Activity_Assign Python variables

February 22, 2025

1 Activity: Assign Python variables

1.1 Introduction

Variables help security analysts to keep track of a variety of security-related information. For example, analysts may need to create Python variables for the users who are allowed to log in, the number of login attempts that they're permitted, and the current number of attempts that a user has made.

In this lab, you'll practice assigning values to variables and determining their data types.

Tips for completing this lab

As you navigate this lab, keep the following tips in mind:

- `### YOUR CODE HERE ###` indicates where you should write code. Be sure to replace this with your own code before running the code cell.
- Feel free to open the hints for additional guidance as you work on each task.
- To enter your answer to a question, double-click the markdown cell to edit. Be sure to replace the "[Double-click to enter your responses here.]" with your own answer.
- You can save your work manually by clicking File and then Save in the menu bar at the top of the notebook.
- You can download your work locally by clicking File and then Download and then specifying your preferred file format in the menu bar at the top of the notebook.

1.2 Scenario

You are a security analyst who is responsible for writing code that will automate analysis of login attempts made to a specific device. As the first step, you'll need to create variables to keep track of information relevant to the login process. This information includes the device ID, list of approved usernames, maximum login attempts allowed per user, current login attempts made by a user, and login status.

Throughout this lab, you'll assign these variables and check the data types of the variables.

1.3 Task 1

In your work as an analyst, imagine there is a device only users specified on an allow list can access, and its device ID is `"72e08x0"`.

In the following code cell, assign this value to a variable named `device_id`. Then, display the contents of the variable and observe the output.

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

[8]:
```
# Assign the `device_id` variable to the device ID that only specified users␣
↪can access

device_id = "72e08x0"

# Display `device_id`

print (device_id)
```

```
72e08x0
```

Hint 1

Use quotation marks around the value that you assign to the `device_id` variable.

Hint 2

Assign the value `"72e08x0"` to the variable `device_id`. To do this, place the value to right of the `=` operator.

Hint 3

To display the contents of `device_id`, place the name of this variable inside the `print()` function.

1.4 Task 2

Now that the variable `device_id` is defined, you can return its data type.

In this task, use a Python function to find the data type of the variable `device_id`. Store the data type in another variable called `device_id_type`. Then, display `device_id_type` to examine the output.

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

[2]:
```
# Assign the `device_id` variable to the device ID that only specified users␣
↪can access

device_id = "72e08x0"

# Assign `device_id_type` to the data type of `device_id`

device_id_type = type (device_id)

# Display `device_id_type`
```

```
print(device_id_type)
```

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

Hint 1

The `type()` function allows you to get the data type of a given value.

Hint 2

Use `type(device_id)` to get the data type of the `device_id` variable.

Hint 3

To display `device_id_type`, place the name of this variable inside the `print()` function.

Question 1 Based on the output above, what do you observe about the data type of `device_id`?

The data type results in <class 'str' which lets us know that it, device_id is a string value.

1.5 Task 3

As you continue your work, you're provided a list of usernames of users who are allowed to access the device. The usernames with this access are `"madebowa"`, `"jnguyen"`, `"tbecker"`, `"nhersh"`, and `"redwards"`.

In this task, create a variable called `username_list`. Assign a list with the approved usernames to this variable. Then, display the value of the `username_list` variable.

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

[4]:
```
# Assign `username_list` to the list of usernames who are allowed to access the␣
↪device

username_list = ["madebowa", "jnguyen", "tbecker", "nhersh", "redwards"]

# Display `username_list`

print(username_list)
```

```
['madebowa', 'jnguyen', 'tbecker', 'nhersh', 'redwards']
```

Hint 1

To create a list in Python, use square brackets. Inside the square brackets, write the elements of the list, with a comma between elements.

Hint 2

To assign a value to a variable in Python, place the name of the variable to the left of the `=` operator, and place the value to the right of the `=` operator.

In this task, make sure to place `["madebowa", "jnguyen", "tbecker", "nhersh", "redwards"]` to the right of the `=` operator.

Hint 3

To display `username_list`, place the name of this variable inside the `print()` function.

1.6 Task 4

In this task, find the data type of the `username_list`. Store the type in a variable called `username_list_type`. Then, display `username_list_type` to examine the output.

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

[7]:
```
# Assign `username_list` to the list of usernames who are allowed to access the␣
 ↪device

username_list = ["madebowa", "jnguyen", "tbecker", "nhersh", "redwards"]

# Assign `username_list_type` to the data type of `username_list`

username_list_type = type (username_list)
# Display `username_list_type`

print (username_list_type)
```

```
<class 'list'>
```

Hint 1

The `type()` function allows you to get the data type of a given value.

Hint 2

To get the data type of `username_list`, call the `type()` function and pass in `username_list`.

Hint 3

In Python, you can use the `print()` function to display the value of a variable. To display `username_list_type`, call the `print()` function and pass in `username_list_type`.

Question 2 Based on the output above, what do you observe about the data type of `username_list`?

The following above shows that username_list_type is a type of list, confirming that username_list carries a list.

1.7 Task 5

Now, imagine that you've been informed that the previous list is not up-to-date and that there is another employee that now has access to the device. You're given the updated list of usernames with access, including the new employee, as follows: `"madebowa"`, `"jnguyen"`, `"tbecker"`, `"nhersh"`, `"redwards"`, and `"lpope"`.

In this task, reassign the variable `username_list` to the new list. Run the code to display the list before and after it's been updated to observe the difference.

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

[8]:
```
# Assign `username_list` to the list of usernames who are allowed to access the␣
 ↪device

username_list = ["madebowa", "jnguyen", "tbecker", "nhersh", "redwards"]

# Display `username_list`

print(username_list)

# Assign `username_list` to the updated list of usernames who are allowed to␣
 ↪access the device

username_list = ["madebowa", "jnguyen", "tbecker", "nhersh", "redwards",␣
 ↪"lpope"]

# Display `username_list`

print(username_list)
```

```
['madebowa', 'jnguyen', 'tbecker', 'nhersh', 'redwards']
['madebowa', 'jnguyen', 'tbecker', 'nhersh', 'redwards', 'lpope']
```

Hint 1

When reassigning a variable to a new value in Python, place the name of the variable to the left of the `=` operator, just as you would when assigning the variable for the first time.

Hint 2

To reassign `username_list` to the updated list, place `username_list` to the left of the `=` operator.

Question 3 Based on the output above, what do you observe about the contents of `username_list`?

The following shows the old list and new updated list.

1.8 Task 6

In this task, define a variable called `max_logins` that represents the maximum number of login attempts allowed per user. Store the value `3` in this variable. Then, store its data type in another variable called `max_logins_type`. Display `max_logins_type` to examine the output.

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

[9]:
```
# Assign `max_logins` to the value 3

max_logins = 3

# Assign `max_logins_type` to the data type of `max_logins`

max_logins_type = type(max_logins)

# Display `max_logins_type`

print (max_logins_type)
```

```
<class 'int'>
```

Hint 1

When assigning a value to a variable in Python, use the `=` operator. Place the name of the variable to the left of the `=` operator, and place the value to the right of the `=` operator.

Hint 2

To assign `3` to `max_logins`, place `max_logins` to the left of the `=` operator, and place `3` to the right of the `=` operator.

To assign `max_logins_type`, place `max_logins_type` to the left of the `=` operator before the `type()` function call.

Hint 3

In Python, you can use the `print()` function to display the value of a variable. To display `max_logins_type`, call `print()` and pass in `max_logins_type`.

Question 4 Based on the output above, what do you observe about the data type of `max_logins`?

The following code outputs 'int' which confirms that the max_logins are integer values.

1.9 Task 7

In this task, define a variable called `login_attempts` that represents the current number of login attempts made by a user. Store the value `2` in this variable. Then, store its data type in a variable called `login_attempts_type`. Display `login_attempts_type` to observe the output.

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

[10]:
```
# Assign `login_attempts` to the value 2

login_attemps = 2

# Assign `login_attempts_type` to the data type of `login_attempts`

login_attemps_type = type (login_attemps)

# Display `login_attempts_type`

print(login_attemps_type)
```

```
<class 'int'>
```

Hint 1

When assigning a value to a variable in Python, use the `=` operator. Place the name of the variable to the left of the `=` operator, and place the value to the right of the `=` operator.

Hint 2

To assign `2` to `login_attempts`, place `login_attempts` to the left of the `=` operator, and place `2` to the right of the `=` operator.

To assign `login_attempts_type`, place `login_attempts_type` to the left of the `=` operator, and place a call to the `type()` function to the right of the `=` operator.

When calling `type()`, make sure to pass in `login_attempts`.

Hint 3

In Python, you can use the `print()` function to display the value of a variable. To display `login_attempts_type`, call `print()` and pass in `login_attempts_type`.

Question 5 Based on the output above, what do you observe about the data type of `login_attempts`?

Baed on the following output, the data shows that login_attemps is an interger value.

1.10 Task 8

In this task, you'll determine the Boolean value that represents whether the current number of login attempts a user has made is less than or equal to the maximum number of login attempts allowed.

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

[12]:
```
# Assign `max_logins` to the value 3

max_logins = 3

# Assign `login_attempts` to the value 2

login_attempts = 2

# Determine whether the current number of login attempts a user has made is␣
 ↪less than or equal to the maximum number of login attempts allowed,
# and display the resulting Boolean value

print(login_attemps<=max_logins)
```

```
True
```

Hint 1

In Python, you can use the `<=` comparison operator to determine whether one value is less than or equal to another value.

Hint 2

To determine whether the current number of login attempts a user has made is less than or equal to the maximum number of login attempts allowed, use the `<=` operator. Place `login_attempts` to the left of the `<=` operator, and place `max_logins` to the right of the `<=` operator.

To make sure the resulting Boolean value is displayed, write this code inside of the parantheses where `print()` is called.

Question 6 What is the output? What does this mean?

The following shoes a Boolean value of True. This is because the login_attemps has not in fact attempted 3 or more login attempts.

1.11 Task 9

This code continues to check for the Boolean value of whether `max_logins` is less than or equal to `login_attempts`. In this task, reassign other values to `login_attempts`. For example, you might choose a value that is higher than the maximum number of attempts allowed. Observe how the output changes.

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

[16]:
```
# Assign `max_logins` to the value 3

max_logins = 3
```

```
# Assign `login_attempts` to a specific value

login_attempts = 4

# Determine whether the current number of login attempts a user has made is␣
 ↪less than or equal to the maximum number of login attempts allowed,
# and display the resulting Boolean value

print(login_attempts <=max_logins)
```

```
False
```

Hint 1

To assign `login_attempts` to a specific value, make sure to write the value to the right of the `=` operator.

Question 7 Based on the different values you assigned to `login_attempts`, what did you observe about the output?

Based on the different values I assigned to login_attempts, I quickly noticed the Boolean value is false. This confirms that the number of login attemps a user made is higher than the maximum number of atttemps allowed.

1.12 Task 10

Finally, you can also assign a Boolean value of `True` or `False` to a variable.

In this task, you'll create a variable called `login_status`, which is a Boolean that represents whether a user is logged in. Assign `False` to this variable and store its data type in a variable called `login_status_type` and display it.

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

[17]:
```
# Assign `login_status` to the Boolean value False

login_status = True

# Assign `login_status_type` to the data type of `login_status`

login_status_type = type(login_status)

# Display `login_status_type`

print(login_status_type)
```

```
<class 'bool'>
```

Hint 1

To assign the Boolean value `False` to the `login_status` variable, make sure to write `False` to the right of the `=` operator.

Hint 2

Note that Boolean values should not have quotation marks around them in code.

Question 8 Based on the output above, what do you observe about the data type of `login_status`?

Based on the output above, the data type of login_status is confirmed to be a Boolean value

1.13 Conclusion

What are your key takeaways from this lab?

There are a handful of variables we can use as cybersecurity analyst with Python and although tricky at first, you can use it to your advantage to simplify your work. Some variables bool=Boolean, int=integer, stg= string, lst=list, print() displays the output, </> more /less than, and so many more things.

[ ]: