Work with built-in functions

Previously, you explored built-in functions in Python, including *print()*, *type()*, *max()*, and *sorted()*. **Built-in functions** are functions that exist within Python and can be called directly. In this reading, you'll explore these further and also learn about the *min()* function. In addition, you'll review how to pass the output of one function into another function.

print()

The *print()* function outputs a specified object to the screen. The *print()* function is one of the most commonly used functions in Python because it allows you to output any detail from your code.

To use the *print()* function, you pass the object you want to print as an argument to the function. The *print()* function takes in any number of arguments, separated by a comma, and prints all of them. For example, you can run the following code that prints a string, a variable, another string, and an integer together:

```
month = "September"
print("Investigate failed login attempts during", month, "if more than", 100)
```

```
Investigate failed login attempts during September if more than 100
```

type()

The *type()* function returns the data type of its argument. The *type()* function helps you keep track of the data types of variables to avoid errors throughout your code.

To use it, you pass the object as an argument, and it returns its data type. It only accepts one argument. For example, you could specify *type("security")* or *type(7)*.

Passing one function into another

When working with functions, you often need to pass them through *print()* if you want to output the data type to the screen. This is the case when using a function like *type()*. Consider the following code:

```
print(type("This is a string"))
```

```
<class 'str'>
```

It displays *str*, which means that the argument passed to the *type()* function is a string. This happens because the *type()* function is processed first and its output is passed as an argument to the *print()* function.

max() and min()

The *max()* function returns the largest numeric input passed into it. The *min()* function returns the smallest numeric input passed into it.

The *max()* and *min()* functions accept arguments of either multiple numeric values or of an iterable like a list, and they return the largest or smallest value respectively.

In a cybersecurity context, you could use these functions to identify the longest or shortest session that a user logged in for. If a specific user logged in seven times during a week, and you stored their access times in minutes in a list, you can use the *max()* and *min()* functions to find and print their longest and shortest sessions:

```
time_list = [12, 2, 32, 19, 57, 22, 14]
print(min(time_list))
print(max(time_list))
```

```
2
57
```

sorted()

The *sorted()* function sorts the components of a list. The *sorted()* function also works on any iterable, like a string, and returns the sorted elements in a list. By default, it sorts them in ascending order. When given an iterable that contains numbers, it sorts them from smallest to largest; this includes iterables that contain numeric data as well as iterables that contain string data beginning with numbers. An iterable that contains strings that begin with alphabetic characters will be sorted alphabetically.

The *sorted()* function takes an iterable, like a list or a string, as an input. So, for example, you can use the following code to sort the list of login sessions from shortest to longest:

```
time_list = [12, 2, 32, 19, 57, 22, 14]
print(sorted(time_list))
```

```
[2, 12, 14, 19, 22, 32, 57]
```

This displays the sorted list.

The *sorted()* function does not change the iterable that it sorts. The following code illustrates this:

```
time_list = [12, 2, 32, 19, 57, 22, 14]
print(sorted(time_list))
print(time_list)
```

```
[2, 12, 14, 19, 22, 32, 57]
[12, 2, 32, 19, 57, 22, 14]
```

The first *print()* function displays the sorted list. However, the second *print()* function, which does not include the *sorted()* function, displays the list as assigned to *time_list* in the first line of code.

One more important detail about the *sorted()* function is that it cannot take lists or strings that have elements of more than one data type. For example, you can't use the list *[1, 2, "hello"]*.

Key takeaways

Built-in functions are powerful tools in Python that allow you to perform tasks with one simple command. The *print()* function prints its arguments to the screen, the *type()* function returns the data type of its argument, the *min()* and *max()* functions return the smallest and largest values of an iterable respectively, and *sorted()* organizes its argument.

Resources for more information

These were just a few of Python's built-in functions. You can continue learning about others on your own:

- The Python Standard Library documentation

: A list of Python's built-in functions and information on how to use them

---