The code is divided into three main parts:

* **users.py**
* **theDevices.py**
* **main.py**

Each of these parts plays a specific role in how the application functions. Additionally, there are two supporting text files: **users.pkl** contains user authentication, and **devices.txt** contains information about devices.

**Components and Steps**:

1. **User Class (users.py)**

This class enables user authentication. It loads user data, manages login attempts, and offers methods for login. Users can attempt to log in, with a limited number of 3 attempts. The pre-registered user with username: **admin** and password: **Pyth0n2023**

**Initialization:**

Constructed **\_\_init\_\_(self)** initializes by loading user data from the *"users.pkl"* file and sets the attempt to log in default to 3.

**Load:**

The**load\_users(self)** to load user data from the *"users.pkl"* file.

**Login:**

The **login(self)**to enter username and password. The method verifies if the provided username and password match the loaded user data and check the limited number of attempts to log in.

**Register:**

The**register\_user(self)** to register new user credential and then invoke the save function with the user credentials as an argument.

**Save User:**

The**save\_users(self)** to save user credentials in a *"users.pkl"* file.

1. **DeviceManagement Class (theDevices.py)**

This class handles operations associated with devices, offering functions for viewing, adding, deleting, updating, and searching devices.

**Initialization:**

Constructed **\_\_init\_\_(self)** method to initialize the devices attribute, setting it with data retrieved from the *"devices.txt"* file.

**Load:**

The **load\_devices(self)** to load the list of devices from the *"devices.txt"* file.

**Save:**

The **save\_device(self)** to save the updated list of devices back to the *"devices.txt"* file.

**View:**

The **view\_devices(self)** to display the names of all devices in the list.

**Add:**

The **add\_device(self, name)** to add a new device assigned in *“name”* and, while also verifying if the device already exists and subsequently add it if it’s not present.

**Delete:**

The **delete\_device(self, device\_to\_delete)** to remove a device from the list based on a provided *“code”.*

**Update:**

The **update\_device(self, device\_to\_update, new\_name)** to update a device's name in the list from “*device\_to\_update”* to “*new\_name”*.

**Validate:**

The **validate\_device(self, old\_name)** to validate the input device code exists in *“devices.txt”* file.

**Search:**

The **search\_device(self, keyword)** to search for devices containing a specified “*keyword”*.

**Menu:**

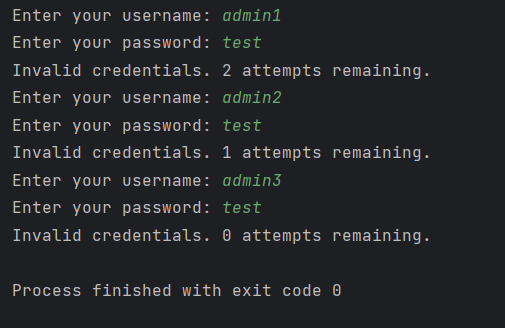
The **main\_menu(self)** to display the devices menu and handle user input, perform the selected action, and offer a choice to continue or exit.

1. **Main Function (main.py):**

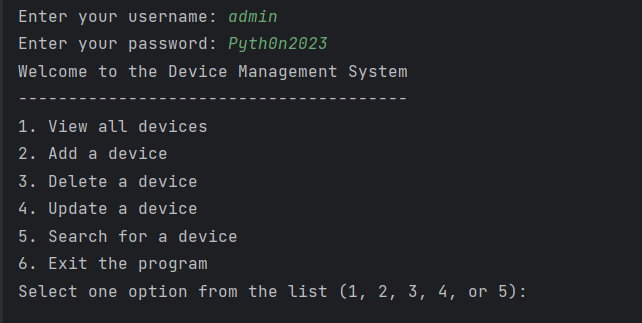
The main function facilitates user interaction by displaying options and handling their selections. It initializes instances of the **User** and **DeviceManagement** classes. When users successfully log in, the device management menu will be shown.

**Screenshots that show the outputs of each function:**

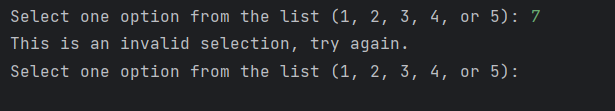
Login Attempt:



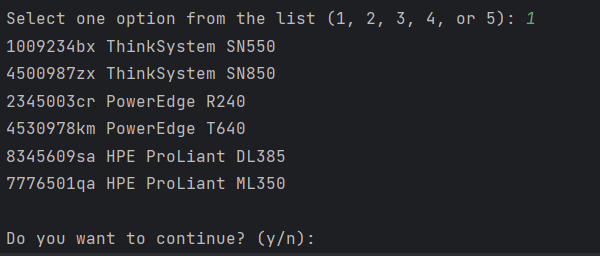
Login and display the device management menu:



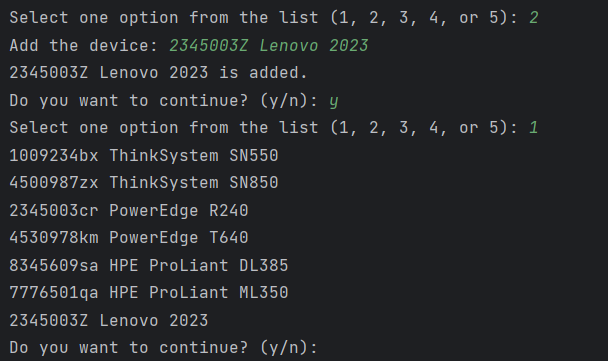
Validate user selection:



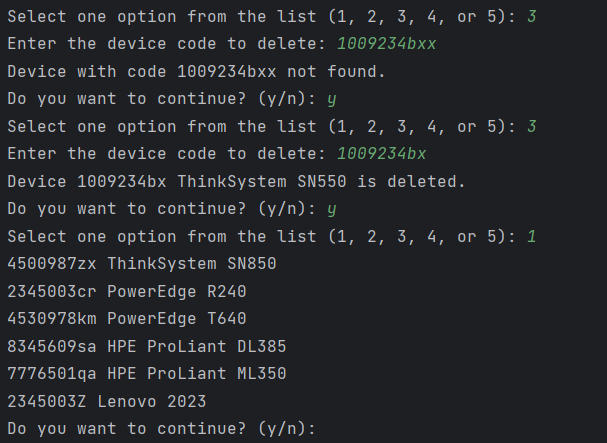
View all devices:



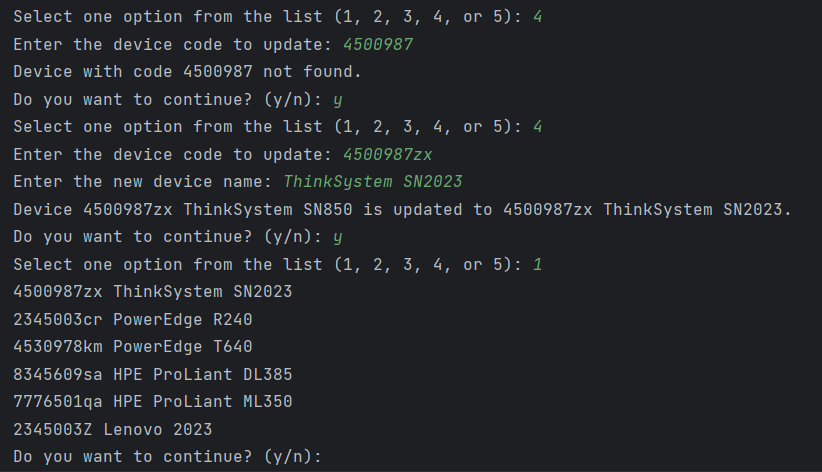
Add a device:



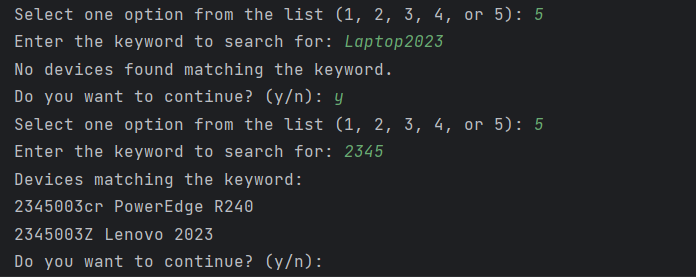
Delete a device:



Update a device:



Search for a device:



Exit the program:

