LOGIN LOGOUT EVENT TRACKER

November 7, 2020

1 Event Tracker: To keep track of Login and Logout from a system

Hello, coders! Below we have code similar to what we wrote in the last video. Go ahead and run the following cell that defines our `get_event_date`, `current_users` and `generate_report` methods.

```
In [27]: def get_event_date(event):
           return event.date

         def current_users(events):
           events.sort(key=get_event_date)
           machines = {}
           for event in events:
             if event.machine not in machines:
               machines[event.machine] = set()
             if event.type == "login":
               machines[event.machine].add(event.user)
             elif event.type == "logout":
               machines[event.machine].remove(event.user)
           return machines

         def generate_report(machines):
           for machine, users in machines.items():
             if len(users) > 0:
               user_list = ", ".join(users)
               print("{}: {}".format(machine, user_list))
```

No output should be generated from running the custom function definitions above. To check that our code is doing everything it's supposed to do, we need an `Event` class. The code in the next cell below initializes our `Event` class. Go ahead and run this cell next.

```
In [28]: class Event:
           def __init__(self, event_date, event_type, machine_name, user):
             self.date = event_date
             self.type = event_type
             self.machine = machine_name
             self.user = user
```

Ok, we have an `Event` class that has a constructor and sets the necessary attributes. Next let's create some events and add them to a list by running the following cell.

```
In [29]: events = [
             Event('2020-01-21 12:45:56', 'login', 'myworkstation.local', 'jordan'),
             Event('2020-01-22 15:53:42', 'logout', 'webserver.local', 'jordan'),
             Event('2020-01-21 18:53:21', 'login', 'webserver.local', 'lane'),
             Event('2020-01-22 10:25:34', 'logout', 'myworkstation.local', 'jordan'),
             Event('2020-01-21 08:20:01', 'login', 'webserver.local', 'jordan'),
             Event('2020-01-23 11:24:35', 'login', 'mailserver.local', 'chris'),
         ]
```

Now we've got a bunch of events. Let's feed these events into our `custom_users` function and see what happens.

```
In [30]: users = current_users(events)
         print(users)
```

```
{'webserver.local': {'lane'}, 'myworkstation.local': set(), 'mailserver.local': {'chris'}}
```

Uh oh. The code in the previous cell produces an error message. This is because we have a user in our `events` list that was logged out of a machine he was not logged into. Do you see which user this is? Make edits to the first cell containing our custom function definitions to see if you can fix this error message. There may be more than one way to do so. Remember when you have finished making your edits, rerun that cell as well as the cell that feeds the `events` list into our `custom_users` function to see whether the error message has been fixed. Once the error message has been cleared and you have correctly outputted a dictionary with machine names as keys, your custom functions are properly finished. Great!

Now try generating the report by running the next cell.

```
In [31]: generate_report(users)
```

```
webserver.local: lane
mailserver.local: chris
```

Whoop whoop! Success! The error message has been cleared and the desired output is produced. You are all done with this practice notebook. Way to go!