Business Analytics and Text Mining Modeling Using Python

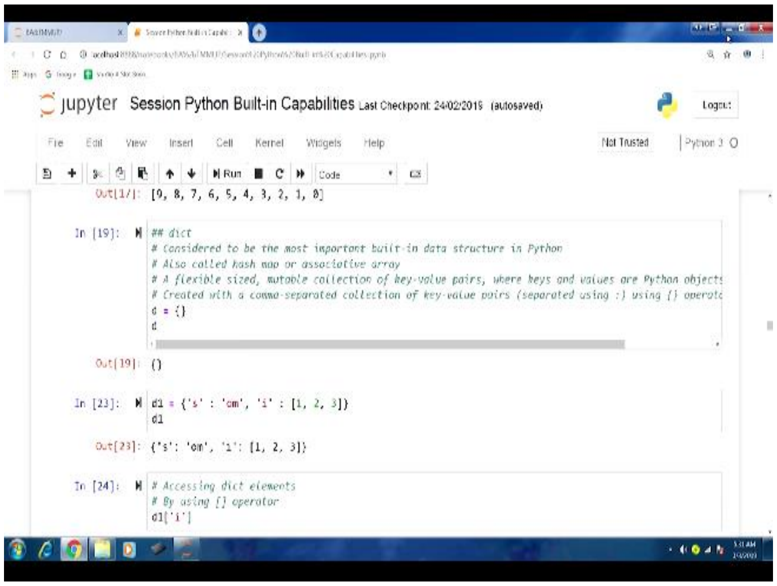
Prof. Gaurav Dixit
Department of Management Studies
Indian Institute of Technology-Roorkee

Lecture-13
Built-in Capabilities of Python-V

Welcome to the course business analytics and text mining modeling using python. So, in previous few lectures we have been talking about some of the built in you know data structures in python. And we have discussed tuples, list, and other things. So, and then in the previous lectures we started on dict as well, so what we will do, we will quickly go through you know, some of that part of that is related to dict.

And then we will pick up from the point you know, where we stopped in the previous lecture. So, we talked about dict consider to be the most important building data section in Python.

(Refer Slide Time: 01:02)



```
Out[17]: [9, 8, 7, 6, 5, 4, 3, 2, 1, 0]

In [19]: ## dict
         # Considered to be the most important built-in data structure in Python
         # Also called hash map or associative array
         # A flexible sized, mutable collection of key-value pairs, where keys and values are Python objects
         # Created with a comma-separated collection of key-value pairs (separated using :) using {} operato
         d = {}
         d

Out[19]: {}

In [23]: d1 = {'s' : 'om', 'i' : [1, 2, 3]}
         d1

Out[23]: {'s': 'om', 'i': [1, 2, 3]}

In [24]: # Accessing dict elements
         # By using [] operator
         d1['i']
```

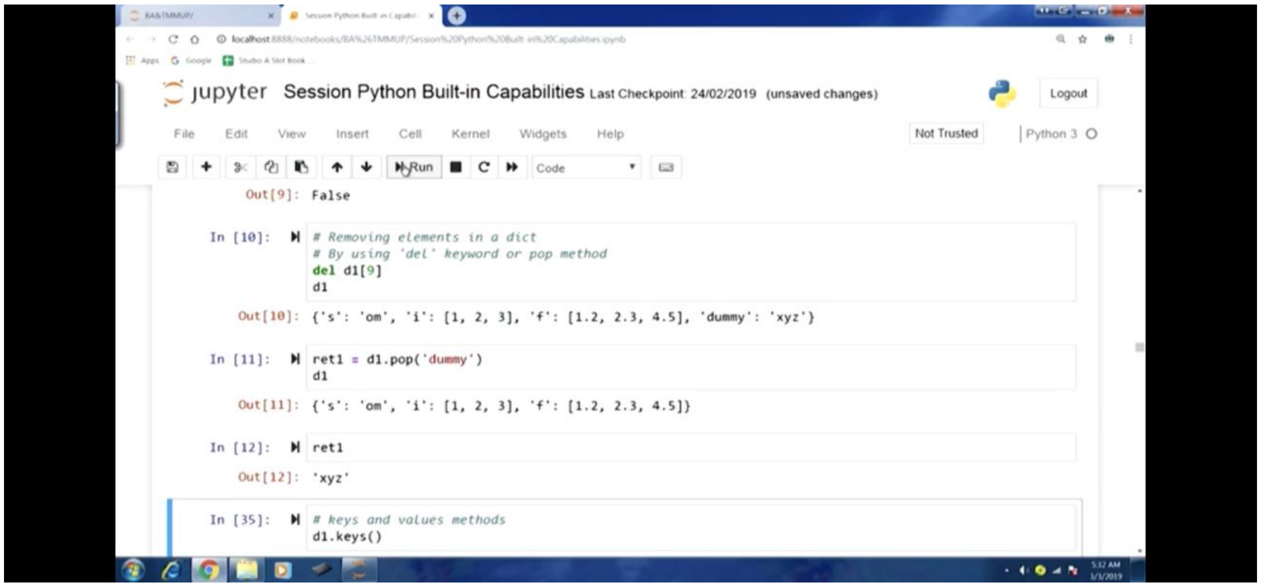
Also call has map or associative array and flexible in a mutable collection of key value pairs. So, the aspects we talked about, and how we can create them using the; you know curly braces, and separated between key value pair is the column. So this aspect we have talked about.

(Video Starts: 01:25)

So, I will quickly go through so that you know, we are able to you know, leave some of the things that we have discussed in the previous lecture. So, I will quickly go through some of the things that we discussed we just few ways to actually create you know, dicts objects, and accessing dict elements and then adding elements in dict. So, all these things, we talked about, modifying elements of dict, checking whether you know, a dict contains a key particular you know, a specific tree that we might be interested in.

So, all these aspect we talked about removing elements in a dict. So, this we have discussed, I know pop method to again remove. So, all this we have this discussed keys and values method we have discussed in the previous lecture. So, we will just go through some of these you know lines of code, we talked about merging 2 dicts. So, here again, you can see the example and we used update method dict. And you can see some of these output, we talked about pairing up 2sequences element wise to create a dict.

(Refer Slide Time: 02:22)


jupyter Session Python Built-in Capabilities Last Checkpoint: 24/02/2019 (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code

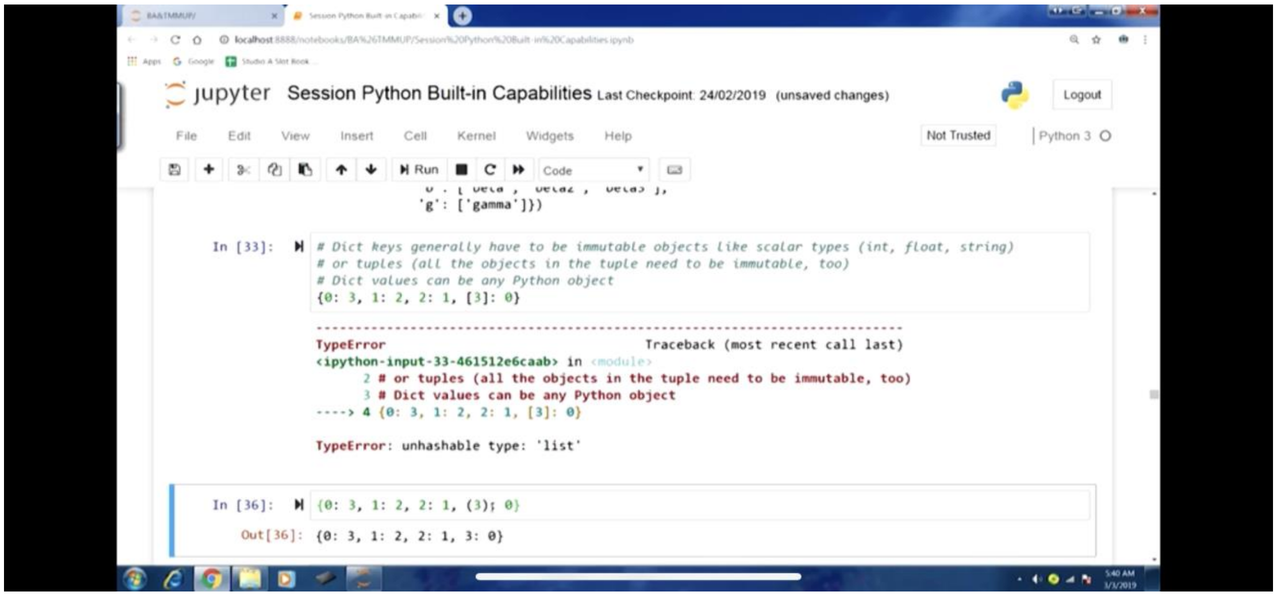
```
Out[9]: False

In [10]: # Removing elements in a dict
         # By using 'del' keyword or pop method
         del d1[9]
         d1
Out[10]: {'s': 'om', 'i': [1, 2, 3], 'f': [1.2, 2.3, 4.5], 'dummy': 'xyz'}

In [11]: ret1 = d1.pop('dummy')
         d1
Out[11]: {'s': 'om', 'i': [1, 2, 3], 'f': [1.2, 2.3, 4.5]}

In [12]: ret1
Out[12]: 'xyz'

In [35]: # keys and values methods
         d1.keys()
```

So, this particular aspect also we discussed, then using dict function to create a dict object. So, this also something that we discussed. Now, I think in the last part of it, we were discussing useful methods get a pop. So, here you can see d1.get and passing on the key value to actually, you know, to access the value part of it, and we talked about these aspects, and we talked about typically by default, none is return if that value is that particular key value, key is not part of a dict, so, we can change that default dict value.

So, the aspect we discussed, similarly for the pop method also, if a particular you know, key is not part of the dict object and again, we can change the default rate and value here, so, you can see. Now we will discuss some useful function here. So, one of them is set default. So, again this function is quite useful in the context when we are trying to set values while generating a dict. So, let us take an example, suppose, we want to categorize a list of terms by their first letters.

And we want to create a dict of list. So, where the key part of the object is going to be based on the first letter and the terms that you know, use that first letter, they would be you know, in the form of a list. So, in this fashion if we want to create the set default method can be really useful. So, the same thing can actually be achieved using you know for loop and you know, if else block along with that.

(Refer Slide Time: 05:40)


jupyter Session Python Built-in Capabilities Last Checkpoint: 24/02/2019 (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code

```
          'g': ['gamma']})

In [33]:  # Dict keys generally have to be immutable objects like scalar types (int, float, string)
          # or tuples (all the objects in the tuple need to be immutable, too)
          # Dict values can be any Python object
          {0: 3, 1: 2, 2: 1, [3]: 0}

          ---------------------------------------------------------------------------
          TypeError                                 Traceback (most recent call last)
          <ipython-input-33-461512e6caab> in <module>
                2 # or tuples (all the objects in the tuple need to be immutable, too)
                3 # Dict values can be any Python object
          ----> 4 {0: 3, 1: 2, 2: 1, [3]: 0}

          TypeError: unhashable type: 'list'

In [36]:  {0: 3, 1: 2, 2: 1, (3); 0}

Out[36]:  {0: 3, 1: 2, 2: 1, 3: 0}
```

However you would realize that set default method, you know, you will have to if you use this particular you know, method you will have to use, you know, few lines of code and you will be able to optimize your code, readability and other things, any other benefits we will have. So, the string that we have is terms so, let us say we have these terms alpha beta gamma alpha 2 beta2 beta3.

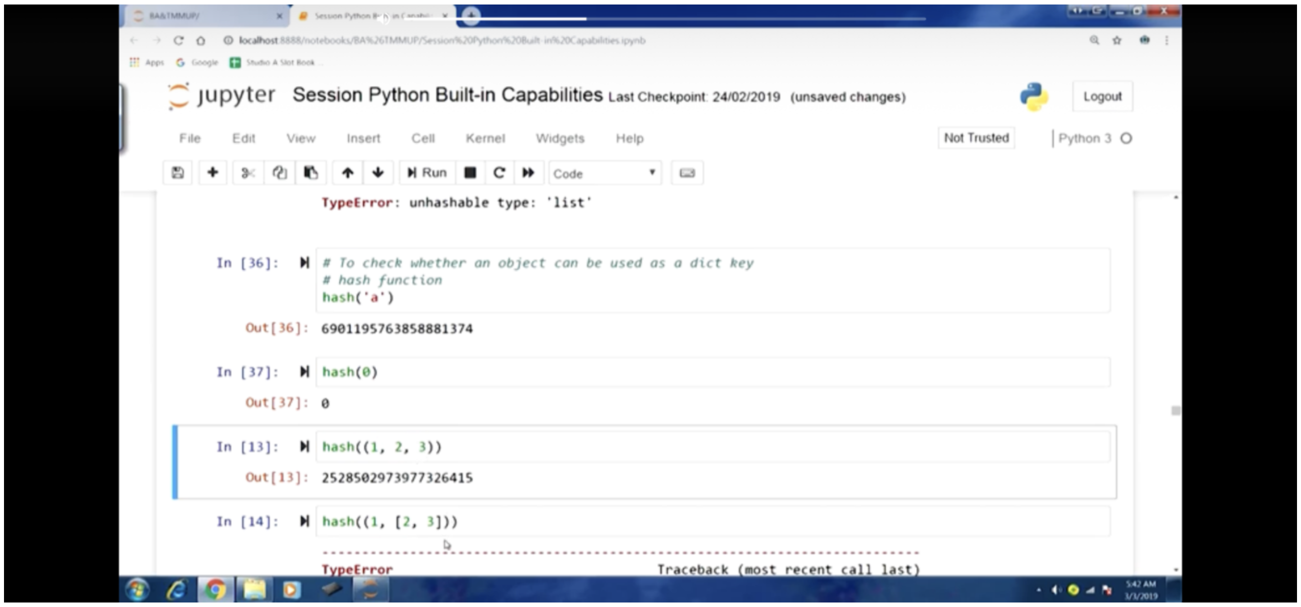
And if we want to create a dict object based on this particular, you know, list of strings. So, what we do first we can create empty dict here empty dict object, and then in the loop, what we are doing is, for every term that is part of this list of a string that means, for us every string, we are running this loop and in that, you know, we are calling this set default method, and we are picking up this first character of each of these strings.

And then we are attending that we are appending the you now we are appending the terms along with that. So, the terms would be part of the list the value part of the you know, key value pair in the dict. And the first character is going to be part of first letter is going to be key part of key value pair. So, let us run this and you can see in the output, we have a and then alpha alpha2 and similarly for the next one, b beta beta2 beta3.

So in this fashion set default can be really useful in many scenarios that typically we encounter in data analytics area. Now, there is another mechanism to achieve the same thing that is default dict class, you know, that we have from collections module. So, this can also be used to set values while generating a dict. So here again, we are importing this from collections, this module, we are importing this particular class default dict.

And you can see first letter, initializing is using this particular function default dict and in the parenthesis we are passing this list, and we are running a loop here again. And for every string that is part of that list of string, we are again, extracting the first character and then appending the elements and creating in this fashion. So if I run this, again, you will see that similar kind of output we are able to generate.

(Refer Slide Time: 09:00)


jupyter Session Python Built-in Capabilities Last Checkpoint: 24/02/2019 (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code

```
TypeError: unhashable type: 'list'

In [36]:  # To check whether an object can be used as a dict key
          # hash function
          hash('a')
Out[36]:  6901195763858881374

In [37]:  hash(0)
Out[37]:  0

In [13]:  hash((1, 2, 3))
Out[13]:  2528502973977326415

In [14]:  hash((1, [2, 3]))
-----------------------------------------------------------------------------
TypeError                              Traceback (most recent call last)
```

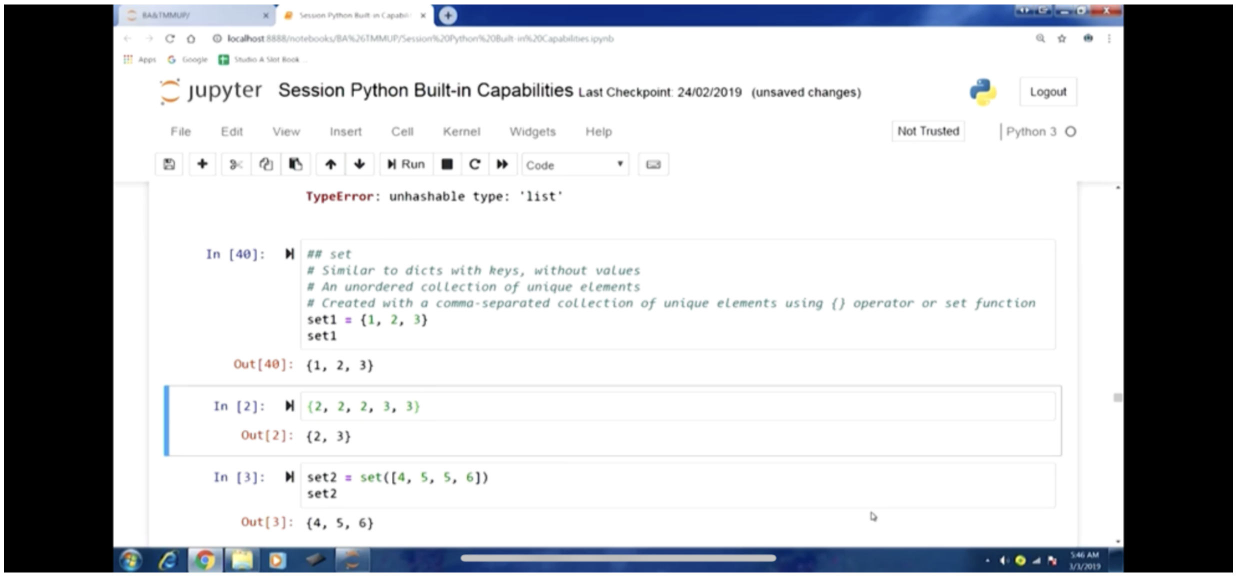
Now let us move on few more things about dict objects. So if we focus let us talk about first the keys parts keys are generally have to be immutable objects, just like a scalar type. So it could be in ford string, or tuples. So we talked about mutable and immutable objects. So these are the examples of some of the immutable objects. So the keys part of the key value pair, it is supposed to be the immutable object.

And so all the objects are here, you know, when we are using tuple all the objects that are part of that tuple they also need to be immutable. So we will check, you know some of these things through examples as well. Now, as far as the dict the value part is concerned in the dict objects that can be any python object. So let us take an example here. So here, you can see, we have 0 3, so 0 can be a key.

And then we have 1:2, 1 can be a key scalar type, so 2:1, it is ok. Now in the fourth element that we have in this example, we have 3 as the single element in a list. So this is immutable object. So this is not supposed to be a key in the dict object. So if I run this, I will get an type error. So you can see, you can see unhashable type list. So a list object cannot be a key of a dict object.

Similarly, I can have another example. So in this example, you can see, first element is fine, fist pair is fine, second pair is fine, third pair is fine. Now in the fourth pair, you can see I am using tuple with a single element. So this is also fine because the element within this tuple is immutability scalar type. So if I run this, no problem, and you can see output from the 34. And dict object is created.

(Refer Slide Time: 11:37)


jupyter Session Python Built-in Capabilities Last Checkpoint: 24/02/2019 (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code

```
TypeError: unhashable type: 'list'

In [40]:  ## set
          # Similar to dicts with keys, without values
          # An unordered collection of unique elements
          # Created with a comma-separated collection of unique elements using {} operator or set function
          set1 = {1, 2, 3}
          set1
Out[40]:  {1, 2, 3}

In [2]:   {2, 2, 2, 3, 3}
Out[2]:   {2, 3}

In [3]:   set2 = set([4, 5, 5, 6])
          set2
Out[3]:   {4, 5, 6}
```

Now, let us take another example. So if in this example, you notice, first 3 elements, first 3 pairs are fine. Now on the fourth pair the key part, I am using a tuple, so tuple is allowed to be a key, but the only element that is there in the tuple is actually a list. So we said that, if you are using tuple as key, even the elements of that tuple are supposed to be immutable. So if I run this part, you can see again, I am getting a type error.

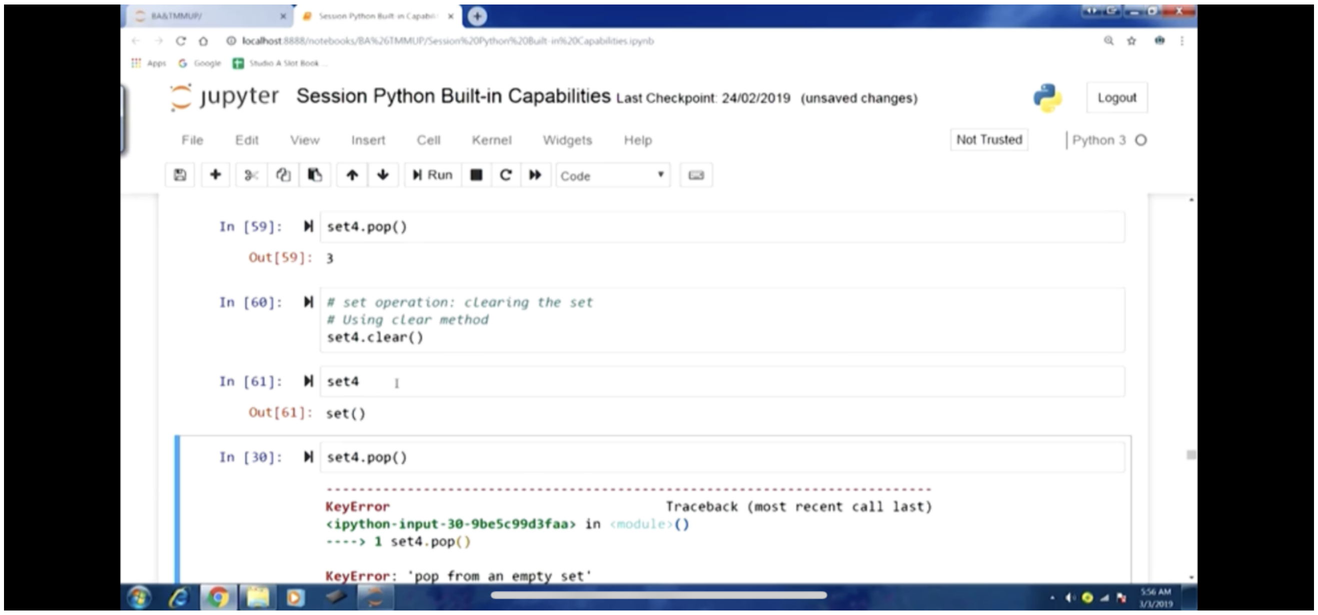
So you can see in the output I am getting unhashable type list here. So in this fashion, you will get the idea, you know, whenever we are creating a dict object, what can be the key, what can be the value part, and how we can work with that. So if in certain situation we are required to, you know, we have some variables in the form of list object, then we can convert them into tuple and then accordingly, we can use them.

It is important for us to understand how different data structures are to be used, and what can work and what cannot work. So those things. Now, if you want to check whether a particular object can be used as a dict key or not, so for that also, we have a hash function here. So let us use this. So if we are using this, you know, a string, the scalar type, and we are passing on into the hash, so we get a output, you know, which looks like a hash value has index, then this particular object can be used as a dict key.

So if I run this hash, and just you know, one element, I am passing scalar type a string value, if I run this, I will get an hash here. Now similarly, if I in the another example I am passing on this element 0 integer type and this will also be allowed, so I will get it as well. Now, if I take another example, now you see, here I am passing on this argument, this variable is a tuple with 3 elements, 1 2 3, all the scalar types.

So this is also okay. So if I run this, you can see hash values written here. So this is also ok. Now in the next example, what I am doing is I am passing a tuple, but the second element of the tuple is a list. So which is the immutable object. So therefore, you know, if I pass this variable as an argument here, I will get an error, you can see type error, and you can see unhashable type placed.

(Refer Slide Time: 14:57)


jupyter Session Python Built-in Capabilities Last Checkpoint: 24/02/2019 (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code

```
In [59]:  set4.pop()
Out[59]:  3

In [60]:  # set operation: clearing the set
          # Using clear method
          set4.clear()

In [61]:  set4
Out[61]:  set()

In [30]:  set4.pop()

          ---------------------------------------------------------------------------
          KeyError                                  Traceback (most recent call last)
          <ipython-input-30-9be5c99d3faa> in <module>()
          ----> 1 set4.pop()

          KeyError: 'pop from an empty set'
```

So, using this function also, we can find out whether a particular object can become a can be used as a dict key. So, certain situation, we might have to perform this kind of check. So this can be really useful this particular function has can be really useful. Now, we will move on to the next data structure in python, so, this is called set. So, this is quite similar to what we understand by said in you know algebra.

So, in terms of similarity, semantics, and similarity, it is quite similar to dicts dict keys and without values. So you can see we have just understood what kind of objects can be a dict keys. So, if we do not have the values part and just have the keys part, so you know, dicts and you know, set would be here, but you know, in a similar fashion. So, in another way, we can define set as an on unordered collection of unique elements.

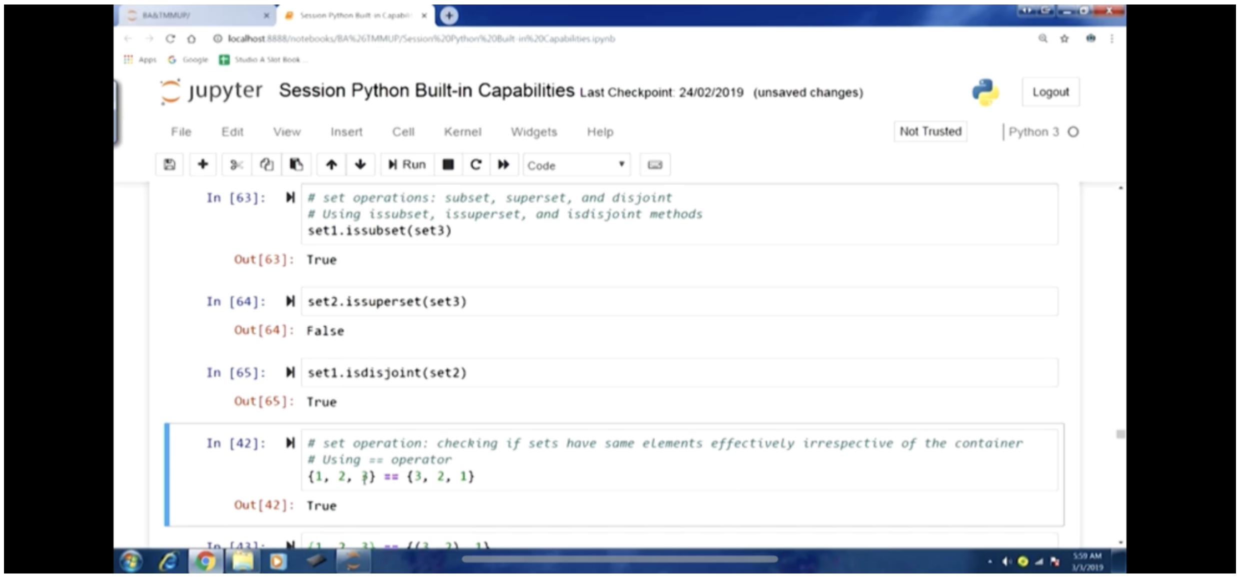
So, all the elements are to be unique. So, just like you know, typically that we expect here, so, how this is created. So, again, this is also you know, a collection. So, this is created with a comma separated collection of unique elements using you know, curly braces operator or set function, you see these sequences that we have talked about tuples and list, as we have discussed before , they are 0 index.

So whenever we want to access the individual elements of you know those sequences tuples list, now we can use the you know brackets operator, pass on the dices 0 1 2 3 and we are able to, you know, access. So in a sense, they are ordered here, if you see, just like dicts set are also, in the definition of set itself, we have said this is unordered collection of you know, unique elements. So slightly different in that sense.

So let us see how this can be created. So you can see here is in curly braces, and we have a comma separated collection of these unique elements. So I will use few examples to again, you know, explain all of these you know aspects. So let us run this. And you can see we have created a set object, which is, you know, presented using curly braces, and we have 3 elements 1, 2, 3. Now, if I try to when I said unique elements, so if I try to create a set object, where, you know, 1 or 2 elements are repeated quite often.

So let us take this example. So within curly braces, I am writing 2, 2, 2, and then 3, 3. So, if I run this, you will see in the output I have just the 2 elements 2, 3. So only the; you know, repeated instances are not taken, and the elements are supposed to be you know unique, once each element 2 and 3 only once they have been taken, and set object has been created. So the all the elements in a set has to be you know, have to be unique elements.

(Refer Slide Time: 23:53)



jupyter Session Python Built-in Capabilities Last Checkpoint: 24/02/2019 (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code

```
In [63]:  # set operations: subset, superset, and disjoint
          # Using issubset, issuperset, and isdisjoint methods
          set1.issubset(set3)

Out[63]:  True

In [64]:  set2.issuperset(set3)

Out[64]:  False

In [65]:  set1.isdisjoint(set2)

Out[65]:  True

In [42]:  # set operation: checking if sets have same elements effectively irrespective of the container
          # Using == operator
          {1, 2, 3} == {3, 2, 1}

Out[42]:  True
```

Similarly, you can see in the next example, what I am doing is I am using set function. So here in the set function, again I am passing a list as an argument. And in that list, again, the unique elements aspect of set, and again, passing 4 and 5 is twice there, 5, 5, 6. So if I run this, and we will get a set object, with only 3 elements, they will be fast 4 elements in the list, we got a set object with just 3 elements 4, 5, 6. here.

So you can see how this particular data structure set is different from, you know, other data structure that we have talked about so far, let us talk about some of the operation set operation that we can perform on set objects. So let us first start with union of sets here. So in this, we can use union method, or this operator to actually create a union. So what I am doing is set1 and set2 we have already created.

So for the set1 I am calling this union method, and it will return me union of these 2 sets set1 and set2 recording that return value in set3. So, if I run this, that in set1 we had 1 2 3 and in set2 we have 4 5 6 and union of sets we have all these elements, because all of them were unique and these 2 sets set1 and set2 they did not have any common you know elements. So therefore, you know, all the elements are part of the final union set.

So, similarly, using the operator also, I can perform this. So, you can see set1 or set2 and you can see in the output, we have got the same result. Now, similarly, we can apply it to follow more than 2 sets set1 set2 set3, if I run this again, you can see you know, we have got the same result here again. So, let us you know move on to the next operation. So, this operation is you know intersection of sets.

So, here, again, we have 2 ways we can you know complete it, we can either use the intersection method or the ampersand operator here. So, again in this fashion, you can call set1.intersection and passing set3, and the intersection set would be created and would be returned to set4. So, if I run this. So, you remember that set1 had just 2 element 1 2 3 and set3 had 6 elements 1 2 3 4 5 6 you can see in the output 43.

And so the intersection would have the whole loft set1 element, if I run this, you can see we have got set4 which is the intersection of set1 and set3 and it is having all the elements same as set1. Now, the same thing, you know, we can perform using the ampersand operator here. So set2 ampersand set3, and you can see in the output we have 4 5 6 because as you might remember the set2 we had 4 5 6 and set3 we had all 6 element 1 2 3 4 5 6.

So, in the intersection we have got 4 5 6 here. So if I run this the same output is repeated. Similarly, we can also perform intersection operation, you know, among more than 2 sets, and let us say set1 and present set2 and a present set3. Now set1 had 1 2 3, set2 have 4 5 6 and set3 had all of these 6 elements. So, if you look at it, the common elements, there are no common elements here.

So, if I run this, I would not get any output here. Now, let us move forward. Now the next you know useful set operation is updating a set. So, how this is performed. So, we have the update method or you know, we can use this or or equal to operator here. So, in a sense these are in place counterparts of union method and all operator, that we have just gone through. So, you know, earlier we were recording the return value, we are restoring the return value.

Now here it would be in place, so if I want to form a union I can just call so set4.update set2 and you know set2 we had 4 5 6 at 4 I think we had all the you know 1 2 3. So, if I just run this so set4 will actually change it will become 1 2 3. So this is in place thing. So, we do not need to store the return value here. So, sometimes the in place counterparts can be really useful. So, most of the operators they have their in place counterparts.

So, some of the important you know methods and operators will go through. So, you can again see you are again using set4 here or equal to and you know 7 8 9. So, if I run this these 3 elements will also be would also become part of set4 itself, because this is in place you know an operator. So let us run this. Let us look at the value of set4 and you can see 9 elements have become part of this.

Let us move on and a few more you know method we would like to cover here. So another one is adding elements. So, for this we can use added method. So, for example, let us take set4 and dot add, so, if we want to add another element 10 then we can just run this set4.add10 and the set4 is going to be updated in this fashion. Similarly, we can also work on another method that is whenever we want to remove certain elements from a set.

So, for this we can use we have 2 ways to do to perform this first one is remove method, the second one is the pop method here. So, I can call these methods like this set4 is the object you know the set object that we have and dot and then remove and the element that we would like to remove from the set. So, if I run this, you will see that the last element 10 is gone from the set. Similarly, pop can also be used to remove element.

However pop method removes an arbitrary element. So, if I just pass so we do not need to pass on any arguments, because it will pick some arbitrary element and you know, I will have to call like this set4.pop, let us run this and the element that is removed that is the output. So, one first element has been removed. And similarly if you know so let us look at the updated set object set4 you can see first element is gone 1 is gone.

And we are 2 3 on words tuple line. So, if I call the pop method again, again one more arbitrary value could be removed. So, let me run this and you can see this time 2 is gone. And similarly if I run it again, then you can see this time 3 is gone. So in this fashion you know it will remove some of the other element. Let us move on. So, another set operation is clearing the set. So for that we have another method clear method.

So whenever we are required in certain situations, if you want to clear the set, so we can use this method set4.clear do not need to pass any arguments here. So, if I run this, and if I look at the updated set object, so you can see set4 there is nothing is there. Now, in this case, for this particular set object is not having any elements if I try to remove element using let us say pop method, if I run this I will get an error, you can see pop from an empty set.

So, this is something again here also that will get, so that is another difference from the remove method here that will get this error. Now, let us move on to a few other operations that are part of you know, that can be used on set objects. So, let us talk about subsets, superset and this joint set. So, sometimes you might be required to find out whether a particular you know given set is a subset of another you know, another given set.

And sometimes whether a particular given set is superset of another you know given set that we have or whether they are this joint right. So, these are some of the common operations that we typically perform and we are dealing with set objects. So, for this we have these 3 methods call is subset is superset and is disjoint methods. So, we can use these methods to find out whether, you know, a particular you know, subject is you know, subset or superset or disjoint you know set.

So, let us call set1.issubset you know we are passing set3. So, if you remember set1 was 1 2 3 and set3 had all the elements. So, this is subset. So, I think it will return true, you can see here output is true. Similarly, if I want to find out whether set2 is a superset of set3, so, set2 I think it had 4 5 6 these 3 elements, and if we want to find out we can use this method.is superset and set 3 had all 6 elements, if I run this will get false .

Similarly, if I want to find out whether set1 and set2 are disjoint. So, as we set1 add 3 elements 1 2 3 and set2 had 3 elements 4 5 6 so no common thing there. So, I think we will get 2 as the output if I run this you can see output number 65 we got 2. Similarly there is you know, another set operation that can be useful sometimes this is sometimes we might be required to check if you know 2 sets, they have same elements effectively irrespective of the container.

So, let us say scalar types 1 2 3, the other you know set object might have 3 2 1 2 1 3 and the structure of the way it might be part of that you know the other object might also be slightly different. So, it might instead of scalar type it might be part of the tuple. So, for this we can use the double equal to operator here.

So, let us run this example. So, here we are checking this within curly braces this set object having 3 elements 1, 2, 3, whether this is you know, whether this one and on the right hand side, we have another set object 3 elements first one 3, then 2, then 1. Remember that when we talk about set, we said that this is unordered, you know collection. So, in that sense, it makes sense for us to find out whether 2 sets have the same elements in a effective sense.

So we can use this double equal to operator, so if I run this here, I will get 2 here. Now, if I make some changes in this example 1 2 3 and in the right hand side of you know the set object that we have the first 2 elements, I have put them into tuple. Now, when I put them into tuple the structure of the element itself has changed. So, now you know the contents you know the element will remain same.

So, because of that, if I run w equal to operate and get false. However in third example that I have, if I change the right hand side set object and if I you know just single element 3 I have, you know, I make it a tuple object. So, just this tuple first one right hand side, and this is tuple having 3. Now effectively if I look at it, you know the same elements are there. So if I run double equal to operator again, I will get 2.

(Video Ends: 28:44)

At this point, I would like to stop here and some of the remaining aspects related to set and other data structure will cover in the next lecture, thank you.

Keywords: Data Structure, text mining modeling, Python, subset, superset, disjoint methods.