Solution Breakdown

| Method/Class (Line #) | Code | Intention |
|---|---|---|
| Formatting transaction summary line (22) | def summary_line(code, to_acc, amount, from_acc, name):<br>  if code == LOGOUT:<br>  return "EOS\n"<br>  elif code == CREATEACC:<br>  return "NEW {} 000 0000000 {}\n".format(to_acc, name)<br>  elif code == DELETEACC:<br>  return "DEL {} 000 0000000 {}\n".format(to_acc, name)<br>  elif code == DEPOSIT:<br>  return "DEP {} {} 0000000 ***\n".format(to_acc, amount)<br>  elif code == WITHDRAW:<br>  return "WDR 0000000 {} {} ***\n".format(amount, from_acc)<br>  elif code == TRANSFER:<br>  return "XFR {} {} {} ***\n".format(to_acc, amount, from_acc)<br>  else:<br>  return "ERROR" | Returns a formatted transaction summary line given the 5 data pieces. |
| Account number validity (39) | def valid_number(number):<br>  return re.match(r'^[1-9][0-9]{6}$', number) | Ensures valid account number string. |
| Account name validity (43) | def valid_name(name):<br>  return re.match(r'^[a-zA-Z\d][\w\s]{1,28}[a-zA-Z\d]$', name) | Ensure valid account number string. |
| Start system (47) | def main():<br>  state = NOT_LOGGED_IN<br>  transaction_log = []<br>  valid_accounts_list = {} | Core logic held within session and setting up system admin. |
| Initial state (57) | while state == NOT_LOGGED_IN:<br>  total_withdrawn = {}<br>  total_deposited = {}<br>  total_transferred_out = {}<br>  print("----------")<br>  print("Welcome to Quinterac!")<br>  login = input("Enter '{}' to begin.\n".format(LOGIN)) | Always logged out until logged in. |
| Start User Session (69) | if login == LOGIN:<br>  state = NO_MODE | User can only login when logged out and initiates a new session. |
| Selecting mode-agent or machine (72) | while state == NO_MODE:<br>  print("----MODE----")<br>  mode_choice = input("Please choose either {} or {} mode:\n".format(MACHINE, AGENT))<br>  if mode_choice == MACHINE or mode_choice == AGENT:<br>  print("Using {} mode.".format(mode_choice)) | User selects mode and can only select machine, agent or logout. |

| | | |
|---|---|---|
| | print("Attempting to read valid accounts file.")<br>try:<br>with open(PATH_VALID_ACCOUNTS, 'r') as valid_accounts_file:<br>print("Successfully read valid accounts file.")<br>for account in valid_accounts_file:<br>current = account.split()[0]<br>if current != "0000000":<br>valid_accounts_list[current] = None<br>state = mode_choice<br>except Exception as error:<br>print("Error reading valid accounts file.")<br>print(error)<br>elif mode_choice == LOGOUT:<br>print("Logging out.")<br>transaction_log.append(summary_line(LOGOUT, "", "", "", ""))<br>state = NOT_LOGGED_IN<br>with open(PATH_TRANSACTION_SUMMARY, 'a+') as transaction_summary_file:<br>for line in transaction_log:<br>transaction_summary_file.write(line)<br>else:<br>print("Error selecting mode.")<br>print("{} is not a valid login mode".format(mode_choice)) | |
| Agent state (109) | while state == AGENT:<br>print("----AGENT----")<br>transaction = input("Please input an {} transaction or log out:\n".format(AGENT))<br>if transaction == LOGOUT:<br>print("Logging out.")<br>transaction_log.append(summary_line(LOGOUT, "", "", "", ""))<br>state = NOT_LOGGED_IN<br>with open(PATH_TRANSACTION_SUMMARY, 'a+') as transaction_summary_file:<br>for line in transaction_log:<br>transaction_summary_file.write(line)<br>elif transaction == CREATEACC:<br>number_valid = False<br>while not number_valid:<br>account_number = input("Please input account number:\n")<br>if not valid_number(account_number):<br>print("Invalid account number. Must be exactly 7 digits, not beginning with 0.")<br>elif account_number in valid_accounts_list:<br>print("Account number is taken. Choose something unique.")<br>else:<br>number_valid = True<br>name_valid = False<br>while not name_valid:<br>account_name = input("Please input account name:\n")<br>if not valid_name(account_name):<br>print("Invalid account name. Must be between 3 and 30 characters, not starting or ending with a space.") | Sets transactions allowed for agent state with accompanying actions for each. |

```
                    else:
                        name_valid = True
                transaction_log.append(summary_line(CREATEACC,
account_number, "", "", account_name))
                print("Successfully created account {} -
{}".format(account_number, account_name))
            elif transaction == DELETEACC:
                number_valid = False
                while not number_valid:
                    account_number = input("Please input account
number:\n")
                    if not valid_number(account_number):
                        print("Invalid account number. Must be exactly 7
digits, not beginning with 0.")
                    else:
                        number_valid = True
                name_valid = False
                while not name_valid:
                    account_name = input("Please input account
name:\n")
                    if not valid_name(account_name):
                        print("Invalid account name. Must be between 3
and 30 characters, not starting or ending with a space.")
                    else:
                        name_valid = True
                del valid_accounts_list[account_number]
                transaction_log.append(summary_line(DELETEACC,
account_number, "", "", account_name))
                print("Successfully deleted account {} -
{}".format(account_number, account_name))
            elif transaction == WITHDRAW:
                number_valid = False
                while not number_valid:
                    account_number = input("Please input account
number:\n")
                    if not valid_number(account_number):
                        print("Invalid account number. Must be exactly 7
digits, not beginning with 0.")
                    else:
                        number_valid = True
                amount_valid = False
                while not amount_valid:
                    amount = input("Please input an amount to withdraw
(in cents):\n")
                    if int(amount) > 99999999:
                        print("Invalid amount. Unable to make withdrawls
above $999,999.99.")
                    else:
                        amount_valid = True
                transaction_log.append(summary_line(WITHDRAW, "",
amount, account_number, ""))
                print("Successfully withdrew {} from {}".format(amount,
account_number))
            elif transaction == DEPOSIT:
                number_valid = False
                while not number_valid:
                    account_number = input("Please input account
number:\n")
                    if not valid_number(account_number):
                        print("Invalid account number. Must be exactly 7
```

| | | |
|---|---|---|
| | digits, not beginning with 0.")<br>else:<br>number_valid = True<br>amount_valid = False<br>while not amount_valid:<br>amount = input("Please input an amount to withdraw (in cents):\n")<br>if int(amount) > 99999999:<br>print("Invalid amount. Unable to make deposits above $999,999.99.")<br>else:<br>amount_valid = True<br>transaction_log.append(summary_line(DEPOSIT, account_number, amount, "", ""))<br>print("Successfully deposited {} into {}".format(amount, account_number))<br>elif transaction == TRANSFER:<br>number_sender_valid = False<br>while not number_sender_valid:<br>account_number_sender = input("Please input sender account number:\n")<br>if not valid_number(account_number_sender):<br>print("Invalid sender account number. Must be exactly 7 digits, not beginning with 0.")<br>else:<br>number_sender_valid = True<br>number_recipient_valid = False<br>while not number_recipient_valid:<br>account_number_recipient = input("Please input recipient account number:\n")<br>if not valid_number(account_number_recipient):<br>print("Invalid sender account number. Must be exactly 7 digits, not beginning with 0.")<br>else:<br>number_recipient_valid = True<br>amount_valid = False<br>while not amount_valid:<br>amount = input("Please input an amount to withdraw (in cents):\n")<br>if int(amount) > 99999999:<br>print("Invalid amount. Unable to make deposits above $999,999.99.")<br>else:<br>amount_valid = True<br>transaction_log.append(summary_line(TRANSFER, account_number_sender, amount, account_number_recipient, ""))<br>print("Successfully transferred {} from {} into {}".format(amount, account_number_sender, account_number_recipient))<br>else:<br>print("Invalid transaction code '{}'.".format(transaction)) | |
| Machine state (217) | while state == MACHINE:<br>print("---MACHINE---")<br>transaction = input("Please input an {} transaction or log out:\n".format(AGENT))<br>if transaction == LOGOUT:<br>print("Logging out.")<br>transaction_log.append(summary_line(LOGOUT, "", "", | Sets transactions allowed for machine state with accompanying |

| | | |
|---|---|---|
| | ```"", ""))<br>state = NOT_LOGGED_IN<br>with open(PATH_TRANSACTION_SUMMARY, 'a+') as transaction_summary_file:<br>for line in transaction_log:<br>transaction_summary_file.write(line)<br>elif transaction == WITHDRAW:<br>number_valid = False<br>while not number_valid:<br>account_number = input("Please input account number:\n")<br>if not valid_number(account_number):<br>print("Invalid account number. Must be exactly 7 digits, not beginning with 0.")<br>else:<br>number_valid = True<br>if not account_number in total_withdrawn:<br>total_withdrawn[account_number] = 0<br>amount_valid = False<br>while not amount_valid:<br>amount = input("Please input an amount to withdraw (in cents):\n")<br>if int(amount) > 100000:<br>print("Invalid amount. Unable to make withdrawls above $1,000.00.")<br>elif total_withdrawn[account_number] + int(amount) > 500000:<br>print("Invalid amount. Withdrawl would exceed $5,000.00 daily limit.")<br>else:<br>amount_valid = True<br>total_withdrawn[account_number] += int(amount)<br>transaction_log.append(summary_line(WITHDRAW, "", amount, account_number, ""))<br>print("Successfully withdrew {} from {}".format(amount, account_number))<br>elif transaction == DEPOSIT:<br>number_valid = False<br>while not number_valid:<br>account_number = input("Please input account number:\n")<br>if not valid_number(account_number):<br>print("Invalid account number. Must be exactly 7 digits, not beginning with 0.")<br>else:<br>number_valid = True<br>if not account_number in total_deposited:<br>total_deposited[account_number] = 0<br>amount_valid = False<br>while not amount_valid:<br>amount = input("Please input an amount to withdraw (in cents):\n")<br>if int(amount) > 200000:<br>print("Invalid amount. Unable to make deposits above $2,000.00.")<br>elif total_deposited[account_number] + int(amount) > 500000:<br>print("Invalid amount. Deposit would exceed daily limit of $5,000.00.")<br>else:``` | actions for each. |

| | | |
|---|---|---|
| | amount_valid = True<br>total_deposited[account_number] += int(amount)<br>transaction_log.append(summary_line(DEPOSIT, account_number, amount, "", ""))<br>print("Successfully deposited {} into {}".format(amount, account_number))<br>elif transaction == TRANSFER:<br>number_sender_valid = False<br>while not number_sender_valid:<br>account_number_sender = input("Please input sender account number:\n")<br>if not valid_number(account_number_sender):<br>print("Invalid sender account number. Must be exactly 7 digits, not beginning with 0.")<br>else:<br>number_sender_valid = True<br>if not account_number_sender in total_transferred_out:<br>total_transferred_out[account_number_sender] = 0<br>number_recipient_valid = False<br>while not number_recipient_valid:<br>account_number_recipient = input("Please input recipient account number:\n")<br>if not valid_number(account_number_recipient):<br>print("Invalid sender account number. Must be exactly 7 digits, not beginning with 0.")<br>else:<br>number_recipient_valid = True<br>amount_valid = False<br>while not amount_valid:<br>amount = input("Please input an amount to withdraw (in cents):\n")<br>if int(amount) > 1000000:<br>print("Invalid amount. Unable to make transfers above $10,000.00.")<br>elif total_transferred_out[account_number_sender] + int(amount) > 1000000:<br>print("Invalid amount. Transfer would exceed daily limit of $10,000.00.")<br>else:<br>amount_valid = True<br>total_transferred_out[account_number_sender] += int(amount)<br>transaction_log.append(summary_line(TRANSFER, account_number_sender, amount, account_number_recipient, ""))<br>print("Successfully transferred {} from {} into {}".format(amount, account_number_sender, account_number_recipient))<br>else:<br>print("Invalid transaction code '{}'.".format(transaction)) | |
| Initialize system | if __name__ == "__main__":<br>main() | Main is called. |