Developing An Algorithm

Defining Approved Users and Devices

In the first part of this project, I created an algorithm to automate the process of validating users and their assigned devices. I began by defining two Python lists: approved_users and approved_devices. Each username in the approved_users list corresponds directly to a specific device ID in the approved_devices list. This pairing forms the foundation of the authentication logic, similar to how organizations maintain user and device access mappings in real-world security systems.

```
# Assign `approved_users` to a list of approved usernames

approved_users = ["elarson", "bmoreno", "sgilmore", "eraab", "gesparza"]

# Assign `approved_devices` to a list of device IDs that correspond to the␣
 ↪usernames in `approved_users`

approved_devices = ["8rp2k75", "hl0s5o1", "4n482ts", "a307vir", "3rcv4w6"]

# Define a function named `login` that takes in two parameters, `username` and␣
 ↪`device_id`

def login(username, device_id):
```

Creating the Login Function

Next, I defined a function called login() that takes two parameters — username and device_id. This function serves as the core of the algorithm, allowing for automated checks each time a user attempts to log in. By passing in the username and the associated device ID, the function can dynamically determine if access should be granted or denied without manual verification, simulating how automated security systems streamline authentication.

Inside the login() function, I used conditional statements to automate decision-making. The function first checks if the username exists in the approved_users list. If it does, the algorithm retrieves the index of that username and compares the corresponding device ID from the approved_devices list. If both values match, it prints a message confirming that the user is authorized and using their assigned device. If the device ID doesn't match, the function automatically flags the attempt as an unauthorized access.

Developing An Algorithm

```
    # If `username` belongs to `approved_users`,

    if username in approved_users:

        # then display "The user ______ is approved to access the system.",

        print("The user", username, "is approved to access the system.")

        # assign `ind` to the index of `username` in `approved_users`,

        ind = approved_users.index(username)

        # and execute the following conditional
        # If `device_id` matches the element at the index `ind` in␣
↪`approved_devices`,

        if device_id == approved_devices[ind]:

          # then display "______ is the assigned device for ______"

          print(device_id, "is the assigned device for", username)

        # Otherwise,

        else:

          # display "______ is not their assigned device"

          print(device_id, "is not their assigned device.")

    # Otherwise (part of the outer conditional and handles the case when␣
↪`username` does not belong to `approved_users`),

    else:

        # Display "The user ______ is not approved to access the system."

        print("The username", username, "is not approved to access the system.")

# Call the function you just defined to experiment with different username and␣
↪device_id combinations

login("bmoreno", "hl0s5o1")
login("elarson", "r2s5r9g")
login("abernard", "4n482ts")
```

```
The user bmoreno is approved to access the system.


hl0s5o1 is the assigned device for bmoreno
The user elarson is approved to access the system.
r2s5r9g is not their assigned device.
The username abernard is not approved to access the system.
```

Developing An Algorithm

This automated algorithm reflects how security systems validate login credentials and enforce **user-device pairing policies** to prevent unauthorized access. It also demonstrates the power of Python logic to simplify repetitive security tasks and ensure consistent validation across multiple users.