ECE 122: Introduction to Programming for ECE- Spring 2020

Project 2: My Company Employee Database (a complete procedural programming example)

Due Date: **Deadline: see website, class policy and moodle for submission**
This is an individual project (discussions are encouraged but no sharing of code)

Description

The goal of the project is to design and implement a basic company employee database system in which administrators (you) can inquire the current list of employees, and add or remove employees. Employees are identified by their ID number and they have different attributes: name, age, job title, years of service, and info if they are from Massachusetts. We will work with the following database which contains 16 employees here in alphabetical name order):

ID	Name	Age	Job	#Years	Is from Mass?
E04	Andrea	37	Sale	8	True
E11	Bob	28	Engineer	4	False
E09	Brooke	34	Engineer	5	True
E12	Connor	27	Engineer	4	True
E16	James	25	Accountant	3	False
E24	Jenna	24	Engineer	1	False
E02	John	45	Manager	17	True
E03	Julie	37	Engineer	10	True
E01	Kate	48	Manager	24	False
E08	Keith	28	Engineer	6	False
E23	Kelly	25	Engineer	1	False
E17	Luke	33	Sale	3	False
E18	Mark	34	Sale	3	True
E22	Pat	26	Engineer	2	True
E15	Taylor	32	Accountant	4	True
E05	Tony	34	Sale	8	False

The project must include three files:

1. `Employee.py` file/module that contains the user-defined type (object data) Employee.
2. `database.py` file/module that contains **all** the necessary functions to operate the database.
3. `company.py` file containing the main program.

Submission/Grading Proposal

Only **one zip file** must be submitted on moodle. Make sure you know how to create and submit a zip file before the submission date (and that your zip file is no empty once submitted). This project will be graded out of 100 points:

1. Your program should implement all basic functionality/Tasks and run correctly (90 points).
2. Overall programming style: program should have proper identification, and comments. (10 points).

The project is designed to be incremental, you can then debug, test and run your code after each new task/option is implemented. However, after Task 1 done all the other Tasks/options can be completed in any order. Use your preferred IDE to read, write and save your files. If you are not using IDLE, however, make sure your program is running using the command prompt (your grade will be reduced by 50% if this is not the case or if the graders have to go the extra-step to make sure it runs using the command prompt). Do not forget to comment your code. Make sure you obtain the **same output** for the **same input** in the examples (this includes syntax, blank spaces, and skipping blank lines). Your program will also be tested with different inputs by the graders. Finally, you are free to consider additonal functions if you wish to do so. However, you cannot use programming concepts different than the ones we have seen in class so far: if, for/while, functions, print/input, and data object.

How to Start- Task-0 [10pts]

At the first execution of the program `company.py`, the output should include a menu containing 9 options:

```
Employee Management Tool
========================

Menu (Year:2020)
================
1-List Employees        2-General Information   3-Employee Information
4-Filter per job        5-Filter per #years     6-Remove Employee
7-Add Employee          8-Change Date           0-Exit
Enter Command:
```

Once option 0 (for Exit) is selected, the program stops. And it should print:

```
Enter Command: 0
Goodbye!
```

How to proceed?

1. In the `company.py` file: display the welcoming message "Employee Management Tool" etc.

2. Use a call to a function `display_menu` that will print the menu.

3. Consider using a `while` loop (such as `while true` that will keep printing the menu and keep asking the user to enter a new command; this while loop should exit if the entry is 0 and print a 'Goodbye!' message;

4. the function to print the menu, `display_menu`, is already provided to you in the file `database.py`;
5. We note that the header of the main file already contains the instruction `import database` which will allow you to call the functions in the `database` file using the dot operator.

All the other command options are reviewed below.

Task-1- [20pts]

Let us now see what should happen when option 1 is selected.

```
Enter Command: 1

ID      Name    #years  job
---------------------------
E04     Andrea  8       Sale
E11     Bob     4       Engineer
E09     Brooke  5       Engineer
E12     Connor  4       Engineer
E16     James   3       Accountant
E24     Jenna   1       Engineer
E02     John    17      Manager
E03     Julie   10      Engineer
E01     Kate    24      Manager
E08     Keith   6       Engineer
E23     Kelly   1       Engineer
E17     Luke    3       Sale
E18     Mark    3       Sale
E22     Pat     2       Engineer
E15     Taylor  4       Accountant
E05     Tony    8       Sale

Menu (Year:2020)
================
1-List Employees        2-General Information   3-Employee Information
4-Filter per job        5-Filter per #years     6-Remove Employee
7-Add Employee          8-Change Date           0-Exit
Enter Command:
```

Here the entire list of employees is displayed (along with some info: id, name, years of service, and job title). At the end the menu selection is printed again and the program is waiting for you to make another choice.

How to proceed?

1. In the `company.py` file you need first to implement a call to a function `initialize` that will return a list of items (data objects of type Employee...explained further below). Then, the option 1 must be implemented inside the `while` loop. The option could include a call to function `display` (you can use as arguments the list of employees).
2. In the `Employee.py` file. You need to implement a class that define the type data object `Employee`. The latter should include the constructor (i.e. function `__init__`). You will

need several attributes, you could use for example: `idnumber`, `name`, `age`,`job`,`years`, and `is_from_mass`. All the attributes could be initialized to `None` by default.

3. In the `database.py` file: (1) Implement the `initialize` function that creates and returns a list of Employee data objects. Ideally we would like to read all the employee data from a file (so we could easily consider hundreds of employees if needed) but we will do that later in the semester. Here, you will need to fill up by hand (hard coded) all the attributes of the objects for our selected 16 employees presented at the beginning of the project.

   (2) The function `display` that displays all the employee information as presented in the output example (you can use `\t` to separate each attribute). We note that the header of the file contains the instruction `from Employee import Employee` which will allow you to use the Employee data type.

Task-2- [10pts]

When option 2 is selected, the program should give you back some statistics: the number of employees, the ratio of employees by job position, the mean values of the employees' ages and years of services.

```
Enter Command: 2

#Employee 16
Manager: 12.5%
Sale: 25.0%
Accountant: 12.5%
Engineer: 50.0%

Age Mean: 32.3125
#Years Mean: 6.4375

Menu (Year:2020)
================
1-List Employees        2-General Information   3-Employee Information
4-Filter per job        5-Filter per #years     6-Remove Employee
7-Add Employee          8-Change Date           0-Exit
Enter Command:
```

How to proceed?

1. In the `company.py` file: implement the option 2 that contains a call to a function `info` (you can use as arguments the list of employees)

2. In the `database.py` file: the function `info` that displays the employee statistics as presented in the output example. You will need a for loop that scans through all employee objects in the list. Hint: to find out about the number of employees doing a particular job, you could set up some local counters that could be incremented at each loop (using conditional statements).

Task-3- [10pts]

Let us now see what is happening when option 3 is selected (here we do it two times).

```
Enter Command: 3
Enter employee ID: E04
Andrea is 37 years old and has a Sale position
Andrea is from Massachusetts

Menu (Year:2020)
================
1-List Employees        2-General Information   3-Employee Information
4-Filter per job        5-Filter per #years     6-Remove Employee
7-Add Employee          8-Change Date           0-Exit
Enter Command: 3
Enter employee ID: E07
Employee not found!

Menu (Year:2020)
================
1-List Employees        2-General Information   3-Employee Information
4-Filter per job        5-Filter per #years     6-Remove Employee
7-Add Employee          8-Change Date           0-Exit
Enter Command:
```

The program is asking the user to enter a particular employee ID number and it will then display more personal information about this particular employee (or returned "Employee not found!" if the ID is not in the database).
How to proceed?

1. In the `company.py` file. Implement the option 3 that is asking the user to enter an employee ID. I suggest then the following: (1) include a call to a function `search_employee` that would return True or False (True if this employee exists in the database). (2) if True, call another function `display_employee` that displays the information for the selected employee as presented in the output example.

2. In the `database.py` file, you need to implement the two functions above. Both functions, could use as arguments: the list of employees, and the id number that has been chosen by the user. Hint: While `search_employee` should be straightforward to implement (you can scan the list and check for the idnumber attribute), the function `display_employee` need first to search for the correct index in the list (that corresponds to the user id entry).

Task-4- [10pts]

Let us now see what is happening when option 4 is selected.

```
Enter Command: 4
```

```
ID      Name    #years  job
---------------------------
E02     John    17      Manager
E01     Kate    24      Manager
E04     Andrea  8       Sale
E17     Luke    3       Sale
E18     Mark    3       Sale
E05     Tony    8       Sale
E16     James   3       Accountant
E15     Taylor  4       Accountant
E11     Bob     4       Engineer
E09     Brooke  5       Engineer
E12     Connor  4       Engineer
E24     Jenna   1       Engineer
E03     Julie   10      Engineer
E08     Keith   6       Engineer
E23     Kelly   1       Engineer
E22     Pat     2       Engineer

Menu (Year:2020)
================
1-List Employees        2-General Information   3-Employee Information
4-Filter per job        5-Filter per #years     6-Remove Employee
7-Add Employee          8-Change Date           0-Exit
Enter Command:
```

As you can see, option 4 filters the result by job title displaying first the employee managers, followed by sales, accountants and engineers. The names in each category are still placed in alphabetical order.
How to proceed?:

1. In the `company.py` file: the option 4 that could include a call to a function `job_filter` that will display the new information.

2. In the `database.py` file implements the method `job_filter` that uses as argument the list of employees. Hint: You do not need to implement any types of sorting techniques (we will see sorting in Chapter 5), this filtering can actually be done pretty easily using 2 `for` loops. The first loop could scan a list of job titles that you will create, and the second could scan your employee list. You would just need a condition to test the value of the employee attribute `job` before printing.

Task-5- [10pts]

Let us now see what is happening when option 5 is selected.

```
Enter Command: 5

ID        Name        #years  job
---------------------------
E01     Kate    24      Manager
```

```
E02     John    17      Manager
E03     Julie   10      Engineer
E04     Andrea  8       Sale
E05     Tony    8       Sale
E08     Keith   6       Engineer
E09     Brooke  5       Engineer
E11     Bob     4       Engineer
E12     Connor  4       Engineer
E15     Taylor  4       Accountant
E16     James   3       Accountant
E17     Luke    3       Sale
E18     Mark    3       Sale
E22     Pat     2       Engineer
E24     Jenna   1       Engineer
E23     Kelly   1       Engineer

Menu (Year:2020)
================
1-List Employees        2-General Information   3-Employee Information
4-Filter per job        5-Filter per #years     6-Remove Employee
7-Add Employee          8-Change Date           0-Exit
Enter Command:
```

As you can see, the list of employees is now filtered by the (decreasing) number of years of service. This is very similar to task 4, where you can now use a function `years_filter` instead. The result can be achieved using a similar two `for` loop approach. I let you figure out this one.

As a side remark, you note that all the ID are in some type of order (order of arrival in the company), that would mean that some employees may have left the company and their ID has not be reassigned yet.

Task-6- [10pts]

Using option 6, you will have the possibility to remove an employee from the list, if you provide his id number. Let us suppose Kelly does not like her job after 1 year at the company and decide to quit. Here we see the output results after the option 6, followed by 1 and 2 are selected.

```
Enter Command: 6
Enter Employee ID: E23

Menu (Year:2020)
================
1-List Employees        2-General Information   3-Employee Information
4-Filter per job        5-Filter per #years     6-Remove Employee
7-Add Employee          8-Change Date           0-Exit
Enter Command: 1

ID        Name        #years        job
---------------------------
E04     Andrea  8       Sale
E11     Bob     4       Engineer
E09     Brooke  5       Engineer
```

```
E12     Connor  4       Engineer
E16     James   3       Accountant
E24     Jenna   1       Engineer
E02     John    17      Manager
E03     Julie   10      Engineer
E01     Kate    24      Manager
E08     Keith   6       Engineer
E17     Luke    3       Sale
E18     Mark    3       Sale
E22     Pat     2       Engineer
E15     Taylor  4       Accountant
E05     Tony    8       Sale

Menu (Year:2020)
================
1-List Employees        2-General Information   3-Employee Information
4-Filter per job        5-Filter per #years     6-Remove Employee
7-Add Employee          8-Change Date           0-Exit
Enter Command: 2

#Employee 15
Manager: 13.333333333333334%
Sale: 26.666666666666668%
Accountant: 13.333333333333334%
Engineer: 46.666666666666664%

Age Mean: 32.8
#Years Mean: 6.8

Menu (Year:2020)
================
1-List Employees        2-General Information   3-Employee Information
4-Filter per job        5-Filter per #years     6-Remove Employee
7-Add Employee          8-Change Date           0-Exit
Enter Command:
```

As soon as you enter Kelly ID `E23`, her entry is removed from the list of employees in the database.

How to proceed?

1. In the `company.py` file implement the option 6 that should first include a call to the function `search_employee` you implemented in Option 3. Indeed, if the user enters a bad ID number, the code should return again "Employee Not found!". If the employee is in the database, you can then proceed with removing this employee using a new function `remove_employee`.
2. In the `database.py`, you need to implement `remove_employee`, again using two arguments: list of employees, and the id number that has been chosen by the user. Once the correct index found in the list, you can use the built-in `del` function (seen in class) to remove an item with a given index from a list (it will automatically left shift all the other items with higher indexes).

Task-7- [10pts]

Now with Option 7, we would like to add a new employee to the database. In the example below, we first try to add the ID `E01` which happens to exist already so the code is returning `Employee exists already!`, then we add the employee ID `E06` named “Eric” (age 22, and Engineer). As you can see the number of years of service for Eric is 0.

```
Enter Command: 7
Enter Employee ID: E01
Employee exists already!

Menu (Year:2020)
================
1-List Employees        2-General Information   3-Employee Information
4-Filter per job        5-Filter per #years     6-Remove Employee
7-Add Employee          8-Change Date           0-Exit
Enter Command: 7
Enter Employee ID: E06
Enter Name: Eric
Enter Age: 22
Job Select: 1-Manager, 2-Sale, 3-Accountant, 4-Engineer: 4

Menu (Year:2020)
================
1-List Employees        2-General Information   3-Employee Information
4-Filter per job        5-Filter per #years     6-Remove Employee
7-Add Employee          8-Change Date           0-Exit
Enter Command: 1

ID        Name        #years        job
---------------------------
E04     Andrea  8       Sale
E11     Bob     4       Engineer
E09     Brooke  5       Engineer
E12     Connor  4       Engineer
E16     James   3       Accountant
E24     Jenna   1       Engineer
E02     John    17      Manager
E03     Julie   10      Engineer
E01     Kate    24      Manager
E08     Keith   6       Engineer
E23     Kelly   1       Engineer
E17     Luke    3       Sale
E18     Mark    3       Sale
E22     Pat     2       Engineer
E15     Taylor  4       Accountant
E05     Tony    8       Sale
E06     Eric    0       Engineer

Menu (Year:2020)
================
1-List Employees        2-General Information   3-Employee Information
4-Filter per job        5-Filter per #years     6-Remove Employee
7-Add Employee          8-Change Date           0-Exit
Enter Command: 2
```

```
#Employee 17
Manager: 11.76470588235294%
Sale: 23.52941176470588%
Accountant: 11.76470588235294%
Engineer: 52.94117647058824%

Age Mean: 31.705882352941178
#Years Mean: 6.0588235294117645

Menu (Year:2020)
================
1-List Employees        2-General Information   3-Employee Information
4-Filter per job        5-Filter per #years     6-Remove Employee
7-Add Employee          8-Change Date           0-Exit
Enter Command:
```

What you need to implement:

1. Implement the option 7 that should first include a call to the function `search_employee` you implemented in Option 3. Indeed, if the user enters a bad ID number, the code should return again "Employee exists already!". This time if the search was unsuccessful, you can then proceed with adding this employee. Basically, you will need to use a new function `create_employee` that returns a new data object Employee. You will then need to append it to the list (updating then the new database).

Task-8- Bonus [5pts]

Finally, in Task 8, and as you have seen since the beginning, the menu displays the Year 2020. You need to modify the code to allow for projection in the future. For example in year 2030, every employee will be 10 years older and have 10 more years of services. Here an example:

```
Enter Command: 8
Projection date (>=2020): 2030

Menu (Year:2030)
================
1-List Employees        2-General Information   3-Employee Information
4-Filter per job        5-Filter per #years     6-Remove Employee
7-Add Employee          8-Change Date           0-Exit
Enter Command: 2

#Employee 16
Manager: 12.5%
Sale: 25.0%
Accountant: 12.5%
Engineer: 50.0%

Age Mean: 42.3125
#Years Mean: 16.4375

Menu (Year:2030)
```

```
================
1-List Employees        2-General Information   3-Employee Information
4-Filter per job        5-Filter per #years     6-Remove Employee
7-Add Employee          8-Change Date           0-Exit
Projection date (>=2020): 2025

Menu (Year:2025)
================
1-List Employees        2-General Information   3-Employee Information
4-Filter per job        5-Filter per #years     6-Remove Employee
7-Add Employee          8-Change Date           0-Exit
Enter Command: 5

ID        Name        #years        job
---------------------------
E01     Kate    29      Manager
E02     John    22      Manager
E03     Julie   15      Engineer
E04     Andrea  13      Sale
E05     Tony    13      Sale
E08     Keith   11      Engineer
E09     Brooke  10      Engineer
E11     Bob     9       Engineer
E12     Connor  9       Engineer
E15     Taylor  9       Accountant
E16     James   8       Accountant
E17     Luke    8       Sale
E18     Mark    8       Sale
E22     Pat     7       Engineer
E24     Jenna   6       Engineer
E23     Kelly   6       Engineer

Menu (Year:2025)
================
1-List Employees        2-General Information   3-Employee Information
4-Filter per job        5-Filter per #years     6-Remove Employee
7-Add Employee          8-Change Date           0-Exit
Enter Command:
```