Assign and reassign variables in Python

Previously, you've explored variables and how to assign and reassign them in Python. In this reading, you'll expand your understanding of these topics. You'll also learn about the general practice of naming variables so that you can avoid syntax errors and improve code readability.

What are variables?

In a programming language, a **variable** is a container that stores data. It's a named storage location in a computer's memory that can hold a value. It stores the data in a particular data type, such as integer, string, or Boolean. The value that is stored in a variable can change.

You can think of variables as boxes with labels on them. Even when you change the contents of a box, the label on the box itself remains the same. Similarly, when you change the value stored in a variable, the name of the variable remains the same.

Security analysts working in Python will use a variety of variables. Some examples include variables for login attempts, allow lists, and addresses.

Working with variables

In Python, it's important to know both how to assign variables and how to reassign them.

Assigning and reassigning variables

If you want to create a variable called *username* and assign it a value of "*nzhao*", place the variable to the left of the equals sign and its value to the right:

 # Assign '*username*'

username = "*nzhao*"

If you later reset this username to "*zhao2*", you still refer to that variable container as *username*.

 # Reassign '*username*'

username = "*zhao2*"

Although the contents have changed from "*nzhao*" to "*zhao2*", the variable *username* remains the same.

Note: You must place "*nzhao*" and "*zhao2*" in quotation marks because they're strings. Python automatically assigns a variable its data type when it runs. For example, when the *username* variable contains the string "*nzhao*", it’s assigned a string data type.

Assigning variables to variables

Using a similar process, you can also assign variables to other variables. In the following example, the variable *username* is assigned to a new variable *old_username*:

 # Assign a variable to another variable

username = "*nzhao*"

old_username = username

Because *username* contains the string value of "*nzhao*" and *old_username* contains the value of *username*, *old_username* now contains a value of "*nzhao*".

Putting it together

The following code demonstrates how a username can be updated. The *username* variable is assigned an initial value, which is then stored in a second variable called *old_username*. After this, the *username* variable is reassigned a new value. You can run this code to get a message about the previous username and the current username:

```
username = "nzhao"
old_username = username
username = "zhao2"
print("Previous username:", old_username)
print("Current username:", username)
```

Output

```
Previous username: nzhao
Current username: zhao2
```

Best practices for naming variables

You can name a variable almost anything you want, but there are a few guidelines you should follow to ensure correct syntax and prevent errors:

- Use only letters, numbers, and underscores in variable names. Valid examples: *date_3*, *username*, *interval2*
- Start a variable name with a letter or underscore. Do not start it with a number. Valid examples: *time*, *_login*
- Remember that variable names in Python are case-sensitive. These are all different variables: *time*, *Time*, *TIME*, *timE*.
- Don't use Python's built-in keywords or functions for variable names. For example, variables shouldn't be named *True*, *False*, or *if*.

Additionally, you should follow these stylistic guidelines to make your code easier for you and other security analysts to read and understand:

- Separate two or more words with underscores. Valid examples: *login_attempts, invalid_user, status_update*
- Avoid variables with similar names. These variables could be easily confused with one another: *start_time*, *starting_time*, *time_starting*.
- Avoid unnecessarily long names for variables. For instance, don't give variables names like *variable_that_equals_3*.
- Names should describe the data and not be random words. Valid examples: *num_login_attempts*, *device_id*, *invalid_usernames*

Note: Using underscores to separate multiple words in variables is recommended, but another convention that you might encounter is capitalizing the first letter of each word except the first word. Example: *loginAttempt*

Key takeaways

It's important for security analysts to have a fundamental understanding of variables. Variables are containers of data. They are assigned values and can also be reassigned other values or variables. It's helpful to remember the best practices for naming variables in order to create more functional, readable code.

Revision #1
Created 2023-12-09 02:29:12 UTC by naruzkurai
Updated 2023-12-19 03:38:35 UTC by naruzkurai