Design and Analysis of
Data Structure and Algorithms

Python coursework
Animal Sanctuary
November 2019

Minh Anh Le

Explanation on the naming convention in this project:

I have decided to go with Java naming convention “mixedCase” instead of “joined_lower” mainly because of personal preference and so far I have not found any disadvantages to using those in an individual project.

Referencing from Style Guide Python Codes:

“The naming conventions of Python's library are a bit of a mess, so we'll never get this completely consistent”

Of course it is best to abide with whatever the prevailing style for a codebase / project / team happens to be. As the Python Style Guide points out, *internal consistency* matters most.

Below will be the explanation and pseudo-code of the functions created inside the program:

- Initialization of the DLLs in the program
- Add a new entry (for the list of Pets, Wild Animals)
- Search for an entry
- Edit a specific field of an entry
- Generate a list
- Writing into CSV files/Edit a row in CSV files

General structure of the project:

There will be several doubly linked lists initialized at the start of the program, and depends on which task the user want to execute the corresponding linked list will be called with the appropriate function to get the job done. All of the functions are created in the DoublyLinkedList class, only the Pet class and WildAnimal class are seperated into two other files.

Initialization:

- When the program is compiled, it will search for the csv files put in the same directory, create a temporary object from the data gathered from the file, then that object is inserted into the Doubly Linked List using the created insert function

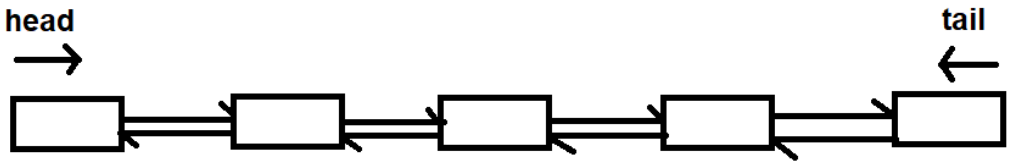
Add a new entry:

- The program will ask the user to type in every attribute of the new Pet/Wild Animal (Any attribute can be left blank if not needed). Then similarly to initialization, a temporary object is created, that object is inserted to the DLL, and finally the new entry is writen into the csv file.

```
After user types in data of the new entry
      create new temporary object
            add object to DLL and write new entry to csv file
```

Search for an entry:

- This function is the main reason DLL is chosen for this project. Initially, the idea is to use **threads** in Python to run 2 functions **at the same time**, one would search the DLL from the head of the list and the other would start from the tail of the list going backwards, making the complexity O(n/2).



- But during the development, I have found out that because of the **Global Intepreter Lock**, these threads won't execute at the exact same time, even if the machine has multiple CPUs. More details on GIL can be found here.
- Instead, a **pseudo-thread function** has been implemented, where there is two pointer running into the midddle of the list from both ends, and whichever finds the desired value first will print it out and break the loop.

```
while head and tail does not meet
      if head is the desired value or tail is the desired value
            return the head or the tail correspondingly
            break out of loop
      increment head and decrement tail simultaneously
```

Edit an attribute of an entry:

- The program will loop the corresponding DLL to find the desired Sanctuary Identification, and from then it will access the attribute the user wants to change and alter the value by the input of the user from keyboard.

```
While list is not traversed completely
      if the desired entry ID is found
            change the attribute of that entry in the DLL
            write the new change back into csv file
```

Generate a list:

- The program will loop through the corresponding DLL to find the entries that qualified the conditions of the new list the user is creating, and whichever satisfies those conditions will be added to a new Doubly Linked List. For the case of abuser and abandoner, there is a function in place to remove any duplicates.

```
While list is not traversed completely
      if the entry satisfies the conditions
            add entry to result list
```

- For the conditions of each list:

  + List of cats/dogs ready for adoption:
    They're vaccinated (**Vaccinated** field = "Yes"), Neuteured (**Neuteured** field = "Yes") and finally there exists a microchip ID in the **Microchip** field.

  + List of animals that are ready to be returned to owner:
    For the wild animals, their **reason for admission** will be "Lost" and they still remain in the sanctuary (**Departure date** is blank), also their **medication** has to be finished as well. For the pets, it would not make sense if the pets are returned to people who have abused or abandoned them, so their reasons for admission **must not be "stray","abandoned" or "abused"** and they haven't

left the sanctuary (**Departure date** left blank). Both wild animals and pets must have been vaccinated.

+ List of animal abusers:

The program will traverse the Treatment file and any "**Responsible for Abuse**" field that is filled will be extracted, any duplicate will be removed after. The list will be provided after sorting is done.

+ List of animal abandoners:

The program will traverse the Treatment file and any "**Responsible for Abandoning**" field that is occupied will be extracted, any duplicate will be removed after. The list will be provided after sorting is done.

Writing into CSV files:

- For this function I have imported the csv library in Python to use the read and write functions.

Possible improvements:

- The method for traversing the DLL in some functions can be improved, for example instead of traversing in one direction only, doing so from both ends can theoretically increase the efficiency.
- The insert function for putting new nodes into the doubly linked list can be improved so that the nodes are sorted when they are inserted. Because of inexperience and bad practice the function could not be improved in time for submission.
- The program can do with a basic but considered more intuitive UI using Python GUI libraries.
- A solution is needed to merge similar functions but have to be seperated (e.g initialization functions for Pet and Wild Animal have to be seperated) for various objects because of the difference in the number of attributes, for example a Pet has 11 attributes while a Wild Animal has 8.