Activity_Develop an algorithm

April 11, 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.

1.2 Scenario

In this lab, I'm working as a security analyst and am responsible for developing an algorithm that connects users to their assigned devices. I'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

I'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 me to verify if the username and device ID entered by a user correspond to each other.

[2]:
```
# 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[2])
```

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

print(approved_devices[2])
```

```
tshah
2ye3lzg
```

1.4 Task

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, I am given a username and device ID of this new user, stored in the variables `new_user` and `new_device`, respectively. I'll use the `.append()` method to add these variables to the `approved_users` and `approved_devices` respectively.

[4]:
```
# 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']
```

1.5 Task

An employee has left the team and should no longer have access to the system. In the following code cell, I am given the username and device ID of the user to be removed, stored in the variables `removed_user` and `removed_device` respectively. I'll use the `.remove()` method to remove each of these elements from the corresponding list.

[6]:
```
# 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']
```

1.6 Task

As part of verifying a user's identity in the system, I'll need to check if the user is one of the approved users. I'll 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."`.

[10]:
```
# 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 = "dabaly"

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

```
The username dabaly is not approved to access the system.
```

1.7 Task

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.

[11]:
```
# 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
```

1.8 Task

This task is to help me to build my understanding of list operations for the algorithm that I'll eventually build. It will demonstrate how one can find an index in one list and then use this index to display connected information in another list. First, I'll use the `.index()` method again to find the index of `username` in the `approved_users` and store that in a variable named `ind`. Then, I'll connect `ind` to the `approved_devices` and display the device ID located at the index `ind`.

[13]:
```
# 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 = "eraab"

# 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])
```

```
a307vir
```

1.9 Task

My next step in creating the algorithm is to determine if a username and device ID correspond. To do this, I'll 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. I'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.

[16]:
```
# Assign `approved_users` to a list of approved usernames

approved_users = ["elarson", "bmoreno", "dabaly", "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 = "dabaly"

# 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 in approved_devices:
    print("The username", username, "is approved to access the system.")
    print(device_id, "is the assigned device for", username)
```

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

1.10 Task

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

[21]:
```
# Assign `approved_users` to a list of approved usernames

approved_users = ["elarson", "bmoreno", "dabaly", "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 = "dabaly"

# Assign `device_id` to a device ID

device_id = "randomdeviceid"

# 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 dabaly is approved to access the system, but randomdeviceid is not
their assigned device.
```

1.11 Task

In this task, I'll complete my algorithm by developing a function that uses some of the code 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, I must define a function named `login` that takes in two parameters, `username` and `device_id`.

[25]:
```
# 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("dabaly", "randomid")
login("sgilmore", "n482ts")
login("sgilmore", "4n482ts")
```

```
The username dabaly is not approved to access the system.
The user sgilmore is approved to access the system.
n482ts is not their assigned device.
The user sgilmore is approved to access the system.
4n482ts is the assigned device for sgilmore
```

1.12 Conclusion

What are your key takeaways from this lab?

[Indexing a list is similar to indexing a string. Index values start at 0. The .append() method helps one add new elements to the end of lists. The .remove() method helps one remove elements from lists. The .index() method can be used on different types of sequences. They can be used not only with strings, but also with lists. With a list, the .index() method allows one to identify the position where a specified element is located in that list. If two lists contain information that correspond to each other in a specific order, one can use indices to pair elements from the lists together(concatenate). Functions can be used to develop algorithms. When defining a function, you must specify the parameters it takes in and the actions it should execute.]