Project

I'm working on a script to automate checking operating system updates. My code uses a conditional statement to determine if a device needs an update by comparing its operating system against the known up-to-date version. If the system isn't OS 2, the script will flag that an update is required, helping streamline the process of maintaining system security.

In [4]:
```
# Assign a variable named `system` to a specific operating system, represented as a string
# This variable indicates which operating system is running
# Feel free to run this cell multiple times; each time try assigning `system` to different values ("OS 1", "OS 2", "

system = "OS 2"

# If OS 2 is running, then display a "no update needed" message

if system == "OS 2":
    print("no update needed")
```

```
no update needed
```

In this task, I'm testing my script with different operating system values. I'll assign OS 1, OS 2, and OS 3 to the operating_system variable one at a time, keeping the conditional statement unchanged. By running the script with each value, I'll observe how it correctly identifies which systems need updates and which are up-to-date. This helps me verify that my conditional logic works as intended for all possible scenarios.

In [4]:
```
# Assign `system` to a specific operating system
# This variable represents which operating system is running
# Feel free to run this cell multiple times; each time try assigning `system` to different values ("OS 1", "OS 2", "

system = "OS 1"

# If OS 2 is running, then display a "no update needed" message

if system == "OS 2":
    print("no update needed")
```

Question 1

What happens when OS 2 is running? What happens when OS 1 is running?

```
When the system is identified as "OS 2", the code executes and displays the message "no update needed". However, if
the system is running "OS 1", the code doesn't produce any output. The print function is only triggered when the
condition system == "OS 2" is met and returns True.
```

In this task I created a system update notification script that checks the operating system type. The script displays a no update needed message only when the system is OS 2, remaining silent for OS 1. This approach provides targeted update communication based on the specific system running.

In [7]:
```
# Assign `system` to a specific operating system
# This variable represents which operating system is running

system = "OS 2"

# If OS 2 is running, then display a "no update needed" message
# Otherwise, display a "update needed" message

if system == "OS 2":
    print("no update needed")
else:
    print("update needed")
```

```
no update needed
```

> **Hint 1**
>
> Use the `else` keyword.

Question 2

In this setup what happens when OS 2 is running? And what happens when OS 2 is not running?

```
In this configuration, the system checks the current operating system. If OS 2 is active, it shows "no update
needed". When OS 3 (or any other operating system) is running, the script indicates "update needed".
```

In this task, I'm improving the conditional logic for the system update notification script. I'm adding two `elif` statements after the initial `if` statement to handle specific operating systems. The first `elif` checks for `OS 1`, and the second for `OS 3`, both displaying `update needed` when true. I'm then testing the script with various system values, including `OS 1`, `OS 2`, `OS 3`, and other strings like `OS 4`. This approach ensures more precise control over update notifications for different operating systems and handles unexpected inputs more effectively.

In [12]:
```
# Assign `system` to a specific operating system
# This variable represents which operating system is running

system = "OS 1"

# If OS 2 is running, then display a "no update needed" message
# Otherwise if OS 1 is running, display a "update needed" message
# Otherwise if OS 3 is running, display a "update needed" message

if system == "OS 2":
    print("no update needed")
elif system == "OS 1":
    print("update needed")
elif system == "OS 3":
    print ("update needed")
```

```
update needed
```

Question 3

Under this setup what happens when OS 2 is running? What happens when OS 1 is running? What happens when OS 3 is running? What happens when neither of those three operating systems are running?

```
This configuration implements a conditional display system based on the active operating system. It outputs "no
update needed" exclusively when OS 2 is in use. For OS 1 or OS 3, the system shows "update needed". If the current
operating system is not one of these three, the display remains blank.
```

In this task, I'll consolidate two `elif` statements into a single, more concise condition using a logical operator. The goal is to simplify the code while maintaining the original logic for displaying update status based on the operating system.

In [24]:
```
# Assign `system` to a specific operating system
# This variable represents which operating system is running

system = "OS 4"

# If OS 2 is running, then display a "no update needed" message
# Otherwise if either OS 1 or OS 3 is running, display a "update needed" message

if system == "OS 2":
    print("no update needed")
elif system == "OS 1" or system == "OS 2":
    print("update needed")
```

Hint 1

Hint 2

Question 4

What do you observe about this conditional?

```
This updated conditional maintains the same functionality as the previous version, but with more streamlined
syntax. By utilizing the 'or' operator, we've combined two separate conditions into a single 'elif' statement.
```

In this next step, I'm tasked with creating a login verification system for a specific device. I'll start by setting up two variables for approved users and another for the current login attempt. My goal is to write a conditional statement that compares the username of the person trying to log in against the list of authorized users. This will help determine whether access should be granted or denied based on the user's credentials.

In [27]:
```
# Assign `approved_user1` and `approved_user2` to usernames of approved users

approved_user1 = "elarson"
approved_user2 = "bmoreno"

# Assign `username` to the username of a specific user trying to log in

username = "bmoreno"

# If the user trying to log in is among the approved users, then display a message that they are approved to access
# Otherwise, display a message that they do not have access to this device

if username == approved_user1 or approved_user2:
    print("This user has access to this device.")
else:
    print("This user does not have access to this device.")
```

```
This user has access to this device.
```

In this task, I'll upgrade the login system to handle five approved users using an allow list called `approved_list`. I'll employ the `in` operator to check if a username exists in this list. My code will display `This user has access to this device` for approved users and `This user does not have access to this device` for others. After implementing this, I'll test the system with both approved and unapproved usernames to verify its functionality.

```
In [31]: # Assign `approved_list` to a list of approved usernames

         approved_list = ["elarson", "bmoreno", "tshah", "sgilmore", "eraab"]

         # Assign `username` to the username of a specific user trying to log in

         username = "elarson"

         # If the user trying to log in is among the approved users, then display a message that they are approved to access t
         # Otherwise, display a message that they do not have access to this device

         if username in approved_list:
             print ("This user has access to this device.")
         else:
             print ("This user does not have access to this device")
```

```
This user has access to this device.
```

In this task, I'll create a conditional statement to monitor login times using a Boolean variable `organization_hours`. The system will display `Login attempt made during organization hours` when `organization_hours` is True, and `Login attempt made outside of organization hours` when it's False. This will help track whether users are accessing the system within or outside of designated timeframes.

```
In [31]: # Assign `approved_list` to a list of approved usernames

         approved_list = ["elarson", "bmoreno", "tshah", "sgilmore", "eraab"]

         # Assign `username` to the username of a specific user trying to log in

         username = "elarson"

         # If the user trying to log in is among the approved users, then display a message that they are approved to access t
         # Otherwise, display a message that they do not have access to this device

         if username in approved_list:
             print ("This user has access to this device.")
         else:
             print ("This user does not have access to this device")
```

```
This user has access to this device.
```

Hint 1

Hint 2

Use the `print()` function to display messages.

Question 5

What happens when an approved user tries to log in? What happens when an unapproved user tries to log in?

```
The login system checks user credentials against an approved list, displaying "This user has access to this device"
for authorized users and "This user does not have access to this device" for unauthorized attempts.
```

In this task, I'll run a cell containing combined code from previous exercises. This code includes conditionals for checking user authorization and login timing. I'll test various combinations of `username` and `organization_hours` values to observe how different inputs affect the system's output.

In [41]:
```
# Assign `approved_list` to a list of approved usernames

approved_list = ["elarson", "bmoreno", "tshah", "sgilmore", "eraab"]

# Assign `username` to the username of a specific user trying to log in

username = "elarson"

# If the user trying to log in is among the approved users, then display a message that they are approved to access
# Otherwise, display a message that they do not have access to this device

if username in approved_list:
    print("This user has access to this device.")

else:
    print("This user does not have access to this device.")

# Assign `organization_hours` to a Boolean value that represents whether the user is trying to log in during organiz

organization_hours = False

# If the entered `organization_hours` has a value of True, then display "Login attempt made during organization hour
# Otherwise, display "Login attempt made outside of organization hours."

if organization_hours == True:
    print("Login attempt made during organization hours.")
else:
    print("Login attempt made outside of organization hours.")
```

```
This user has access to this device.
Login attempt made outside of organization hours.
```

In this task, I'll combine two separate conditions into a single conditional statement using a logical operator. This will streamline the code to provide a unified message about the login attempt. I'll insert the appropriate operator where indicated, then test the updated code with various input combinations to observe its behavior.

In [42]:
```
# Assign `approved_list` to a list of approved usernames

approved_list = ["elarson", "bmoreno", "tshah", "sgilmore", "eraab"]

# Assign `username` to the username of a specific user trying to log in

username = "bmoreno"

# Assign `organization_hours` to a Boolean value that represents whether the user is trying to log in during organiz

organization_hours = True

# If the user is among the approved users and they are logging in during organization hours, then convey that the us
# Otherwise, convey that either the username is not approved or the login attempt was made outside of organization h

if username in approved_list and organization_hours == True:
    print("Login attempt made by an approved user during organization hours.")
else:
    print("Username not approved or login attempt made outside of organization hours.")
```

```
Login attempt made by an approved user during organization hours.
```

Hint 1

Hint 2

Question 8

In this setup, what happens when the user trying to log in is an approved user and doing so during organization hours? What happens when the user either is not approved or attempts to log in outside of organization hours?

```
This code combines user approval and login timing checks into a single conditional statement. It displays a
specific message for approved users logging in during organization hours, and a different message if either
condition is not met. This approach streamlines the previous code while maintaining the same functionality.
```

Conclusion

****What are your key takeaways from this lab?****

1)Python allows combining multiple conditions using logical operators (and, or)
2)Logical operators help create more concise and readable code
3)Access control involves checking multiple criteria (user approval, timing)
4)Boolean variables are useful for tracking binary states like authorization