# -\*- coding: utf-8 -\*-

"""

Created on Thu Nov 17 08:29:35 2022

@author: badru

"""

class Event:# defining class Event

 def \_\_init\_\_(self, event\_date, event\_type, machine\_name, user):# Creating object by specifying the attributes for the Event class with init method

    self.date = event\_date # using self to access variables within the class-date variable

    self.type = event\_type #to access type variable

    self.machine = machine\_name # to access machine variable

    self.user = user  # to aces user variable

events = [ # defining the events with all the attributes mentioned in the class

    Event('2020-01-21 12:45:56', 'login', 'myworkstation.local', 'Saad'), #user-saad event attributes

    Event('2020-01-22 15:53:42', 'logout', 'Webserver.local', 'Fahad'),#user fahad event attributes

    Event('2020-01-21 18:53:21', 'login', 'Webserver.local', 'lane'),#user lane event attributes

    Event('2020-01-22 10:25:34', 'logout', 'myworkstation.local', 'Saad'),#user saad event attributes

    Event('2020-01-21 08:20:01', 'login', 'Webserver.local', 'Fahad'),#user fahad event attributes

    Event('2020-01-23 11:24:35', 'logout', 'mailserver.local', 'Chris'),#user chris event attributes

]

def get\_event\_date(event):#defining function that returns date- this is the helper function used to sort the list

    return event.date   #returns the date stored in the event object

def current\_users(events):#Defining current users- processing function defining function that returns machines

    events.sort(key=get\_event\_date)#sorting the events and passing the functino created in first step as the key

    machines={}#"""defining a dictioninary - to store names using the machine as the key and the current user as the value

    for event in events:#using for loop to check to iterate through a list of events

        if event.machine not in machines:#conditional loop to check if the machine is present in the dictionary

            machines[event.machine] = set()#using empty set to store the events

        if event.type == "login":#conditional statement to check if the user has logged in

            machines[event.machine].add(event.user)#adding the user to machines if logged in

        elif event.type == "login" and "logout":#conditional statement to check if the user has logged in and out

            machines[event.machine].remove(event.user)#removing the users from machines set

    return machines #returns disctionary

def generate\_report(machines):#defining function to generate reports to print the report

  for machine, users in machines.items():#using for loop - to iterate over keys and values in the dictionary -method items is used

    if len(users)>0:#This check-prevents lists with zero users from being printed

        user\_list=", ".join(users)#generates the string for logged in users for a machine with join method

    #The join() function of str gathers the user attributes (which is a string)

    #into a single string, with commas separating the users

    print("{}:{}".format(machine,user\_list))# using format method to generate the string we want.

#   Separating functions is helpful when debugging or making other changes,

#as it keeps functions from getting ‘tangled’. It also makes it easier to

#adapt functions for other uses.

users = current\_users(events)# calling the current\_users function and saving it in users dictionary

print(users)# prints dictionary or the users

generate\_report(users)# calling the generate report function to generate reports