

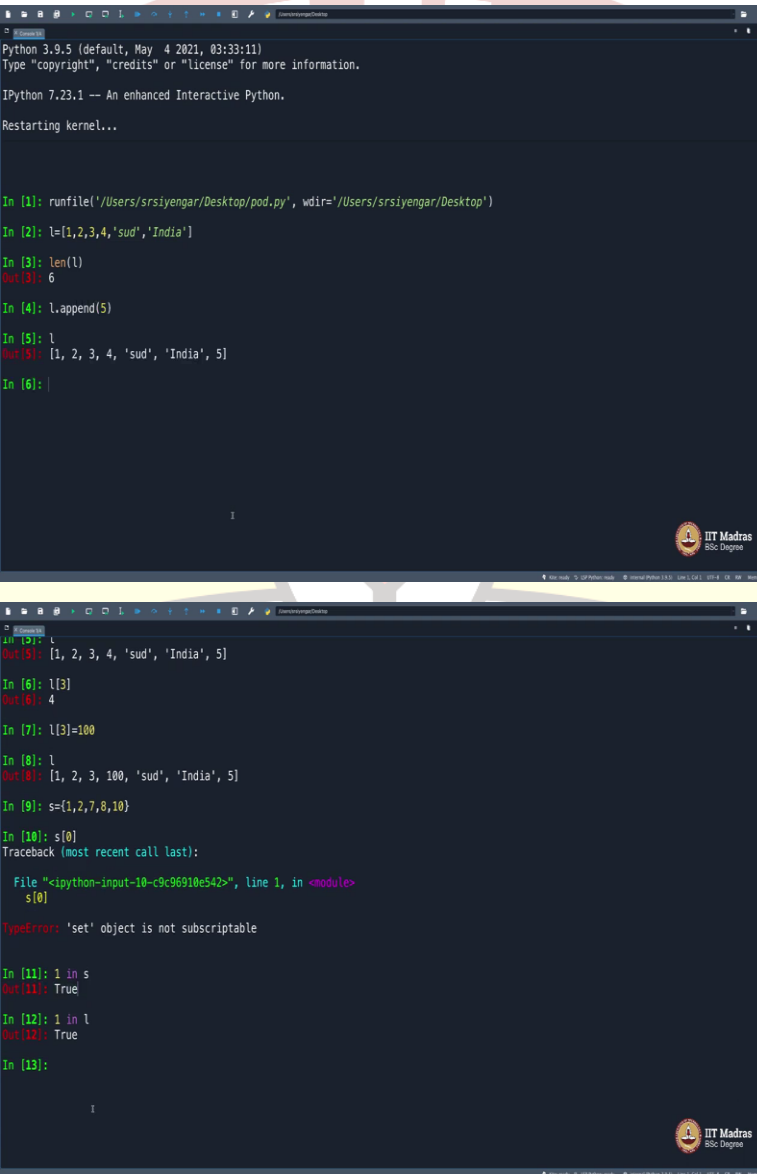


IIT Madras

ONLINE DEGREE

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

(Refer Slide Time: 00:16)



```
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]: runfile('/Users/srsiyengar/Desktop/pod.py', wdir='/Users/srsiyengar/Desktop')
In [2]: l=[1,2,3,4,'sud','India']
In [3]: len(l)
Out[3]: 6
In [4]: l.append(5)
In [5]: l
Out[5]: [1, 2, 3, 4, 'sud', 'India', 5]
In [6]: |

In [5]: l
Out[5]: [1, 2, 3, 4, 'sud', 'India', 5]
In [6]: |

In [7]: l
Out[7]: [1, 2, 3, 4, 'sud', 'India', 5]
In [8]: l[3]
Out[8]: 4
In [9]: l[3]=100
In [10]: l
Out[10]: [1, 2, 3, 100, 'sud', 'India', 5]
In [11]: s={1,2,7,8,10}
In [12]: s[0]
Traceback (most recent call last):
  File "<ipython-input-10-c9c96910e542>", line 1, in <module>
    s[0]
TypeError: 'set' object is not subscriptable

In [11]: 1 in s
Out[11]: True
In [12]: 1 in l
Out[12]: True
In [13]: |
```

```
Out[11]: True
In [12]: 1 in l
Out[12]: True
In [13]: d={}
In [14]: d['sudarshan']=9898989898
In [15]: d['ramya']=7878787878
In [16]: d['ravi']=9879766965
In [17]: print(d)
{'sudarshan': 9898989898, 'ramya': 7878787878, 'ravi': 9879766965}
In [18]: d['sudarshan']
Out[18]: 9898989898
In [19]: print(l)
[1, 2, 3, 100, 'sud', 'India', 5]
In [20]: l[0]
Out[20]: 1
In [21]: d[0]
Traceback (most recent call last):
  File "<ipython-input-21-123a9cc6df61>", line 1, in <module>
    d[0]
KeyError: 0
In [22]: d
Out[22]: {}

In [22]: d[1]
Traceback (most recent call last):
  File "<ipython-input-22-abe28337115>", line 1, in <module>
    d[1]
KeyError: 1

In [23]: print(d)
{'sudarshan': 9898989898, 'ramya': 7878787878, 'ravi': 9879766965}
In [24]: d['ramya']
Out[24]: 7878787878
In [25]: d['iit']
Traceback (most recent call last):
  File "<ipython-input-25-7632d9381124>", line 1, in <module>
    d['iit']
KeyError: 'iit'

In [26]: print(d)
{'sudarshan': 9898989898, 'ramya': 7878787878, 'ravi': 9879766965}
In [27]:
```

So, we saw what is the list. List is simply put some numbers here or any other string as you want and this creates a list. Length of list, it shows you the list, and you can append something onto the list and you have 5 here. You just appended 5 here. So, you can even change the value of, let us say, 1 of 3. What is 1 of 3, 1 of 3 is 4, you can say 1 of equals 100 and that gets changed, you see.

But the set, as you know, set was, what was the set 1, 2, 7, 8, 10, let us say, and s did not have this facility. You could not index s. So, I told you that there are differences between the, between l and s, you observed that you can find out an element in s, you can also find out an element in l, but this works really fast. If the size is over a billion, this is super duper fast.

Now, let us go ahead and discuss something that is slightly complex at the level of a computer, but is actually easy on your mind. It is called a dictionary data structure. And trust me, this is going to be slightly difficult on your minds to begin with. Although, I say it is easy, but it takes some time for this concept to become easy on your mind. So stay patient. As and always, only when we demonstrate the usage of a concept, will you really realize that now I understand why this is used.

So, I will tell you the idea behind this concept called dictionaries and then put that to use that will make it very clear to you. So, if I say `d` is equal to simply open and close bracket, it becomes a dictionary. When you include single numbers like this, it becomes a set. But when you only start and end an empty bracket like this, it becomes a dictionary. And you can say `d` of anything you want, let us say Sudarshan, is equal to my let us say phone number, 98989898, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, perfect. So, this is actually not my phone number. Do not give me a ring. This is just an example.

So, `d` of let us say someone else by name, Ramya and her phone number, let us say is so much 1, 2, 3, 4, 5, 1, 2, 3, 4, 5. So and then, Ravi, his phone number happens to be 1, 2, 3, 4, 5, 6, 7, 8, 9, let me put 9 here and make it a 10 digit number. This takes values like this. And when you print `d`, it shows like this. And whenever you want to find out the value of any entry here, you simply put `d` and in bracket you put that and you will get the value here. Is that clear?

So, do you remember how the list works? I had created a list `l` here. List simply contains entries like this. If you say `l` of 0, it will show you the first element. But here in case of a dictionary is a `d` of 0, it simply does not know what to do. It will go and see if 1 is present in `d`. You can only put `d` big bracket. Inside this bracket you must specify whatever you have here, let us say, Ramya. If you put something that is not present, you only have Sudarshan, Ramya and Ravi here. If you let us say put IIT here, it will throw an error saying there is nothing like that here, key error, IIT that is not available.

So, of what use will this be? Of what use is this `d` here, which you created by saying `d` is equal to so much and then you said `d` of something, what was that? What does `d` of Ravi equal so much, `d` of Ramya equal so much, `d` of Sudarshan equal so much, you created it and finally you had your `d`, which is this, of what uses this facility? I mean, sounds like a very useless whatever facility in

Python. I do not see any immediate use for this. But remember, from your computational thinking course, there was extensive discussion on dictionaries.

If I can remind you, Professor Madhavan, in fact, tries to take the example of the bills in a shop, in a supermarket, and then tries to take the sequence number, rather the bill number, and then figures out what all transactions were done. In fact, he explains what is a table, and then creates a dictionary of entries. If it sounds, if I sound jargonic, please go back and then watch your computational thinking part of, dictionary discussion in that course.

(Refer Slide Time: 06:08)

Max in a single iteration and max in two iterations (non-nested)

It was Monday morning. Swaminathan was reluctant to open his eyes. He considered Monday specially unpleasant in the calendar. After the delicious freedom of Saturday and Sunday, it was difficult to get into the Monday mood of work and discipline. He shuddered at the very thought of school: that dismal yellow building; the fire-eyed Vedanayagam, his class-teacher; and the Head Master with his thin long cane....

Max-frequency 6

It	2	specially	1	get	1	building	1
was	3	unpleasant	1	into	1	fire-eyed	1
Monday	3	in	1	mood	1	Veda-	
morning	1	the	6	work	1	nayagam	1
Swaminathan	1	calendar	1	discipline	1	class-	
reluctant	1	After	1	shuddered	1	teacher	1
to	2	delicious	1	at	1	Head	1
open	1	freedom	1	very	1	Master	1
his	2	of	3	thought	1	with	1
eyes	1	Saturday	1	school	1	thin	1
He	2	and	3	that	1	long	1
considered	1	Sunday	1	dismal	1	cane	1
		difficult	1	yellow	1		

But for the time being, I will now open the video of Professor Madhavan which I kept it ready right now just so that, I was actually wondering from where to introduce dictionaries to you people. And I was very happy that it was discussed in the competition thinking course. I am going to particularly take this example that is discussed here.

Says, it was a Monday morning Swaminathan was reluctant to open his eyes. He considered Monday specially unpleasant in the calendar. After the delicious freedom of Saturday and Sunday, it was difficult to get into the Monday mood of work and discipline. He shuddered at the very thought of school, that dismal yellow building, the fire-eyed Vedanayagam, his class teacher and the headmaster with his thin long cane, made a wonderful work by R.K. Narayan.

If you did not know this is a pick from Malgudi Days Swami and friends. If you do not know what that is, you probably should read the story, better even watch the series. It was made in the

1980s, a very popular tele serial it was. It is probably available online right now. Anyway details aside. I felt good reading it like a English lesson. In fact, if I remember it right, we have had this lesson in one of our, in our college days, if I remember it right. So, cut the carp.

So, let us look at what is being told here. Here, Professor Madhavan very nicely explains how you can find out the most frequently occurring word. So, it occurs twice, he says, because it is here, it is also here. Was appears thrice, was is here, was is here, where else is was, one here, one here, and maybe there is one more? Where is it? Did I miss it? Let me go through it slowly, was, was, 2 was and there are 3 was, very thought of school that dismal yellow building, the fire-eyed program, his class teacher and the head master with his thin long cane, why is was repeated thrice. The calendar of the delicious freedom of Saturday and Sunday, it was difficult. Here it is.

So, you see, I purposefully did this exercise, because I do not want to cut this part where I stumbled to see where it was. That is how humans count. That is how humans compute. And that is why you need a computer. I mean, even if it is a long, long, long text, I would not, in fact, it will get complex for my naked eye to figure out how many times a word is repeated in a small text as small as this. Imagine it was several bytes of data with over thousands of words, some 100,000 words. How will you figure out the repetition? How many times a word is repeated?

So, you need a computer and Professor Madhavan nicely explains how you can write a piece of code and figure out the max frequently occurring word. In fact, you see the word is actually the here. But the program, if you remember it right, displays the frequency, max frequency. If I am sounding Greek and Latin, you probably should go to YouTube and search for this lesson, watch it and then come back. Now, how do I code this on a computer? This very problem.

(Refer Slide Time: 9:45)

```
KeyError: 'iit'

In [26]: print(d)
{'sudarshan': 9898989898, 'ramya': 7878787878, 'ravi': 9879766965}

In [27]: malgudi
Out[27]:
['It',
 'was',
 'Monday',
 'morning',
 'Swaminathan',
 'was',
 'reluctant',
 'to',
 'open',
 'his',
 'eyes',
 'he',
 'considered',
 'Monday',
 'specially',
 'unpleasant',
 'in',
 'the',
 'calendar',
 'After',
 'the',
 'delicious',
 'freedom',
 'of',
 'Saturday',
 'and',
 'Sunday',
 'it',
 'was',
 'difficult',
 'to',
 'get',
 'into',
 'the',
 'Monday',
 'mood',
 'of',
 'work',
 'and',
 'discipline',
 'He',
 'shuddered',
 'at',
 'the',
 'very',
 'thought',
 'of',
 'school',
 'the',
 'dismal',
 'yellow',
 'building',
 'the',
 'fire',
 'eyed',
 'Vedanayagan',
 'his',
 'class',
 'teacher',
 'and',
 'headmaster',
 'with',
 'his',
 'thin',
 'long',
 'cane']

In [31]: l=[1,2,3,5,79]

In [32]: for i in range(len(l)):
...:     print(i)
...:
0
1
2
3
4

In [33]: for x in l:
...:     print(x)
...:
1
2
3
5
79

In [34]:
```

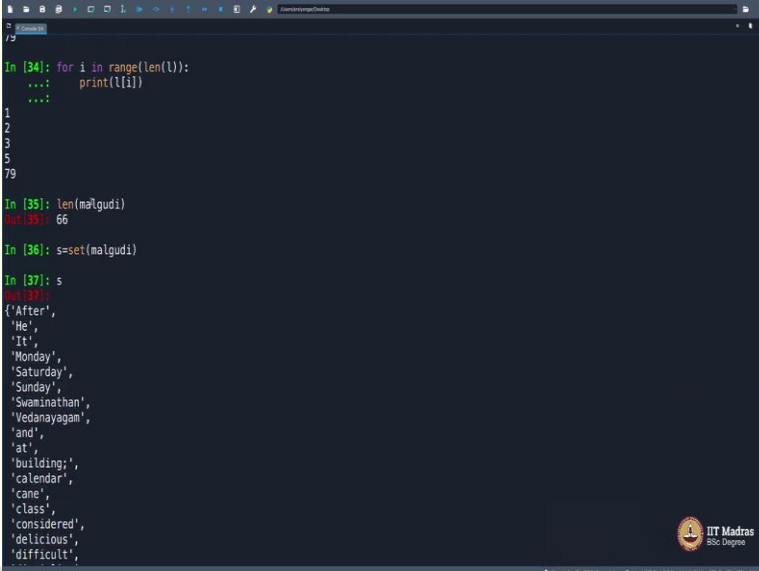
And for that, I have it prepared. In fact, in Malgudi, you must be thinking what magic is this, how could he get this the moment he typed Malgudi. It is not a built in entity in Python. I stored these words in a list called Malgudi. So, what is what is Malgudi have? Malgudi is a list where the zeroth element is it and the first element is was. In fact, you can print the entire list. If you say print Malgudi, it prints the entire list like this one next to the other.

So, it was Monday morning Swaminathan has reluctant, was reluctant to open his eyes, he considered Monday etcetera, etcetera. I have removed all the dots, commas and punctuation and things like that. And even these 3 dots here, 4 dots here I have removed. I have made it only

words. So, that it is easy for me to search through. So, how will I search through the words here? Is there an easy way? Let me see.

So, all this while I was using a list, whenever I use list of numbers like this, I would say for i in range len of l print i would print the numbers like this. So, there is another way of doing this. You can always say for x or i or whatever, x in l print x. This is very easy. Now, that I introduce sets to you, sets, the word in there is used for belongingness. But here when I say for x in l, I mean go through every element of l through this for loop. And then whatever you give here, that x variable will come and sit here and it will print you the, give you the values of that particular variable.

(Refer Slide Time: 11:52)



```
In [34]: for i in range(len(l)):
...:     print(l[i])
...:
1
2
3
5
79

In [35]: len(malgudi)
Out[35]: 66

In [36]: s=set(malgudi)

In [37]: s
Out[37]:
{'After',
'He',
'It',
'Monday',
'Saturday',
'Sunday',
'Swaminathan',
'Vedanayagam',
'and',
'at',
'building',
'calendar',
'cane',
'class',
'considered',
'delicious',
'difficult',
```



```
In [38]: print(s)
{'and', 'specially', 'to', 'After', 'freedom', 'difficult', 'shuddered', 'was', 'of', 'headmaster', 'cane', 'discipline',
'building', 'Sunday', 'teacher', 'he', 'his', 'unpleasant', 'work', 'the', 'Saturday', 'it', 'dismal', 'reluctant', 'calendar',
'Vedanayagam', 'yellow', 'school', 'It', 'eyed', 'at', 'mood', 'class', 'He', 'in', 'long', 'Swaminathan', 'morning', 'delicious',
'considered', 'get', 'open', 'fire', 'with', 'thin', 'eyes', 'thought', 'Monday', 'very', 'into'}

In [39]: print(malgudi)
['It', 'was', 'Monday', 'morning', 'Swaminathan', 'was', 'reluctant', 'to', 'open', 'his', 'eyes', 'he', 'considered', 'Monday',
'specially', 'unpleasant', 'in', 'the', 'calendar', 'After', 'the', 'delicious', 'freedom', 'of', 'Saturday', 'and', 'Sunday', 'it',
'was', 'difficult', 'to', 'get', 'into', 'the', 'Monday', 'mood', 'of', 'work', 'and', 'discipline', 'He', 'shuddered', 'at', 'the',
'very', 'thought', 'of', 'school', 'the', 'dismal', 'yellow', 'building', 'the', 'fire', 'eyed', 'Vedanayagam', 'his', 'class',
'teacher', 'and', 'headmaster', 'with', 'his', 'thin', 'long', 'cane']

In [40]: len(s)
Out[40]: 50

In [41]: len(malgudi)
Out[41]: 66

In [42]: malgudi[10]
Out[42]: 'eyes'

In [43]: malgudi[30]
Out[43]: 'to'

In [44]: malgudi[31]
Out[44]: 'get'

In [45]: malgudi[47]
Out[45]: 'school:'

In [46]: malgudi[47]='school'

In [47]: |

In [46]: malgudi[47]='school'

In [47]: malgudi[51]
Out[47]: 'building:'

In [48]: malgudi[51]='building'

In [49]: print(malgudi)
['It', 'was', 'Monday', 'morning', 'Swaminathan', 'was', 'reluctant', 'to', 'open', 'his', 'eyes', 'he', 'considered', 'Monday',
'specially', 'unpleasant', 'in', 'the', 'calendar', 'After', 'the', 'delicious', 'freedom', 'of', 'Saturday', 'and', 'Sunday', 'it',
'was', 'difficult', 'to', 'get', 'into', 'the', 'Monday', 'mood', 'of', 'work', 'and', 'discipline', 'He', 'shuddered', 'at', 'the',
'very', 'thought', 'of', 'school', 'the', 'dismal', 'yellow', 'building', 'the', 'fire', 'eyed', 'Vedanayagam', 'his', 'class',
'teacher', 'and', 'headmaster', 'with', 'his', 'thin', 'long', 'cane']

In [50]: s=set(malgudi)

In [51]: len(s)
Out[51]: 50

In [52]: len(malgudi)
Out[52]: 66

In [53]: for x in malgudi:
....:     print(x)
```

So, for `i in range len of l` print `i` this is also fine. I am sorry, what am I doing here. We should not print `i`. I am very, very sorry. The small mistake here. I should say for `i in range len of l` say print `l of i` that prints the value of `l`, values of `l`. This was just printing `i`, you see. For `i in range len of l` will be 1, 2, 3, 4, 5, so print from 0 to 5. So, it will be printing 0, 1, 2, 3, 4. So, you should rather print `l of i`. This is same as saying for `x in l` print `x`, a quick tip for you guys. From now onwards, we will try using this as it is easy on your minds.

So, now Malgudi has so many words. How many words does Malgudi have, `len of malgudi`, it has 66 words. However, there could be repetitions. So, we know sets. So, I can say `s` equals set of Malgudi, now repetitions will be gone, you see, print `s`. All repetitions are gone right now in

flower brackets. But if you print Malgudi, you will get everything with repetitions. As you can see my world was which I stumbled. You see, was is repeated twice here, but was is somewhere once only displayed here and I need to find that. Let me find where is was here. Oh my god. You see how difficult is searching manually. Was is here.

So, what is the story so far? The story so far is you create a list like this, you display it, you can do it like this or you can also do it like this. Tip, that is all. This entire circles here was just a small tip that instead of doing, I am sorry, not this, this was a mistake, instead of doing, writing this code, you can instead write this code. I hope it is clear for you. If it is not clear, pause and try to write this code and write this code, you will realize they both do the same thing.

Story so far, I had written a list containing all the words that Professor Madhavan discussed in computational thinking course. In one of his lessons, he quoted this paragraph to find out the most frequently occurring word and it is basically the frequency, not exactly the word, but he wanted to, he was interested to find out the frequency of the occurrence of a word to its max. Written out to be the and it was 6.

I have now created the list, manually I created Malgudi is equal to it and then appended was, appended Monday, appended morning, it was manually done. Believe in me, if you suspect then, so you are right. In fact, I wrote a small piece of code, where I could remove all the unwanted stuff and then keep these things. Anyway that is out of the scope of our discussion right now. So, I am not going to discuss that. We still have some problems here. Namely school dot, school colon, we should fix that. But anyways, do not worry. Let us go ahead right now and then see that Malgudi has length of Malgudi 66. If you make it a set, it becomes smaller.

So, what is the length of s here? It is 50. But length of Malgudi is 66. So, far so good. You understand Malgudi is displayed here. Let me now display it once again. Malgudi is simply this, while s is this. There are a couple of problems here as I just now saw. You have school column. Let me make it school so that there are no mistakes. How do I do that? Malgudi of, what is that, 10 is eyes. Where is eyes, maybe what word is school here. 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, roughly some 30, I guess. Let me just check. What is Malgudi of 30? It is two. Malgudi of 31 is get. Where is get?

So, there is a nice way to do this. Let we use function here. I can always remove and then append, no that may not be a good idea. So, let me just figure out. If this is 31, get is 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47 is school. So, I will just say 47 is equal to school and just so that I removed this. Anyway, I do not want to edit this part, where I want you people to understand that you can also do something like this. When you cleanse the data, you tend to have such errors.

But you can sometimes manually fix or in fact write another piece of code to fix this. That is not going to be difficult. This is 47, 48, 49, 50, 51. So, Malgudi of 51 it says building colon, I want to make it simply building. Any other place that requires any change, looks like we are through. There is no problem.

Now, let me print Malgudi and I get clean data. I am sorry for this. That was a oversight error. I did not know how this colon and semicolon popped in here, but we have fixed that right now. So, Malgudi is so much s equals set of Malgudi is the same old thing, length of s will be so much, len of Malgudi is so much.

Now, slowly look at the magic that is going to happen. One reason why I like Python is the way you code is the way you speak in English. Now, what do I mean by that with that? That I mean, if I said for x in malgudi print x, what does this mean? It means, for all values x in malgudi, you simply print those values. It prints all those values is in your mind. So, now what I will do is I will create a dictionary.

(Refer Slide Time: 18:26)

```
In [54]: d={}
In [55]: for x in s:
...:     d[x]=0
...:
In [56]: print(s)
{'and', 'specially', 'to', 'After', 'freedom', 'difficult', 'shuddered', 'was', 'of', 'headmaster', 'cane',
'discipline', 'Sunday', 'teacher', 'he', 'his', 'unpleasant', 'work', 'the', 'Saturday', 'it', 'school', 'dismal',
'reluctant', 'calendar', 'Vedanayagam', 'yellow', 'It', 'eyed', 'at', 'mood', 'class', 'He', 'building', 'in', 'long',
'Swaminathan', 'morning', 'delicious', 'considered', 'get', 'open', 'fire', 'with', 'thin', 'eyes', 'thought', 'Monday',
'very', 'into'}

In [57]: print(malgudi)
['It', 'was', 'Monday', 'morning', 'Swaminathan', 'was', 'reluctant', 'to', 'open', 'his', 'eyes', 'he', 'considered',
'Monday', 'specially', 'unpleasant', 'in', 'the', 'calendar', 'After', 'the', 'delicious', 'freedom', 'of', 'Saturday',
'and', 'Sunday', 'it', 'was', 'difficult', 'to', 'get', 'into', 'the', 'Monday', 'mood', 'of', 'work', 'and',
'discipline', 'He', 'shuddered', 'at', 'the', 'very', 'thought', 'of', 'school', 'the', 'dismal', 'yellow', 'building',
'the', 'fire', 'eyed', 'Vedanayagam', 'his', 'class', 'teacher', 'and', 'headmaster', 'with', 'his', 'thin', 'long',
'cane']

In [58]: print(d)
{'and': 0, 'specially': 0, 'to': 0, 'After': 0, 'freedom': 0, 'difficult': 0, 'shuddered': 0, 'was': 0, 'of': 0,
'headmaster': 0, 'cane': 0, 'discipline': 0, 'Sunday': 0, 'teacher': 0, 'he': 0, 'his': 0, 'unpleasant': 0, 'work': 0,
'the': 0, 'Saturday': 0, 'it': 0, 'school': 0, 'dismal': 0, 'reluctant': 0, 'calendar': 0, 'Vedanayagam': 0, 'yellow':
0, 'It': 0, 'eyed': 0, 'at': 0, 'mood': 0, 'class': 0, 'He': 0, 'building': 0, 'in': 0, 'long': 0, 'Swaminathan': 0,
'morning': 0, 'delicious': 0, 'considered': 0, 'get': 0, 'open': 0, 'fire': 0, 'with': 0, 'thin': 0, 'eyes': 0,
'thought': 0, 'Monday': 0, 'very': 0, 'into': 0}

In [59]:

In [59]: d['get']
Out[59]: 0

In [60]: for x in malgudi:
...:     d[x]=d[x]+1
...:

In [61]: print(d)
{'and': 3, 'specially': 1, 'to': 2, 'After': 1, 'freedom': 1, 'difficult': 1, 'shuddered': 1, 'was': 3, 'of': 3,
'headmaster': 1, 'cane': 1, 'discipline': 1, 'Sunday': 1, 'teacher': 1, 'he': 1, 'his': 3, 'unpleasant': 1, 'work': 1,
'the': 6, 'Saturday': 1, 'it': 1, 'school': 1, 'dismal': 1, 'reluctant': 1, 'calendar': 1, 'Vedanayagam': 1, 'yellow':
1, 'It': 1, 'eyed': 1, 'at': 1, 'mood': 1, 'class': 1, 'He': 1, 'building': 1, 'in': 1, 'long': 1, 'Swaminathan': 1,
'morning': 1, 'delicious': 1, 'considered': 1, 'get': 1, 'open': 1, 'fire': 1, 'with': 1, 'thin': 1, 'eyes': 1,
'thought': 1, 'Monday': 3, 'very': 1, 'into': 1}

In [62]: d['his']
Out[62]: 3

In [63]: malgudi
Out[63]:
['It',
'Monday',
'morning',
'Swaminathan',
'was',
'reluctant',
'to',
'open',
'his',
'eyes',
'the']
```

Let me increase the font, so that it is easy on your eyes. Let me create a dictionary and for all the values in s, why s, because there could be repetitions here. So, s is what? S is same as Malgudi, but repetitions removed. For x in s, I will say d of x is equal to 0. What do I mean by this? This is a dictionary as you know. It will take x in s. What was s? Remember, s was all the words.

Note, you have to open your terminal and code with me. If you do not code with me, if you are not coding already, things are going to get slightly complicated. You will not be able to appreciate magic. Magic will be understandable if you are coding with me, otherwise magic will

remain mysterious. You will not understand what I am doing. So, I say print s, you get the set. All these are non-repeated words from your Malgudi list, from this list.

So, when I say for x in s, it goes through all values of s, these words and then creates a dictionary with that word followed by 0. Let me print d and z, you see. You remember, I typed Sudarshan, Ramya, Ravi and put phone numbers by using colon there. So, it says and is 0, specially is 0, to is 0, after is 0, freedom is 0, difficulty is 0, so on and so forth. Now, when I say d of get within quotes, it will see where is get here, get is somewhere here. It will give me the value. Anyway every value is 0, so it is going to say 0 here.

But then pause for a second and observe this code for x in Malgudi, Malgudi means what, the actual text, this text, I am going through it one by one. What I do is d of x is equal to, it was 0. Now, I say, d of x equals whatever d of x was add 1 to it. What do you mean by this? By this you mean if for x in Malgudi, for a word in this list, as you go from it was Monday morning Swaminathan, you go to d of that word, d of it.

And what was d of it. d of it should be somewhere here. It gets difficult to search the words here. You see, d of it is 0 here. So, d of it will be whatever d of it was 0 plus 1. So, d of it will actually become 1 here when you do this. And that is true for all words. Was will become 1. But then the second time you see was, here it will make was as 2. Pause for a second, think what I am doing here. Pretty straightforward English statement written as a two line Python code. I say enter here and you will see print d.

You have what, the number of occurrences of and, specially, to, after, freedom, difficult, shuddered, was has come thrice, you see. I was stumbling to find out how many times was, was repeated. Of is again thrice, headmaster comes ones, cane comes ones, discipline comes ones, so on, his comes three. Let us do one thing. Let us go to Professor Madhavan's lecture and try to figure out if it is the same.

So, his is actually it is showing us 3, but here it says 2, I do not know why. Let us see. Maybe there is some mistake here or mistake from our side or from this is his only ones here. D of x equals, our code cannot be wrong. So, let us see where the problem is. D of his is I observed just now randomly that d of his turning out to be 3. So, in Malgudi, let us say how many times his has come.

(Refer Slide Time: 23:06)

```
In [64]: print(malgudi)
['It', 'was', 'Monday', 'morning', 'Swaminathan', 'was', 'reluctant', 'to', 'open', 'his', 'eyes', 'he', 'considered',
'Monday', 'specially', 'unpleasant', 'in', 'the', 'calendar', 'After', 'the', 'delicious', 'freedom', 'of', 'Saturday',
'and', 'Sunday', 'it', 'was', 'difficult', 'to', 'get', 'into', 'the', 'Monday', 'mood', 'of', 'work', 'and',
'discipline', 'He', 'shuddered', 'at', 'the', 'very', 'thought', 'of', 'school', 'the', 'dismal', 'yellow', 'building',
'the', 'fire', 'eyed', 'Vedanayagam', 'his', 'class', 'teacher', 'and', 'headmaster', 'with', 'his', 'thin', 'long',
'cane']

In [65]: max=0

In [66]: d={}

In [67]: for x in s:
...:     d[x]=0
...:

In [68]: print(d)
{'and': 0, 'specially': 0, 'to': 0, 'After': 0, 'freedom': 0, 'difficult': 0, 'shuddered': 0, 'was': 0, 'of': 0,
'headmaster': 0, 'cane': 0, 'discipline': 0, 'Sunday': 0, 'teacher': 0, 'he': 0, 'his': 0, 'unpleasant': 0, 'work': 0,
'the': 0, 'Saturday': 0, 'it': 0, 'school': 0, 'dismal': 0, 'reluctant': 0, 'calendar': 0, 'Vedanayagam': 0, 'yellow':
0, 'It': 0, 'eyed': 0, 'at': 0, 'mood': 0, 'class': 0, 'He': 0, 'building': 0, 'in': 0, 'long': 0, 'Swaminathan': 0,
'morning': 0, 'delicious': 0, 'considered': 0, 'get': 0, 'open': 0, 'fire': 0, 'with': 0, 'thin': 0, 'eyes': 0,
'thought': 0, 'Monday': 0, 'very': 0, 'into': 0}

In [69]: for x in malgudi:
...:     d[x]=d[x]+1
...:     if d[x]>max:
...:         max=d[x]
...:

In [70]: print(d)
{'and': 3, 'specially': 1, 'to': 2, 'After': 1, 'freedom': 1, 'difficult': 1, 'shuddered': 1, 'was': 3, 'of': 3,
'headmaster': 1, 'cane': 1, 'discipline': 1, 'Sunday': 1, 'teacher': 1, 'he': 1, 'his': 3, 'unpleasant': 1, 'work': 1,
'the': 6, 'Saturday': 1, 'it': 1, 'school': 1, 'dismal': 1, 'reluctant': 1, 'calendar': 1, 'Vedanayagam': 1, 'yellow':
1, 'It': 1, 'eyed': 1, 'at': 1, 'mood': 1, 'class': 1, 'He': 1, 'building': 1, 'in': 1, 'long': 1, 'Swaminathan': 1,
'morning': 1, 'delicious': 1, 'considered': 1, 'get': 1, 'open': 1, 'fire': 1, 'with': 1, 'thin': 1, 'eyes': 1,
'thought': 1, 'Monday': 3, 'very': 1, 'into': 1}

In [71]: print(max)
6

In [72]: d={}

In [73]: for x in s:
...:     d[x]=0
...:

In [74]: max=0

In [75]: answer_word=''

In [76]: for x in malgudi:
...:     d[x]=d[x]+1
...:     if d[x]>max:
...:         max=d[x]
...:         answer_word=x
```

```
In [75]: answer_word=''

In [76]: for x in malgudi:
...:     d[x]=d[x]+1
...:     if (d[x])>max:
...:         max=d[x]
...:         answer_word=x
...:

In [77]: print(d)
{'and': 3, 'specially': 1, 'to': 2, 'After': 1, 'freedom': 1, 'difficult': 1, 'shuddered': 1, 'was': 3, 'of': 3,
'headmaster': 1, 'came': 1, 'discipline': 1, 'Sunday': 1, 'teacher': 1, 'he': 1, 'his': 3, 'unpleasant': 1, 'work': 1,
'the': 6, 'Saturday': 1, 'it': 1, 'school': 1, 'dismal': 1, 'reluctant': 1, 'calendar': 1, 'Vedanayagan': 1, 'yellow':
1, 'it': 1, 'eyed': 1, 'at': 1, 'mood': 1, 'class': 1, 'He': 1, 'building': 1, 'in': 1, 'long': 1, 'Swaminathan': 1,
'morning': 1, 'delicious': 1, 'considered': 1, 'get': 1, 'open': 1, 'fire': 1, 'with': 1, 'thin': 1, 'eyes': 1,
'thought': 1, 'Monday': 3, 'very': 1, 'into': 1}

In [78]: print(max)
6

In [79]: print(answer_word)
the

In [80]:
```

```
'much',
'You',
'would',
'certainly',
'have',
'been',
'burned',
'had',
'you',
'lived',
'a',
'few',
'centuries',
'ago',
'It',
'is',
'true',
'that',
...]

In [84]: len(sherlock)
Out[84]: 109021

In [85]: len(malgudi)
Out[85]: 66

In [86]: for x in sherlock:
...:     print(x)
```



```
account
of
the
adventure
of
Traceback (most recent call last):
  File "<ipython-input-86-360cb5b4e846>", line 1, in <module>
    print(x),
  File "ipykernel/iostream.pyc", line 401, in write
  File "ipykernel/iostream.pyc", line 281, in schedule
  File "/Applications/Spyder.app/Contents/Resources/lib/python3.9/zmq/sugar/socket.py", line 505, in send
    return super(Socket, self).send(data, flags=flags, copy=copy, track=track)
  File "zmq/backend/cython/socket.pyx", line 718, in zmq.backend.cython.socket.Socket.send
  File "zmq/backend/cython/socket.pyx", line 765, in zmq.backend.cython.socket.Socket.send
  File "zmq/backend/cython/socket.pyx", line 242, in zmq.backend.cython.socket._send_copy
  File "zmq/backend/cython/checkrc.pxd", line 13, in zmq.backend.cython.checkrc._check_rc
KeyboardInterrupt

In [87]: for x in sherlock:
...:     print(x, end='')
...:
```

```
theseesnoobjectiontoyourcooperationandthateventhinkthatitmightbeofsomeassistanceIwillcallatfouroclockintheafternoonan
dshouldyouhaveanyotherengagementatthattimeIhopethatyouwillpostponeitasthismatterisofparamountimportanceYoursfaithfullyRO
BERTSISIMONItisdatedfromGrosvenorMansionswrittenwithaquillpenandthenoblelordhashadthemisfortunetogetasnearofinkupontheou
tersideofhisrightlittlefingerremarkedHolmesashefoldeduptheepistleHesaysfouroclockItisthreenowHewillbehereinanhourTraceba
ck (most recent call last):
  File "<ipython-input-87-50fc8cc22b2e>", line 1, in <module>
    print(x, end='')
  File "ipykernel/iostream.pyc", line 401, in write
  File "ipykernel/iostream.pyc", line 281, in schedule
  File "/Applications/Spyder.app/Contents/Resources/lib/python3.9/zmq/sugar/socket.py", line 505, in send
    return super(Socket, self).send(data, flags=flags, copy=copy, track=track)
  File "zmq/backend/cython/socket.pyx", line 718, in zmq.backend.cython.socket.Socket.send
  File "zmq/backend/cython/socket.pyx", line 765, in zmq.backend.cython.socket.Socket.send
  File "zmq/backend/cython/socket.pyx", line 242, in zmq.backend.cython.socket._send_copy
  File "zmq/backend/cython/checkrc.pxd", line 13, in zmq.backend.cython.checkrc._check_rc
KeyboardInterrupt

In [88]: for x in sherlock:
...:     print(x, end=' ')
...:
```

```
paperwork and many fees to meet and keep up with these requirements we do not solicit donations in locations where we have not received written confirmation of compliance To SEND DONATIONS or determine the status of compliance for any particular state visit www.gutenberg.org/donate While we cannot and do not solicit contributions from states where we have not met the solicitation requirements we know of no prohibition against accepting unsolicited donations from donors in such states who approach us with offers to donate International donations are gratefully accepted but we cannot make any statements concerning tax treatment of donations received from outside the United States U S laws alone swamp our small staff Please check the Project Gutenberg Web pages for current donation methods and addresses Donations are accepted in a number of other ways including checks online payments and credit card donations To donate please visit www.gutenberg.org/donate Section General Information About Project Gutenberg-tm electronic works Professor Michael S Hart was the originator of the Project Gutenberg-tm concept of a library of electronic works that could be freely shared with anyone For forty years he produced and distributed Project Gutenberg-tm eBooks with only a loose network of volunteer support Project Gutenberg-tm eBooks are often created from several printed editions all of which are confirmed as not protected by copyright in the U S unless a copyright notice is included Thus we do not necessarily keep eBooks in compliance with any particular paper edition Most people start at our Web site which has the main PG search facility www.gutenberg.org This Web site includes information about Project Gutenberg-tm including how to make donations to the Project Gutenberg Literary Archive Foundation how to help produce our new eBooks and how to subscribe to our email newsletter to hear about new eBooks

In [89]: d={}

In [90]: max=0

In [91]: ans=''

In [92]: for x in sherlock:
...:     d[x]=0
...:

In [93]: print(d)
```

```
'RIGHT': 0, 'REPLACEMENT': 0, 'REFUND': 0, 'elect': 0, 'lieu': 0, 'electronically': 0, 'opportunities': 0, 'AS': 0, 'OTHER': 0, 'WARRANTIES': 0, 'KIND': 0, 'EXPRESS': 0, 'IMPLIED': 0, 'INCLUDING': 0, 'BUT': 0, 'MERCHANTABILITY': 0, 'FITNESS': 0, 'PURPOSE': 0, 'disclaimers': 0, 'implied': 0, 'warranties': 0, 'limitation': 0, 'types': 0, 'disclaimer': 0, 'violates': 0, 'interpreted': 0, 'maximum': 0, 'permitted': 0, 'invalidity': 0, 'unenforceability': 0, 'void': 0, 'provisions': 0, 'INDEMNITY': 0, 'indemnify': 0, 'employee': 0, 'production': 0, 'promotion': 0, 'harmless': 0, 'b': 0, 'alteration': 0, 'modification': 0, 'additions': 0, 'deletions': 0, 'Defect': 0, 'Mission': 0, 'synonymous': 0, 'readable': 0, 'widest': 0, 'computers': 0, 'obsolete': 0, 'exists': 0, 'Volunteers': 0, 'financial': 0, 'critical': 0, 'goals': 0, 'ensuring': 0, 'Sections': 0, 'non': 0, 'profit': 0, 'educational': 0, 'corporation': 0, 'organized': 0, 'Mississippi': 0, 'exempt': 0, 'Internal': 0, 'Revenue': 0, 'Service': 0, 'EIN': 0, 'federal': 0, 'identification': 0, 'Contributions': 0, 'deductible': 0, 'Fairbanks': 0, 'Alaska': 0, 'mailing': 0, 'PO': 0, 'Box': 0, 'AK': 0, 'locations': 0, 'Salt': 0, 'Lake': 0, 'UT': 0, 'Email': 0, 'Gregory': 0, 'Newby': 0, 'Chief': 0, 'Executive': 0, 'Director': 0, 'gnewby': 0, 'pglaf': 0, 'Donations': 0, 'survive': 0, 'licensed': 0, 'accessible': 0, 'outdated': 0, 'maintaining': 0, 'IRS': 0, 'regulating': 0, 'charities': 0, 'charitable': 0, 'Compliance': 0, 'paperwork': 0, 'solicit': 0, 'confirmation': 0, 'SEND': 0, 'DONATIONS': 0, 'contributions': 0, 'solicitation': 0, 'prohibition': 0, 'accepting': 0, 'unsolicited': 0, 'donors': 0, 'International': 0, 'gratefully': 0, 'Web': 0, 'pages': 0, 'donation': 0, 'addresses': 0, 'checks': 0, 'Professor': 0, 'Michael': 0, 'Hart': 0, 'originator': 0, 'network': 0, 'Thus': 0, 'necessarily': 0, 'edition': 0, 'PG': 0, 'includes': 0, 'subscribe': 0, 'email': 0, 'newsletter': 0}

In [94]: for x in sherlock:
...:     d[x]=d[x]+1
...:     if d[x]>max:
...:         max=d[x]
...:     ans=x
...:

In [95]: print(max)
5431

In [96]: print(d)
```

```
'viewing': 1, 'royalty': 1, 'profits': 1, 'derive': 1, 'calculated': 1, 'calculate': 1, 'applicable': 3, 'taxes': 1,
'owed': 1, 'agreed': 1, 'donate': 5, 'Royalty': 2, 'payments': 3, 'legally': 1, 'periodic': 1, 'information': 5,
'donations': 12, 'notifies': 1, 'require': 1, 'physical': 2, 'medium': 5, 'discontinue': 1, 'accordance': 2,
'replacement': 3, 'Trademark': 1, 'LLC': 1, 'Contact': 1, 'volunteers': 5, 'employees': 2, 'expend': 1, 'transcribe': 1,
'proofread': 1, 'Despite': 1, 'Defects': 1, 'inaccurate': 1, 'corrupt': 1, 'transcription': 1, 'errors': 1,
'infringement': 1, 'defective': 2, 'damaged': 1, 'disk': 1, 'computer': 2, 'virus': 1, 'codes': 1, 'damage': 1,
'equipment': 3, 'LIMITED': 3, 'WARRANTY': 2, 'DISCLAIMER': 1, 'DAMAGES': 2, 'Right': 1, 'Replacement': 1, 'Refund': 1,
'disclaim': 1, 'liability': 2, 'damages': 2, 'costs': 2, 'AGREE': 2, 'THAT': 2, 'HAVE': 1, 'NO': 2, 'REMEDIES': 1,
'FOR': 3, 'NEGLIGENCE': 1, 'STRICT': 1, 'LIABILITY': 1, 'BREACH': 2, 'CONTRACT': 1, 'EXCEPT': 1, 'THOSE': 1, 'PROVIDED':
1, 'PARAGRAPH': 1, 'FOUNDATION': 1, 'TRADEMARK': 1, 'OWNER': 1, 'AND': 1, 'ANY': 3, 'DISTRIBUTOR': 1, 'UNDER': 1,
'AGREEMENT': 1, 'WILL': 1, 'NOT': 2, 'BE': 1, 'LIABLE': 1, 'ACTUAL': 1, 'DIRECT': 1, 'INDIRECT': 1, 'CONSEQUENTIAL': 1,
'PUNITIVE': 1, 'INCIDENTAL': 1, 'EVEN': 1, 'IF': 1, 'GIVE': 1, 'NOTICE': 1, 'POSSIBILITY': 1, 'SUCH': 1, 'DAMAGE': 1,
'RIGHT': 1, 'REPLACEMENT': 1, 'REFUND': 1, 'elect': 1, 'lieu': 2, 'electronically': 2, 'opportunities': 1, 'AS': 1,
'OTHER': 1, 'WARRANTIES': 2, 'KIND': 1, 'EXPRESS': 1, 'IMPLIED': 1, 'INCLUDING': 1, 'BUT': 1, 'MERCHANTABILITY': 1,
'FITNESS': 1, 'PURPOSE': 1, 'disclaimers': 1, 'implied': 1, 'warranties': 1, 'limitation': 3, 'types': 1, 'disclaimer':
2, 'violates': 1, 'interpreted': 1, 'maximum': 1, 'permitted': 2, 'invalidity': 1, 'unenforceability': 1, 'void': 1,
'provisions': 1, 'INDEMNITY': 1, 'indemnify': 1, 'employee': 1, 'production': 1, 'promotion': 1, 'harmless': 1, 'b': 1,
'alteration': 1, 'modification': 1, 'additions': 1, 'deletions': 1, 'Defect': 1, 'Mission': 1, 'synonymous': 1,
'readable': 2, 'widest': 2, 'computers': 2, 'obsolete': 1, 'exists': 1, 'Volunteers': 1, 'financial': 1, 'critical': 1,
'goals': 1, 'ensuring': 1, 'Sections': 1, 'non': 1, 'profit': 1, 'educational': 1, 'corporation': 1, 'organized': 1,
'Mississippi': 1, 'exempt': 2, 'Internal': 1, 'Revenue': 1, 'Service': 1, 'EIN': 1, 'federal': 2, 'identification': 1,
'Contributions': 1, 'deductible': 1, 'Fairbanks': 2, 'Alaska': 1, 'mailing': 1, 'PO': 1, 'Box': 1, 'AK': 1, 'locations':
2, 'Salt': 1, 'Lake': 1, 'UT': 1, 'Email': 1, 'Gregory': 1, 'Newby': 1, 'Chief': 1, 'Executive': 1, 'Director': 1,
'gnewby': 1, 'pqlaf': 1, 'Donations': 2, 'survive': 1, 'licensed': 1, 'accessible': 1, 'outdated': 1, 'maintaining': 1,
'IRS': 1, 'regulating': 1, 'charities': 1, 'charitable': 1, 'Compliance': 1, 'paperwork': 1, 'solicit': 2,
'confirmation': 1, 'SEND': 1, 'DONATIONS': 1, 'contributions': 1, 'solicitation': 1, 'prohibition': 1, 'accepting': 1,
'unolicited': 1, 'donors': 1, 'International': 1, 'gratefully': 1, 'Web': 3, 'pages': 1, 'donation': 1, 'addresses': 1,
'checks': 1, 'Professor': 1, 'Michael': 1, 'Hart': 1, 'originator': 1, 'network': 1, 'Thus': 1, 'necessarily': 1,
'edition': 1, 'PG': 1, 'includes': 1, 'subscribe': 1, 'email': 1, 'newsletter': 1}

In [97]:
```

So, this is easy ones, twice, thrice. It has come thrice. But it was Monday morning Swaminathan was looking to open his eyes once he considered Monday specially unpleasant after delicious etcetera. It was Monday mood of work, he shattered the very thought of school that may be the words used here and the words that I have used are probably different. So, let me just cross check that print Malgudi.

It was Monday morning Swaminathan was reluctant to open his eyes. He considered Monday specially unpleasant in the calendar after the delicious freedom of Saturday and Sunday. It was difficult to get into the Monday mood of work and discipline. He shuddered at the very thought of school, the dismal-yellow building, the fire-eyed Vedanayagam, his class teacher, there is thrice. In fact, there is his here, his here and his here. It should be 3, I wonder why there is 2 here. Probably it was a mistake here and not with our code. You see that is the advantage of writing a piece of code.

Of course, in making this PowerPoint, it was not easy manually for the person who did this PowerPoint to not have seen this error. Anyway, never mind. We all make mistakes. I make so many mistakes when I type my piece of code, never mind. So, our code is perfect. We observe that his indeed appears thrice. And as you observe this very thing we created here which is here. And now you may want to tell me, what is the most frequently occurring word here? It is the.

But then how do you find that out? It is not easy to find that out. How will you find that out? Let us write the same code that computational thinking course taught you. Whatever was the pseudo

code that was taught in this course, I am going to try writing the same old program here and try to figure out whether I can find out the max frequently occurring word. Is it, are you finding it complicated or you finding it not so easy on your mind? Do not worry at all.

The best way to learn is to restart the video. See it once again. I think this particular video you want to watch it thrice or even 4 times for you to make sense if you have not understood dictionary before. If it is the first time you are observing what is a dictionary, you may want to see this video couple of times more. Certainly, one pass you will not understand anything.

So, where were we? We need to figure out what is the max frequently occurring word. So, how do we do that? So, what I will do is I will say max equals 0. In fact, you can write a piece of code on the editor too, but I will do it on the console like this, max equals 0 for x in malgudi. So, you should probably redo that dictionary. Dictionary, I should start from the beginning. Why, because I have put all the values here. Now, I will redo the entire code from the beginning so that you understand what is what.

So, d equals so much, for x in s, every single word in this malgudi, where I have removed the repetitions and created a set instead. You see the application of list and a set. If this was too bigger list, going through every single word and even the repetitions does not make sense, you see, because you need to include the word only once. I initialize my dictionary. So, as you can see, every single word comes and it says 0, 0, 0, 0, because I have not gone through every single word and counted them how many times they occur.

Now, I am going to do that. Look at this. I will go slowly here. Max is 0, dictionary initiated, dictionary values are appended to the dictionary and I count the first one will be the word, second one will be the number of occurrence of that word. Now, I must find which word is most frequently occurring.

So, what I will do is for x in malgudi, I will go through every single word here. So, if d of x equals d of x plus 1, so I increment the occurrence of that word. For instance, I see it here, as I explained earlier, so it will be one occurrence or two occurrence or three. It will get incremented by one here. It is case sensitive. This, it is different from this set. So, I am assuming that. So, small letter it is different from capital letter it. So, it will become one right now. And then was will become 1. When was is encountered twice, it becomes twice and so on and so forth.

So, then what I will do is, I will say here, watch carefully, if d of x , whatever you saw here was greater than max , then what you do is max equals d of x . Why would you do that? That is because, if your d of x is greater than the maximum number seen so far, then you declare that as the maximum value, that is all. Stare at this for a minute, write this down on a sheet of paper and observe what we are doing.

In plain English all I am saying is go through all values in the, in this list `malgudi` one at a time from left to right, slowly, it was Monday morning, each word you go through. For every word what you do is you increment the dictionary by 1, d of x will be 1. If it is 0 that value becomes 1. It becomes 1 if it encountered once. If it is encountered twice, it becomes 2 and so on and so forth here.

As you are loop in through this for loop, it also checks if any of this is greater than the max known so far. That is what is being taught here. If you remember, in single iteration, in this iteration itself, I am finishing it, you will, this will not make sense to you if you have not watched this video. So, watch this video first please. So, and then I do this, and I say enter, and I come out of this, and boom, print d , you will get the updated values, and then print. What am I going to type now? Max , you will get the max value two, which is 6 here.

You may wondering, sir, why is it saying 6? I also want the exact word too. No, that is not what we are doing. We only want to display the max element. That is what even this lecture talks about. But let us go one step ahead and try to figure out if we can actually display the word as well. So, how do we do that? Let me start from beginning once again.

Let me initialize d . What did I do here? I initialize d and I initialize the values to 0. So, for x in s , d of x is equal to 0. And after that, I, my max was initialized to 0 as well, and then this for loop happened. So, what I am going to now do is the answer word I am going to initialize. This will have the answer right now. So, I would say for x in `malgudi`, let me scroll up.

This code I am going to rewrite it, d of x is equal to d of x plus 1 if d of x is greater than max then max equals d of x and your answer word is equal to the value x . I started this. And every time max gets changed, answer word is also getting changed, plain English. Think about it. This looks very complicated. This looks like kind of a language that you have never encountered so far. But bear with me, things will get easier as you keep practicing it. So, I say enter here. Then I

will print d, good, updated. I will print x, I am sorry, I will print max. I will get 6. What is the word? Print answer word, the word is 'the'.

Now, again, pause here if you have not understood. But if you have understood, let me actually show you the magic here. What I have done? I have taken not just this Malgudi file here with all these words, I have taken the whole of Sherlock Holmes as a list. Did you observe that? The moment I said Sherlock, this came. Do not try to type it on your terminal. You will not get it. We wrote a small script to gather all the words and put them as a list here. If you want, I can send you this list of words. Maybe I will post it on the forum. Or you can do it all by yourself. You can take a big list of words of your choice.

So, Sherlock Holmes, the entire work is available by Arthur Conan Doyle. You can take it from there. It is a plain text file and then create this list. What is the length of Sherlock? It is close to 100,000. That is why I was referring to 100,000 in my, the beginning of this lecture. So, it has 100,000 words, a little more than 100,000 words. And how big was your malgudi, around 100, I mean, 66, very, very small. For Sherlock, what I will do is I will run the same code for Sherlock. What is the code? I cannot scroll up like this. So, I will try to figure out what is the max frequently occurring words in Sherlock.

So, you know to print. Let me show you something. For x in Sherlock, all words in Sherlock, I will say print x. I think if you put a comma here, it will show one word next to the other. It is not showing. I think I should say end equals this, yes. So, it is showing all the words one next to the other. Maybe you can try to show it by, this is the problem of taking a big file. If you want to stop it, it will take a few seconds. If you want to abort it, it will take a few minutes.

So, if you put a space here, I think it will show you one word. It is the perfect. So, this is the entire novel. Let us go through it. Read it. You probably should be a robot to read it at this pace. Anyways, it goes on and on. It is a very, very, very big novel. It did not, you did not see the entire novel actually. To go through all the 100,000 words, it actually if you even spend a microsecond per word, it is going to take a long time to finish it. Microsecond a word, microsecond is 10 to the power of minus 6, it probably, it is of that order.

So, it will probably take a second if you can read one word in a microsecond. But of course, you can only read 3-4 words in a second, which means reading this entire thing will take a few days,

maybe even close to a month. I do not know. My estimation could be wrong. You can estimate it, but it is a long, long time. So, you have Sherlock as a list. I displayed it. Now, let me go through the whole of the Sherlock document just the way I did for malgudi and try to figure out what is the most frequently occurring word here. Same old process. What was that?

Let me remember the code, `d` equals open bracket, close bracket. Obviously, my computer will be a little slow here, because I have put a lot of stuff in my memory. It will cry a little, do not mind. And then what I should do is again, initialize `max` to 0 and then answer a word, whatever the previous thing I should initialize that. This is not required actually. You can directly assign it there if you know what I mean. If you do not know what I mean, do not break your head.

So, what I should do now is for `x` in Sherlock, `d` of `x` is equal to 0. So, what am I doing here? I am initializing the dictionary first. Let me do this. So, print `d`, here is all the words in Sherlock novel. And you see I am getting a little tensed. That is because it is a huge file. I mean, although the code will be exactly the same, I am a little tensed, whether it will really work for this bigger code. So, you observed that I have initialized my dictionary.

So, what should I do next? I have initialized `max` is it. I have initialized `max` already. So, what I should do is for `x` in `d`, what was that, for `x` in the, I am sorry, for `x` in Sherlock, for all the values in this list Sherlock what you do is look at the word `x` in Sherlock, as you go through all words of Sherlock, and then increment it by 1, thus telling me the total number of occurrences of individual words. You see Michael here, how many times did it occur? The word computers, how many times it occur? So, then same old code. Why am I struggling? I mean, it is the same code.

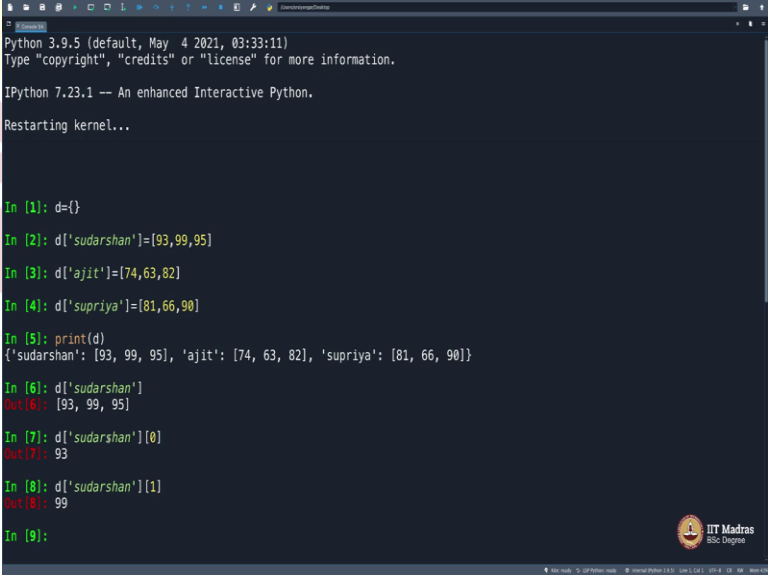
So, `d` of `x` equals `d` of `x` plus 1. And then what I should do is, `max`. So, if `d` of `x` is greater than `max`, then your `max` will be `d` of `x`. And your answer word, what does that answer, I guess, answer will actually be equal to your `x`. Enter, another enter, this will now boom, over all 100,000 words, it saw in a fraction of a second. That is the advantage of Python. It is very, very fast when it comes to word processing. In fact, you can take big files, the whole of Twitter data if you take and try to do similar operation on it, it does it really fast. It is optimized to do it. I can say all these things provided the answer is right. Let me see. What is print `max`?

Some word is repeated 5431 times. What is that word? Let me print the dictionary. See, all these words are repeated once, twice, thrice, seven times, but some word is repeated, one word is repeated many times. What is that word? Let me see. How do we see that? The word `ans` would have captured it. It is the word, 'the'. So, it seems to be the most frequently occurring word in English, I believe, because when we took this small passage, it was the most frequently occurring word here in your previous course. In this course, we tried doing the same operation, same piece of code on a huge file. Still it seems to be the most frequently occurring word. With this, I conclude the idea of a dictionary.

I just now introduced, what is a dictionary, what is a list. This should be, and what is a set, this should be enough to begin with. There is a small concept called a tuple, which I will explain next in a short video. But apart from that, the idea of a list, a set and a dictionary is a very important one in Python. You will use lists a lot. You will use dictionaries a lot. We just saw the power of dictionaries. Whatever we did was a good application of dictionary. In fact, most of the graphs that you learnt in computational thinking can be achieved using dictionary.

I will show you a small illustration of what you can do with a dictionary. Let me clear the screen. Let me also clear the cache, so that everything goes away so that my terminal is faster than the noise. So, what shall we do?

(Refer Slide Time: 40:54)



```
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]: d={}
In [2]: d['sudarshan']=[93,99,95]
In [3]: d['ajit']=[74,63,82]
In [4]: d['supriya']=[81,66,90]
In [5]: print(d)
{'sudarshan': [93, 99, 95], 'ajit': [74, 63, 82], 'supriya': [81, 66, 90]}
In [6]: d['sudarshan']
Out[6]: [93, 99, 95]
In [7]: d['sudarshan'][0]
Out[7]: 93
In [8]: d['sudarshan'][1]
Out[8]: 99
In [9]:
```

```
In [7]: d['sudarshan'][0]
Out[7]: 93

In [8]: d['sudarshan'][1]
Out[8]: 99

In [9]: d['sudarshan'][2]
Out[9]: 95

In [10]: d={}

In [11]: d['sudarshan']=[93,99,95,'sudarshan@iitrpr.ac.in']
In [12]: d['ajit']=[74,63,82,'ajit.rao1234@gmail.com']
In [13]: d['supriya']=[81,66,90,'supriyahs78921@gmail.com']
In [14]: d['supriya'][1]
....:
....:
Out[14]: 66

In [15]: d['supriya'][3]
Out[15]: 'supriyahs78921@gmail.com'

In [16]:
```

Let me create a new dictionary, d. And then let us say d of Sudarshan is equal to my marks in physics, chemistry and mathematics, 93, mathematics was 99, and chemistry was 95. So, at least in virtual world, I mean, unreal world let me bloat on my marks, although this is not the marks that I ever secured. So, Sudarshan has secured so much. And let us say Ajit has secured, if you capitalize this, you should retain the capitalization throughout.

I am not capitalizing it for the sake of simplicity. He has secured 74, 63 and 82. Obviously, everyone else should have secured less than me, because I want to be the first ranker in the class. So, d, let us says Supriya has secured 81, 66, 80, let us say, 90. Now, when you display d, it will show you like this. And if you want to know what is Sudarshan's marks, so this is physics, chemistry, mathematics, PCM. I think I said PMC. But anyways let us retain this as, let us call it, physics, chemistry and mathematics.

Sudarshan's marks will be displayed like this. But then d of this is actually a list you see. This is a list. So, we can simply say d of Sudarshan of 0 that will be my physics marks of 1 that will be my chemistry marks, of 2 that will be my mathematics marks. So, you can go through these values by simply calling d of Sudarshan. In fact, in the previous case where we saw the Sherlock example, this was simply the number of occurrences of the word so much.

But right now I am saying this is the marks secured by Sudarshan. You can make it one level more complicated. Let me show you. And then include more details, you see. You can always say d of Sudarshan is equal to, it is of course the marks 93, 99 and 95 and the last entry will be

his email address in codes because it is a string, email address. And you can do that for the others two, let us say Ajit. Ajit's email address, let us say Ajit Rao at some gmail dot com etcetera etcetera. some number and gmail dot com.

Of course, this is not a gmail. And I can say Supriya is equal to her marks, 81, 66, 90 comma Supriya, let us say, Supriya HS 7891221 gmail dot com is her email address. Now, when I say d of Supriya of 1, I should get 66. I will get 66 here. This should give me 66. Yes, this is giving me 66. And if I said 3, 0, 1, 2, 3 should be her email address. You see, it should be her email address and it does precisely that. And you can put a big list here as big as you want.

You can say d of Sherlock and put all the words in Sherlock and put d of RK Narayan and put all the words in his novel, but word, the sky is the limit. You can go on and on. In fact, d of this can be another dictionary here itself, enter another dictionary by using a flower bracket here. Sounds complicated, do not break your head. All that you need to know right now, again, I repeat, summary. The story so far is you know what is a list, you know what is a set. You have seen an application of dictionary with this example. If you know this much that is more than enough to begin with. The rest comes next.

You must learn the bare minimum entities and go one top up at a time in understanding what is what. Unless you encounter a situation where you require a, where there is this dyer requirement of dictionaries or whatever we have taught, until then you will not understand what is what. So what this video a couple of times. It is not easy on the, for the first time code a along as I am coding and you will be able to understand the code completely. There is nothing complicated in what we did so far.

So, now, with this, I am ending my part of this week's lectures. On to Mr. Omkar who will try to write more code, in fact, show you some nice aspects of dictionary quite a follow up of your computational thinking class and also more to do with tuples I will explain it to you in the next video in a couple of minutes. But otherwise we are done for this week from my side. On to Omkar.