Previously, you explored how to define and call your own functions. In this reading, you'll revisit what you learned about functions and examine how functions can improve efficiency in a cybersecurity setting.

Functions in cybersecurity

A **function** is a section of code that can be reused in a program. Functions are important in Python because they allow you to automate repetitive parts of your code. In cybersecurity, you will likely adopt some processes that you will often repeat.

When working with security logs, you will often encounter tasks that need to be repeated. For example, if you were responsible for finding malicious login activity based on failed login attempts, you might have to repeat the process for multiple logs.

To work around that, you could define a function that takes a log as its input and returns all potentially malicious logins. It would be easy to apply this function to different logs.

Defining a function

In Python, you'll work with built-in functions and user-defined functions. **Built-in functions** are functions that exist within Python and can be called directly. The `print()` function is an example of a built-in function.

User-defined functions are functions that programmers design for their specific needs. To define a function, you need to include a function header and the body of your function.

Function header

The function header is what tells Python that you are starting to define a function. For example, if you want to define a function that displays an `"investigate activity"` message, you can include this function header:

```
def display_investigation_message():
```

The `def` keyword is placed before a function name to define a function. In this case, the name of that function is `display_investigation_message`.

The parentheses that follow the name of the function and the colon (`:`) at the end of the function header are also essential parts of the syntax.

Pro tip: When naming a function, give it a name that indicates what it does. This will make it easier to remember when calling it later.

Function body

The body of the function is an indented block of code after the function header that defines what the function does. The indentation is very important when writing a function because it separates the definition of a function from the rest of the code.

To add a body to your definition of the `display_investigation_message()` function, add an indented line with the `print()` function. Your function definition becomes the following:

```
def display_investigation_message():

    print("investigate activity")
```

Calling a function

After defining a function, you can use it as many times as needed in your code. Using a function after defining it is referred to as calling a function. To call a function, write its name followed by parentheses. So, for the function you previously defined, you can use the following code to call it:

```
display_investigation_message()
```

Although you'll use functions in more complex ways as you expand your understanding, the following code provides an introduction to how the `display_investigation_message()` function might be part of a larger section of code. You can run it and analyze its output:

```
1
def display_investigation_message():
```

```
    print("investigate activity")

application_status = "potential concern"

email_status = "okay"

if application_status == "potential concern":

    print("application log:")

    display_investigation_message()

if email_status == "potential concern":

    print("email log:")

    display_investigation_message()
```

Run

The `display_investigation_message()` function is used twice within the code. It will print `"investigate activity"` messages about two different logs when the specified conditions evaluate to `True`. In this example, only the first conditional statement evaluates to `True`, so the message prints once.

This code calls the function from within conditionals, but you might call a function from a variety of locations within the code.

Note: Calling a function inside of the body of its function definition can create an infinite loop. This happens when it is not combined with logic that stops the function call when certain conditions are met. For example, in the following function definition, after you first call `func1()`, it will continue to call itself and create an infinite loop:

```
def func1():
```

```
    func1()
```

Key takeaways

Python’s functions are important when writing code. To define your own functions, you need the two essential components of the function header and the function body. After defining a function, you can call it when needed.