LAB_014_Activity

April 11, 2024

1 Activity: Debug Python Code

1.1 Introduction

One of the biggest challenges faced by analysts is ensuring that automated processes run smoothly. Debugging is an important practice that security analysts incorporate in their work to identify errors in code and resolve them so that the code achieves the desired outcome.

1.2 Scenario

The codecells in this lab contained errors which I identified and fixed hence the completed lab does not have any errors.

1.3 Task

The following code cell contained a syntax error. In this task, I ran the code, identified why the error is occuring, and modified the code to resolve it.

[2]:
```python
# For loop that iterates over a range of numbers
# and displays a message each iteration

for i in range(10):
    print("Connection cannot be established")
```

```
Connection cannot be established
Connection cannot be established
Connection cannot be established
Connection cannot be established
Connection cannot be established
Connection cannot be established
Connection cannot be established
Connection cannot be established
Connection cannot be established
Connection cannot be established
```

1.4 Task

In the following code cell, I was provided a list of usernames. There was an issue with the syntax. I ran the cell, observed what happens, and modified the code to fix the issue.

[5]:
```
# Assign `usernames_list` to a list of usernames

usernames_list = ["djames", "jpark", "tbailey", "zdutchma", "esmith",␣
 ↪"srobinso", "dcoleman", "fbautist"]

# Display `usernames_list`

print(usernames_list)
```

```
['djames', 'jpark', 'tbailey', 'zdutchma', 'esmith', 'srobinso', 'dcoleman',
'fbautist']
```

1.5 Task

In the following code cell, there was a syntax error. I ran the cell, identified what was causing the error, and fixed it.

[8]:
```
# Display a message in upper case

print("update needed".upper())
```

```
UPDATE NEEDED
```

1.6 Task

In the following code cell, I was provided a `usernames_list`, a `username`, and code that determines whether the username is approved. There were two syntax errors and one exception. My task was to find them and fix the code. A helpful debugging strategy I used was to focus on one error at a time and ran the code after fixing each one.

[12]:
```
# Assign `usernames_list` to a list of usernames that represent approved users

usernames_list = ["djames", "jpark", "tbailey", "zdutchma", "esmith",␣
 ↪"srobinso", "dcoleman", "fbautist"]

# Assign `username` to a specific username

username = "dabaly"

# For loop that iterates over the elements of `usernames_list` and determines␣
 ↪whether each element corresponds to an approved user
```

```
for name in usernames_list:

    # Check if `name` matches `username`
    # If it does match, then display a message accordingly

    if name == username:
        print("The user is an approved user")
else:
    print("The user is not an approved user")
```

```
The user is not an approved user
```

1.7 Task

In this task, I examined the following code and identified the type of error that occured. Then, I adjusted the code to fix the error.

[13]:
```
# Assign `usernames_list` to a list of usernames

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

# Assign `username` to a specific username

username = "eraab"

# Determine whether `username` is the final username in `usernames_list`
# If it is, then display a message accordingly

if username == usernames_list[4]:
    print("This username is the final one in the list.")
```

```
This username is the final one in the list.
```

1.8 Task

In this task, I examined the following code. The code imports a text file into Python, reads its contents, and stores the contents as a list in a variable named `ip_addresses`. It then removes elements from `ip_addresses` if they are in `remove_list`. There were two errors in the code: first a syntax error and then an exception related to a string method. My goal was to find these errors and fix them.

[14]:
```
# Assign `import_file` to the name of the text file

import_file = "allow_list.txt"
```

```
# Assign `remove_list` to a list of IP addressess that are no longer allowed to␣
↪access the network

remove_list = ["192.168.97.225", "192.168.158.170", "192.168.201.40", "192.168.
↪58.57"]

# With statement that reads in the text file and stores its contents as a list␣
↪in `ip_addresses`

with open(import_file, "r") as file:
    ip_addresses = file.read()

# Convert `ip_addresses` from a string to a list

ip_addresses = ip_addresses.split()

# For loop that iterates over the elements in `remove_list`,
# checks if each element is in `ip_addresses`,
# and removes each element that corresponds to an IP address that is no longer␣
↪allowed

for element in remove_list:
    if element in ip_addresses:
        ip_addresses.remove(element)

# Display `ip_addresses` after the removal process

print(ip_addresses)
```

```
['ip_address', '192.168.25.60', '192.168.205.12', '192.168.6.9',
'192.168.52.90', '192.168.90.124', '192.168.186.176', '192.168.133.188',
'192.168.203.198', '192.168.218.219', '192.168.52.37', '192.168.156.224',
'192.168.60.153', '192.168.69.116']
```

1.9 Task

In this final task, there are three operating systems: OS 1, OS 2, and OS 3. Each operating system needs a security patch by a specific date. The patch date for OS 1 is `"March 1st"`, the patch date for OS 2 is `"April 1st"`, and the patch date for OS 3 is `"May 1st"`.

The following code stores one of these operating systems in a variable named `system`. Then, it uses conditionals to output the patch date for this operating system.

However, this code had logic errors. My goal was to assign the `system` variable to different values, run the code to examine the output, identify the error, and fix it.

```
[17]: # Assign `system` to a specific operating system as a string

      system = "OS 3"

      # Assign `patch_schedule` to a list of patch dates in order of operating system

      patch_schedule = ["March 1st", "April 1st", "May 1st"]

      # Conditional statement that checks which operating system is stored in␣
       ↪`system` and displays a message showing the corresponding patch date

      if system == "OS 1":
          print("Patch date:", patch_schedule[0])

      elif system == "OS 2":
          print("Patch date:", patch_schedule[1])

      elif system == "OS 3":
          print("Patch date:", patch_schedule[2])
```

```
Patch date: May 1st
```