forth. Recursively, you argue here, and that is precisely what we did just now.

I hope, it was clear. If it was not clear, please see the code once again, redo it, as I am explaining. You type it, open the terminal and then do it, you will have errors, you will have lack of clarity, but with time you will achieve clarity. Thank you very much. That ends this week's discussion with recursion.

(Refer Slide Time: 17:57)

```
7 if (begin==end):
8     if (L[begin]==k):
9         return 1
10    else:
11        return 0
12    #if begin and end are consecutive, then check them
13    #individually.
14    if (end-begin==1):
15        if (L[begin]==k) or (L[end]==k):
16            return 1
17        else:
18            return 0
19    #if end-begin>1
20    if (end-begin>1):
21        #compute the middle element
22        mid=(begin+end)//2
23        if (L[mid]==k):
24            #discard the right and retain the left.
25            end=mid-1
26        if (L[mid]<k):
27            #discard the left and retain the right.
28            begin=mid+1
29        if (L[mid]==k):
30            return 1
31    if (end-begin<0):
32        return 0
33
34    return rbinarysearch(L,k,begin,end)
35
36
37
38
```

```
srsiyengar@Sudarshans-iMac pod % ipython
Python 3.9.5 (default, May 4 2021, 03:36:27)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.24.1 -- An enhanced Interactive Python. Type '?' for help.

In [1]: import rbinarysearch

In [2]: rbinarysearch.rbinarysearch([1,7,10,16,100,108,1008],7,0,6
...: )
Out[2]: 1

In [3]: rbinarysearch.rbinarysearch([1,7,10,16,100,108,1008],-1,0,
...: )
Out[3]: 0

In [4]: L=list(range(1000*1000*100))

In [5]: rbinarysearch.rbinarysearch(L,1000000,0,len(L)-1)
Out[5]: 1

In [6]: rbinarysearch.rbinarysearch(L,1000000,0,len(L)-1)
Out[6]: 1

In [7]: rbinarysearch.rbinarysearch(L,-1,0,len(L)-1)
Out[7]: 0

In [8]:
```

IIT Madras  
BSc Degree

I only have a small thing to illustrate right now, which I will illustrate very quickly the dangers of recursion, especially in Python.

(Refer Slide Time: 18:06)

```
In [9]: reset
Once deleted, variables cannot be recovered. Proceed (y/[n])? y

In [10]: def sum(n):
...:     if (n==0):
...:         return 0
...:     else:
...:         return n+sum(n-1)
...:

In [11]: sum(10)
Out[11]: 55

In [12]: sum(0)
Out[12]: 0

In [13]: sum(1)
Out[13]: 1

In [14]: sum(2)
Out[14]: 3

In [15]: sum(3)
Out[15]: 6

In [16]: sum(4)
Out[16]: 10

In [17]: sum(900)
Out[17]: 405450

In [18]:
```

IIT Madras  
BSc Degree

So let me zoom this. Clear this, and reset this. So, you remember, we tried sorting using Python and recursively. We tried defining sum of n numbers and we used recursion to do this. How did we do it? We started off with, how was that? We said, if n is equal to 0 then return 0 else return n plus sum of n minus 1.

Let us see if this works or not. Sum of 10 is 55, beautiful. Sum of 0 should be 0. Sum of 1 should be 1, sum of 2 should be 3, sum of 3 should be 3 plus 2 plus 1, 6, sum of 4 should be 4



plus 3 plus 2 plus 1 is equal to 10 and so on and so forth. Now, see what will happen when sum of 900 is computed, it gives the answer. Very good.

(Refer Slide Time: 19:20)

```
In [17]: sum(900)
Out[17]: 405450

In [18]: sum(950)
Out[18]: 451725

In [19]: sum(990)
Out[19]: 490545

In [20]: sum(999)
Out[20]: 499500

In [21]: sum(1000)
Out[21]: 500500

In [22]: sum(1001)
Out[22]: 501501

In [23]: sum(1002)
Out[23]: 502503

In [24]: sum(1003)
Out[24]: 503506

In [25]: 
[1] 0:Python*Z "Sudarshans-iMac.local" 00:36 22-Jun-21
```

Let me zoom in. What happens if I say 950, it computes the first 950 numbers and gives me the answer. What if I say 990, it still it works, 999, it works. What if I say 1,000 it works, 1001, works, 1002 works, 1003 works.

(Refer Slide Time: 19:55)

```
In [24]: sum(1003)
Out[24]: 503506

In [25]: sum(100000)

RecursionError                                Traceback (most recent call last)
<ipython-input-25-b0c222b0a3d3> in <module>
----> 1 sum(100000)

<ipython-input-10-90034c06e597> in sum(n)
      3     return 0
      4     else:
----> 5         return n+sum(n-1)
      6

... last 1 frames repeated, from the frame below ...

<ipython-input-10-90034c06e597> in sum(n)
      3     return 0
      4     else:
----> 5         return n+sum(n-1)
      6

RecursionError: maximum recursion depth exceeded in comparison

In [26]: 
[1] 0:Python*Z "Sudarshans-iMac.local" 00:36 22-Jun-21
```

When I say sum, let us say how much 100,000 it says maximum recursion depth exceeded in comparison. You cannot go on like this. There is a stage after which it cannot do more recursion. On your computers, the limit by default is 999. I was expecting that here too, but it

went beyond a few, beyond the like actual recursion limit, but otherwise the recursion limit is simply 999.

(Refer Slide Time: 20:32)

```
In [19]: sum(990)
Out[19]: 490545

In [20]: sum(999)
Out[20]: 499500

In [21]: sum(1000)
Out[21]: 500500

In [22]: sum(1001)
Out[22]: 501501

In [23]: sum(1002)
Out[23]: 502503

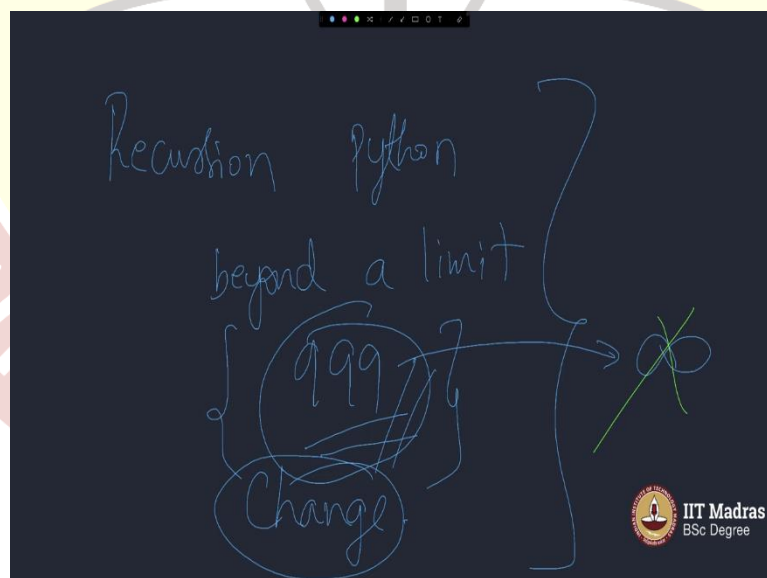
In [24]: sum(1003)
Out[24]: 503506

In [25]: sum(100000)
-----
RecursionError                                Traceback (most recent call last)
<ipython-input-25-b0c222b0a3d3> in <module>
----> 1 sum(100000)

<ipython-input-10-90034c06e597> in sum(n)
3     return 0
```

In fact, this will not work on your computer, when you say sum of 1000, it will spit out an error. So, what is it that I am trying to say?

(Refer Slide Time: 20:43)



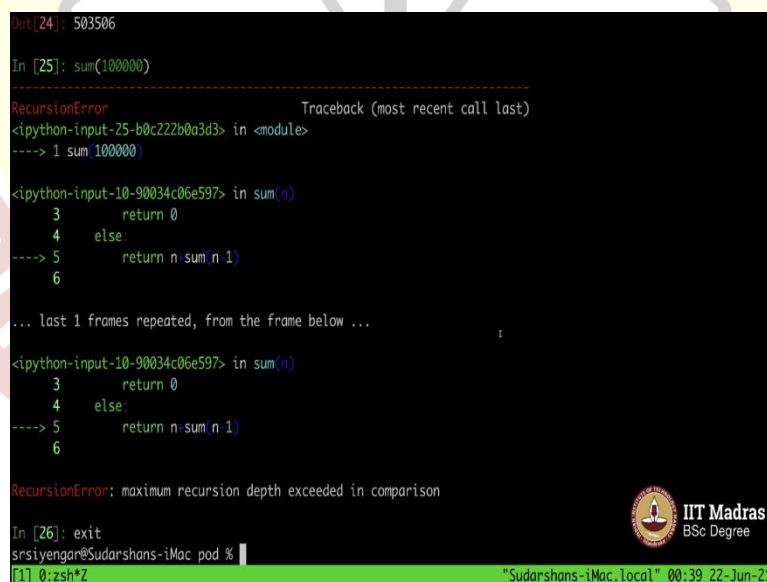
I am trying to say that recursion will not always work in Python, beyond the limit. Beyond limit, the default limit is actually 999, but you can change this. It is very easy to change, I do not want to get into technicalities it is pretty easy. It is just a one-line command, you must basically import a function called sys library function called sys and then there is a one-line command.

You figure it out, but then point to note recursion cannot go on and on. In binary search, we used recursion, it did not cross the limit, so it worked. Well, if it crosses the limit, it will throw out an error, which is okay. This is purposefully done because recursion sometimes can keep going on and on and on and on, not stop.

So, python puts a check point that 999 is a limit. You better know it says Python. If you want more, you must change it, you must meddle with our settings and then change the depth. And by the way, you cannot change it to infinity because you know cannot change it to infinity for a simple reason that you are limited by your computer resources. I mean, so you cannot go beyond what your memory can permit.

Recursion is a very complicated process, it is important for us to put out this is called a nudge effect. Purposefully, you limit, so the idea is assume you are working on your calories. I ensure that I do not give you beyond 1,000 calories, you come for dinner, I say, look, 1,000 calories are done. Now, if you want more, you must give it to me in writing that you want more food. It is very mean for me to do that. I am just giving you like as example that is what Python does. It does not let you go beyond a limit. Did not understand what I said, do not break your head much.

(Refer Slide Time: 22:41)



```
Out[24]: 503506
In [25]: sum(100000)
-----
RecursionError                                Traceback (most recent call last)
<ipython-input-25-b0c22b0a3d3> in <module>
----> 1 sum(100000)

<ipython-input-10-90034c06e597> in sum(n)
      3     return 0
      4     else:
----> 5         return n + sum(n-1)
      6

... last 1 frames repeated, from the frame below ...

<ipython-input-10-90034c06e597> in sum(n)
      3     return 0
      4     else:
----> 5         return n + sum(n-1)
      6

RecursionError: maximum recursion depth exceeded in comparison

In [26]: exit
srshyengar@Sudarshans-iMac pod %
[1] 0:zsh*Z
```

If you understood what was binary search recursively that should be more than enough. Let me exit the terminal and big bye to you all for this week. We are done with recursion. We are done with binary search. I had to do file handling, but we have covered quite a bit right now. I will skip file handling and do it the next week. So, time to say thank you all very much. See you next week. Bye bye.