**Lists in Python**

##### Speaker: Curt Knowles, Instructor

##### [00:00:00.000]

Hi everyone in this video we're going to take a look at working with lists in Python. Some of the things we're going to do here is to show you how to use lists when writing Python programs. How to describe an item in a list, how it is accessed, how to describe the use of certain list methods for appending items, inserting items, removing items, finding the index of an item in a list. Also, how to find the minimum/maximum items, count of items in lists.

##### [00:00:39.380]

How to find the reverse order, to sort the list and to do a copy of the list as well. So let's first look at what is a list in Python. It's a data structure that is a collection. It's a data structure that holds a collection of values. List are collections that are ordered and there are changeable.

##### [00:01:07.710]

Each item in the list can be changed. Lists also allow for duplicate member values. Because lists are ordered, you can access the individual values in the collection or in the list using index numbers. And when using index numbers, remember that index numbers for Python lists are zero based. So the first item in the list is actually at index number zero, not index number one.

##### [00:01:41.780]

You can also access list values by referencing negative index numbers. For example, -1 refers to the last item in the list, -2 to the next to last item, and so forth. So if we look at how do you create a list? First, we have the syntax to create a list where you create a variable. This variable will end up being a variable of type list, and you set the list inside of square brackets, and you just identify each item in the list separated by commas.

##### [00:02:24.870]

Here's an example of creating a list called temps that holds five values that you can see there. An example of setting a list called inventory that holds three values. Notice that a list can hold any data type of values - five float values in the case of temps, three string values in the case of inventory. It can also hold mixed data types.

##### [00:02:57.300]

So in the movie list, I have three items or elements. One is a string, one is an integer and one is a float. Also, to declare an empty list with no items in it, you can just use the square brackets with nothing in between. You can also alternatively create a list, if you want a list of multiple of the same item values, you can use the multiplication operator to set the value.

##### [00:03:27.690]

In this case, we want a list that contains five zeros, so we're going to set the scores list equal to the list of zero times five, and you would end up with scores containing these five elements in the scores list. Looking at our temps list again, let's drill down and look at some different operations to be performed on the temps list. First of all the index values that are mentioned recently. You would refer to them as temps[0] through temps[4], so 48.0 would be temps[0], 30 .5 would be temps[2], 20 .2 would be temps[2] and so on.

##### [00:04:14.480]

You can also refer to them beginning at the end as negative one and count down to the beginning.

##### [00:04:21.470]

So temps[-1] is 42, temps[-2] is 100, temps[-3] is 20.2 and so on. So if we take our temps list and we want to refer to one item value in the list and set that equal to a variable, we'll call our variable temp - that is different than temps. temps is the list.

##### [00:04:48.450]

temp is one individual value in the list, so we're going to set it equal to temps[0] that would set the variable temp equal to the float value 48.0. Likewise, temps[4] would be the last item in this list - 42. If I tried to reference temps[5], the Python interpreter when I run the program will give me an index error that says index out of range. That's because there is no index five that exists in the temps list.

##### [00:05:21.260]

It only goes 0,1,2,3, and 4. If I have a list called inventory that contains six strings. If I set item equals to inventory[5], I can count 0,1,2,3,4,5 that's "scroll". So item will get the string value "scroll". Item 3 if you count that out would be "bread". Item 6 again would give an index out of range because there is no index 6 item in that list. Now you can set an item equal to a new value in the list.

##### [00:06:09.110]

The list itself is called immutable. You cannot change the entire list to another entire list, but you can change individual items in the list their values - the items in the list are called mutable because they're changeable. So with temps[3], if I want to set that to a new value, I just set an assignment statement. temps[3] equals 98. That will take the third indexed item in temps - 0,1,2,3 - the 100 and it would change it to a 98.

##### [00:06:46.260]

inventory[4] - setting it to ration would change potion to ration.

##### [00:06:52.990]

Here are some other methods for modifying a list. We're going to look at example a few of these. You have the append item in which you put in as a parameter argument, the value or the item that you want to append it will append the item to the end of the list. Then you have insert which will insert the item at the position at the left of the index that you specify in the parameter in the list. Remove item, remove the item from the list and shifts all index positions to the right of the index all the way to the end of the list.

##### [00:07:34.210]

It shifts them all up one position to fill in the gap. Index will return the index position of the item in the list and pop index would remove the item located at the index that's specified in the square brackets. It would remove it from the list and shift all index positions up to fill in the gap. So here's examples of using append, insert, and remove. We have a stats list consisting of four floats, inventory list consisting of five strings.

##### [00:08:11.540]

So if I want to append 99.5 to stats, it will add 99.5 to the end of the list. If I want to insert the item "robe" into my inventory list at index position three, it will count zero one, two, and it will insert "robe" after "shoes" at index position three. So "robe" gets inserted at index position three, and then it moves "bread" and "potion" down to index positions four and five. If I want to remove "shoes" from the inventory list, then I would go to wherever "shoes" is in the list,

##### [00:08:54.580]

remove that from the list, and I would shift everything below that - "robe", "bread" and "potion" up one to fill in the gap, so now "robe" would be at index two, "bread" at three, "potion" at four.

##### [00:09:09.950]

The pop method is an alternative way to remove an item from the list. If I have my inventory list of four strings and I pop and I do not give an index for the pop or a parameter for pop. That's saying pop the last item off the list, so "bread" would be popped off the list at the end. If I say pop[1], it will count to the index one - "hat" is index one, so it will pop "hat" off the list and move "robe" up one to index one.

##### [00:09:47.270]

Okay, so now looking at the index and the pop method, let's say that you know that you instead of you want to pop an item off the list, but you don't know what index it's at in the list, but you know what the item name is or the item value is.

##### [00:10:05.160]

So you can pop that - use pop to remove that item, even though you don't know the index, using two statements. First, you can use the index statement to find out where the items at in the list, set that equal to an Interger variable, in this case the variable i. So I look for the index of "hat" - it's at index one, so it sets i to 1, and then I can do a pop feeding in my variable i, which equals 1, as the index, and that will look to item one hat and will pop it off the list.

##### [00:10:44.830]

There's another built in function to get the length of a list, and that's the len function. So if I have my list inventory has four strings, then if I do a len(inventory) that will return four, in this case I return four and I put it in the variable len\_inventory. Now you also sometimes need to go look in a list for a value, or you need to look to see if a value exists in a list. So you use the "in" keyword to check whether or not an item is in the list.

##### [00:11:25.210]

Again, my list is inventory.

##### [00:11:27.120]

I'm looking for the item "bread". I say if the item, which is now "bread", is in the inventory list, then remove that item from the list and it will remove "bread" from the list. I can also print a list to the console in two different ways. First, I can issue just a simple print statement passing in the list as the parameter. It will print the entire list to the screen, but it will do it all on one line, and it will include the commas, the single quotes around each item and the square brackets.

##### [00:12:08.020]

So sometimes that might be the most user friendly way to display the list to the user. So what we can do is use a for loop to loop through the list, printing one item at a time to make it a little more user friendly. So here I have my inventory list with "staff", "hat" and "shoes", and I can say "for item in inventory". So I'm using my in statement with a for loop instead of an if statement, and I'm saying "for item in inventory, print the item".

##### [00:12:43.340]

So what this will do is it will traverse through every item in the inventory, starting with the left index zero and go all the way to the end, and it will perform whatever statement or statements you put inside the for loop on that particular item in the list. In this case, we're printing an item. So in essence, we have three print statements that are happening because we go through the for loop three different times. Right? So you'll have a carriage return after each print statement.

##### [00:13:14.690]

So this would print "staff", "hat", and "shoes" on three different lines without all of the square brackets, commas, quotes, and it looks more user friendly to show the list this way. So again, to process items in list, you can do it with a for loop. So here we have a list of scores. And what I'm trying to do here is I'm trying to find out how many total points have I made. Let's say these are four test scores.

##### [00:13:45.320]

How many total points have I made on all four of these tests? So what I can do is I can start with an accumulator variable called "total", in which I'm going to accumulate all score values, and I can say for score... so "score" is just an arbitrary variable that you're using to describe one element in the scores list. So I can say "for score in scores", and it will start with the left most. It will put 70 into score.

##### [00:14:15.530]

First, the total is the total plus the score. So 0 + 70 total will now be 70. Then I go through again and it takes the next item in the list - 80 - assigns it to score. Now total will be 70 + 80 or 150, and it will keep going through the list until it hits 100. Then it will drop out and then you will print the total in that list, which would be 340 when you add them all together.

##### [00:14:43.980]

You can also do this with a while loop. You just have to set up not only an accumulator variable but a counter variable. So for the while loop, what you're doing is you're not automatically traversing through the loop with a for statement and a score variable. You're actually looking for the value of i and you're referencing each individual item in the list using its index. You're using i as a counter index going through the list.

##### [00:15:15.410]

So you would want to start at 0 because the first index in the list is 0 and you would increment that each time through the while loop and you would set up the while so that your index variable would keep on operating until it is, as long as it's less than the length of scores. Now why do we say less than - why not less than or equal to? Because remember the scores list here. If you look at the length and you run the len function on it, the length is going to be four, but you don't want to traverse 0 through 4, including 4, because if you try to access scores[4], that's index out of range.

##### [00:15:57.080]

All you have is index is 0,1,2,3.

##### [00:16:00.540]

So you want to traverse the while statement from 0 to 3, not 0 to 4, because that's the only indexes you have in the list.

##### [00:16:13.200]

Here are three more list methods - the count, the reverse, and the sort. The count returns the number of items in the list. The reverse returns the number of the number of a certain item value in the list. The reverse will return the list to you in reverse order, and the sort will return the list in sorted order ascending by value. There's also a built-in function called sorted, which is an alternative way to return the list in a sorted order ascending by value.

##### [00:16:53.720]

We're going to take a look at each one of these examples, so we have a numlist. Looks like ten integers. So what this statement will do? numlist.count(14). It will count the number of times the value 14 exists in the list, so it exists once, twice - two times, and it will turn value 2 into count variable.

##### [00:17:22.750]

Now if I say numlist.reverse, it will return to me the reverse order. You can see this is the reverse order of numlist, and then the sort - numlist.sort - will return the sorted order of numlist and it will include duplicates. If there are duplicates.

##### [00:17:45.010]

Now look at the two sort methods "sort" and "sorted". First of all, we have sort. Again, I put these examples here because you have to be careful again with the string sort sequence. So if you do foodlist sort and you have a capital letter in there that's going to come before any lower case letters due to our ASCII sort sequence with letters. So Pear would actually come first here before apple, banana and orange

##### [00:18:17.380]

if you sort the food list. However, there's this optional key parameter that you can send to the sort method and you can say key equals string.lower or str.lower. That's saying, "even if it's a capital letter, starts with a capital, sort it as a lower letter. Keep the value as a capital letter, but sort it as a lower letter.

##### [00:18:42.930]

So in that case, Pear would then be put at the end. The sorted function. If we have our foodlist again, we can set a variable equal to sorted. We call sorted function and we pass the entire list into it and we can put that in another list variable called sorted\_foodlist. And then if we print that, it will print out our sorted list. sorted also has the key optional parameter that we can sort everything as a lower case and do the same thing and print the variable list that is returned and it will give us the sorted list in correct order.

##### [00:19:30.960]

Okay, a couple of more built in functions for the list structure - min and max. It does exactly what you think it would do. min will return the lowest value in the list. max will return the highest value in the list. There are also two other functions, choice and shuffle. Choice will choose a random item from the list.

##### [00:19:51.920]

shuffle will shuffle the items in the list in a random fashion. So if I have my numlist and I set a variable minimum equal to the min(numlist) that will give me the 2, the lowest value in the list. If I set maximum variable equal to max(numlist). That will give me 84, which is the highest number in the list. To use choice and shuffle, you have to do what's called an import statement.

##### [00:20:22.100]

There is a library that exists in Python called the random library that includes the abilities to be able to choice and shuffle lists. So you have to import the random library before you use choice and shuffle. So if I add my numlist again, I can set a variable choice equal to random (this is the library I imported).choice. The choice function belongs not to the list but to the random library, so I can say random.choice and I can send in the entire list and it will get a random item from the list. shuffle - I can call random.shuffle(numlist) and it will shuffle the items of the list randomly.

##### [00:21:10.130]

Okay. And one of the last things we're going to look at here with lists is copying lists and the deepcopy function. So let's take a couple of examples. Let's say I have a list called list\_one. It consists of 1,2,3,4,5.

##### [00:21:26.910]

I want to copy list\_one to a second list called list\_two, and then I want to change one of the items in list\_two, at index one, the 2. I want to change it to 4. Okay, so my list would now be 1,4,3,4,5. Now I'm going to print list\_one.

##### [00:21:54.250]

Keep in mind the list that I changed was list\_two, not list\_one. So I'm going to print list\_one and then print list\_two. But notice they both print with the 4, even though only what changed was list\_two. The reason for this is because when you just do a copy like this, you're not really copying the list.

##### [00:22:18.940]

You're pointing the variable list\_two to the address where list\_one exists in memory. So really, whenever you change list\_two, you're changing list\_one because this is just a pointer. It's not a real list. Both list\_one and list\_two are pointing to the same memory address where these five numbers are located.

##### [00:22:42.580]

So when you change list\_two, you change also this one. Okay, so sometimes you may actually really want to make a copy of the list so that you have two different lists. So the way to do that, now what I just described to you is called a shallow copy of a Python list. So the way to really copy a list so that you have two different lists is to do a deepcopy of the list. To do this, you have to import a library called copy.

##### [00:23:11.730]

These libraries. You don't have to install them on your computer. When you installed Python, it installed the standard Python libraries like copy and random and math, so you can just use them by using the import statement. But the import statement has to come first before any calls to any functions in those libraries. So you import copy library, you set your list\_one equal to 1,2,3,4,5.

##### [00:23:41.840]

Then you call the copy library's deepcopy function by saying copy.deepcopy, and you feed in the list that you really want to copy. We copy that into list\_two. And then again, we change the first index of list\_two to 4, and now we're going to print our two lists. And you can see that now when I changed index one to 4, it only changed it in list\_two.

##### [00:24:10.580]

It did not change it in list\_one. That is because list\_two is a separate list pointing to a different place in memory. So you have two copies of the 1,2,3,4,5 list at two different places in memory. So now you can change list\_two without changing list\_one. And one more thing here is being able to slice up a list into parts. So the syntax for slicing a list, it almost looks similar to a for loop.

##### [00:24:46.460]

With your parameters, you give it the list name, and then you give it a start and an end and a step just like the range function for a for loop. So the start is the index in the list that you want to start your slice at. The end is the index where you want to end the slice. And the step is how many positions you want to step for each element in the slice. Now with the end argument, it works the same as the for loop.

##### [00:25:22.670]

Now, if you remember in the for loop, if we say start 0/end 2, it's actually going to end at 1. It doesn't include the 2. The 2 is the endpoint. That's where we cut it off. We don't include the 2.

##### [00:25:37.650]

We take it all the way to the one, the one before the actual end argument. Okay. So keep that in mind when we do this. So with numbers list, I have six integers, and if I want to slice numbers starting at 0, ending at 2, it's going to take the 0 index and the 1 index, and then it's going to end at 2 and not include 2. So I get 52 and 54 as a new list.

##### [00:26:09.230]

If I take a slice and I don't give a start parameter that automatically defaults to start at 0. So this is the same thing as saying 0:2, and we would get the same answer. Now, if I don't give an end, this is saying slice all the way up to and including the end element in list. So I'm going to start at index 4. So let's count index 0.

##### [00:26:36.220]

Index 1 is 54. Index 2 is 56. Index 3 is 58. Index 4 is 60. I'm going to start at 60, and then I'm going to take from 60 all the way up to and including the end last element of the list.

##### [00:26:52.460]

So my resulting list would be 60, 62. Okay. Now here's code that copies with the step argument. In this case, we want to start at 0, end at 4, and we want to step by 2. So we're going to start with 0.

##### [00:27:09.350]

That's the 52. We're going to skip 54 because we're stepping by 2 and we're going to get 56. Now we're on index number 2 here. Okay. So we'll take the 56.

##### [00:27:20.350]

We're going to skip again and go to 60. Now we're on index 4. This is actually our end argument. So we're not going to include 60. We're going to stop it right there.

##### [00:27:30.950]

So the resulting list will be 52 and 56. Now here's an example where we have no start, no end and we're stepping by -1. When you step by a negative, that means you're starting at the end and coming forward. Okay. So no start means start at the beginning.

##### [00:27:49.020]

No end means go all the way up to and including the last element in the list. So we're going to actually start at 62 and we're going to go -1, and we're going to include all of these elements in our list up to and including the 52. So you can see the end result list here. Okay. Now and I believe this is the last slide we need to look at.

##### [00:28:21.740]

How do you concatenate or join two lists together? Here I have two lists - inventory in chest, and they each have two strings. And I'm going to use the plus symbol, the addition symbol, and we're using them on strings here. So it's not doing math. It's just concatenating or joining.

##### [00:28:43.680]

Okay. So I can say I can set up a third list called combined, and I can set it equal to Inventory + chest. It will take "staff" and "robe" plus "scroll" and "pestle" and make a new list, including them all. Now, if I print(inventory), it'll print "staff" and "robe" right here. And now, if I do inventory += chest, remember the += means to plus this to itself. So this is the same as saying "inventory = inventory + chest".

##### [00:29:21.780]

So I'm actually changing the inventory list here. So I'm taking inventory, and now I'm setting it equal to inventory + chest. inventory + chest is "staff" and "robe" and "scroll" and "pestle". So it will take all four of those now and put it into the list inventory. inventory now contains four elements. Then if I print inventory, you can see it will contain four elements or items that were concatenated together.

##### [00:29:53.040]

Okay. That concludes this lecture on lists and working with lists in Python. You'll be using these pretty extensively in this week's lab questions and in the quiz. There's some questions about lists as well. If you have any questions on using lists or getting started with lists in Python, let me know.

##### [00:30:16.840]

Send me an email or we can set up time to talk and I can walk you through some things in PyCharm. Thank you very much. Have a good evening.