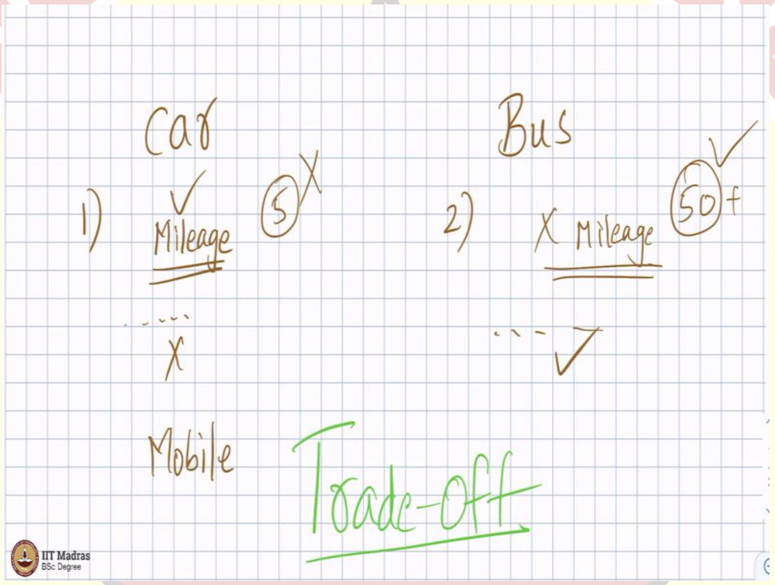
IIT Madras

ONLINE DEGREE

Programming in Python
Professor. Sudarshan Iyengar
Department of Computer Science & Engineering
Indian Institute of Technology, Ropar
Lists and Sets

Welcome to the week 6 of Python Programming course. We will start with a small example which has nothing to do with programming to begin with, but understanding this idea helps us explain a concept a bit.

(Refer Slide Time: 00:15)

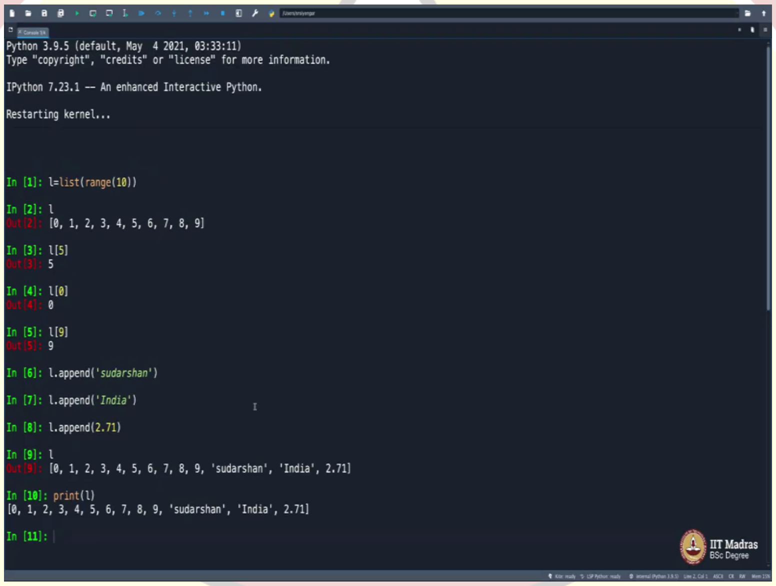


So, assume you have a car and a bus, you see, what is it that you have in a car that you do not have in a bus? What is it that you have in a bus that you do not have in a car? If you think for a minute, you will realize that a car gives good mileage, while a bus does not give very good mileage. It is only common sensical, simply because car is a typical five seater, a bus is a 50 seater plus, probably even more. But then this is actually an advantage. This is certainly an advantage. This is a plus here, while this becomes a minus here.

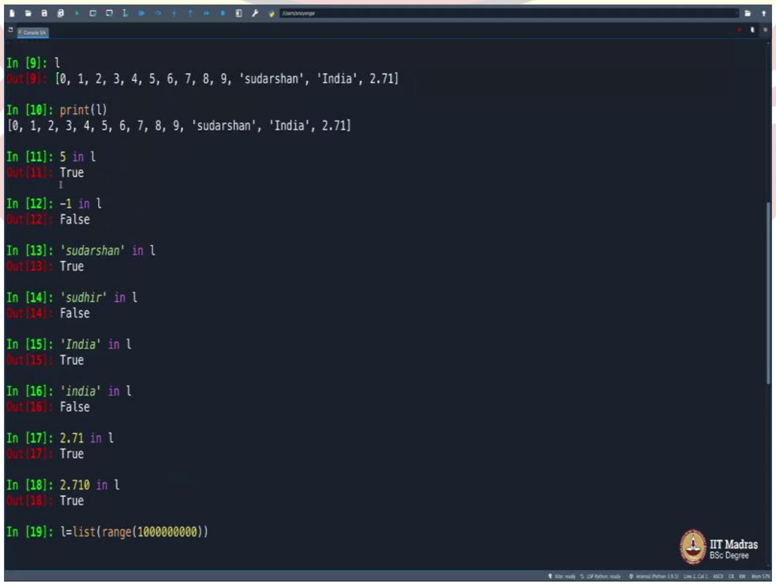
But then there is something nice about the car. It gives you mileage, while bus does not give you good mileage. There are more benefits of a car. You see, it is very sort of, what do I say, I can call it mobile. You can take it to a narrow road and still cruise through the road and then come back safely. But then you take a bus, you may get stuck in the inner roads. If the road is very narrow, you probably will not be able to turn it properly. So, what is it that you observe here? We observe what is called a trade off.

So, if you want to gain something in life, you may have to give up on some other things in life. We all know this well known idiom. So, as you can observe here, car and bus has its own pros and cons. Similarly, in Python, we will observe some aspects of Python where there will be something positive and there are some other aspects where there are some negatives of it. So, we cannot have everything in our facility. I may sound very Greek right now, but it will be clear to you in the next video.

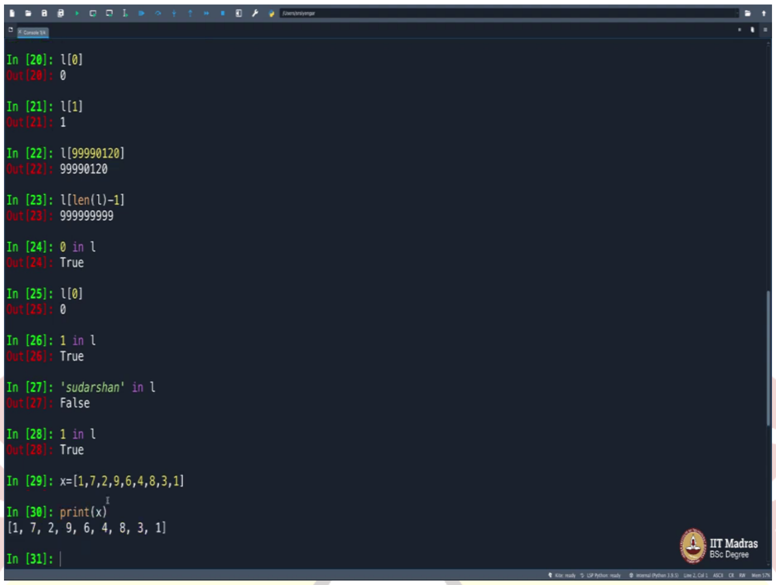
(Refer Slide Time: 2:53)



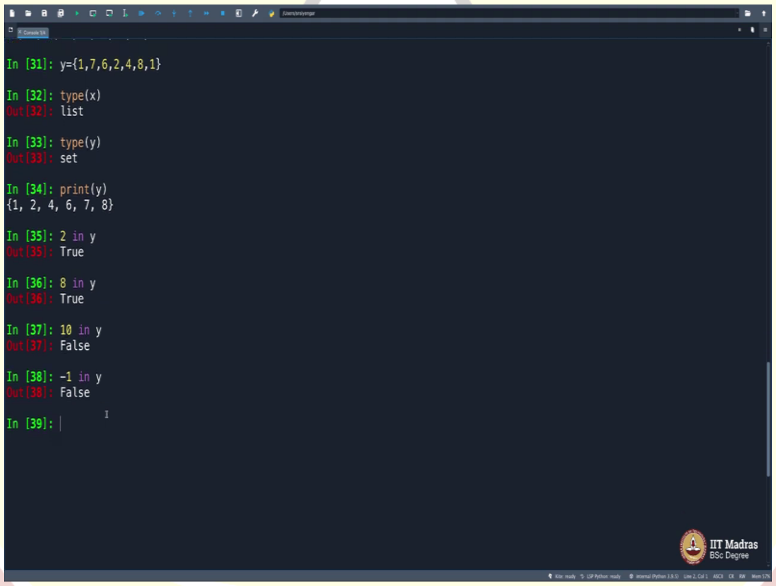
```
Python 3.9.5 (default, May  4 2021, 03:33:11)
Type "copyright", "credits" or "license" for more information.

IPython 7.23.1 -- An enhanced Interactive Python.

Restarting kernel...



In [1]: l=list(range(10))

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

In [3]: l[5]
Out[3]: 5

In [4]: l[0]
Out[4]: 0

In [5]: l[9]
Out[5]: 9

In [6]: l.append('sudarshan')

In [7]: l.append('India')

In [8]: l.append(2.71)

In [9]: l
Out[9]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 'sudarshan', 'India', 2.71]

In [10]: print(l)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 'sudarshan', 'India', 2.71]

In [11]:
```



```
In [9]: l
Out[9]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 'sudarshan', 'India', 2.71]

In [10]: print(l)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 'sudarshan', 'India', 2.71]

In [11]: 5 in l
Out[11]: True

In [12]: -1 in l
Out[12]: False

In [13]: 'sudarshan' in l
Out[13]: True

In [14]: 'sudhir' in l
Out[14]: False

In [15]: 'India' in l
Out[15]: True

In [16]: 'india' in l
Out[16]: False

In [17]: 2.71 in l
Out[17]: True

In [18]: 2.710 in l
Out[18]: True

In [19]: l=list(range(1000000000))
```



```
In [20]: l[0]
Out[20]: 0

In [21]: l[1]
Out[21]: 1

In [22]: l[99990120]
Out[22]: 99990120

In [23]: l[len(l)-1]
Out[23]: 999999999

In [24]: 0 in l
Out[24]: True

In [25]: l[0]
Out[25]: 0

In [26]: 1 in l
Out[26]: True

In [27]: 'sudarshan' in l
Out[27]: False

In [28]: 1 in l
Out[28]: True

In [29]: x=[1,7,2,9,6,4,8,3,1]

In [30]: print(x)
[1, 7, 2, 9, 6, 4, 8, 3, 1]

In [31]:
```



```
In [31]: y={1,7,6,2,4,8,1}

In [32]: type(x)
Out[32]: list

In [33]: type(y)
Out[33]: set

In [34]: print(y)
{1, 2, 4, 6, 7, 8}

In [35]: 2 in y
Out[35]: True

In [36]: 8 in y
Out[36]: True

In [37]: 10 in y
Out[37]: False

In [38]: -1 in y
Out[38]: False

In [39]:
```

So, let me consider a list l, which is, let us say, a list of range of 1 million or let us say only 10 to begin with. So, l is so much. And I know I can access a particular element in l, l of 5 is 5, l of 0 is the first element and l of 9 is the last element. And in fact, you can append anything to this list. I can append my name here. I can append whatever I want here. I can even append a real number, something like this. So, if I say print l, you will observe that, let me scroll up, you will observe that it shows me the whole of the list l.

So, let me increase the length of the list l, before which let me introduce a small command to you people. When I say 5 in l, it says true because you see 5 is in l. So you now know the command in. 5 in l turns out to be true if 5 is in l. If it is not in l, it turns out to be false. For instance, per se, minus 1 in l, it spits out an error, saying it is false. Or let us say I say Sudarshan in l, true, Sudhir in l, false, India in l, true. Is it case sensitive? It indeed is. As you can see, it is false. India is false. 2.71 in l, true, 2.710 in l, what is your guess? Will it say true or false? It says true, because this is a real number. 2.71 is same as 2.70. Is not that interesting?

So, the more important thing here is that you can use the in command. So, now just observe something really fascinating happens right now. When I say, instead of l, let us say l is a list of range of really big a number, let us say 1 billion. This is 1 billion, 1 followed by nine zeros. So, it takes some time. It roughly takes some, I believe, some 5 to 10 seconds to create this list, because it is a huge number. It must create from l of 0 to l of 1 billion and that is a very huge number. It will obviously take some time.

And as you can observe, this l will be overwritten because you have used the list here. You have defined the list l here already to be so much with so many appends, which means l is so much. Again, when you say l equals something, it will rewrite it. So, our list is ready. Let me see what is an l of 0? Obviously, l of 0 is 0, l of 1 is 1. That is how I created the list. You can create whatever you want, but it chose to create it like this. So, what is l of 9990120, whatever, whatever, it will give the value.

What is the last element in the list? You know how to do it? What will happen if I close this? It will throw an error, because this will be out of range. I should say minus 1. So, this is 1 less than a billion. Now, let me check if 0 is in l. 0 is in l. Why, because l of 0 is indeed 0. So, 0 is present in l. 1 in l, obviously. Do you think Sudarshan is in l? No, not anymore, because I changed it. Now, here is a problem. When I say Sudarshan in l, it will search for the whole of l.

Remember, our names search program that we wrote, that took a long time, if the list was very big, in previous to previous week. So, this is precisely a very similar algorithm that Python is using to search for Sudarshan in l. It is still searching. Why, because l is a very huge list of, it is done now, of 1 billion entries. It goes through everything and says it is false.

But then why did, it turn out to be true so quickly. It said true in no time when I said 1 in l, that is because it quickly found 1 it was in the beginning. It goes through the entire list. In case you punch in something that is not in the list, it may want to go through the entire list and say, no, it is not there at the end.

So, because it should see through every single entry in l, here is a moment where we should pause and then realize that a computer is fast. But this fast is not infinitely fast. If you torture it with a big number like this, the computer definitely will cry. If you have a super duper computer, then what I will do is I will increase the number of zeros here. And there is a stage where it will start crying. By crying, I mean, it will take a long time.

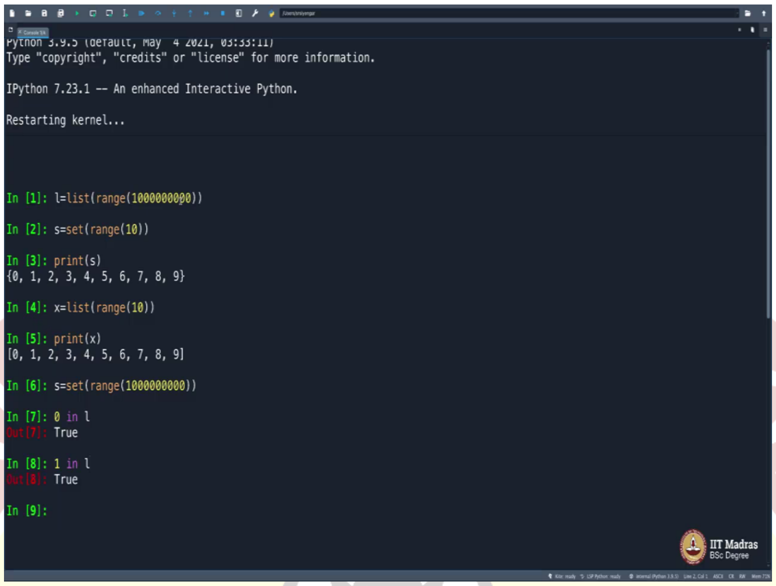
Let us go ahead and let us try to understand if there is a way in which we can search through the list quickly. Now, whenever you said, let us say x is equal to a list 1, 7, 2, 9, 6, 4, 8, 3, and let us say I will put another 1 here, this becomes a list. You print x and you get a list. But whenever you create a list and instead of these square brackets, these are called the square brackets, these are called a small bracket. This is a small bracket. This is a square bracket. And this is a flower bracket. So, I believe you are taught all these things in your school days.

So, if I say y is equal to a flower bracket instead and I say 1, 7, 6, 2, 4, 8, then close it. Maybe I will put another 1 here and then close it. It accepts. It does not really require a bracket. But then this does not become a list. So, what else does it become? We know how to check the data type. You see, if I say type x, it says x is of type list. When I say type of y, what has happened so far, let us be clear, I created a list like thing, but I did not use the big bracket. I used a flower bracket. In Python that stands for a set.

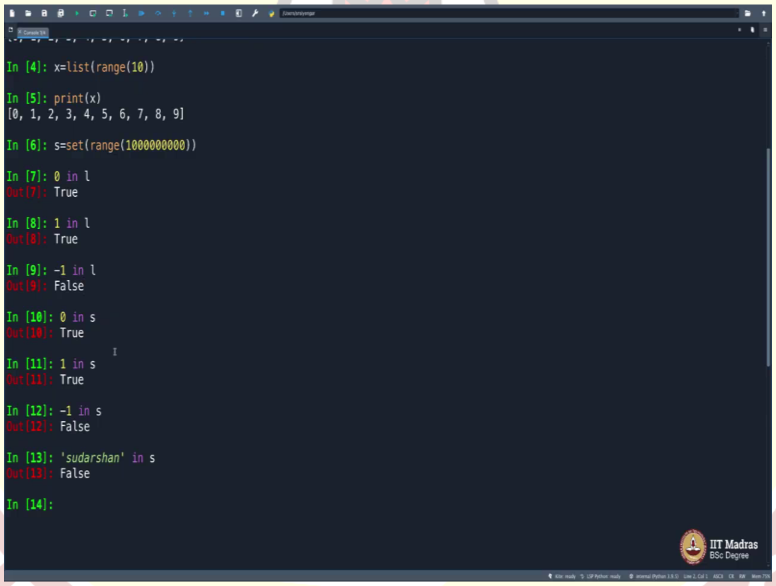
What is a set? Let me print y and show you what a set is. You see, set was, you wrote y as 1, 7, 6, 2, 4, 8, 1, but you remember the definition of the set. Set does not have repeated elements. So, it removed the repeated 1 here and did not display it here, whereas in x 1 was repeated and it did display 1 repeated here. But when you declare it as flower brackets, it calls itself a set. Flower bracket means that entity becomes a set and in a set reputations are not allowed.

So, when I say repetitions are not allowed, it is a man made convention that in a set it is a bunch of elements which are not repeated. And on this set also you can check 2 in y, true, 8 in y, true, 10 in y, no, false, minus 1 in y, no, false. So, the in operator works on y also. Now, we are slowly getting to the example of car and bus, the analogy of car and bus. We cannot have everything in life, you see. Is not that right? So, what I will now do is, very patiently, I will create, let me clear this entire shell for a second, I am clearing the shell.

(Refer Slide Time: 11:29)



```
Python 3.9.5 (default, May  4 2021, 03:33:11)
Type "copyright", "credits" or "license" for more information.

IPython 7.23.1 -- An enhanced Interactive Python.

Restarting kernel...



In [1]: l=list(range(1000000000))

In [2]: s=set(range(10))

In [3]: print(s)
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

In [4]: x=list(range(10))

In [5]: print(x)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

In [6]: s=set(range(1000000000))

In [7]: 0 in l
Out[7]: True

In [8]: 1 in l
Out[8]: True

In [9]:
```



```
In [4]: x=list(range(10))

In [5]: print(x)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

In [6]: s=set(range(1000000000))

In [7]: 0 in l
Out[7]: True

In [8]: 1 in l
Out[8]: True

In [9]: -1 in l
Out[9]: False

In [10]: 0 in s
Out[10]: True

In [11]: 1 in s
Out[11]: True

In [12]: -1 in s
Out[12]: False

In [13]: 'sudarshan' in s
Out[13]: False

In [14]:
```

And let me create once again, a list as big as, let us say, 1 billion entries. How do I do that, l equals list of range of 1, 2, 3, 4, 5, 6, 7, 8, 9 perfect and close the bracket. And I get a list. It will take some time to create it. Let us spend the time. And I will meanwhile tell you what I am going to do next. I am going to create a set of the same size, of size 1 million, which means it will simply be what s will be, all the elements 0, 1, 2, 3 up to 999999999, which is 1 less than 1 billion. So, we are through.

If I say s equals set of range of I will say 10 to begin with, just so that you understand what I am doing and I will print s. It will have these things. If I said, x equals list of range of 10, it will this how as you would have expected. It has a list 0 to 9. Now, this has the element 0 to 9. Of course, this has no repetitions. I did not include any repetitions here. When I said range of 10, it was from 0 to 9. And I converted it. Range is another data type. In fact, I converted that to a set data type.

What is the big deal about it? Now observe this. Instead of 10 here, I will copy paste this very same code, but I will put 1, 2, 3, 1, 2, 3 and 1, 2, 3 and then hit enter and this will take actually more time than l. So, on your computer when you try this, it may take even more time than what it is taking on my computer or maybe slightly less time. But one thing is for sure that this will take more time than this when you create an l.

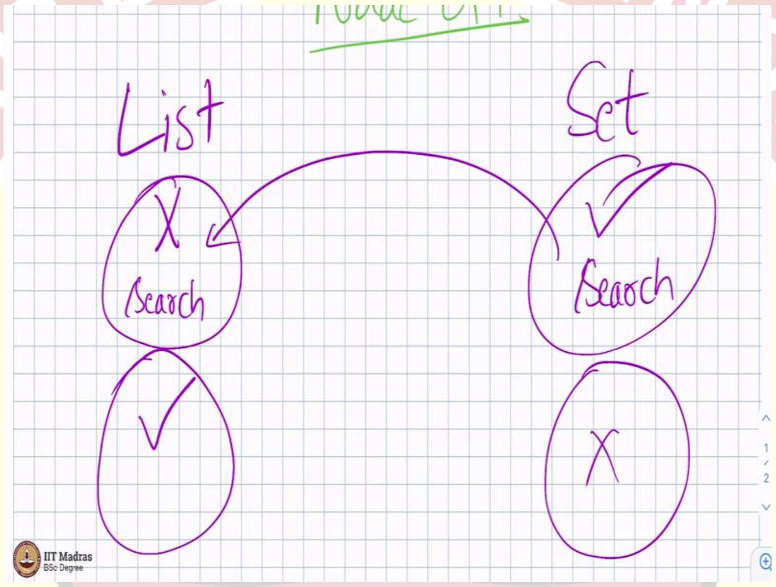
So, let us wait. Well, finally, the program is done. S is now a set containing 1 billion entries. In fact, I had to pause the video and then resume it because it took a whopping lot of time. It took close to a couple of minutes. It may take a lot of time on your computer. So, try to see, put as many zeros so that creating a list takes a few seconds, then creating a set will take even more than the list. You may wonder why, now let us see the, what I call, I think this is magic for you all. So, just look at this.

When I say 0 in l, it is true, because 0 is in l, as you can see. L contains the first 1 billion entries from 0 to 1 billion minus 1. So, when I say 1 in l, it is true, as we discussed earlier. When I say minus 1 in l, it will take a long, long time. Why is that? That is because it has to go through all of this 1 billion entries and check if minus 1 is present anywhere or not. Of course, it is not present because it is from 0 to 999999 that we know, but the computer does not know. It will only see it as a list and will search through it completely and gets tired and finally says false minus 1 is not l. So, let us wait for it to say that.

Again, it took me close to a minute for this to return false, which means the computer did work hard. You see, at least, it took a lot of time to tell me it is not there, but then look at the magic here. Now, when I say 0 in s, which is true, 1 in s, it is true, and minus 1 in s, what is your guess? Again, s is what? S is not just as big as 10. I have redone that s is in the range of 1 billion as big as the list, except that it is of the data type set. This should also take a long time is my guess. But when I press enter, you see, boom, it quickly says false.

Sudarshan in s, it is false. So, what exactly is happening here that the data structure s, I mean, the data type s is so fast, but l is so slow. When it comes to the idea of belongingness, when I say, is minus 1 in s, it quickly says false. But the same question in l when I asked, it took me close to a minute to return the answer. So, probably you need to type this as I am talking on your terminal and check it for yourself the speed with which s says true or false, while l does not. So, why is this happening?

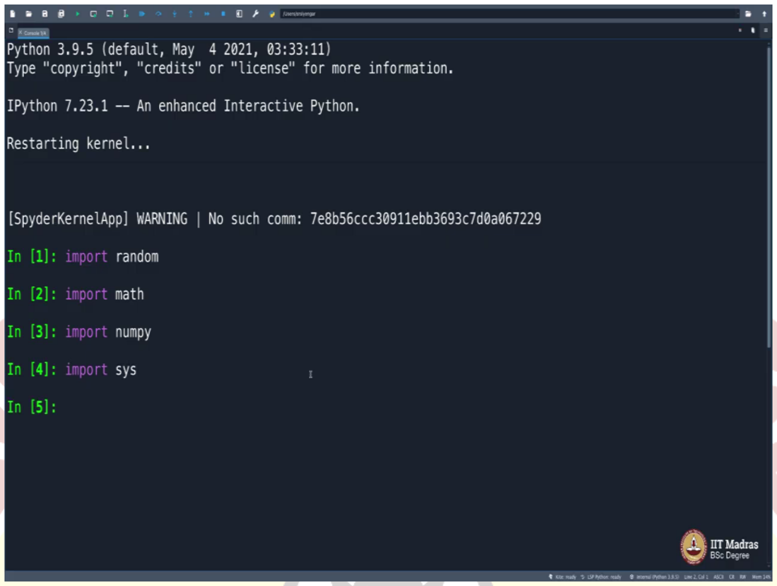
(Refer Slide Time: 16:41)

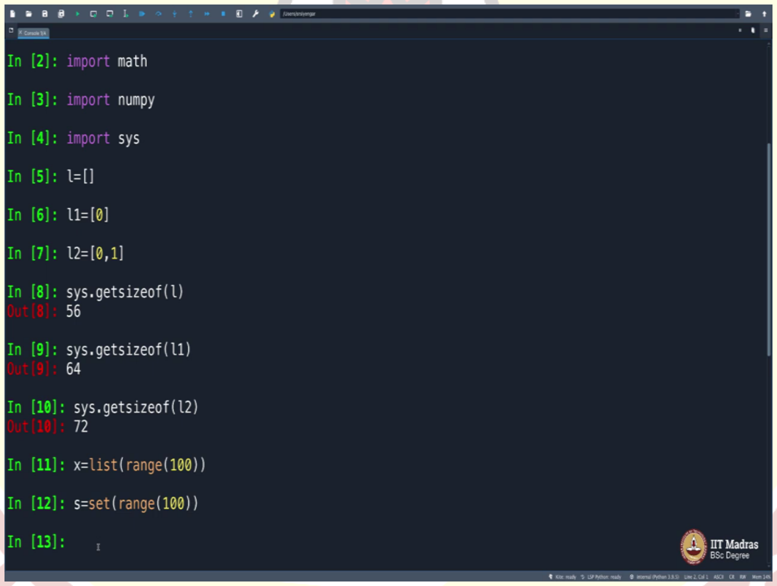


Let us continue to look at this problem that I was discussing, the question of car and bus and the trade off. You see, that is precisely what happened between our list and our set. Here in the list something was difficult, but in the set that something was easy. As you saw searching in the list was very difficult, but in the set searching was very easy. But then why not use this technique here in the list? No, that is not possible.

That is not possible, because there is something in the list that is easy which is not easy in a set. As I say, life is full of trade-offs, so is it is a computer. There are trade-offs here too. You can only achieve a few things that you want. If you want to achieve everything, then it will become very costly for you. So, let us go back and then try to see what exactly is nice about a list that is not there in a set.

(Refer Slide Time: 17:55)


```
Python 3.9.5 (default, May  4 2021, 03:33:11)
Type "copyright", "credits" or "license" for more information.

IPython 7.23.1 -- An enhanced Interactive Python.

Restarting kernel...


[SpyderKernelApp] WARNING | No such comm: 7e8b56ccc30911ebb3693c7d0a067229

In [1]: import random

In [2]: import math

In [3]: import numpy

In [4]: import sys

In [5]:
```


```
In [2]: import math

In [3]: import numpy

In [4]: import sys

In [5]: l=[]

In [6]: l1=[0]

In [7]: l2=[0,1]

In [8]: sys.getsizeof(l)
Out[8]: 56

In [9]: sys.getsizeof(l1)
Out[9]: 64

In [10]: sys.getsizeof(l2)
Out[10]: 72

In [11]: x=list(range(100))

In [12]: s=set(range(100))

In [13]:
```

So, we still are questioning what is so great about lists, why cannot we simply use sets. Let me answer that question on this terminal here by showing you a nice fact. Remember, import that we used to do import random, we also did import math, if you remember, there was also import numpy if you remember in the previous weeks that we taught, there is also something called import sys. We do not need numpy, math and random right now. We only need import sys today.

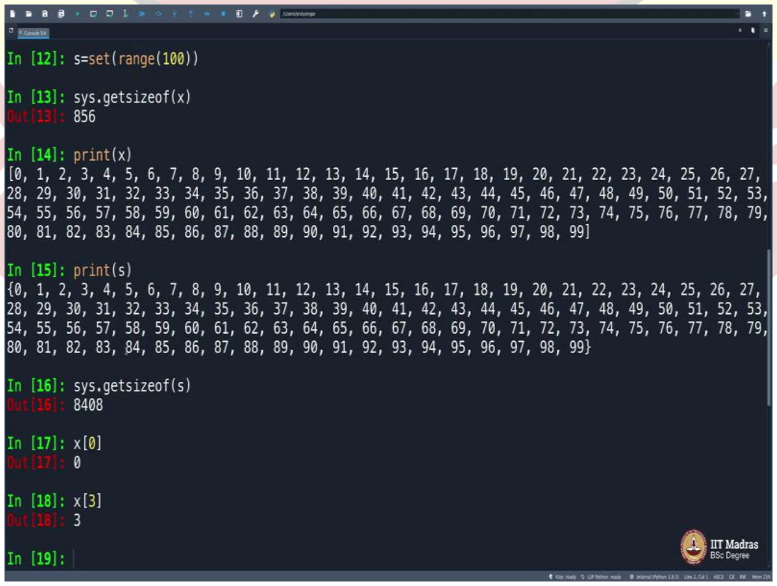
So, what I will do is I will say list is equal to empty list. And I will create another list l1 and I will put one number there. I create another list l2, put another 0 comma 1 there, maybe two entities, and then sys has a very nice function is called getsizeoff. When I put l, it simply tells me what is the size of l in the memory of your computer, whatever that means to you?

Do not break your head much about what his memory and all that. You would not have understood that part yet. Maybe in one of the courses I had you might understand it. But as of now, this is the amount of space you keep. In a book you dedicate some space for writing something and then move to the next page. So, similarly, even your computer allocate some space. This is actually in bytes. It has allocated 56 bytes for l.

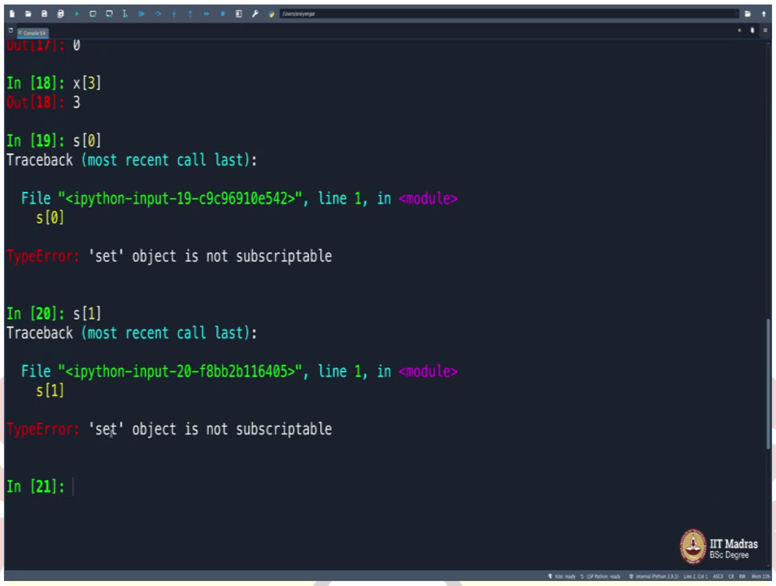
Let us see how much it is allocated for l1. It has 64. Obviously, it should be big, because l1 contains something. L did not contain anything. Actually it should be 0, you see, because it did not contain anything, but then that does not happen. Computer always allocates some space for the list itself. There will be some bookkeeping that it will do. Do not worry about the jargons bookkeeping and even the word memory is a little confusing here. Do not break your head. All I am trying to say is it allocate some space. If I said, l2, obviously it will be more.

So, now let me say x as a list of range 100 entities and a set s of list, I am sorry, it is a set, range 100 entities. Let us forget the rest. I only have x, which is a list. L was used already. That is why I did not use l here. I used x here. X is a list of range 100. S is a set of range 100. So, what are you tempted to see right now.

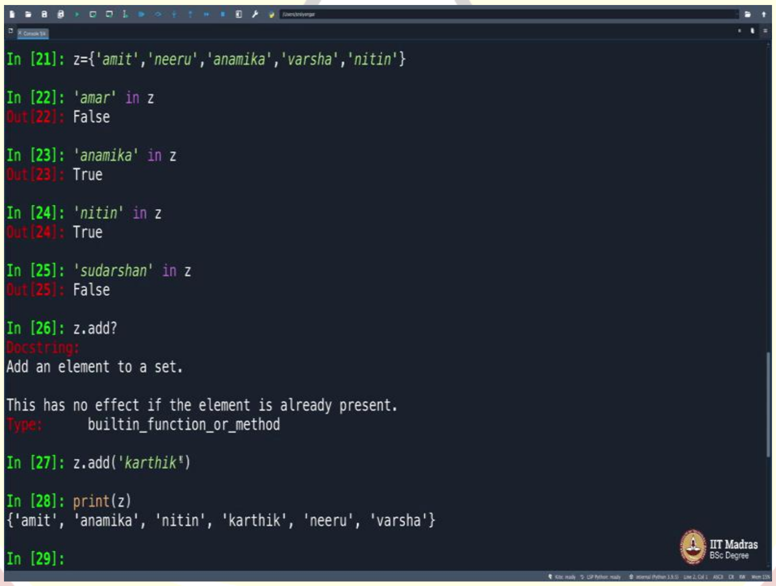
(Refer Slide Time: 20:54)



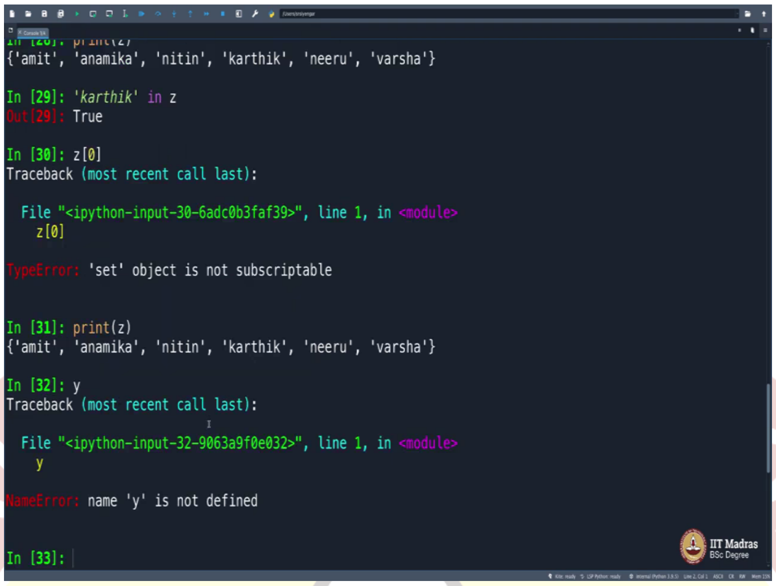
```
In [12]: s=set(range(100))

In [13]: sys.getsizeof(x)
Out[13]: 856

In [14]: print(x)
[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, 47, 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, 92, 93, 94, 95, 96, 97, 98, 99]

In [15]: print(s)
{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, 47, 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, 92, 93, 94, 95, 96, 97, 98, 99}

In [16]: sys.getsizeof(s)
Out[16]: 8408

In [17]: x[0]
Out[17]: 0

In [18]: x[3]
Out[18]: 3

In [19]:
```



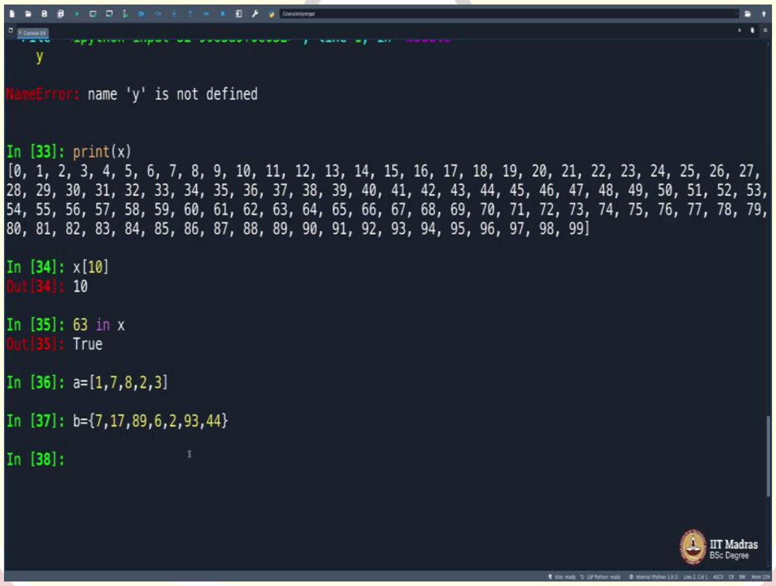
```
Out[17]: 0

In [18]: x[3]
Out[18]: 3

In [19]: s[0]
Traceback (most recent call last):

  File "<ipython-input-19-c9c96910e542>", line 1, in <module>
    s[0]

TypeError: 'set' object is not subscriptable


In [20]: s[1]
Traceback (most recent call last):

  File "<ipython-input-20-f8bb2b116405>", line 1, in <module>
    s[1]

TypeError: 'set' object is not subscriptable


In [21]:
```



```
In [21]: z={'amit','neeru','anamika','varsha','nitin'}

In [22]: 'amar' in z
Out[22]: False

In [23]: 'anamika' in z
Out[23]: True

In [24]: 'nitin' in z
Out[24]: True

In [25]: 'sudarshan' in z
Out[25]: False

In [26]: z.add?
Docstring:
Add an element to a set.

This has no effect if the element is already present.
Type:      builtin_function_or_method

In [27]: z.add('karthik')

In [28]: print(z)
{'amit', 'anamika', 'nitin', 'karthik', 'neeru', 'varsha'}

In [29]:
```



```
{'amit', 'anamika', 'nitin', 'karthik', 'neeru', 'varsha'}

In [29]: 'karthik' in z
Out[29]: True

In [30]: z[0]
Traceback (most recent call last):

  File "<ipython-input-30-6adc0b3faf39>", line 1, in <module>
    z[0]

TypeError: 'set' object is not subscriptable


In [31]: print(z)
{'amit', 'anamika', 'nitin', 'karthik', 'neeru', 'varsha'}

In [32]: y
Traceback (most recent call last):

  File "<ipython-input-32-9063a9f0e032>", line 1, in <module>
    y

NameError: name 'y' is not defined


In [33]:
```



```
    y

NameError: name 'y' is not defined


In [33]: print(x)
[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, 47, 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, 92, 93, 94, 95, 96, 97, 98, 99]

In [34]: x[10]
Out[34]: 10

In [35]: 63 in x
Out[35]: True

In [36]: a=[1,7,8,2,3]

In [37]: b={7,17,89,6,2,93,44}

In [38]:
```

You would want to say what is sys getsizeof x? It is 856. But for s, pause take a guess, s also stores the same 100 numbers. Let me let me help you out by printing x 100 numbers, printer s, 100 numbers, just that is within the flower bracket, this is within the square bracket. My guess is it should be the same. But actually it is not. It takes roughly 10 times more the size as that of a list.

Again, I repeat, the motive is not to confuse you with this sys getsizeof s and x, I am using a very common sensical, I am trying to answer a very common sensical question here. I am only seeing what is the big difference between x and s. You saw right? Searching was so easy in a set, while very difficult in x. Maybe they are doing, Python is doing something more. It is creating a bus instead of a car. Your list was just a car, but this is a big bus. It is doing something that this cannot do.

But then the question is, is there something that the set cannot do which the list does, you can, you will see it in a minute. X of 0 is the first element, x of 3 is the third element and so on. But if I say s of 0, it says some error. If you say s of 1, it throws up an error. In fact, as you can read, it says set object is not subscriptable.

This technical term simply means that you cannot use it like a list. You cannot call the third element of the list. There is nothing called a third element here. These are all a set of elements. I can only tell you, if an element is present here or not, I cannot tell you, what is the first element or second element. That is the downside of s. What is s? S is a set. That is the downside of a set. But an upside of set is you saw. Minus 1 in s when we did, when we searched for an element with 1 billion entries, we observed that it searches in no time, while a list takes a long time.

You might ask me, where exactly should I use a set, where exactly should I use a list? There are many instances where you may want to use a set. Assuming I have the names, I will call it let us say z is equal to all those people whom I have met so far, which is, let us say, Amit and then Neeru and then, let us say, Anamika, let us say, Varsha, and then Nitin and so on. These people I have met. I am maintaining a list of, list as an English list, actually I have called it a set here. According to Python, I am making an entry of who all I met.

Here I do not need to know who was the first person I met, who was the second person I met, but I need to know one thing. I need to know if I have met Amar or not. It will say no. If I have met Anamika or not, it says yes. If I have met Nitin or not, it says yes. Have I met, Sudarshan? It says no, because Sudarshan is not here. So, what if I add Sudarshan here? I can always add Sudarshan here. There is a z dot add. Let us see what it does. Add an element to a set. By add it means it does not mean plus, it means include.

If I say add and say Sudarshan, let me not include Sudarshan. That will confuse you, because this is a list of people I have met. It does not make sense to meet myself unless there is someone else by name Sudarshan. I will say Karthik. And when I say print z, I get all these entries. I can even say how I seen Karthik any time in z, it says yes. But as you know, the z does not have first element or second element and things like that.

Even if you have a huge list of people, how big is z? Z is only some 6 entries here. Even if it had 6 billion entries, it would very quickly tell me if something is there or not. To check whether something is there or not, you are maintaining a big database then you use a set. If you want to refer to the elements in your whatever data set, then you use a list. A list namely, y in the example that we use, was it y, what was that? I forget it. I am so sorry. What was that, x. If it was x, then you can do x of 0, x of 1, x of 10. You can even say is 63 in x, it will return, but it will be very, very slow, so slow that you cannot handle that data.

You can wonder why cannot we mix this list and set both and then try to have a very generic type of data that is why you will study a subject called data structures. Let me type that here. Of course, it will not return anything. It is not a command. It is just for a reference. Data structures in the forthcoming semesters where you will understand what kind of data structure is your list, what kind of data structure is your set, when to use what and more of it is actually sort of philosophical and you will understand it at a very mathematical level than a computational level. Details aside, I did not mean to confuse you people.

The summary of what we discussed is basically there is a set. Let us say a equals 1, 7, 8, 2, 3 and there is a list, a is a list, on a set a is a list, b is another set containing some other elements 6, 2, 93, 44 and so on. Here you can do a few things that you cannot do here. You can do something here very fast that you cannot do here. If you have understood this much that should be enough. With time you will realize where to use a set and when to use a list.