List operations in Python

Another data type we discussed previously is the list.
Lists are useful because they allow you to store multiple pieces of data in a single variable.
In the security profession, you will work with a variety of lists.
For example, you may have a list of IP addresses that have accessed a network, and another list might hold information on applications that are blocked from running on the system.
Let's recap how to create a list in Python.
In this case, the items in our list are the letters A through E.
We separate them by commas and surround them with square brackets.
We can also assign our list to a variable to make it easier to use later.
Here, we've named our variable my_list.
When we access specific elements from lists, we use syntax similar to when we access the specific elements from strings.
We place its index value in brackets after the variable that stores the list.
So this would access the second item in the list.
This is because in Python, we start counting the elements in the list at zero and not at one.
So the index for the first element is zero and the index for the second element is one.
Let's try extracting some elements from a list.
We'll extract the second element by putting 1 in brackets after the variable.
We place this in a print() function to output the results, and after we run it, Python outputs the letter "b".
Similar to strings, we can also concatenate lists with the plus sign.
List concatenation is combining two lists into one by placing the elements of the second list directly after the elements of the first list.
Let's work with this in Python.
Let's concatenate two lists.
First, we define the same list as in the previous example and store it in the variable my_list.
Now, let's define an additional list with the numbers 1 through 4.
Finally, let's concatenate the two lists with a plus sign and print out the result.
And when we run it, we have a final concatenated list.
Having discussed the similarities, let's now explore the differences between lists and strings.
We mentioned earlier that strings are immutable, meaning after they are defined, they cannot be changed.
Lists, on the other hand, do not have this property, and we can freely change, add, and remove list values.
So, for example, if we have a list of malicious IP addresses, then every time a new malicious IP address is identified, we can easily add it to the list.
Let's first try changing a specific element in a list in Python.
We start with the list used in the previous example.
To change an element in a list, We combine what we learned about bracket notation with what we learned about variable assignment.
Let's change the second element in my_list, which is the string "b", to the number 7.

We place the object we want to change on the left-hand side of the variable assignment.
In this case, we'll change the second element in my_list.
Then we place an equals sign to indicate we are reassigning this element of the list.
Finally, we place the object to take its place on the right-hand side.
Here, we'll reassign the second list element to a value of 7.
Now let's print out the list and run the code to examine the change.
Perfect!
The letter "b" is now changed to the number 7.
Now, let's take a look at methods for inserting and removing elements in lists.
The first method we'll work with in this video is the insert method.
The insert method adds an element in a specific position inside a list.
The method takes two arguments: the first is the position we're adding the element to, and the second is the element we want to add.
Let's use the insert method.
We'll start with the list we defined in our my_list variable.
Then we type my_list.insert and pass in two arguments.
The first argument is the position where we want to insert the new information.
In this case, we want to insert into index 1.
The second argument is the information we want to add to the list; in this case, the integer 7.
Now let's print my_list.
Our list still begins with "a", the element with an index of 0, and now, we have the integer 7 in the next position, the position represented with an index of 1.
Notice that the letter "b", which was originally at index 1, did not get replaced like when we used bracket notation.
With the insert method, every element beyond index 1 is simply shifted down by one position.
The index of "b" is now 2.
Sometimes we might want to remove an element that is no longer needed from a list.
To do this, we can use the remove method.
The removed method removes the first occurrence of a specific element in the list.
Unlike insert, the argument of removed is not an index value.
Instead, you directly type the element you want to remove.
The remove method removes the first instance of it in the list.
Let's use the remove method to delete the letter "d" from our list.
We'll type the name of our variable my_list, then add the remove method.
We want to remove "d"from this list.
So, we'll place it in quotation marks as our argument.
Then we'll print my_list.
And let's run this.
Perfect!
"d" has now been removed from the list.
Just like with strings, being able to search through lists is a necessary skill for security analysts.
I'm looking forward to expanding our understanding as we move forward in this course.
nother data type we discussed previously is the list.
Lists are useful because they allow you to store multiple pieces of data in a single variable.
In the security profession, you will work with a variety of lists.
For example, you may have a list of IP addresses that have accessed a network, and another list might hold information on applications that are blocked from running on the system.

Let's recap how to create a list in Python.
In this case, the items in our list are the letters A through E.
We se.

Revision #1
Created 24 December 2023 12:39:59 by naruzkurai
Updated 27 December 2023 11:35:24 by naruzkurai