Return statements

We previously learned how we can pass arguments into a function.
We can do more than pass information into a function.
We can also send information out of one!
Return statements allow us to do this.
A return statement is a Python statement that executes inside a function and sends information back to the function call.
This ability to send information back from a function is useful to a security analyst in various ways.
As one example, an analyst might have a function that checks whether someone is allowed to access a particular file and will return a Boolean value of "True" or "False" to the larger program.
We'll explore another example.
Let's create a function related to analyzing login attempts.
Based on the information it takes in, this function will compute the percentage of failed attempts and return this percentage.
The program could use this information in a variety of ways.
For example, it might be used to determine whether or not to lock an account.
So let's get started and learn how to return information from a function.
Just like before, we start by defining our function.
We'll name it calculate fails() and we'll set two parameters related to login attempts: one for total_attempts and one for failed_attempts.
Next, we'll tell Python what we want this function to do.
We want this function to store the percentage of failed attempts in a variable called fail_percentage.
We need to divide failed_attempts by total_attempts to get this percentage.
So far, this is similar to what we've learned previously.
But now, let's learn how to return the fail percentage.
To do this, we need to use the keyword return.
Return is used to return information from a function.
In our case, we'll return the percentage we just calculated.
So after the keyword return, we'll type fail_percentage.
This is our variable that contains this information.
Now, we're ready to call this function.
We'll calculate the percentage for a user who has logged in 4 times with 2 failed attempts.
So, our arguments are 4 and 2.
When we run this, the function returns the percentage of failed attempts.
It's5, or 50 percent, but in some Python environments, this might not be printed to the screen.
We cannot use the specific variable named fail_percentage outside of the function.
So, in order to use this information in another part of the program, we would need to return the value from the function and assign it to a new variable.
Let's check this out.
This time, when the function is called, the value that's returned is stored in a variable called percentage.

Then, we can use this variable in additional code.
For example, we can write a conditional that checks if the percentage of failed attempts is greater than or equal to 50 percent.
When this condition is met, we can tell Python to print an "Account locked" message.
Let's run this code.
And this time, the percentage isn't returned to the screen.
Instead, we get the “Account locked” message.
Coming up, we'll discuss more functions, but this time, we'll go over a few that are ready for use and built in to Python!

Revision #1
Created 19 December 2023 06:37:04 by naruzkurai
Updated 27 December 2023 11:35:24 by naruzkurai