Activity_Develop an algorithm

January 9, 2024

1 Activity: Develop an algorithm

1.1 Introduction

An algorithm is a set of steps that can be used to solve a problem. Security analysts develop algorithms to provide the solutions that they need for their work. For example, an analyst may work with users who bring them devices. The analyst may need an algorithm that first checks if a user is approved to access the system and then checks if the device that they have brought is the one assigned to them.

In this lab, you'll develop an algorithm in Python that automates this process.

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

In this lab, you're working as a security analyst and you're responsible for developing an algorithm that connects users to their assigned devices. You'll write code that indicates if a user is approved on the system and has brought their assigned device to the security team.

1.3 Task 1

You'll work with a list of approved usernames along with a list of the approved devices assigned to these users. The elements of the two lists are synchronized. In other words, the user at index `0` in `approved_users` uses the device at index `0` in `approved_devices`. Later, this will allow you to verify if the username and device ID entered by a user correspond to each other.

First, to explore how indices in lists work, run the following code cell as is and observe the output. Then, replace each `0` with another index and run the cell to observe what happens.

[1]:
```
# Assign `approved_users` to a list of approved usernames

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

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

approved_devices = ["8rp2k75", "hl0s5o1", "2ye3lzg", "4n482ts", "a307vir"]

# Display the element at the specified index in `approved_users`

print(approved_users[0])

# Display the element at the specified index in `approved_devices`

print(approved_devices[0])
```

```
elarson
8rp2k75
```

Question 1 What did you observe about the output when `approved_users[0]` is displayed and when `approved_devices[0]` is displayed? What happens when you replace each `0` with another index?

index outputs the correct element in the list

1.4 Task 2

There's a new employee joining the organization, and they need to be provided with a username and device ID. In the following code cell, you are given a username and device ID of this new user, stored in the variables `new_user` and `new_device`, respectively. Use the `.append()` method to add these variables to the `approved_users` and `approved_devices` respectively. Afterwards, display the `approved_users` and `approved_devices` variables to confirm the added information. Be sure to replace each `### YOUR CODE HERE ###` with your own code before you run the following cell.

[1]:
```
 Assign `approved_users` to a list of approved usernames

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

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

approved_devices = ["8rp2k75", "hl0s5o1", "2ye3lzg", "4n482ts", "a307vir"]
```

```
# Assign `new_user` to the username of a new approved user

new_user = "gesparza"

# Assign `new_device` to the device ID of the new approved user

new_device = "3rcv4w6"

# Add that user's username and device ID to `approved_users` and␣
 ↪`approved_devices` respectively

approved_users.append(new_user)
approved_devices.append(new_device)

# Display the contents of `approved_users`

print(approved_users)

# Diplay the contents of `approved_devices`

print(approved_devices)
```

```
['elarson', 'bmoreno', 'tshah', 'sgilmore', 'eraab', 'gesparza']
['8rp2k75', 'hl0s5o1', '2ye3lzg', '4n482ts', 'a307vir', '3rcv4w6']
```

Hint 1

Use the `.append()` method to add `new_user` to `approved_users`.

Use the `.append()` method to add `new_device` to `approved_devices`.

Hint 2

Use the `print()` function to display the contents of `approved_users`.

Use the `print()` function to display the contents of `approved_devices`.

Question 2 After the new approved user is added, what did you observe about the output when `approved_users` is displayed and when `approved_devices` is displayed?

The new user and device got appended totheir particular lists.

1.5 Task 3

An employee has left the team and should no longer have access to the system. In the following code cell, you are given the username and device ID of the user to be removed, stored in the variables `removed_user` and `removed_device` respectively. Use the `.remove()` method to remove each of these elements from the corresponding list. Afterwards, display both the `approved_users` and the `approved_devices` variables to view the removed users. Run the code and observe the results. Be

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

[2]:
```
# Assign `approved_users` to a list of approved usernames

approved_users = ["elarson", "bmoreno", "tshah", "sgilmore", "eraab",␣
↪"gesparza"]

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

approved_devices = ["8rp2k75", "hl0s5o1", "2ye3lzg", "4n482ts", "a307vir",␣
↪"3rcv4w6"]

# Assign `removed_user` to the username of the employee who has left the team

removed_user = "tshah"

# Assign `removed_device` to the device ID of the employee who has left the team

removed_device = "2ye3lzg"

# Remove that employee's username and device ID from `approved_users` and␣
↪`approved_devices` respectively

approved_users.remove(removed_user)
approved_devices.remove(removed_device)

# Display `approved_users`

print(approved_users)

# Diplay `approved_devices`

print(approved_devices)
```

```
['elarson', 'bmoreno', 'sgilmore', 'eraab', 'gesparza']
['8rp2k75', 'hl0s5o1', '4n482ts', 'a307vir', '3rcv4w6']
```

Hint 1

Use the `.remove()` method to remove `removed_user` from `approved_users`.

Use the `.remove()` method to remove `removed_device` from `approved_devices`.

Hint 2

Use the `print()` function to display the contents of `approved_users`.

Use the `print()` function to display the contents of `approved_devices`.

Question 3 After the user who left the team is removed, what did you observe about the output when `approved_users` is displayed and when `approved_devices` is displayed?

Both lists are updated and dont contain the remove item

1.6 Task 4

As part of verifying a user's identity in the system, you'll need to check if the user is one of the approved users. Write a conditional statement that verifies if a given username is an element of the list of approved usernames. If it is, display `"The user ______ is approved to access the system."`. Otherwise, display `"The user ______ is not approved to access the system."`. Be sure to replace each `### YOUR CODE HERE ###` with your own code before you run the following cell.

[3]:
```
# 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"]

# Assign `username` to a username

username = "sgilmore"

# Conditional statement
# If `username` belongs to `approved_users`, then display "The user ______ is␣
 ↪approved to access the system."
# Otherwise display "The user ______ is not approved to access the system."
if username in approved_users:
    print("The username", username, "is approved to access the system.")
else:
    print(f'The user {username} is not approved to access the system.')
```

```
The username sgilmore is approved to access the system.
```

Hint 1

In the `if` condition, be sure to check if `username` belongs to `approved_users`.

Hint 2

After the `if` statement, use the `else` keyword to create an `else` statement that handles the case when `username` is not part of the `approved_users`.

Hint 3

Inside the `else` statement, use the `print()` function to display the message `"The user ______ is not approved to access the system."`.

Refer to the `print()` function call in the `if` statement and observe how commas separate a string containing the first part of the message, the `username` variable, and another string containing the second part of the message.

Question 4 What message do you observe in the output when `username` is `"sgilmore"`?

The user sgijlmore has been granted system access privileges since their username is present on the authorized user list.

1.7 Task 5

The next part of the algorithm uses the `.index()` method to find the index of `username` in the `approved_users` and store that index in a variable named `ind`.

When used on a list, the `.index()` method will return the position of the given value in the list.

Add a statement to display `ind` in the following code cell to explore the value it contains. Be sure to replace the `### YOUR CODE HERE ###` with your own code before you run the following cell.

[4]:
```
# 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"]

# Assign `username` to a username

username = "sgilmore"

# Assign `ind` to the index of `username` in `approved_users`

ind = approved_users.index(username)

# Display the value of `ind`

print(ind)
```

2

Hint 1

Use the `print()` function to display the value of `ind`.

Question 5 What do you observe from the output when `username` is `"sgilmore"`?

it outputs sgilmore index which is at index 2 the third place

1.8 Task 6

This task will allow you to build your understanding of list operations for the algorithm that you'll eventually build. It will demonstrate how you can find an index in one list and then use this index to display connected information in another list. First, use the `.index()` method again to find the index of `username` in the `approved_users` and store that in a variable named `ind`. Then, connect `ind` to the `approved_devices` and display the device ID located at the index `ind`. Afterwards, run the cell to observe the result. Be sure to replace each `### YOUR CODE HERE ###` with your own code before you run the following cell.

[5]:
```
# 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"]

# Assign `username` to a username

username = "sgilmore"

# Assign `ind` to the index of `username` in `approved_users`

ind = approved_users.index(username)

# Display the device ID at the index that matches the value of `ind` in␣
↪`approved_devices`

print(approved_devices[ind])
```

```
4n482ts
```

Hint 1

Use the `.index()` method to get the index value of the `username` in the `approved_users`. Assign `ind` to the result.

Hint 2

To display the correct device ID from `approved_devices`, use `ind` as the index. Place `ind` inside the square brackets to extract the correct element from `approved_devices`.

Question 6 What do you observe from the output when `username` is `"sgilmore"`?

The output displays the approved device that corresponds to the "sgilmore" user, matching on the 3rd element in each of the lists, indicating that this specific device is approved for this specific user

account based on the parallel positions of their names in the respective lists.

1.9 Task 7

Your next step in creating the algorithm is to determine if a username and device ID correspond. To do this, write a conditional that checks if the `username` is an element of the `approved_devices` and if the `device_id` stored at the same index as `username` matches the `device_id` entered. You'll use the logical operator `and` to connect the two conditions. When both conditions evaluate to `True`, display a message that the username is approved and another message that the user has their assigned device. Be sure to replace each `### YOUR CODE HERE ###` with your own code before you run the following cell.

[6]:
```
# 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"]

# Assign `username` to a username

username = "sgilmore"

# Assign `device_id` to a device ID

device_id = "4n482ts"

# Assign `ind` to the index of `username` in `approved_users`

ind = approved_users.index(username)

# Conditional statement
# If `username` belongs to `approved_users`, and if the device ID at `ind` in␣
 ↪`approved_devices` matches `device_id`,
# then display a message that the username is approved,
# followed by a message that the user has the correct device

if username in approved_users and device_id == approved_devices[ind]:
    print("The username", username, "is approved to access the system.")
    print(device_id, "is the assigned device for", username)
```

```
The username sgilmore is approved to access the system.
4n482ts is the assigned device for sgilmore
```

Hint 1

After the logical operator `and`, write the second condition in the `if` statement using a comparison operator to check whether the element at `ind` in `approved_devices` matches `device_id`.

Hint 2

Use the `==` comparison operator to check whether the element at `ind` in `approved_devices` matches `device_id`.

Question 7 What do you observe from the output when `username` is `"sgilmore"` and `device_id` is `"4n482ts"`?

The output shows that the "sgilmore" username and corresponding device are approved, as evidenced by the fact that both the username string and device name occupy the same index position (the 3rd element) in their respective lists - indicating an approval match based on parallel list alignment.

1.10 Task 8

It would also be helpful for users to receive messages when their username is not approved or their device ID is incorrect.

Add to the code by writing an `elif` statement. This `elif` statement should run when the `username` is part of the `approved_users` but the `device_id` doesn't match the corresponding device ID in the `approved_devices`. The statement should also display two messages conveying that information.

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

(After you run the code once with a `device_id` of `"4n482ts"`, you might want to explore what happens if you assign a different value to `device_id`.)

[7]:
```
# 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"]

# Assign `username` to a username

username = "sgilmore"

# Assign `device_id` to a device ID

device_id = "4n482ts2"

# Assign `ind` to the index of `username` in `approved_users`
```

```
ind = approved_users.index(username)

# If statement
# If `username` belongs to `approved_users`, and if the element at `ind` in␣
 ↪`approved_devices` matches `device_id`,
# then display a message that the username is approved,
# followed by a message that the user has the correct device

if username in approved_users and device_id == approved_devices[ind]:
    print("The user", username, "is approved to access the system.")
    print(device_id, "is the assigned device for", username)

# Elif statement
# Handles the case when `username` belongs to `approved_users` but element at␣
 ↪`ind` in `approved_devices` does not match `device_id`,
# and displays two messages accordingly

elif username in approved_users and device_id != approved_devices[ind]:
    print("The user", username, "is approved to access the system, but",␣
 ↪device_id, "is not their assigned device.")
```

```
The user sgilmore is approved to access the system, but 4n482ts2 is not their
assigned device.
```

Hint 1

In the `elif` statement, use the `in` operator to check whether `username` belongs to `approved_users`, use a comparison operator to check whether the element at `ind` in `approved_devices` doesn't match `device_id`, and use a logical operator to connect these two conditions to check whether both of them are met.

Hint 2

In the `elif` statement, use the `in` operator to check whether `username` belongs to `approved_users`, use the `!=` comparison operator to check whether the element at `ind` in `approved_devices` doesn't match `device_id`, and use the `and` logical operator to connect these two conditions to check whether both of them are met.

Question 8 What do you observe from the output when `username` is `"sgilmore"` and `device_id` is `"4n482ts"`?

The output first shows the match for "sgilmore" and their approved device. But changing the device_id produces an error instead, as the new ID now lacks the same index match across lists requisite for approval binding of a user and device.

1.11 Task 9

In this task, you'll complete your algorithm by developing a function that uses some of the code you've written in earlier tasks. This will automate the login process.

There are multiple ways to use conditionals to automate the login process. In the following code, a nested conditional is used to achieve the goals of the algorithm. There is a conditional statement inside of another conditional statement. The outer conditional handles the case when the `username` is approved and the case when `username` is not approved. The inner conditional, which is placed inside the first `if` statement, handles the case when the `username` is approved and the `device_id` is correct, as well as the case when the `username` is approved and the `device_id` is incorrect.

To complete this task, you must define a function named `login` that takes in two parameters, `username` and `device_id`. Afterwards, call the function and pass in different username and device ID combinations to experiment and observe the function's behavior. Be sure to replace the `### YOUR CODE HERE ###` with your own code before you run the following cell.

[8]:
```
# 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):

    # 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("elarson", "8rp2k75")
login("elarsonasdfasdfadf", "8rp2k75")
login("elarson", "8rp2k75234234234")
```

```
The user elarson is approved to access the system.
8rp2k75 is the assigned device for elarson
The username elarsonasdfasdfadf is not approved to access the system.
The user elarson is approved to access the system.
8rp2k75234234234 is not their assigned device.
```

Hint 1

Use the `def` keyword to start the function definition.

Hint 2

After the `def` keyword, specify the name of the function, followed by parantheses and a colon. Inside the parantheses, specify the parameters that the function takes in.

To call the function, write the name of the function, followed by parantheses, and pass in the username and device ID that you want to experiment with.

Hint 3

After the `def` keyword, write `login(username, device_id):` to complete the function definition header.

To call the function, write `login()`, and pass in the username and device ID that you want to experiment with, separated by a comma. Keep in mind that the arguments you pass in are string data.

Question 9 After Python enters the inner conditional, what happens when the `device_id` is correct, and what happens when the `device_id` is incorrect?

If the device ID is correct, it prints approval as the user and device match properly. If incorrect, it raises an error saying that user isn't approved for that unmatched device ID based on the index positions not aligning between the separate account and device lists.

1.12 Conclusion

What are your key takeaways from this lab?

- Developing algorithms requires considering many possible use cases
- Python fundamentals like lists vs tuples suit different data needs
- Analyze edge cases thoroughly to build robust, comprehensive logic

[ ]: