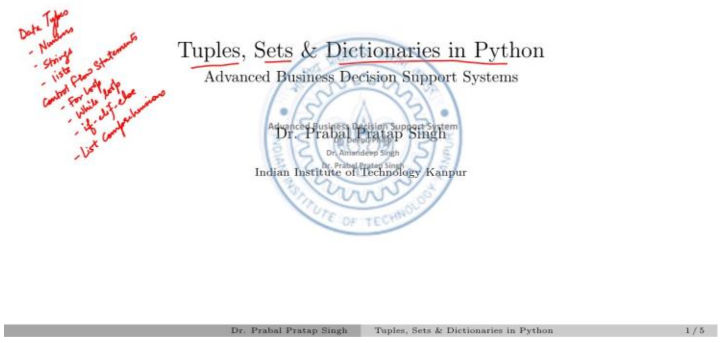
Advanced Business Decision Support Systems
Professor Deepu Philip
Department of Industrial Engineering and Management Engineering
Indian Institute of Technology, Kanpur
Professor Amandeep Singh
Imagineering Laboratory
Dr. Prabal Pratap Singh
Indian Institute of Technology, Kanpur

Lecture 31
Tuples, Sets & Dictionaries in Python

Hello everyone, I welcome you all to the lecture series on Advanced Business Decision Support Systems. I am Doctor Prabal Pratap Singh from IIT Kanpur and we are discussing the fundamentals of Python programming language.



Data Types
- Numbers
- Strings
- lists
Control Flow Statements
- For loop
- While loop
- if-elif-else
- List Comprehensions

Tuples, Sets & Dictionaries in Python

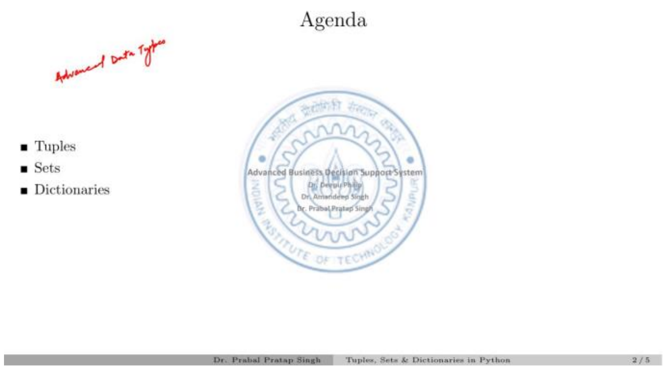
Advanced Business Decision Support Systems

Dr. Prabal Pratap Singh

Indian Institute of Technology Kanpur

Dr. Prabal Pratap Singh   Tuples, Sets & Dictionaries in Python   1 / 5

So, till now, we have discussed data types like Numbers, Strings, List and we have also seen the control flow statements like for Loop, While loop, if, Elif and Else statements. Further, we have also seen some Syntactic sugar like list comprehensions. And now today, we are moving to our next data types that are Tuple, Sets and Dictionaries in Python.



Agenda

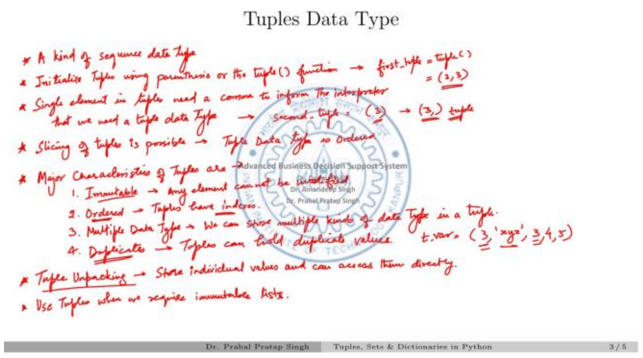
Advanced Data Types

- Tuples
- Sets
- Dictionaries

Advanced Business Decision Support System
Dr. Deepu Philip
Dr. Amandeep Singh
Dr. Prabal Pratap Singh
INDIAN INSTITUTE OF TECHNOLOGY KANPUR

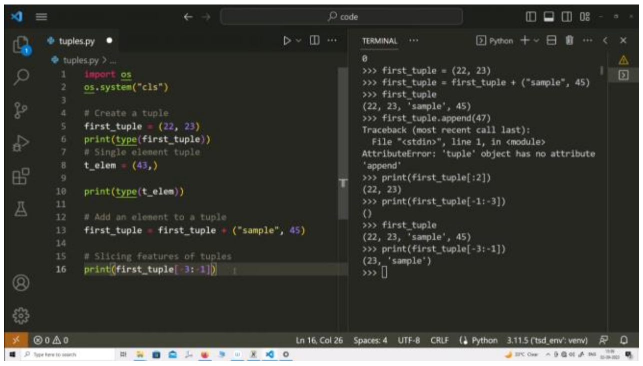
So, the very basic agenda for today is talking about the advanced Data Types. These are useful when we need some few special properties which are not available in our general purpose array like list. So, as you all know that, when we discussed list, we saw that the content of the list is mutable that is, we can alter the content whenever we like.

Also, these lists are ordered, so they all have indexes, each index have a value, but sometimes, we don't want our things to be ordered or we sometimes want that the content of some array should not contain any duplicate values or we may want to have the immutability property in our arrays. So, when these are the requirements, then we need things like Tuple, Sets and Dictionaries.



Tuples Data Type

- A kind of sequence data type
- Initialize Tuples using parenthesis or the tuple() function → first_tuple = tuple() = (2,3)
- Single element in tuples need a comma to inform the interpreter that we need a tuple data type → second_tuple = (3) → (3,) tuple
- Slicing of tuples is possible → Tuple Data type is Ordered
- Major Characteristics of Tuples are →
  1. Immutable → Any element cannot be ~~[illegible]~~ modified.
  2. Ordered → Tuples have indexes.
  3. Multiple Data Type → We can store multiple kinds of data type in a tuple.
  4. Duplicate → Tuples can hold duplicate values. t_var = (3, 'xyz', 3, 4, 5)
- Tuple Unpacking → Store individual values and can access them directly.
- Use Tuples when we require immutable lists.



```python
import os
os.system("cls")

# Create a tuple
first_tuple = (22, 23)
print(type(first_tuple))
# Single element tuple
t_elem = (43,)

print(type(t_elem))

# Add an element to a tuple
first_tuple = first_tuple + ("sample", 45)

# Slicing features of tuples
print(first_tuple[-3:-1])
```

```
0
>>> first_tuple = (22, 23)
>>> first_tuple = first_tuple + ("sample", 45)
>>> first_tuple
(22, 23, 'sample', 45)
>>> first_tuple.append(47)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'tuple' object has no attribute
'append'
>>> print(first_tuple[:2])
(22, 23)
>>> print(first_tuple[-1:-3])
()
>>> first_tuple
(22, 23, 'sample', 45)
>>> print(first_tuple[-3:-1])
(23, 'sample')
>>> 
```

So, let us start with Tuples. So, tuples is also a kind of Sequence data type. And, we can initialize tuples using parenthesis or the tuple function. So, we will see in our Visual Studio Code how to initialize a basic tuple but for example, if we want to say first tuple let us say, this is our variable and it can be initialized by just writing this statement or writing (2, 3). So, this is our basic tuple which contains two elements 2 and 3.

So what if only one element is present in a tuple? So, in those scenarios, this is a special scenario for tuple. So, single element in tuples need a comma to inform the interpreter that the tuple is present and that we need a tuple data type. So, what we mean here is, if we let us say, this is our second tuple. So, if we just write 3, so this is not a tuple interpreter we will think of it as a different kind of data type, but if we write this as 3 comma under brackets, brackets are these are the parenthesis. So, now interpreter will see this as a Tuple.

So, this is a special property to initialize a single element in a tuple. And, slicing of tuples is possible. So, what this means is that Tuple data type is ordered. So, since we are talking about different characteristics of tuples, so let us write down the major characteristics. Major characteristics of tuples are: Immutable data type.

So, what does it means is any element cannot be modified but we can add or remove elements from the tuple. The second major property is it is Ordered. Therefore, tuples have indexes. The third property is Multiple data type. So, this property is similar to what list has that is, we can store multiple kinds of data type in a tuple.

Fourth major property is about Duplicates. Since, tuples are ordered, so they have indexes. Therefore, tuples can hold duplicate values. So, what does this mean is let us say, we have a tvar_3 xyz string and then again 3, then 4, then 5. So, now we can see here that, this tuple is holding a number a string and a duplicate value also because we can assess all these elements by its index numbers.

Further, tuples have a distinct property known as Tuple Unpacking. So, we can store individual values and can assess them directly. So, we will see an example by coding it on our VS code regarding Tuple Unpacking and where it is beneficial. And, usually we can use tuples when we require immutable list. So, let us switch back to our VS code.

So, this is our lecture and we have already created a directory. So, we can now open our VS code here and we can create a new file and call it tuples dot py. Now, as a customary thing, we should create our environment here. So, let us create our environment using python m vn tsd _ env and now it has been created and we need to switch it. So, now our VS code is also using the same environment and we can change the panel position to right.

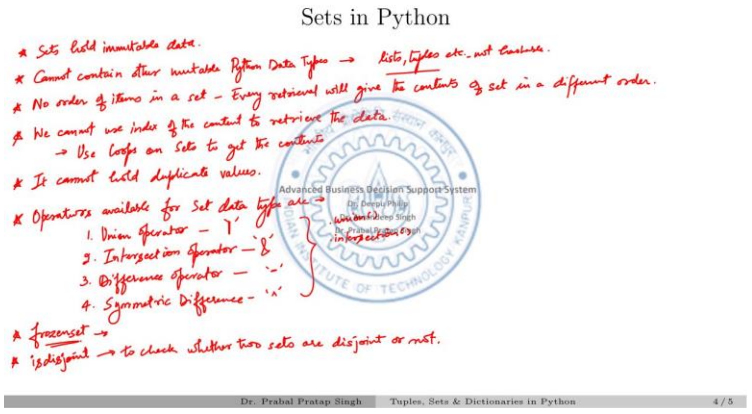
We need to activate this environment. So, now let us start coding. So, we will first import os and then we will write our command to keep using this clear screen feature by using the cls argument. Now, let us create a new tuple. So, we can see the data type of this tuple by using type command on first_ tuple and it is showing that this data type is of class tuple.

Now, what if I just overwrite this first tuple = 22 only and now check the same thing and now it is showing that it is of class int. So, that means that, just using the parenthesis doesn't define a tuple to interpreter. So, a single element tuple can be defined by let us create a new variable t and we can check the data type of this as well by using the type function on t. So, by just using this comma, the interpreter can understand that this is a tuple. So, for creating a single element tuple, we should use this syntax.

Now, next thing we can use is, we can see how to add an element to a tuple. So, we can update our first tuple by using first tuple plus. So, this is a simple addition operator which will add things to our tuple. We can either add a string like sample or we can add a number like 45. So, since we have changed our first tuple to some integer value, so we first need to rerun this line.

Now, it is again a tuple. So, we can run this addition and now you can see that, our first tuple has been updated with a string and a new value and we can. Tuple has no append options. So, we can check the slicing features of tuples. So, let us get two values from this first tuple by using our slicing notation and from start to the second value.

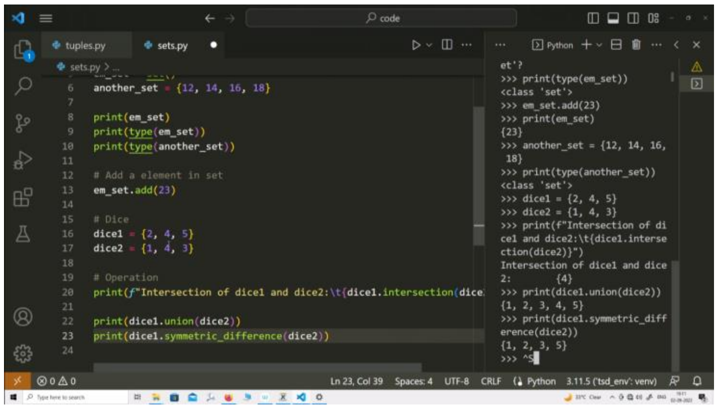
So this will print the first value and the second value which is 22 and 23. We can also get a minus 3 to minus 1 will give the negative indexing.



Sets in Python

* Sets hold immutable data.
* Cannot contain other mutable Python Data Types → lists, tuples etc. not hashable.
* No order of items in a set – Every retrieval will give the contents of set in a different order.
* We cannot use index of the content to retrieve the data.
  → Use loops on Sets to get the contents
* It cannot hold duplicate values.
* Operators available for Set data type are →
  1. Union operator – '|'
  2. Intersection operator – '&'
  3. Difference operator – '-'
  4. Symmetric Difference – '^'

  } .union() .intersection()
* frozenset →
* isdisjoint → to check whether two sets are disjoint or not.



```python
another_set = {12, 14, 16, 18}

print(em_set)
print(type(em_set))
print(type(another_set))

# Add a element in set
em_set.add(23)

# Dice
dice1 = {2, 4, 5}
dice2 = {1, 4, 3}

# Operation
print(f"Intersection of dice1 and dice2:\t{dice1.intersection(dice

print(dice1.union(dice2))
print(dice1.symmetric_difference(dice2))
```

```
et'?
>>> print(type(em_set))
<class 'set'>
>>> em_set.add(23)
>>> print(em_set)
{23}
>>> another_set = {12, 14, 16,
 18}
>>> print(type(another_set))
<class 'set'>
>>> dice1 = {2, 4, 5}
>>> dice2 = {1, 4, 3}
>>> print(f"Intersection of di
ce1 and dice2:\t{dice1.interse
ction(dice2)}")
Intersection of dice1 and dice
2:        {4}
>>> print(dice1.union(dice2))
{1, 2, 3, 4, 5}
>>> print(dice1.symmetric_diff
erence(dice2))
{1, 2, 3, 5}
>>> ^S
```

So, let us move to our next data type which is a Set in python. So, we have seen list which is a general purpose array and tuples which holds immutable data type. Now, we can see what are sets.

So, Sets holds immutable data and it cannot contain other mutable python data types. That means, in a set, we cannot hold list tuples etc. because they are not hashable. Since, a set does not have its own indexing system so it cannot hold any other data type that has its own indexing system. So, next important thing is, there are no order of items in a set which means, that every retrieval action will give the contents of Sets in a different order.

Another important thing is, we cannot use index of the content to retrieve the data. So, how will we able to retrieve the data? We need to use loops on Sets to get the contents and special property of Sets is that it cannot hold duplicate values. So, Sets are nothing but the mathematical set theory operations which usually we perform on different kinds of data Sets. So, like mathematics in python also, there are different operators for Sets. So, these operators available for set data type are, first is Union operator and we can use the pipe character available on our keyboards usually above the enter key.

The second is the Intersection operator and to perform intersection operation, we can use this ampersand (&) symbol. The third is the Difference operator which could be either minus or you can say, hyphen (-) and the last is the Symmetric difference. This is the caret symbol and all these operations can also be performed using their associative named function like dot union, dot intersection, etc.

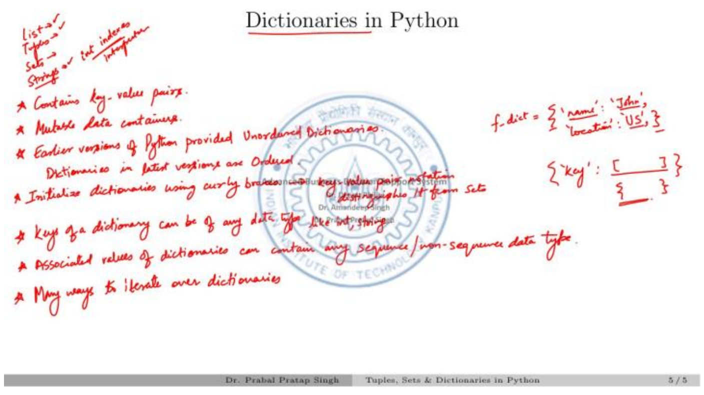
Further, there is one more special keyword available with Sets is the Frozen set. So, what it will do is, it will not change the contents of the set and we can use different functions like is disjoint to check whether two Sets are disjoint or not.

So, let us again switch back to our code. So, let us first see what are the different ways to initialize a set. So, to initialize an empty set, we can write m set = we can use this function and we can check the contents of em set and its type also. So, you can see it is using the class set.

Now, the next basic operation is how to add an element in set. So, we can add an element to this empty set as well. Right and we can print the contents and you can see that, Sets will be denoted by the interpreter using these curly braces and we can also initialize a set by using these curly braces as well. You can say, 12, 14, 16, 18 this is also a set. So, these are two different ways to initialize a set and now let us roll a simple die and do some experiments. So, let us say, we are rolling two dice and we are recording the outcomes in two different Sets.

So, dice 1 have reported 2, we are using the 6 phased dice so 2, 4 and 5 phased dice two has reported 1, 4 and 3. So, let us see how to perform different operations which we have discussed earlier. So, to get the intersection, we can write print using f string intersection of dice 1 and dice 2 is dice 1 dot intersection dice 2. So, if we run this statement, we can see that the interpreter is saying that the intersection of dice 1 and dice 2 is 4 and we know that intersection means the common things that occurred in these two Sets.

So, the only common thing is 4. And, we can also do a union operation by doing dice 1 dot union dice 2 which reported all the elements without using the duplicate element, so 1, 2, 3, 4 and 5. The interpreter is not reporting 4 two times, it is only reporting 4 ones, so this is the property of sets that it will remove the duplicates and programmers often use this important property while coding at different scenarios. We can also see the symmetric difference between these two sets by using dice 1 dot symmetric difference dice 2. So now 1, 2 and 3, 5 because we have removed the intersection element from both the sets. So, this is how we use different properties of Sets in Python.



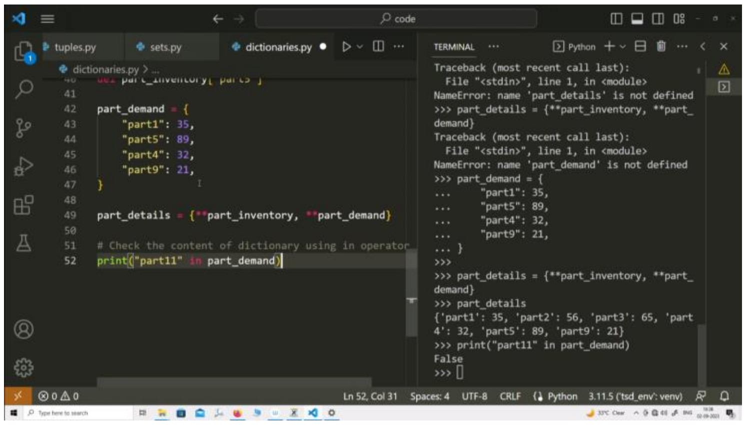
Dictionaries in Python

List → ✓
Tuples → ✓
Sets →
Strings → int indexes / integers

* Contains key-value pairs.
* Mutable data containers.
* Earlier versions of Python provided Unordered Dictionaries. Dictionaries in latest versions are Ordered.
* Initialize dictionaries using curly braces → key-value pair notation distinguishes it from sets
* Keys of a dictionary can be of any data type like int, string
* Associated values of dictionaries can contain any sequence/non-sequence data type.
* Many ways to iterate over dictionaries

f_dict = {'name': 'John', 'location': 'US', }

{'key': [ ] }
{ }



tuples.py
sets.py
dictionaries.py
TERMINAL
Python
Ln 52, Col 31 Spaces: 4 UTF-8 CRLF Python 3.11.5 ('tsd_env': venv)

```python
part_demand = {
    "part1": 35,
    "part5": 89,
    "part4": 32,
    "part9": 21,
}

part_details = {**part_inventory, **part_demand}

# Check the content of dictionary using in operator
print("part11" in part_demand)
```

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'part_details' is not defined
>>> part_details = {**part_inventory, **part_
demand}
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'part_demand' is not defined
>>> part_demand = {
...     "part1": 35,
...     "part5": 89,
...     "part4": 32,
...     "part9": 21,
... }
>>>
>>> part_details = {**part_inventory, **part_
demand}
>>> part_details
{'part1': 35, 'part2': 56, 'part3': 65, 'part
4': 32, 'part5': 89, 'part9': 21}
>>> print("part11" in part_demand)
False
>>> 
```

Now, let us move to our next important data type. So, Dictionaries is a very important data type because until now, we have seen List, Tuples, Sets, Strings. So, first we saw strings and we saw that each content of the string is having its own index, similar thing was with the list, it was having its own index tuples also and all these indexing system utilizes the integer number of indexing and these indexes were provided by interpreter.

So, these are actually the locations for the programmer and they are the references to some locations in the memory sets had no indexing system because it has this property of being unordered but here the problem arise when the programmer wanted to use a named key value pairs in while programming because most of the data contains things that are associated with other things that has its own names. So, to accommodate all these requirements, the developers introduced Dictionaries in Python.

So, Dictionaries contains key value pairs and these are mutable data containers. In earlier in versions, before I think 3.6 earlier versions of Python provided unordered dictionaries but now dictionaries in latest versions are ordered. Further, we can initialize Dictionaries using curly braces. You may say that Sets are also initialized by curly braces but here the content distinguishes between Sets and Dictionaries by the interpreter.

So, key value pair notation distinguishes it from sets. So, basic dictionary like

$$f - dict = \{'name':'John',\ 'location':'US'\}$$

So, this can be a basic dictionary which we can initialize by using these curly braces and this dictionary contains two keys which is Name and Location and each have its own associated value which is the actual name of the person and its location. So, now keys of a dictionary can be of any type like int or string Associated values of dictionaries can contain any sequence or non-sequence data type. So, this the value of a particular key let us say this is a key and here we need to fill a value.

So, this value data type can either be a list or it can either be another dictionary which will make it a nested dictionary or this can also be a set. So, this way this is very general purpose and we can create as per our requirement. So, dictionaries are very useful and there are many ways to iterate over dictionaries. So, let us now again switch back to our VS code and see the benefits of dictionaries. So, first thing again we will see how to initialize empty and filled dictionaries.

So, empty dict. This can be initialized by using this dict function. Now, you can see that interpreter is again showing that, an empty dictionary is shown as just curly braces which is similar to sets but if we use the type statement on this empty dict and we can see here that, it is of class dict. So, this is different from sets. Now, let us create a short example of inventory management and use this part inventory variable to fill the quantity of inventory in a warehouse for let us say, 4 different parts.

So, part 1 contains 23 entities. Part 2 contains 56. Part 3 contains 65. Part 4 contains 77. So, when we run this and now in the terminal, we can print the content of this dictionary by just using the variable name and we can see here that, part 1 is associated with the quantity.

Similarly, other parts are also associated. Now, this is the useful thing that until now, we were using integer indexing to locate our elements but let us now extract our contents of this dictionary by using their named locations. So, we can write part inventory and let us check the inventory of part 4.

So, we will use this square bracket notations and we will only write the name of this part which is a valid index already available in our dictionary. So, it will print the output 77 and we can also change this by just using part inventory, part 4 equals to 110. Now, if you again run this statement, it will show that the content of this dictionary element has been changed which shows that this the dictionary is mutable.

So, we can write here mutable property. Now, how can we assess all the elements? So, we have already learned about the control flow statements like Loops, For loop and other things. So, let us write a simple For loop, for part, in part, inventory print part. So, just to refresh the for loop, this is the keyword, this is the dummy element in which, all the contents of this sequence data type will come one by one and then, we can use this dummy variable to use it as per our need for our computation. So, here we are not using any computation, just printing it.

So, let us see what happens here. So, we wanted to print the complete dictionary elements one by one but what happened is, the for loop is printing these four things part, one part, two part, three and part four which are the indexes of this dictionary. So, this means that, when we simply try to assess all the elements like, the keys and its associated values, this is not the correct way of assessing it. Why? Because whenever we try to assess a dictionary, the interpreter will give only the keys of the dictionary as a iterable sequence data type. So, here part inventory is only receiving the keys as a sequence data type.

Now, to assess the complete values, we need to do something else. So, we can write part comma num in part inventory dot items. And, now if I print. So, now we can assess both the elements, the keys and the associated values. So, what changed here is, I used items method on this dictionary which is available by default. So, this will create actually a tuple and here we are using the tuple unpacking.

So, let me just first show you how tuple unpacking works. So, let us say, the demand is 67, 34 and 43. Now, we know that, this is a tuple and if we just write this as d1 comma d2 comma d3, since we already know that, there are only three values that we need to unpack from the sequence data type. So, we created three new variables and we are now writing here as demand.

Now if you print these three different values d1. So, now our d1 contains 67 and d2 contains 34. So, this is one by one mapping. So, this is called tuple unpacking and in a single line, we assigned three different values to three different new containers d1, d2 and d3. So, this is the same thing that is happening here.

We are using two different containers because we know initially that our dictionary contains keys and values and when they will unpack using these items, it will give a tuple with two different values. So, just to show you, we can also write part inventory dot items.

So, these are giving class of dict items and it is giving tuples right and on this sequence data type the list of tuples, we are just using this for loop. So, this is how we can assess all the elements by using tuple unpacking in dictionaries.

Next thing is, how to add an element to an existing dictionary. So, we already have this part inventory dictionary. Let us use it and create a new part as 5 and assign the inventory as 78.

So, this should add a new element in our part inventory. Let us see part inventory. So, now we have 5 parts, we have added one new part also, we can remove a part by using del command on a particular index of part inventory. Let us again use part 5. So, this should have deleted our part 5 from the inventory dictionary. We can again see, now again, we are back with 4 parts only in our dictionary.

One more thing, we can use is, let us create one more dictionary. So, we are now creating a new dictionary for our part demand and let us write part 1 as 35. So, there are 35 units that are needed for part 1. Part 5 is 89. Part 4 is 32. And, part 9 is 21.

Now, we can merge these two dictionaries into a new dictionary named as part details by using an asterisk operator which is available on keyboard as this. So, if we use part inventory and the same operator on part details, this will now unpack these two dictionaries into this new dictionary.

Let us see the contents, Part demand. Okay, details are not defined. This is part demand. We need to first initialize this as part demand and then run this. Now, content of part details. So, our part details contain part 1 which was available.

Part 2, part 3, part 4, part 5 and part 9. Now, if you see these two initial dictionaries closely. So, part 1 was originally 23 and part 4 was 77. And, this dictionary also contains the information about part 1 and part 4. But this was the information regarding demand and this was information regarding inventory. Now, the programmer merged these two dictionaries into one which removed the initial information about part 1.

Because we know that, these dictionaries will not contain the same keys multiple times. So, that is why, the information regarding part 1 gets overwritten and initially it was 23 but our final part details dictionary contains the part 1 information as 35. Similar was the case with part 4, which is 32. So, this is an important property and we can do one more thing is, we can check the content of dictionary using in operator. So, let us say, the programmer wants to see whether part 11 is in the part demand or not.

So, we can write part 11 in part demand. So, this is a conditional statement and it will be either true or false. So, it is showing false because there are no keys with the name part 11. So, I hope that these three important data types in python are clear now and we will keep on using these important data types in our further lectures and next we will try to see a very important python feature that is known as Functions. So thank you and we will meet in the next lecture. Thank you.