LAB_Activity_CreateAConditionalStatement

February 20, 2024

1 Activity: Create a conditional statement

1.1 Introduction

Conditional statements are a powerful structure that help in achieving automation when you need to make sure conditions are met before certain actions are executed. For example, security analysts can use conditional statements in Python to check if users are approved to access a device.

In this lab, you will practice writing conditional statements in Python.

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're working as a security analyst. First, you are responsible for checking whether a user's operating system requires an update. Then, you need to investigate login attempts to a specific device. You must determine if login attempts were made by users approved to access this device and if the login attempts occurred during organization hours.

1.3 Task 1

You are asked to help automate the process of checking whether a user's operating system requires an update. Imagine that a user's device can be running one of the following operating systems: OS 1, OS 2, or OS 3. While OS 2 is up-to-date, OS 1 and OS 3 are not. Your task is to check whether the user's system is up-to-date, and if it is, display a message accordingly. To do this, complete

the conditional statement using the keyword `if`. Be sure to replace the `### YOUR CODE HERE ###` with your own code before you run the following cell.

[2]:
```
# Assuming user_os contains the user's operating system (replace it with the␣
 ↪actual variable)
user_os = "OS 2"  # You should replace this value with the actual user's␣
 ↪operating system

# Checking whether the user's operating system is up-to-date
if user_os == "OS 2":
    print("Your operating system is up-to-date.")
else:
    print("Your operating system requires an update.")
```

```
Your operating system is up-to-date.
```

Hint 1

Use a comparison operator that will allow you to check whether the `system` is running `"OS 2"`.

Hint 2

Use the `==` comparison operator to check whether the `system` is running `"OS 2"`.

1.4 Task 2

Now try assigning the `system` variable to different values (`"OS 1"`, `"OS 2"`, and `"OS 3"`), run the cell, and observe what happens. Keep the conditional statement as is. Be sure to replace the `### YOUR CODE HERE ###` with your own code.

[4]:
```
# Assigning the system variable to different values and observing the output
user_os = "OS 1"
if user_os == "OS 2":
    print("Your operating system is up-to-date.")
else:
    print("Your operating system requires an update.")

user_os = "OS 2"
if user_os == "OS 2":
    print("Your operating system is up-to-date.")
else:
    print("Your operating system requires an update.")

user_os = "OS 3"
if user_os == "OS 2":
    print("Your operating system is up-to-date.")
else:
    print("Your operating system requires an update.")
```

```
Your operating system requires an update.
Your operating system is up-to-date.
Your operating system requires an update.
```

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

When `user_os` is set to "OS 2," the condition `user_os == "OS 2"` is `True`, so the code block under the `if` statement is executed. In this case, the message "Your operating system is up-to-date." is printed.

When `user_os` is set to "OS 1," the condition `user_os == "OS 2"` is `False`, so the code block under the `else` statement is executed. In this case, the message "Your operating system requires an update." is printed.

To summarize:

- `user_os = "OS 2"`: Output is "Your operating system is up-to-date."
- `user_os = "OS 1"`: Output is "Your operating system requires an update."

This behavior demonstrates the basic use of conditional statements to check different conditions and execute code accordingly.

1.5 Task 3

Nothing is displayed when the `system` is not equal to `"OS 2"`. This is because the condition didn't evaluate to `True`.

It would be beneficial if an alternative message is provided to them when updates are needed.

In the following cell, add the appropriate keyword after the first conditional so that it will display a message that conveys that an update is needed when the `system` is not running OS 2. Be sure to replace each `### YOUR CODE HERE ###` with your own code.

Then, set the value of the `system` variable to indicate that OS 2 is running and run the cell. After observing what happens, set the value of `system` to indicate either that OS 1 is running or that OS 3 is running and run the cell.

[5]:
```
# Assigning `system` to a specific operating system
# This variable represents which operating system is running

system = "OS 1"  # Replace this value with the actual operating system

# 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")
```

```
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?

When system is set to any value other than "OS 2" (for example, "OS 1" or "OS 3"), the condition system == "OS 2" is False, so the code block under the else statement is executed. In this case, it prints "update needed." This setup checks whether the operating system is "OS 2" and provides different messages based on the condition status. If "OS 2" is running, it says "no update needed"; otherwise, it says "update needed." for the machine to be updated. So therefore there needs to be an update.

1.6 Task 4

This setup is still not ideal. If the variable `system` contains a random string or integer, the conditional above would still display `update needed`.

To improve the conditional, you will need to add the `elif` keyword. In the following cell, you will add two `elif` statements after the `if` statement, to create the final code. The first `elif` statement will display `update needed` if `system` is `"OS 1"`. The second `elif` statement will display the same message, if `system` is `"OS 3"`. Complete the second `elif` statement, and then run the cell with the variable `system` set to a different string each time. Observe what happens when each operating system is running. Also try assigning the `system` variable to some strings other than `"OS 1"`, `"OS 2"`, and `"OS 3"` (for example `"OS 4"`).

Be sure to replace each `### YOUR CODE HERE ###` with your own code.

[6]:
```
# Assigning `system` to a specific operating system
# This variable represents which operating system is running

system = "OS 1"  # Replace this value with the actual operating system

# Checking the operating system and displaying the appropriate message
if system == "OS 2":
    print("no update needed")
elif system == "OS 1":
    print("update needed for OS 1")
elif system == "OS 3":
    print("update needed for OS 3")
else:
    print("unknown operating system, unable to determine update status")
```

```
update needed for OS 1
```

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?

In summary this is what i managed to find out when running the code for any of the operating systems.

1. **When `system` is set to "OS 2":**
   - Output: "no update needed"
   - Explanation: The first condition is met (`system == "OS 2"`), indicating that the operating system is up-to-date. The message "no update needed" is printed.
2. **When `system` is set to "OS 1":**
   - Output: "update needed for OS 1"
   - Explanation: The second condition is met (`system == "OS 1"`), indicating that an update is needed for OS 1. The message "update needed for OS 1" is printed.
3. **When `system` is set to "OS 3":**
   - Output: "update needed for OS 3"
   - Explanation: The third condition is met (`system == "OS 3"`), indicating that an update is needed for OS 3. The message "update needed for OS 3" is printed.
4. **When `system` is set to a value other than "OS 1," "OS 2," or "OS 3" (e.g., "OS 4"):**
   - Output: "unknown operating system, unable to determine update status"
   - Explanation: None of the specific operating system conditions are met, so it falls into the `else` block, indicating that the operating system is unknown. The message "unknown operating system, unable to determine update status" is printed for the display output.

1.7 Task 5

Writing code that is readable and concise is a best practice in programming.

The conditional above can be written more concisely.

In the following cell, use a logical operator to combine the two `elif` statements from the previous setup into one `elif` statement. Be sure to replace each `### YOUR CODE HERE ###`. Then, assign the `system` variable to a value and run the cell. Like you did in the previous task, use `"OS 1"`, `"OS 2"`, `"OS 3"`, and other strings.

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

system = "OS 1"  # Replace this value with the actual operating system

# Checking the operating system and displaying the appropriate message
if system == "OS 2":
    print("no update needed")
elif system == "OS 1" or system == "OS 3":
    print(f"update needed for {system}")
else:
    print("unknown operating system, unable to determine update status")
```

`update needed for OS 1`

Hint 1

Use the `or` logical operator.

Hint 2

Use the `or` operator between two conditions that each use the `==` operator.

Question 4 What do you observe about this conditional?

In this conditional issue, the logical OR (`or`) operator is used to combine the conditions for "OS 1" and "OS 3" into a single `elif` statement. The modified code is more concise compared to having two separate `elif` statements for each operating system.

Here's what you observe about this conditional:

1. **Conciseness:** The use of the logical OR operator allows you to express the condition for "OS 1" or "OS 3" in a more compact form.
2. **Readability:** The code is still readable, and the intent is clear. It checks whether the `system` is "OS 2," "OS 1," or "OS 3" and prints the corresponding message.
3. **Efficiency:** Combining conditions using logical operators can improve efficiency by evaluating only one condition when either part of the logical OR is true.

Overall, the modified code is a good example of writing concise and readable code while maintaining the logical structure of the conditional statements writing the code.

1.8 Task 6

Now you'll move on to the next part of your work. You've been asked to investigate login attempts to a specific device. Only approved users should log on to this device.

You'll start with two authorized users, stored in the variables `approved_user1` and `approved_user2`. You'll need to write a conditional statement that compares those variables to a third variable, `username`. This will be the username of a specific user trying to log in. Be sure to replace each `### YOUR CODE HERE ###` with your own code.

[9]:
```
# Assuming approved_user1 and approved_user2 are predefined
approved_user1 = "john_doe"
approved_user2 = "jane_smith"

# Variable representing the username of a specific user trying to log in
username = "jane_smith"  # Replace this value with the actual username

# Checking if the username is an approved user
if username == approved_user1 or username == approved_user2:
    print(f"Welcome, {username}! You are an authorized user.")
else:
    print(f"Access denied. {username} is not an authorized user.")
```

`Welcome, jane_smith! You are an authorized user.`

Hint 1

Use the `if` keyword in the first conditional statement and the `else` keyword in the second conditional statement. Make sure both statements end with the proper syntax of a colon (`:`).

Hint 2

Use a comparison operator that will allow you to check whether the user trying to log in is an approved user.

Hint 3

Use the `==` comparison operator to check whether the user trying to log in is an approved user.

1.9 Task 7

The number of approved users has now expanded to five. Rather than storing each of the approved users' usernames individually, it would be more concise to store them in an allow list called `approved_list`.

The `in` operator in Python can be used to determine whether a given value is an element of a sequence. Using the `in` operator in a condition can help you check whether a specific username is part of a list of approved usernames. For example, in the code below, `username in approved_list` evaluates to `True` if the value of the `username` variable is included in `approved_list`.

Complete the code in the following cell to display the same messages that you used in the previous step. When the condition evaluates to `True`, the following message will be displayed: `"This user has access to this device."` When it evaluates to `False`, the following message will be displayed: `"This user does not have access to this device."` Then, run the cell to observe its behavior. Be sure to replace each `### YOUR CODE HERE ###` with your own code. Afterwards, reassign the `username` variable to a username that is not approved and run the cell to observe what happens.

[10]:
```
# 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"

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

if username in approved_list:
    print(f"{username} is approved to access this device.")
else:
    print(f"{username} does not have access to this device.")
```

`bmoreno is approved to access this device.`

Hint 1

Use the `else` keyword in the second conditional statement.

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?

When an approved user tries to log in (e.g., `username = "bmoreno"`), the condition `if username in approved_list` evaluates to `True`, and the code block under the `if` statement is executed. The message "bmoreno is approved to access this device." is printed.

When an unapproved user tries to log in with a username not in the `approved_list` (e.g., `username = "non_approved_user"`), the condition `if username in approved_list` evaluates to `False`, and the code block under the `else` statement is executed. The message "non_approved_user does not have access to this device." is printed.

In summary:

1. **Approved User (e.g., `username = "bmoreno"`):**
   - Output: "bmoreno is approved to access this device."
2. **Unapproved User (e.g., `username = "non_approved_user"`):**
   - Output: "non_approved_user does not have access to this device."

1.10 Task 8

Now you'll write another conditional statement. This one will use a `organization_hours` variable to check if the user logged in during specific organization hours. When that condition is met, the code should display the string `"Login attempt made during organization hours."`. When that condition isn't met, the code should display the string `"Login attempt made outside of organization hours."`.

The `organization_hours` variable will have a Boolean data type. If `organization_hours` has a Boolean value of `True`, that means the user is logged in during the specified organization hours. If `organization_hours` has a Boolean value of `False`, that means the user is not logged in during those hours.

Complete the conditional in the following cell. Be sure to replace each `### YOUR CODE HERE ###` with your own code before running the following cell.

[11]:
```
# Assign `organization_hours` to a Boolean value that represents whether the␣
↪user is trying to log in during organization hours

organization_hours = True
```

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

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

```
Login attempt made during organization hours.
```

Hint 1

Use the `==` comparison operator to check whether the user is logged in during the specified organization hours. Compare `organization_hours` to the appropriate Boolean value.

Hint 2

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

Hint 3

Use the `else` keyword in the second conditional statement.

Question 6 What happens when the user logs in during organization hours? What happens when they log in outside of organization hours?

When the user logs in during organization hours (when `organization_hours` is `True`), the condition `if organization_hours:` evaluates to `True`, and the code block under the `if` statement is executed. The message "Login attempt made during organization hours." is printed.

When the user logs in outside of organization hours (when `organization_hours` is `False`), the condition `if organization_hours:` evaluates to `False`, and the code block under the `else` statement is executed. The message "Login attempt made outside of organization hours." is printed.

In summary:

1. **User logs in during organization hours (e.g., `organization_hours = True`):**
   - Output: "Login attempt made during organization hours."
2. **User logs in outside of organization hours (e.g., `organization_hours = False`):**
   - Output: "Login attempt made outside of organization hours."

1.11 Task 9

The following cell assembles the code from the previous tasks. It includes the conditional statement that checks if a user is on the allow list and the conditional statement that checks if the user logged in during organization hours.

Run the cell below a few times. Each time, enter a different combination of values for `username` and `organization_hours` to observe how that affects the output.

[16]:
```
# 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"

# If the user trying to log in is among the approved users, then display a␣
 ↪message that they are approved to access this device
# 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 organization hours

organization_hours = True

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

if organization_hours:
    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 during organization hours.
```

Question 7 What happens when the user trying to log in is not among the approved users? What happens when the user trying to log in is among the approved users? What happens when the user tries to log in outside of organization hours?

Based on my analysis of this code this is what i found:

1. **User trying to log in is not among the approved users (e.g., `username = "non_approved_user"`):**
   - Output: "This user does not have access to this device."
   - Explanation: The first conditional statement checks whether the `username` is in the `approved_list`. If the user is not in the list, it prints "This user does not have access to this device."
2. **User trying to log in is among the approved users (e.g., `username = "bmoreno"`):**
   - Output: "This user has access to this device."

- Explanation: The first conditional statement checks whether the `username` is in the `approved_list`. If the user is in the list, it prints "This user has access to this device."

3. **User trying to log in outside of organization hours (e.g., `organization_hours = False`):**
   - Output: "Login attempt made outside of organization hours."
   - Explanation: The second conditional statement checks whether `organization_hours` is `True`. If it is `False`, it prints "Login attempt made outside of organization hours."

In conclusion, summary:

- If the user is not among the approved users, they do not have access to the device.
- If the user is among the approved users, they have access to the device.
- If the user tries to log in outside of organization hours, it is considered outside of organization hours.

1.12 Task 10

You can also provide a single message about the login attempt. To do this, you can join both conditions into a single conditional statement using a logical operator. This will make the code more concise.

Examine the code in the following cell and add the missing operator that would allow for a single message. Be sure to replace each `### YOUR CODE HERE ###` with your own code before running the following cell. Then run the cell, entering different combinations of information, and observe what happens.

[18]:
```
# 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 organization hours

organization_hours = True

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

if username in approved_list and organization_hours:
    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

Use the logical operator that would allow you to check both conditions in the `if` statement (the condition that the entered username is in the approved list and the condition that the login attempt is occurring during organization hours).

Hint 2

Use the `and` logical operator to check both conditions in the `if` statement (the condition that the entered username is in the approved list and the condition that the login attempt is occurring during organization hours).

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?

In my analysis of this code this is what i found while running my code for trial and error:

1. **User trying to log in is an approved user and doing so during organization hours (e.g., `username = "bmoreno"`, `organization_hours = True`):**
   - Output: "Login attempt made by an approved user during organization hours."
   - Explanation: The combined condition `username in approved_list and organization_hours` evaluates to `True`, and the code block under the `if` statement is executed. It prints "Login attempt made by an approved user during organization hours."
2. **User is either not approved or attempts to log in outside of organization hours (e.g., `username = "non_approved_user"`, `organization_hours = False`):**
   - Output: "Username not approved or login attempt made outside of organization hours."
   - Explanation: The combined condition `username in approved_list and organization_hours` evaluates to `False`, and the code block under the `else` statement is executed. It prints "Username not approved or login attempt made outside of organization hours."

In conclusion, summary:

- If the user is an approved user and logs in during organization hours, they will receive a message indicating a successful login during organization hours.
- If the user is either not approved or attempts to log in outside of organization hours, they receive a message indicating that the username is not approved or the login attempt was made outside of organization hours to be denied access.

1.13 Conclusion

What are your key takeaways from this lab?

The key takeaways from this lab include:

1. **Conditional Statements:** Understanding how to use conditional statements (`if`, `else`, `elif`) is crucial for implementing logic in Python programs. Conditional statements allow the program to make decisions based on specified conditions.
2. **Logical Operators:** Logical operators such as `and`, `or`, and `not` can be used to combine multiple conditions in conditional statements, providing flexibility in expressing complex logic.
3. **Variable Types:** Being aware of different variable types, such as Boolean, allows for effective representation of conditions. Booleans are particularly useful in expressing true/false conditions.
4. **List Operations:** Utilizing lists to store and manage collections of data, such as an approved list of usernames, enhances the organization and efficiency of the code.
5. **User Input Handling:** Incorporating user input into the program allows for dynamic and interactive behavior. The program can respond differently based on the input provided by the user.
6. **Concise Code:** Writing concise and readable code is a best practice in programming. Logical operators and efficient use of conditional statements contribute to more concise and clear code.
7. **Testing and Observing Output:** Running the code with different input values allows for testing and observing how the program behaves under various conditions. This iterative process helps in understanding and debugging the code.

Overall, this lab provides practical experience in implementing conditional logic, handling user input, and making decisions based on specific criteria. These skills are fundamental in programming and are widely applicable in various contexts to help improve my skills stronger to better elevate my knowledge deeper into mastering into the IT industry for technical fundamental skills.

[ ]:

[ ]: