Python EMS Project

You need to create an Employee Management System for a company. This EMS should be able to ***add new Employees***, ***update Employee information***, ***remove Employees***, and ***list Employee information***. The user interface (*U/I*) will be the terminal console, where you can print the program information, and take in inputs from the user's standard input (the keyboard). You will need to have a basic menu with options to navigate your application for your users to interact with.

Requirements:

- *Use the principles of OOP in your design.* (*I.e. encapsulate relevant code together*)
    - Break down methods and classes with separation of concerns (not everything in one big main method), so that each part of your code is focused on doing one task, and doing it well.
- *Employee objects should store at minimum*
    - A ***name***
    - The ***employee id*** (*needs to be unique*)
    - The employee's yearly ***Salary***
    - The employee's ***department*** (*can be stored as a string*)
- *EMS Storage*
    - Employees should be ***stored in-memory as a list***
- *New Employees input* (*Create*)
    - New Employees should be ***entered from the Command Line***
- *Exception Handling*

Cognixia®

    - Implement the use of *at least one custom exception*, and use Exception handling in all appropriate places.
- *General Requirements*
    - Use at least *one List Comprehension*
    - Have a *console based U/I menu system* that allows a user to navigate through your app and perform all of the CRUD (*Create Read Update Delete*) operations listed above, at a minimum.

Querying Features:

- Users should be able to *Add* new Employees, *Update* existing Employees, and *Delete* Employees
- Users should be able to execute the following queries:
    - List all Employees by *id*
    - List all Employees by *salary*, low to high ***and*** high to low
    - List all Employees in a *salary range*
    - List all employees *hired within a range of dates*
    - Find all employees by *Department*
    - Find a single employee by *id* and by *name*

Bonus:

- *Create a Department class*
    - Make sure to store information a department might have (like *name*, *budget*, *phone*, etc.). You will also be able to *add*, *update*, *remove*, and *list* the different departments. You should also be able to ***keep track of all the departments in the company and which employees are within them***.
    - Replace the department string on your Employee object with this department class
- *Use File Reading to persist data*
    - Use the file methods to *store and retrieve* (*read and write*) your data for *Employees* and *Departments* to a file format of your choice (*CSV, JSON, Other*)
- *Testing*
    - Set up testing for code using unittest