Develop a parsing algorithm in Python

We're now going to bring all of the pieces together to import a file, parse it, and implement a simple algorithm to help us detect suspicious login attempts.
In this video, we want to create a program that runs every time a new user logs in and checks if that user has had three or more failed login attempts.
First, let's discuss the structure of our inputs to build a strategy to develop our program.
We have a log file stored in atxt format that contains one username per line.
Each username represents a failed login attempt.
So when a user logs in, we want our program to check for their username and count how many times that username is in our log file.
If that username is repeated three or more times, the program returns an alert.
We'll start with code that imports the file of logging attempts, splits it, and stores it into a variable named usernames.
Let's try printing the variable user names to check for its contents.
We'll run this.
Perfect!
This is exactly what we expected.
The variable usernames is ready to be used in our algorithm.
Now let's develop a strategy for counting username occurrences in the list.
We'll start with the first eight elements of the usernames list.
We notice that there are two occurrences of the username "eraab" in the list, but how would we tell Python to count this?
We'll implement a for loop that iterates through every element.
Let's represent the loop variable with an arrow.
We'll also define a counter variable that starts at 0.
So, our for loop starts at the username "elarson." At every element, Python asks, "Is this element equal to the string 'eraab'?" If the answer is yes, the counter goes up by one.
If it isn't, then the counter stays the same.
Since "elarson" is not the same as "eraab," the counter remains 0.
Then, we move on to the next element.
We encounter our first occurrence of "eraab." At this point, the counter increases by 1.
As we move to the next element, we find another occurrence of "eraab," so we increase our counter by 1 again.
That means that our counter is now at 2.
We will continue this process for the rest of the list.
Now that we know the solution, let's talk about how to implement it in Python.
Solving the problem in Python will involve a for loop, a counter variable, and an if statement.
Let's get back into our code.
We'll create a function that counts a user's failed login attempts.
First, let's define our function.

We'll call it login_check().
It takes two parameters.
The first is called login_list.
This will be used for the list of failed login attempts.
The second is called current_user.
This will be used for the user who logs in.
Inside of this function, we start by defining the counter variable and set its value to 0.
Now we start the for loop.
We'll use i as our loop variable and iterate through the login list.
In other words, as the loop iterates, it will run through all the failed login attempts in the list.
Directly inside of the for loop, we start the if statement.
The if statement checks if our loop variable is equal to the current_user we're searching for.
If this condition is true, We want to add 1 to the counter.
We're almost done with our algorithm.
Now, we just need the final if-else statement to print the alert.
If the counter adds up to 3 or more, we need to tell the user that their account is locked so they can't log in.
We'll also type an else statement for users who can log in.
Our algorithm is complete!
Let's try out our new function on an example username.
We can pull out a few of the usernames in the list and try our function on them.
Let's use the first name in the list.
Let's run the code.
According to our code, this user can log in.
They have fewer than three failed login attempts.
Now let's go back to our user "eraab." Remember, they had two entries in the list of the first eight names in our failed login attempts.
Do you think they'll be able to log in?
When we run, we get an "account locked" message.
This means they had three or more failed login attempts.
Excellent work!
You've just developed your first security algorithm involving a log.
As you grow in your skills, you'll learn how to make this algorithm more efficient, but this solution works well for now.
In this video, we recapped everything we learned so far, from list operations to algorithm development, all the way to file parsing.
We did this while building an algorithm we can apply in a security context.

Revision #1
Created 2023-12-28 17:21:11 UTC by naruzkurai
Updated 2023-12-28 17:21:24 UTC by naruzkurai