**Exemplar - Define and call a function**

**Introduction**

As a security analyst, when you're writing out Python code to automate a certain task, you'll often find yourself needing to reuse the same block of code more than once. This is why functions are important. You can call that function whenever you need the computer to execute those steps. Python not only has built-in functions that have already been defined, but also provides the tools for users to define their own functions. Security analysts often define and call functions in Python to automate series of tasks.

In this lab, you'll practice defining and calling functions in Python.

**Tips for completing this lab**

**Scenario**

Writing functions in Python is a useful skill in your work as a security analyst. In this lab, you'll define and a call a function that displays an alert about a potential security issue. Also, you'll work with a list of employee usernames, creating a function that converts the list into one string.

**Task 1**

The following code cell contains a user-defined function named alert().

For this task, analyze the function definition, and make note of your observations.

You won't need to run the cell in order to answer the question that follows. But if you do run the cell, note that it will not produce an output because the function is just being defined here.

*# Define a function named `alert()`*

​

**def** alert():

print("Potential security issue. Investigate further.")

**Hint 1**

**Question 1**

Summarize what the user-defined function above does in your own words. Think about what the output would be if this function were called.

The alert() function displays the string "Potential security issue. Investigate further." to the screen. This function can be used to inform security analysts about potential security issues in a system. If this function were called, the output would show Potential security issue. Investigate further.. Recall that when a string is displayed, the quotes around the string do not appear in the output.

**Task 2**

For this task, call the alert() function that was defined earlier and analyze the output.

Be sure to replace the ### YOUR CODE HERE ### with your own code before running the following cell.

*# Define a function named `alert()`*

​

**def** alert():

print("Potential security issue. Investigate further.")

​

*# Call the `alert()` function*

​

alert()

Potential security issue. Investigate further.

**Hint 1**

**Question 2**

What are the advantages of placing this code in a function rather than running it directly?

Placing the code in a function allows you to effeciently reuse the code. Whenever you need to display the messages about a potential security issue and further investigation, you can just call the alert() function. The alternative would be to write out that print() statement every time, which would be tedious.

**Task 3**

Functions can include other components that you've already worked with. The following code cell contains a variation of the alert() function that now uses a for loop to display the alert message multiple times.

For this task, call the new alert() function and observe the output.

Be sure to replace the ### YOUR CODE HERE ### with your own code before running the following cell.

*# Define a function named `alert()`*

​

**def** alert():

**for** i **in** range(3):

print("Potential security issue. Investigate further.")

​

*# Call the `alert()` function*

​

alert()

Potential security issue. Investigate further.

Potential security issue. Investigate further.

Potential security issue. Investigate further.

**Hint 1**

**Question 3**

How does the output above compare to the output from calling the previous version of the alert() function? How are the two definitions of the function different?

The output shows Potential security issue. Investigate further. three times, each time appearing on a new line. Meanwhile, the output from calling the previous version of alert() shows the message only once. The difference in behavior is due to the for loop used in the second version. This loop iterates over a range of numbers (specified by range(3)) and executes a print() statement in each iteration. This print()statement is the same as the one in the previous function definition.

**Task 4**

In the next part of your work, you're going to work with a list of approved usernames, representing users who can enter a system. You'll be developing a function that helps you convert the list of approved usernames into one big string. Structuring this data differently enables you to work with it in different ways. For example, structuring the usernames as a list allows you to easily add or remove a username from it. In contrast, structuring it as a string allows you to easily place its contents into a text file.

For this task, start defining a function named list\_to\_string(). Write the function header.

Be sure to replace the ### YOUR CODE HERE ### with your own code. Note that running this cell will produce an error since this cell will just contain the function header; you'll write the function body and complete the function definition in a later task.

*# Define a function named `list\_to\_string()`*

​

**def** list\_to\_string():

File "<ipython-input-4-f359e12ed06d>", line 3

def list\_to\_string():

^

SyntaxError: unexpected EOF while parsing

**Hint 1**

**Task 5**

Now you'll begin to develop the body of the list\_to\_string() function.

In the following code cell, you're provided a list of approved usernames, stored in a variable named username\_list. Your task is to complete the body of the list\_to\_string() function. Recall that the body of a function must be indented. To complete the function body, write a loop that iterates through the elements of the username\_list and displays each element. Then, call the function and run the cell to observe what happens.

Be sure to replace each ### YOUR CODE HERE ### with your own code before running the following cell.

*# Define a function named `list\_to\_string()`*

​

**def** list\_to\_string():

​

*# Store the list of approved usernames in a variable named `username\_list`*

​

username\_list **=** ["elarson", "bmoreno", "tshah", "sgilmore", "eraab", "gesparza", "alevitsk", "wjaffrey"]

​

*# Write a for loop that iterates through the elements of `username\_list` and displays each element*

​

**for** i **in** username\_list:

print(i)

​

*# Call the `list\_to\_string()` function*

​

list\_to\_string()

elarson

bmoreno

tshah

sgilmore

eraab

gesparza

alevitsk

wjaffrey

**Hint 1**

**Hint 2**

**Hint 3**

**Question 4**

What do you observe from the output above?

The output shows each element from username\_list on a new line.

**Task 6**

String concatenation is a powerful concept in coding. It allows you to combine multiple strings together to form one large string, using the addition operator (+). Sometimes analysts need to merge individual pieces of data into a single string value. In this task, you'll use string concatenation to modify how the list\_to\_string() function is defined.

In the following code cell, you're provided a variable named sum\_variable that initially contains an empty string. Your task is to use string concatenation to combine the usernames from the username\_list and store the result in sum\_variable.

In each iteration of the for loop, add the current element of username\_list to sum\_variable. At the end of the function definition, write a print() statement to display the value of sum\_variable at that stage of the process. Then, run the cell to call the list\_to\_string() function and examine its output.

Be sure to replace each ### YOUR CODE HERE ### with your own code before running the following cell.

*# Define a function named `list\_to\_string()`*

​

**def** list\_to\_string():

​

*# Store the list of approved usernames in a variable named `username\_list`*

​

username\_list **=** ["elarson", "bmoreno", "tshah", "sgilmore", "eraab", "gesparza", "alevitsk", "wjaffrey"]

​

*# Assign `sum\_variable` to an empty string*

​

sum\_variable **=** ""

​

*# Write a for loop that iterates through the elements of `username\_list` and displays each element*

​

**for** i **in** username\_list:

sum\_variable **=** sum\_variable **+** i

​

*# Display the value of `sum\_variable`*

​

print(sum\_variable)

​

*# Call the `list\_to\_string()` function*

​

list\_to\_string()

elarsonbmorenotshahsgilmoreeraabgesparzaalevitskwjaffrey

**Hint 1**

**Hint 2**

**Question 5**

What do you observe from the output above?

The output shows all the elements from username\_list merged together in one line. In its current format, the output is difficult to read. It's difficult to decipher where one username ends and the next begins.

**Task 7**

In this final task, you'll modify the code you wrote previously to improve the readability of the output.

This time, in the definition of the list\_to\_string() function, add a comma and a space (", ") after each username. This will prevent all the usernames from running into each other in the output. Adding a comma helps clearly separate one username from the next in the output. Adding a space following the comma as an additional separator between one username and the next makes it easier to read the output. Then, call the function and run the cell to observe the output.

Be sure to replace each ### YOUR CODE HERE ### with your own code before running the following cell.

*# Define a function named `list\_to\_string()`*

​

**def** list\_to\_string():

​

*# Store the list of approved usernames in a variable named `username\_list`*

​

username\_list **=** ["elarson", "bmoreno", "tshah", "sgilmore", "eraab", "gesparza", "alevitsk", "wjaffrey"]

​

*# Assign `sum\_variable` to an empty string*

​

sum\_variable **=** ""

​

*# Write a for loop that iterates through the elements of `username\_list` and displays each element*

​

**for** i **in** username\_list:

sum\_variable **=** sum\_variable **+** i **+** ", "

​

*# Display the value of `sum\_variable`*

​

print(sum\_variable)

​

*# Call the `list\_to\_string()` function*

​

list\_to\_string()

elarson, bmoreno, tshah, sgilmore, eraab, gesparza, alevitsk, wjaffrey,

**Hint 1**

**Hint 2**

**Question 6**

What do you notice about the output from the function call this time?

The output shows all the elements from username\_list in one line. This time, there's a comma and a space after each username. This format is much easier to read. It's easier to distinguish one username from the next.

**Conclusion**

**What are your key takeaways from this lab?**

* Python allows you to define and call functions that you create.
* The main components of a function definition header include the function header and the function body.
  + The function header includes the def keyword, followed by the name of the function, followed by parantheses, followed by a colon.
  + The function body includese an indented block of code that instructs the computer on what to do when the function is called.
* String concatenation involves using the addition operator (+) to combine multiple strings together.
  + One use case for string concatenation is combining the strings from a list into one large string.