INDIAN INSTITUTE OF TECHNOLOGY MADRAS

सिद्धिर्भवति कर्मजा

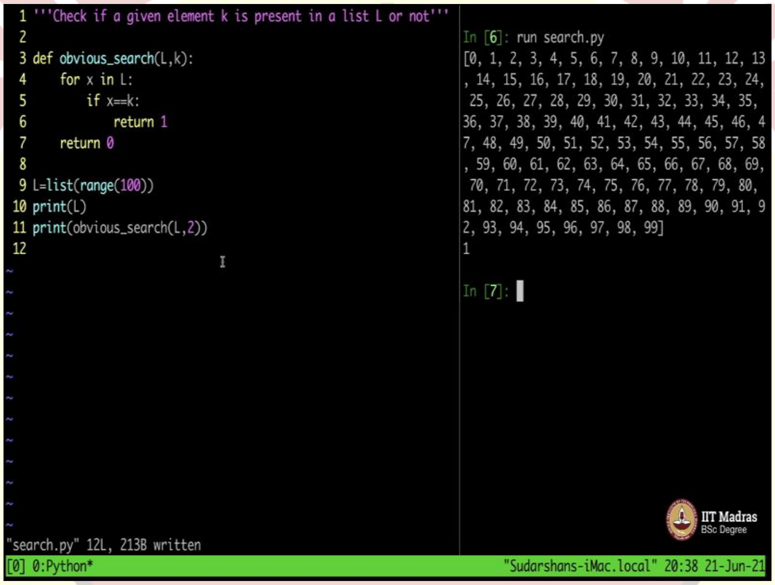
IIT Madras

ONLINE DEGREE

Programming in Python
Professor Sudarshan Iyengar
Department of Computer Science & Engineering
Indian Institute of Technology, Ropar
Mr. Omkar Joshi
Course Instructor
Online Degree Programme
Indian Institute of Technology, Madras

Warm up for Binary Search

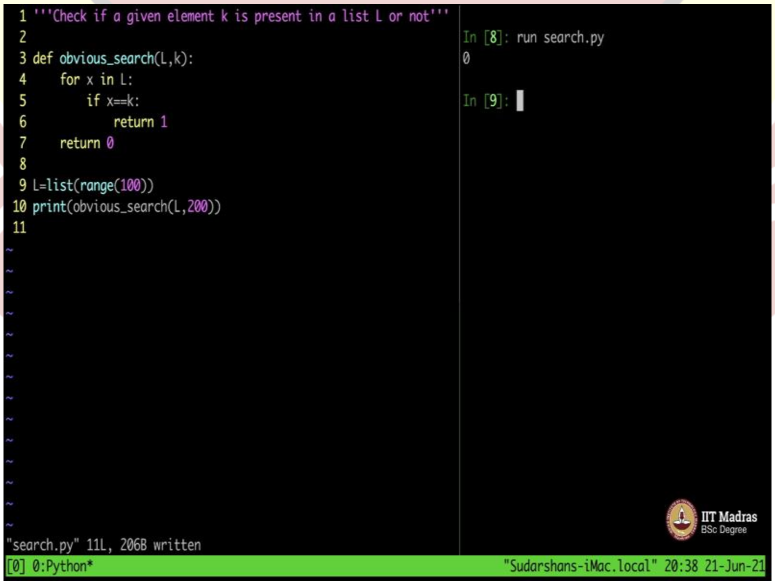
(Refer Slide Time: 00:16)



```
'''Check if a given element k is present in a list L or not'''

def obvious_search(L,k):
    for x in L:
        if x==k:
            return 1
    return 0

L=list(range(100))
print(L)
print(obvious_search(L,2))

"search.py" 12L, 213B written
```

```
In [6]: run search.py
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13
, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24,
 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35,
36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 4
7, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58
, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69,
 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80,
81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 9
2, 93, 94, 95, 96, 97, 98, 99]
1

In [7]:
```

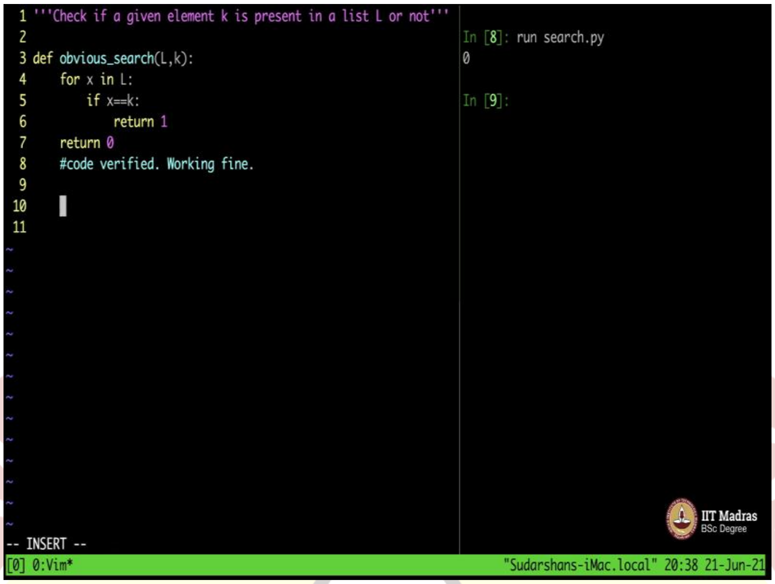
```
[0] 0:Python*                    "Sudarshans-iMac.local" 20:38 21-Jun-21
```



```
'''Check if a given element k is present in a list L or not'''

def obvious_search(L,k):
    for x in L:
        if x==k:
            return 1
    return 0

L=list(range(100))
print(obvious_search(L,200))

"search.py" 11L, 206B written
```

```
In [8]: run search.py
0

In [9]:
```

```
[0] 0:Python*                    "Sudarshans-iMac.local" 20:38 21-Jun-21
```



```
'''Check if a given element k is present in a list L or not'''

def obvious_search(L,k):
    for x in L:
        if x==k:
            return 1
    return 0
    #code verified. Working fine.


```

```
In [8]: run search.py
0

In [9]:
```

```
-- INSERT --
[0] 0:Vim*                                    "Sudarshans-iMac.local" 20:38 21-Jun-21
```

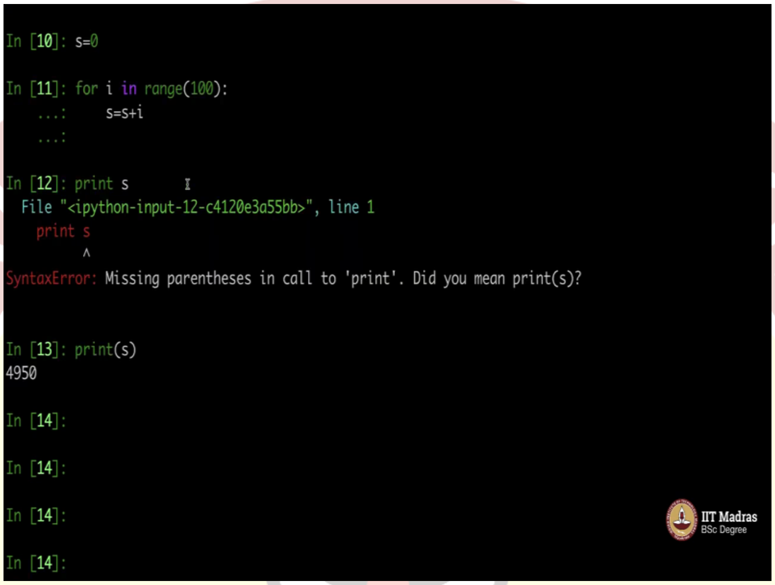
So, let us now try writing a piece of code which will check if a given element k is present in a list L or not. A pretty straightforward code, let us call it the obvious search. In fact, we have done this already. So, I will be a little fast here. So, what do we do? For x in L, you go through all the elements of L. And if in case x is equal to k, then you return true.

And then return 0 here. As you can see, the return 1 here returns true only when you encounter x equals k. When you see some element of L, I mean, if k is encountered in L, it returns 1, if not, it returns 0. So, point to note here is that the moment it returns 1, keep it in mind, in your mind, that it stops. The moment this happens it stops. It never comes down to the next one. It just stops there. So, I hope this works well.

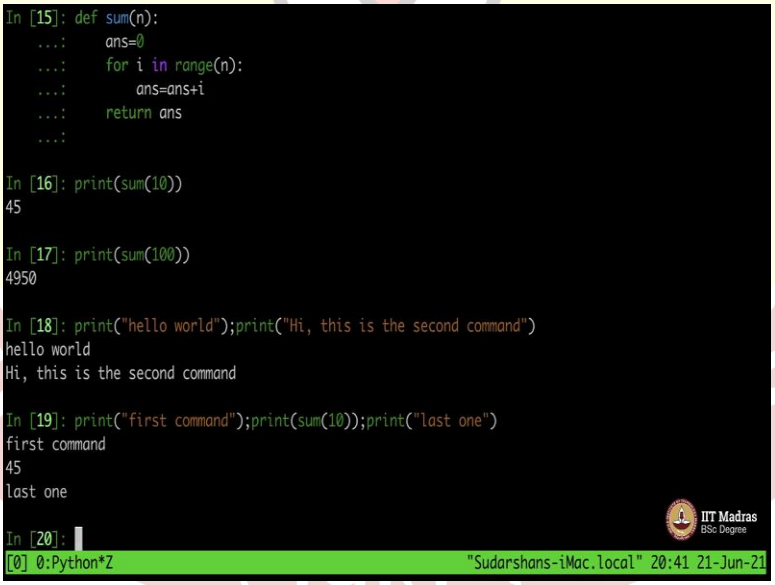
Let us try seeing this with execution of the code. So, what do you do? I say L equals list of range of some 100. Let me print L just so that you see that you have created the list. And what I do is I will also print obvious search L, let me see the element 2, which obviously belongs to L. Let us see. Let me run this search.py. That is the file name. This is how you run on command prompt. As I said, this is not Spyder. This is a straightforward interactive Python, IPython. So, you see run search.py, it displays your what the list. As this reads list range of 100 displays it, it prints L rather, I am sorry, this prints L and then prints obvious search L comma 2. 2 is present, so it returns 1.

So, what we will now do is maybe remove this and then try to see if it says 0 for this. 200 is not present here. Let me clear the screen and then try to execute the code and see yes, it is saying 0, very good. So, I will come here. Code verified, working fine on the examples given.

(Refer Slide Time: 03:30)



```
In [10]: s=0

In [11]: for i in range(100):
    ...:     s=s+i
    ...:

In [12]: print s
  File "<ipython-input-12-c4120e3a55bb>", line 1
    print s
          ^
SyntaxError: Missing parentheses in call to 'print'. Did you mean print(s)?


In [13]: print(s)
4950

In [14]:

In [14]:

In [14]:

In [14]:
```



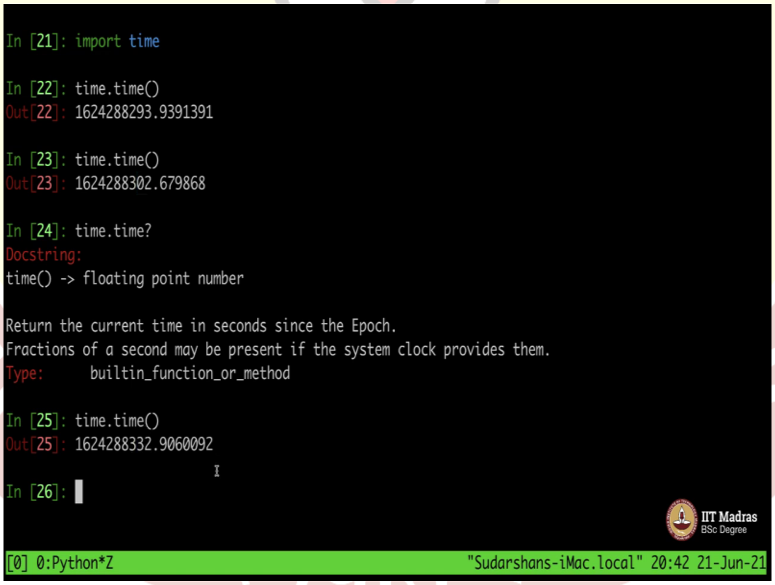
```
In [15]: def sum(n):
    ...:     ans=0
    ...:     for i in range(n):
    ...:         ans=ans+i
    ...:     return ans
    ...:

In [16]: print(sum(10))
45

In [17]: print(sum(100))
4950

In [18]: print("hello world");print("Hi, this is the second command")
hello world
Hi, this is the second command

In [19]: print("first command");print(sum(10));print("last one")
first command
45
last one

In [20]:
[0] 0:Python*Z                                "Sudarshans-iMac.local" 20:41 21-Jun-21
```

So, I will now teach you people a couple of things before going ahead, just concentrate. Let me just zoom this just so that you are able to see the entire screen. Now, you see, I am going to use a couple of ideas and just training your mind before I use it. I want to see how long something takes. For instance, when I say for i in, I will say, let us say, s equals 0 for i in range 100 s equals s plus i and then I will print s, print I am sorry, within parenthesis, print s. It prints 4950, so which is the sum of the first 100 numbers. So, but then I would like to see how much time this takes, this particular thing.

So, what I will do is I will define a function sum of first 10 numbers, let us say. So, answer equals 0, very similar, very same thing. I am trying to write a function for it. You can write it on the interactive Python shell too. You need not necessarily write it in a file and then execute it. Answer equals 0, for i in range n, answer equals answer plus i. And then I will say return answer, boom. So, when I say print sum of 10 it will compute the sum of 10 numbers 0, 1, 2, 3 up to 9. And when I say 100, it will compute so much.

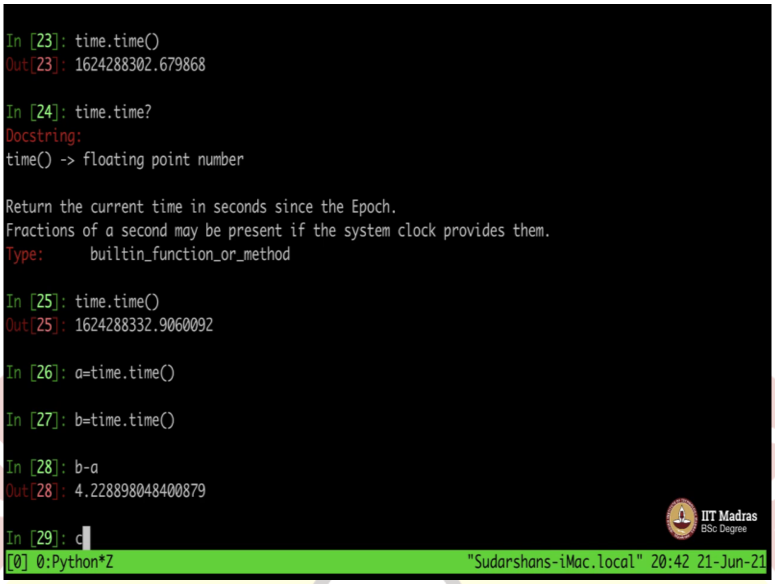
Please not in interactive Python here, I can always say print hello world and then put a semicolon here and then say, print hi, this is a second command and put another semicolon and then execute a third command. So, look at this first command and then I will print sum of 10 and then I will print last one. All these three things get executed here. This is how your IPython works. Even in Spyder, this is true, you can just check that.

(Refer Slide Time: 05:58)



```
In [21]: import time

In [22]: time.time()
Out[22]: 1624288293.9391391

In [23]: time.time()
Out[23]: 1624288302.679868

In [24]: time.time?
Docstring:
time() -> floating point number

Return the current time in seconds since the Epoch.
Fractions of a second may be present if the system clock provides them.
Type:      builtin_function_or_method

In [25]: time.time()
Out[25]: 1624288332.9060092

In [26]: 
```

```
[0] 0:Python*Z                                  "Sudarshans-iMac.local" 20:42 21-Jun-21
```

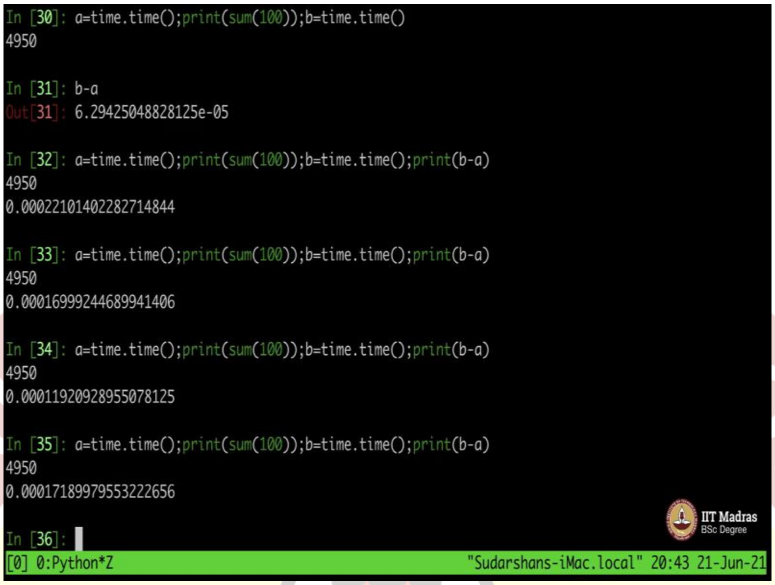


```
In [23]: time.time()
Out[23]: 1624288302.679868

In [24]: time.time?
Docstring:
time() -> floating point number

Return the current time in seconds since the Epoch.
Fractions of a second may be present if the system clock provides them.
Type:      builtin_function_or_method

In [25]: time.time()
Out[25]: 1624288332.9060092

In [26]: a=time.time()

In [27]: b=time.time()

In [28]: b-a
Out[28]: 4.228898048400879

In [29]: c
[0] 0:Python*Z                                "Sudarshans-iMac.local" 20:42 21-Jun-21
```

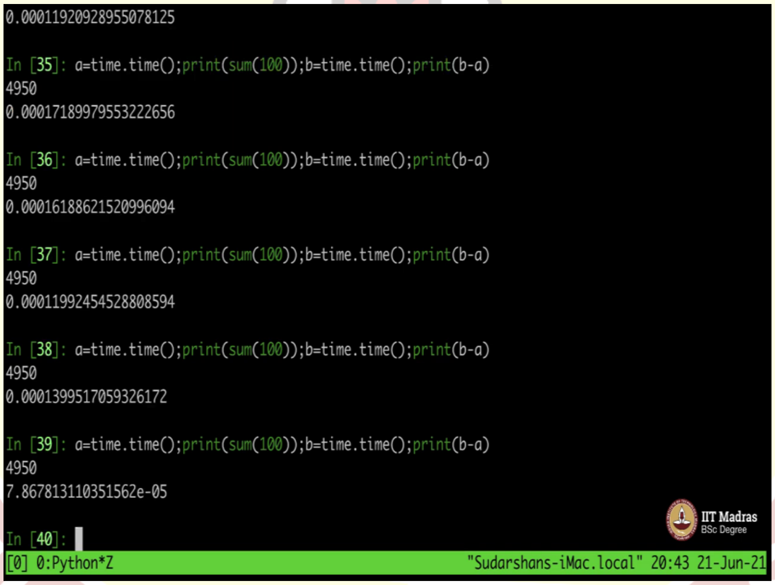
So, what I am now going to do is, let me clear the screen, I had defined the sum. So, what I will do is I will say import time. You will see in a second what I am trying to do. When you, whenever you say time, time, it will tell you that time from the past several years. It actually starts from some date. And then you see it is now a few seconds post that, so on and so forth. So, if you want to know more about it, you can always say returns the current time in seconds since Epoch. Epoch, I think is the time starting from some particular point. You can just Google for it. I do not want to get into the details. It is all there.

The idea is that this is some time that will be displayed whenever you say time, time, see now it is 30 seconds past that. The point is whenever you say a equals time, time, and give it some time and then say b equals time, time, now b minus a will actually be the difference in the execution of this and this. I hope this is clear. This is pretty straightforward. Maybe you should do the coding with me just so that you understand it.

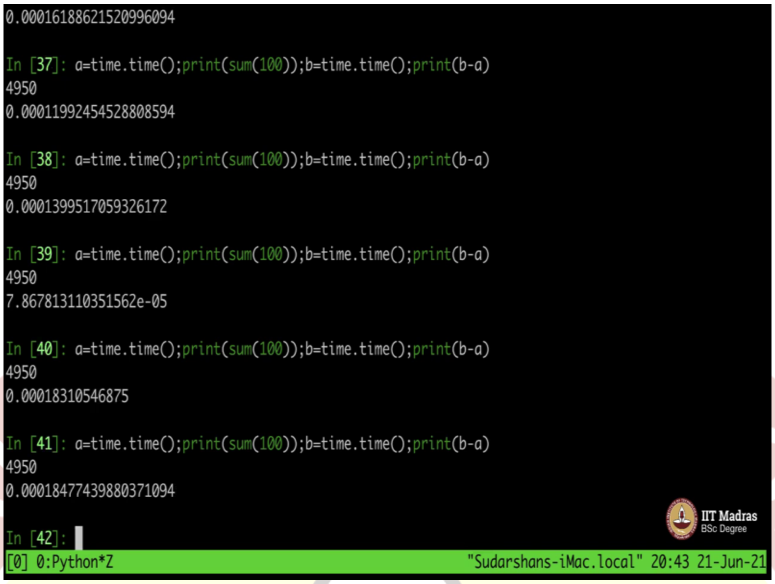
(Refer Slide Time: 07:13)



```
In [30]: a=time.time();print(sum(100));b=time.time()
4950

In [31]: b-a
Out[31]: 6.29425048828125e-05

In [32]: a=time.time();print(sum(100));b=time.time();print(b-a)
4950
0.00022101402282714844

In [33]: a=time.time();print(sum(100));b=time.time();print(b-a)
4950
0.00016999244689941406

In [34]: a=time.time();print(sum(100));b=time.time();print(b-a)
4950
0.00011920928955078125

In [35]: a=time.time();print(sum(100));b=time.time();print(b-a)
4950
0.00017189979553222656

In [36]: 
[0] 0:Python*Z                                    "Sudarshans-iMac.local" 20:43 21-Jun-21
```



```
0.00011920928955078125

In [35]: a=time.time();print(sum(100));b=time.time();print(b-a)
4950
0.00017189979553222656

In [36]: a=time.time();print(sum(100));b=time.time();print(b-a)
4950
0.00016188621520996094

In [37]: a=time.time();print(sum(100));b=time.time();print(b-a)
4950
0.00011992454528808594

In [38]: a=time.time();print(sum(100));b=time.time();print(b-a)
4950
0.0001399517059326172

In [39]: a=time.time();print(sum(100));b=time.time();print(b-a)
4950
7.867813110351562e-05

In [40]: 
[0] 0:Python*Z                                    "Sudarshans-iMac.local" 20:43 21-Jun-21
```

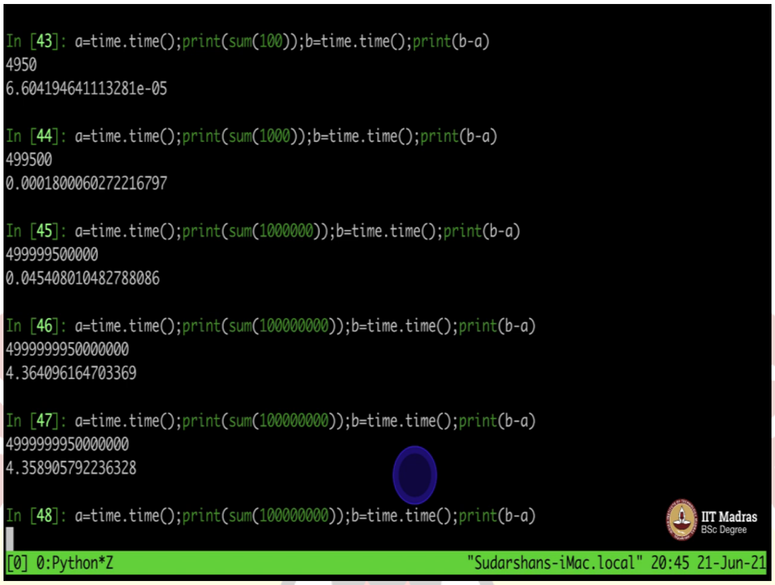


```
0.00016188621520996094

In [37]: a=time.time();print(sum(100));b=time.time();print(b-a)
4950
0.00011992454528808594

In [38]: a=time.time();print(sum(100));b=time.time();print(b-a)
4950
0.0001399517059326172

In [39]: a=time.time();print(sum(100));b=time.time();print(b-a)
4950
7.867813110351562e-05

In [40]: a=time.time();print(sum(100));b=time.time();print(b-a)
4950
0.00018310546875

In [41]: a=time.time();print(sum(100));b=time.time();print(b-a)
4950
0.00018477439880371094

In [42]: 
[0] 0:Python*Z                          "Sudarshans-iMac.local" 20:43 21-Jun-21
```

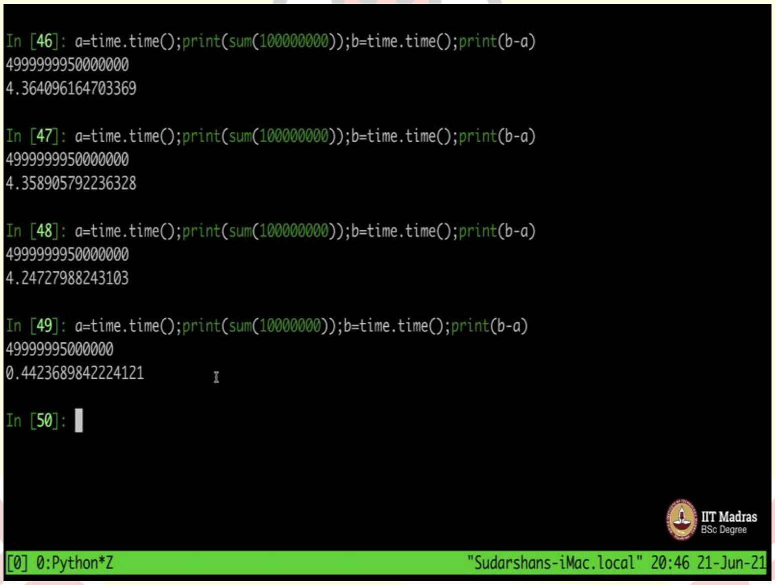
Let me clear the screen and then now say a equals time, time and then put a semicolon, not execute it and then say print sum of 100 and then say b equals time, time, now answer is given here, but a is the time that is recorded before the print statement, b is the time that is recorded after the print statement. So, essentially, b minus a should tell me the time it has taken to execute this. It is 6 to the, 6 into 10 to the power of minus 5 seconds. It is 0.00006 seconds. Roughly close to nothing.

So, but what will happen if, let us say, let me also print b minus a here just so that it prints right there itself. This is the answer and this is the time that it has taken. It is roughly the same time, no matter how many times you execute you see. I mean some minute difference is always there. But it is always in the order of this 0.000 and then something, 0.000 and then something that is what is happening here also. 10 to the power of minus 05 it is in a different format, that is all.

(Refer Slide Time: 08:30)



```
In [43]: a=time.time();print(sum(100));b=time.time();print(b-a)
4950
6.604194641113281e-05

In [44]: a=time.time();print(sum(1000));b=time.time();print(b-a)
499500
0.0001800060272216797

In [45]: a=time.time();print(sum(1000000));b=time.time();print(b-a)
499999500000
0.045408010482788086

In [46]: a=time.time();print(sum(100000000));b=time.time();print(b-a)
4999999950000000
4.364096164703369

In [47]: a=time.time();print(sum(100000000));b=time.time();print(b-a)
4999999950000000
4.358905792236328

In [48]: a=time.time();print(sum(100000000));b=time.time();print(b-a)

[0] 0:Python*Z                                        "Sudarshans-iMac.local" 20:45 21-Jun-21
```



```
In [46]: a=time.time();print(sum(100000000));b=time.time();print(b-a)
4999999950000000
4.364096164703369

In [47]: a=time.time();print(sum(100000000));b=time.time();print(b-a)
4999999950000000
4.358905792236328

In [48]: a=time.time();print(sum(100000000));b=time.time();print(b-a)
4999999950000000
4.24727988243103

In [49]: a=time.time();print(sum(10000000));b=time.time();print(b-a)
49999995000000
0.4423689842224121

In [50]:

[0] 0:Python*Z                                        "Sudarshans-iMac.local" 20:46 21-Jun-21
```

So, now will it be same if I find the sum of, let me clear the screen, sum of 100 numbers for so much, sum of 1000 numbers obviously should take more time. Surprisingly, probably there is a very small difference, because for a computer, finding the sum of 100 numbers and 1000 numbers is not so significantly different. So, let me find the sum of 1 million numbers.

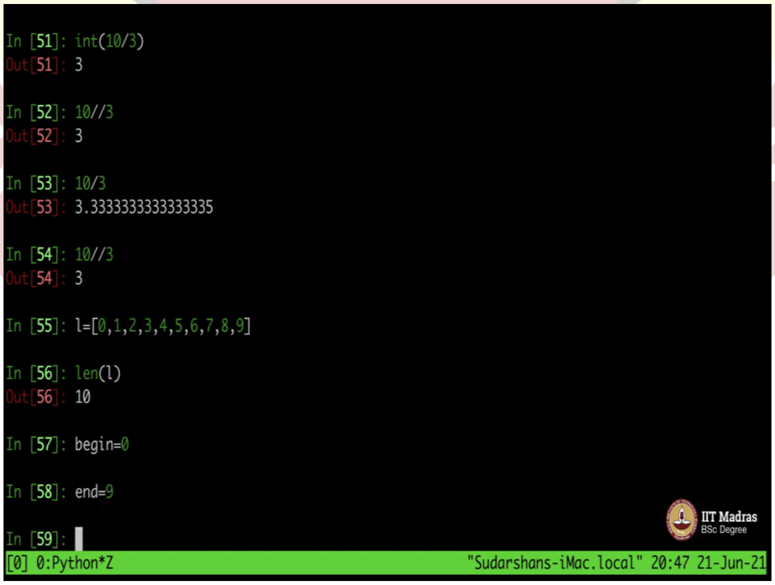
Now, let me see if it still takes something like this. So, you see this is slightly higher. Sum, finding the sum of a 1 million numbers takes close to 0.04 seconds. So, let me increase the complexity. Here again I will come here and then make this sum 10 million, 100 million, maybe this is 100 million as you can see then execute this.

Now, you see it is taking a long time. Probably that was too much. It has taken flat 4.36 seconds. Let me execute once again. It should take around that. Sometimes it is more, sometimes it is less. So, it is exactly the same, very close to the previous one. Again, execute it. If you have a lot of time, maybe you can change this to 1 billion, put one more 0, it becomes a 1 billion. So, I do not have a lot of time. So, I will reduce one 0 and make this 10 million and then execute it. This is much, much better than this, roughly 10 times less, you see.

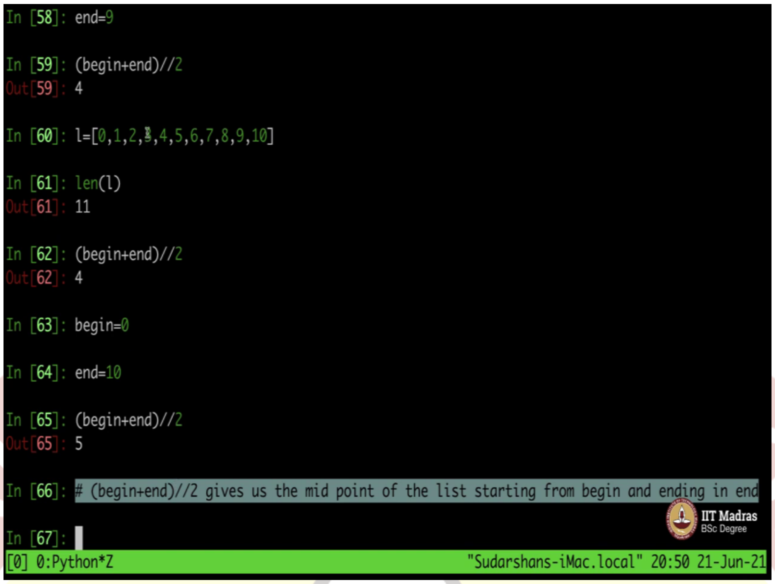
So, you now understand how to use a time function. In fact, there are many other functions in time you see, I mean, you can always take a look at it. And this is the time, time function that I have used here is the simplest of all functions according to me. There are even complicated ways of finding out how much time a particular function takes, but we are not going to get into that. It is slightly advanced.

Maybe by the end of the course, we will be able to teach you what is called profiling in the sense that you can find out how much time your individual line of your code is taking, I mean, that is simply put, it is a lot more complicated than that. Let us not get in there. But the point is you by now know how to find how much time a particular function takes. So far, so good. The next thing I want to tell you make you recollect, because these things are coming for the next upcoming discussion. Let me clear the screen.

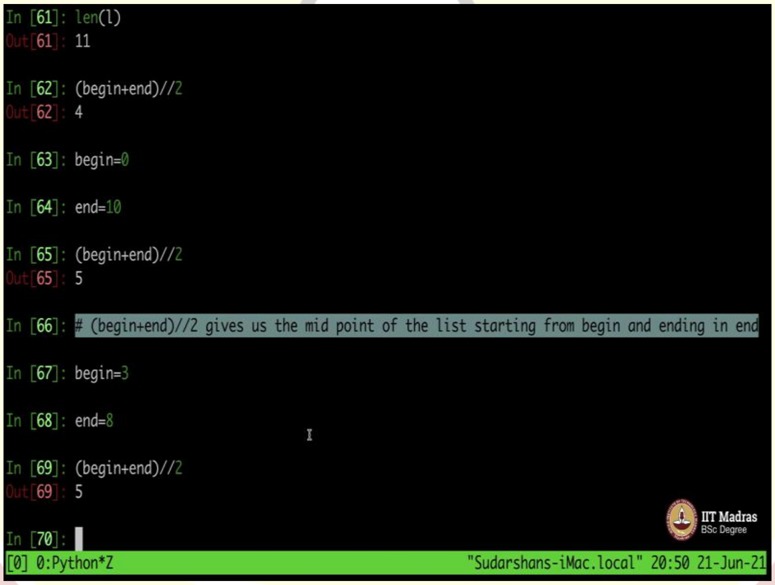
(Refer Slide Time: 11:05)



```
In [51]: int(10/3)
Out[51]: 3

In [52]: 10//3
Out[52]: 3

In [53]: 10/3
Out[53]: 3.3333333333333335

In [54]: 10//3
Out[54]: 3

In [55]: l=[0,1,2,3,4,5,6,7,8,9]

In [56]: len(l)
Out[56]: 10

In [57]: begin=0

In [58]: end=9

In [59]: 
[0] 0:Python*Z                                "Sudarshans-iMac.local" 20:47 21-Jun-21
```



```
In [58]: end=9

In [59]: (begin+end)//2
Out[59]: 4

In [60]: l=[0,1,2,3,4,5,6,7,8,9,10]

In [61]: len(l)
Out[61]: 11

In [62]: (begin+end)//2
Out[62]: 4

In [63]: begin=0

In [64]: end=10

In [65]: (begin+end)//2
Out[65]: 5

In [66]: # (begin+end)//2 gives us the mid point of the list starting from begin and ending in end

In [67]: 
[0] 0:Python*Z                                          "Sudarshans-iMac.local" 20:50 21-Jun-21
```



```
In [61]: len(l)
Out[61]: 11

In [62]: (begin+end)//2
Out[62]: 4

In [63]: begin=0

In [64]: end=10

In [65]: (begin+end)//2
Out[65]: 5

In [66]: # (begin+end)//2 gives us the mid point of the list starting from begin and ending in end

In [67]: begin=3

In [68]: end=8

In [69]: (begin+end)//2
Out[69]: 5

In [70]: 
[0] 0:Python*Z                                          "Sudarshans-iMac.local" 20:50 21-Jun-21
```

Integral part of 10 plus, let us say, 10 by 2 is what, let us say 10 by 3, 10 by 3 is 3.33. So, integral part of that is 3. You know this very well. So, there is another way of doing this. If you put two slashes, that is an easy way to compute, instead of simply saying 10 by 3, which gives you 3.33, when you put two slashes it will give you the integral part. Keep this in your mind. I am going to recollect this as I am going ahead with my program.

Now, let us go ahead. You now know what is this operator and you also know about import time. One thing that I would like to make you realize is the small exercise, let us say, L equals 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, maybe. There are 10 elements here, len of L is 10. So, what will happen if I want to find the middle most element here. So, let us say first element is 0. The last element is 9. How do you compute the middle most element?

All you need to do, you simply do begin plus end divided by 2. That is how you find the midpoint and the midpoint is 4. You might question, why is it 4? You see, there are 1, 2, 3, 4, 5 elements on this side 1, 2, 3, 4, 5 elements on this side. It can as well be 4 or 5. But then when we take begin plus end integral part divided by 2 integral part, it gives you 4. If you add 1 to this, it may give you 5.

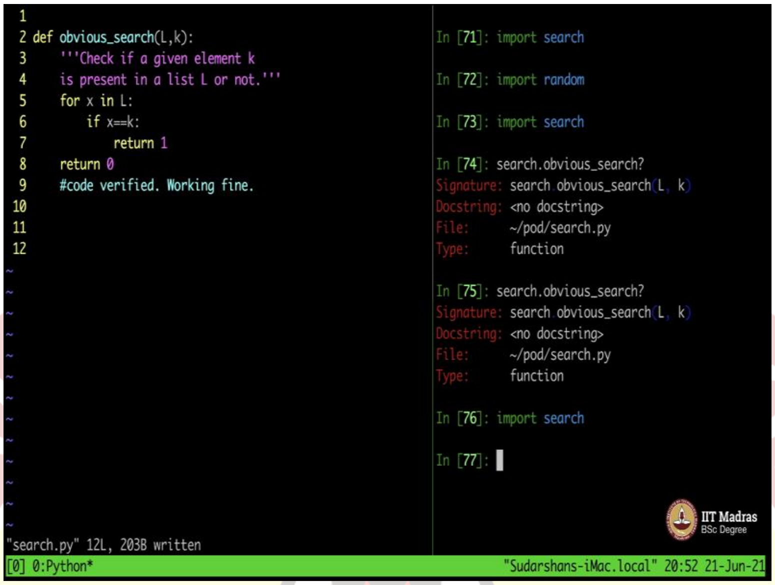
So, ideally, this is the formula to find the midpoint. But what if you had, let us say, odd number of elements. How many elements do you have here? The answer is not 10. Please note 10, 1 to 10 plus another 0, which makes it 11. So, length of L is 11. So, what if we execute the same code begin plus end by 2, it will actually give you the center point 5.

Just try doing that you are getting center point 5 that is because your begin and end continues to be 0 and 9 here, begin is 0, end is 10. So, you see this has nothing to do with the list. I am just trying to tell you how to find the center point of any list. Begin is 0, end is 10. So, begin plus end by 2 is 5. That is the center point of a list that starts with 0 and ends with 10, essentially has 11 elements in it.

Whenever a list has 11 elements in it, begin is, there is 0th element goes up to the 10th element. There are 11 elements you figure out by adding begin plus end by 2. So, what have we inferred? We infer that begin plus end by 2 gives us is the midpoint of the list, which has, of the list starting from begin and ending in end. That is pretty straightforward. You see, you may be wondering, sir what are you trying to say? I am a little confused. All I am trying to say is, what is the midpoint of let us say this? How do you find that?

The midpoint of simply this will be then you should choose begin to be 3, end to be, how much, what was that, what did I select, 3, 4, 5, 6, 7, 8, end is 8, begin is 3, end is 8, the mid should be 5 here. End is 8, so I do this, mid is 5. So, whenever you give begin and end, it will give you the midpoint. Why are we even talking this, these things will come in between when we are writing the code for something that I will tell you right now, I am just getting you people warmed up. Getting you people mentally prepared for what is coming next. So, let us jump ahead and try to see how we can write a piece of code that does better on searching.

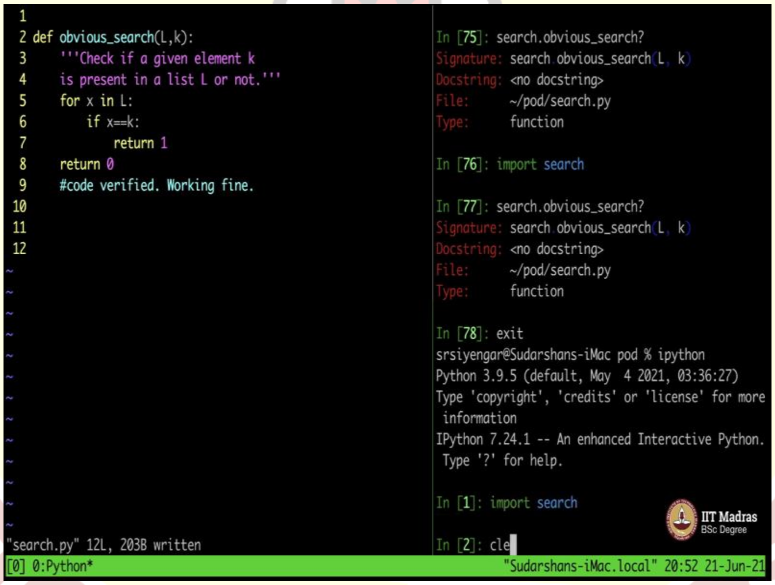
(Refer Slide Time: 15:36)



```

def obvious_search(L,k):
    '''Check if a given element k
    is present in a list L or not.'''
    for x in L:
        if x==k:
            return 1
    return 0
    #code verified. Working fine.



~
~
~
~
~
~
~
~
~
~
~
~
~
"search.py" 12L, 203B written
```

```
In [71]: import search

In [72]: import random

In [73]: import search

In [74]: search.obvious_search?
Signature: search.obvious_search(L, k)
Docstring: <no docstring>
File:      ~/pod/search.py
Type:      function

In [75]: search.obvious_search?
Signature: search.obvious_search(L, k)
Docstring: <no docstring>
File:      ~/pod/search.py
Type:      function

In [76]: import search

In [77]:
```

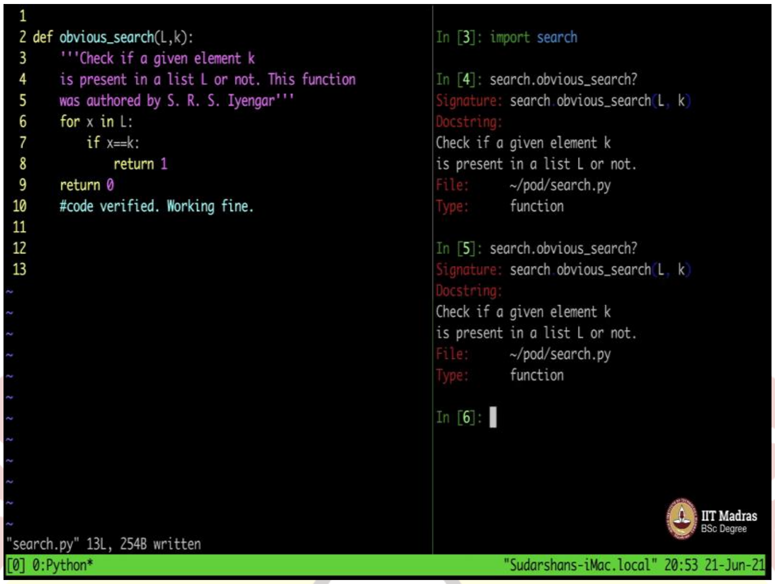
```
[0] 0:Python*                         "Sudarshans-iMac.local" 20:52 21-Jun-21
```



```

def obvious_search(L,k):
    '''Check if a given element k
    is present in a list L or not.'''
    for x in L:
        if x==k:
            return 1
    return 0
    #code verified. Working fine.



~
~
~
~
~
~
~
~
~
~
~
~
~
"search.py" 12L, 203B written
```

```
In [75]: search.obvious_search?
Signature: search.obvious_search(L, k)
Docstring: <no docstring>
File:      ~/pod/search.py
Type:      function

In [76]: import search

In [77]: search.obvious_search?
Signature: search.obvious_search(L, k)
Docstring: <no docstring>
File:      ~/pod/search.py
Type:      function

In [78]: exit
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 search

In [2]: cle
```

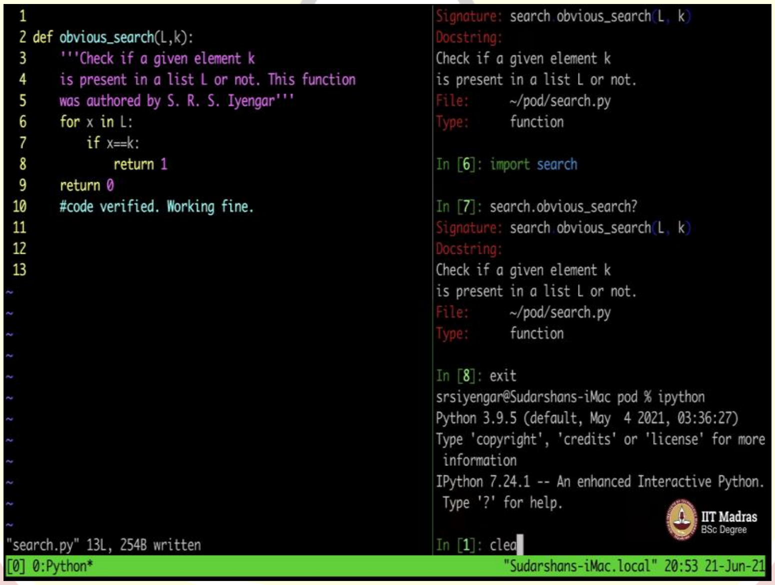
```
[0] 0:Python*                         "Sudarshans-iMac.local" 20:52 21-Jun-21
```



```
def obvious_search(L,k):
    '''Check if a given element k
    is present in a list L or not. This function
    was authored by S. R. S. Iyengar'''
    for x in L:
        if x==k:
            return 1
    return 0
    #code verified. Working fine.
```

```
"search.py" 13L, 254B written
[0] 0:Python*                          "Sudarshans-iMac.local" 20:53 21-Jun-21
```

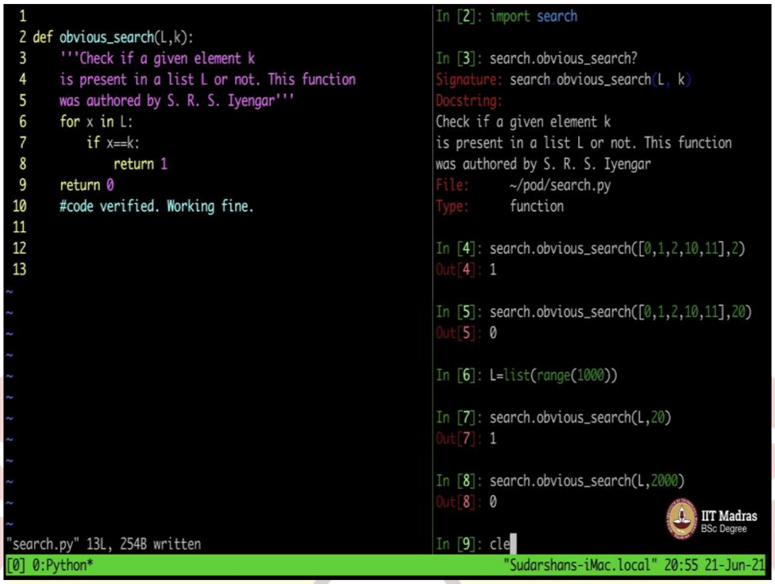
```
In [3]: import search

In [4]: search.obvious_search?
Signature: search.obvious_search(L, k)
Docstring:
Check if a given element k
is present in a list L or not.
File:      ~/pod/search.py
Type:      function

In [5]: search.obvious_search?
Signature: search.obvious_search(L, k)
Docstring:
Check if a given element k
is present in a list L or not.
File:      ~/pod/search.py
Type:      function

In [6]:
```



```
def obvious_search(L,k):
    '''Check if a given element k
    is present in a list L or not. This function
    was authored by S. R. S. Iyengar'''
    for x in L:
        if x==k:
            return 1
    return 0
    #code verified. Working fine.
```

```
"search.py" 13L, 254B written
[0] 0:Python*                          "Sudarshans-iMac.local" 20:53 21-Jun-21
```

```
Signature: search.obvious_search(L, k)
Docstring:
Check if a given element k
is present in a list L or not.
File:      ~/pod/search.py
Type:      function

In [6]: import search

In [7]: search.obvious_search?
Signature: search.obvious_search(L, k)
Docstring:
Check if a given element k
is present in a list L or not.
File:      ~/pod/search.py
Type:      function

In [8]: exit
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]: clea
```



```

def obvious_search(L,k):
    '''Check if a given element k
    is present in a list L or not. This function
    was authored by S. R. S. Iyengar'''
    for x in L:
        if x==k:
            return 1
    return 0
    #code verified. Working fine.



~
~
~
~
~
~
~
~
~
~
~
~
"search.py" 13L, 254B written
```

```
In [2]: import search

In [3]: search.obvious_search?
Signature: search.obvious_search(L, k)
Docstring:
Check if a given element k
is present in a list L or not. This function
was authored by S. R. S. Iyengar
File:      ~/pod/search.py
Type:      function

In [4]: search.obvious_search([0,1,2,10,11],2)
Out[4]: 1

In [5]: search.obvious_search([0,1,2,10,11],20)
Out[5]: 0

In [6]: L=list(range(1000))

In [7]: search.obvious_search(L,20)
Out[7]: 1

In [8]: search.obvious_search(L,2000)
Out[8]: 0

In [9]: cle
```

```
[0] 0:Python*                                  "Sudarshans-iMac.local" 20:55 21-Jun-21
```

What do you mean by better? Let me illustrate that in a minute. Let us recollect this code here, which we wrote for, what, the obvious search. Let me clear the screen and then let me, how I can run this particular code? There is a way to run this particular code on the shell. What you do is you say import and then the file name. What is the file name? This file name actually was search.py. You should be in the same directory and then import you say search. This is like an inbuilt function you see.

So, just how do you say import random, import search also works. Why, because search is a function that you have just now written. I am sorry, you have just now written. So, when you say search dot and then tab, it will show you all the existing functions here. This is the file name itself. And obvious search, is not that awesome! You can actually write your own function. And if you put a question mark here, you should get help. But it does not show anything, because it is the function that you have written.

But if you want some help also displayed here, you can always write a comment here saying this, well, check if, so this very same thing, let us paste it here given element k is, you will understand in a minute what I am trying to do. Whenever, I am sorry came out. Whenever you write a comment in the first line that very thing will be displayed here when you put a question mark, not displaying because I did not import. I imported search once again and then I do this, maybe I should reset it. Just give me a second. I go out and then come back.

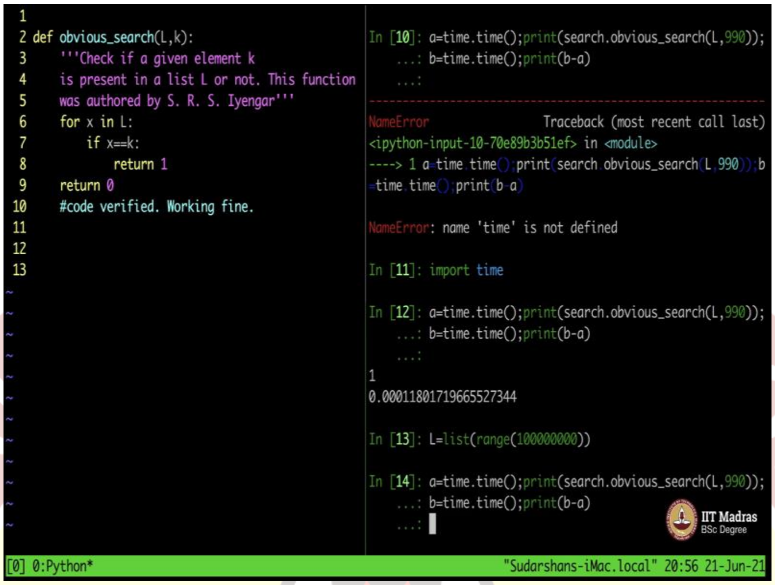
And then again say import search, clear screen, import search and then I will say search, obvious search question mark, it shows me check if an element, if a given element k present in a list L or not. You can you can write whatever you want here. This function was authored by SRS Iyengar, proudly presents one of the simplest Python code ever only four lines. So, when you say this, it will show, you should again import and then it will show, again I said, we should exit, come back, there is also a way in which you can reload it. I do not want to complicate it for you people.

So, once written function will always show you whatever comment that you type in the first line. It does not show it any other comment later on. It only shows the comment in the first line. So, what should I do? I can say search, obvious search from the list 0, 1, 2, 10, 11. The element 2, it is found. If you want to find the element 20 here, no, not found.

As you see, this is a very comfortable technique, the facility where you can write your own function. The first one, as you see here, the first one will be the file name and the second one will be the function name. You might take some time to get a hang of it. But then the bigger idea is that you can sort of play around like this, write your code here and then simply import the file name and then call that particular function with the right parameters. Is not this awesome? You should probably try doing this again.

So, now, what I will do is I will go a step ahead and I will try to create a list of range of let us say 1,000 elements and then I will try to search for the element 20 in this list. Let me put L here so that I can, 20 is obviously present, but you see 2,000 is not present. So far, so good. Let me clear the screen.

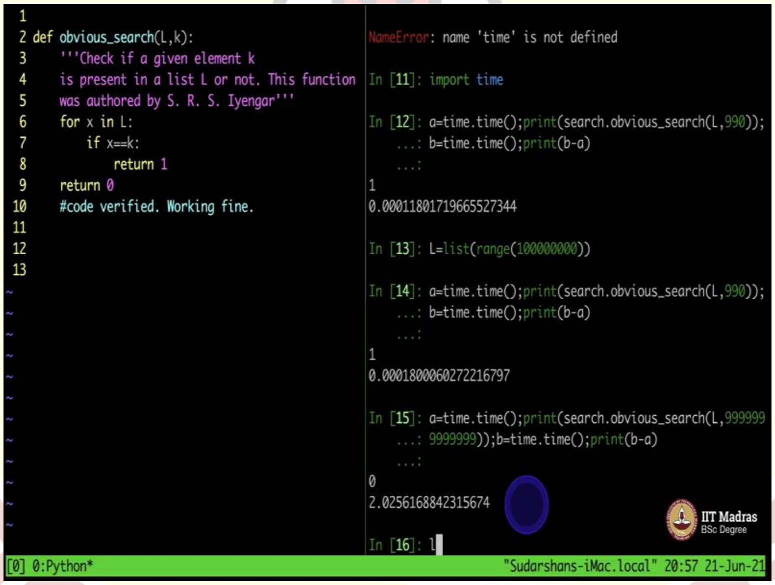
(Refer Slide Time: 19:45)



```
def obvious_search(L,k):
    '''Check if a given element k
    is present in a list L or not. This function
    was authored by S. R. S. Iyengar'''
    for x in L:
        if x==k:
            return 1
    return 0
    #code verified. Working fine.
```

```
In [10]: a=time.time();print(search.obvious_search(L,990));
    ...: b=time.time();print(b-a)
    ...:
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-10-70e89b3b51ef> in <module>
----> 1 a=time.time();print(search.obvious_search(L,990));b
=time.time();print(b-a)

NameError: name 'time' is not defined

In [11]: import time

In [12]: a=time.time();print(search.obvious_search(L,990));
    ...: b=time.time();print(b-a)
    ...:
1
0.00011801719665527344

In [13]: L=list(range(100000000))

In [14]: a=time.time();print(search.obvious_search(L,990));
    ...: b=time.time();print(b-a)
    ...: 
```

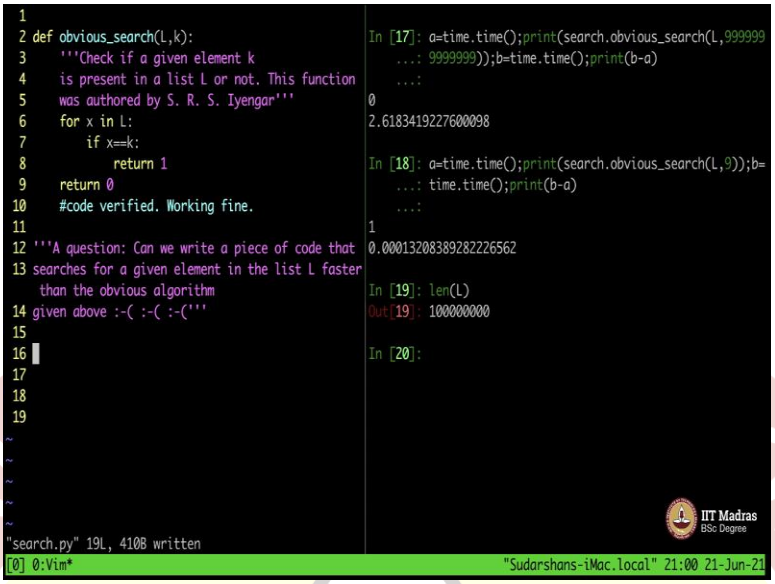
```
[0] 0:Python*                      "Sudarshans-iMac.local" 20:56 21-Jun-21
```



```
def obvious_search(L,k):
    '''Check if a given element k
    is present in a list L or not. This function
    was authored by S. R. S. Iyengar'''
    for x in L:
        if x==k:
            return 1
    return 0
    #code verified. Working fine.
```

```
NameError: name 'time' is not defined

In [11]: import time

In [12]: a=time.time();print(search.obvious_search(L,990));
    ...: b=time.time();print(b-a)
    ...:
1
0.00011801719665527344

In [13]: L=list(range(100000000))

In [14]: a=time.time();print(search.obvious_search(L,990));
    ...: b=time.time();print(b-a)
    ...:
1
0.0001800060272216797

In [15]: a=time.time();print(search.obvious_search(L,999999
    ...: 9999999));b=time.time();print(b-a)
    ...:
0
2.0256168842315674

In [16]: l
```

```
[0] 0:Python*                      "Sudarshans-iMac.local" 20:57 21-Jun-21
```



```

def obvious_search(L,k):
    '''Check if a given element k
    is present in a list L or not. This function
    was authored by S. R. S. Iyengar'''
    for x in L:
        if x==k:
            return 1
    return 0
    #code verified. Working fine.

'''A question: Can we write a piece of code that
searches for a given element in the list L faster
 than the obvious algorithm
given above :-( :-( :-('''




~
~
~
~
~
"search.py" 19L, 410B written
[0] 0:Vim*                                    "Sudarshans-iMac.local" 21:00 21-Jun-21
```

```
In [17]: a=time.time();print(search.obvious_search(L,999999
    ...: 9999999));b=time.time();print(b-a)
    ...:
0
2.6183419227600098

In [18]: a=time.time();print(search.obvious_search(L,9));b=
    ...: time.time();print(b-a)
    ...:
1
0.00013208389282226562

In [19]: len(L)
Out[19]: 100000000

In [20]:
```

I would now like to say a equals, you got it right, time, time, and then I will say print obvious, print search dot obvious search, the same list L and the element, let us say, 990. Is it there or not, let us see. B equals time dot time, it just came to the next line, that is all, and then I will say print b minus a.

Small mistake here, print search dot obvious search and then I should close the bracket here and then the rest is fine. Time is not defined. I must import time here. And then I will redo this. It will show me the time that it took to give me this answer. It just executes in a fraction of a second. You do not even know what just happened.

So, you record the initial time, you record the end time, in between you execute some function and finally, print b minus b, it will tell you the time that this function has taken. So, as and always, let me try increasing the length of the list to 1 million, 10 million, let us say, 100 million. Take some time to even know create this list.

And then I will try to find the element 990 here, boom. That was very fast as though there is no difference between a bigger list and a smaller list. But look at this. If I were to find not the element 990, but an element close to this, length of list, which is 99999999, let us say or let say something that is outside the list. I will make it a big number. Now, look at this. It took more time. It took two seconds.

Why that is, because please note, why, because it went through, let us spend a minute on it. I think let me clear the screen and then redo this so that it is easy on your minds. Waiting for a very big number that is not part of the list. It is not part of the list it says. And then it also says, it is not part of the list it says. It also takes close to 2.6 seconds. That is a lot. Why did it take such a long time, because it went through the entire list that big list that I created, 100 million list, a list of size 100 million. So, it went through the entire thing and said, no, I did not find this number.

As opposed to, let us say, how much time did it take, it has taken 2.6 seconds, if you were to find a smaller number, let us say if you were to find not this big number, which is not in the list, assume you only want to find the element 9 here. 9 is there in the first part of the list, you see. No time it will execute. Why is this? This is because in your very list, the element 9 is 0, 1, 2 up to 9 is here and your 10 million long list it does not go till here. It returns, you see. It returns one the moment it sees it in the first part itself.

But when you include something that is not part, I mean, that is not part of this list if you input a big number here or if you even into anything that is in the end of the list, the end of the list if you include, it ends up taking a long time. That makes sense, does not it?

So, let us get back. So, what did we learn so far, we wrote a simple code for obvious search, we saw a few things, which I said, please, my friends wait, these things will appear very soon in the lecture and we went ahead, and then we showed how we could use this time, library function and this particular obvious such thing is taking a long, long time for a list as big as, how big was the list, I forgot. The list was, how much, 100 million, this is 100 million, am I right? This is 1 million and this 100 million. So, 100 million long list took a long time.

Is there any way we can write a function? Now, here is the question. A question, can we write a piece of code that searches for a given element in the list L faster than the obvious algorithm given above. Does not look like there is an algorithm. Let me take a break now think if there is one such algorithm and then get back. By that I mean you meanwhile digest whatever we have covered so far.

We will come back to this very same code in our next video and try to write a very ingenious algorithm which uses the idea of the example I gave you the Hema Malini actress example, the finding of a word in the dictionary example, the halving of a big number which quickly makes it go close to 0 or a single digit number example, I am going to use that example in writing a very ingenious code that will like work in a matter of few fraction of seconds. I mean, in fraction of a second it will work. We cannot compare that with the obvious search. This is actually a very, very expensive algorithm. It takes a long time. Let us see in our next video.