Justification for the Data-Structures

Approaches for holding the attributes of CSV files; Pet, Wild Animal and Treatment

List of lists

One approach is to use a Python list of lists and index into the list to store the attributes. However, this approach would be difficult to maintain since the programmer must be aware of the relationship between the index and the attribute. For example, **pet[0][1]** represents the sanctuary ID of the first pet in the list. Using list of lists data structure will grow in complexity in terms of the algorithms needed to search the data furthermore, the programme using this approach would be dynamically insufficient and hard to maintain when dealing with 10,000 entries a year.

Classes (Chosen approach)

The Object Orientated approach used for this task is to hold the attributes using Python Classes. This approach allows individual attributes to be directly referenced using an associated name such as; **pet.sanctuary\_id**. Each row in the CSV files; Pet, Wild Animal and Treatment is stored as an Object with assigned attributes (column headings). Each Object is then appended into a Python list. Storing each animal as an Object enables the programme to use simple algorithms to; iterate through the list of Objects, search for a specific attribute, return a specific attribute or update the attribute. Furthermore, storing the Objects in a Python list allows new rows in the CSV file to be created and then appended to the array list as Object of a chosen class.

Python Classes is a dynamic data structure approach as a new column heading can be added by creating a new attribute in a chosen Class and simply adding a new column to the CSV read and write functions. In addition, it’s a way of dealing with complexity and should minimize bigO notation in comparison to just using a Python list of lists. Finally, using Classes has the benefit of allowing overriding of standard methods; such as less than, greater than (for sorting Objects) and is effective in that more overriding methods could easily be added if necessary.

Sorting Algorithms

Insertion Sort

Whilst not being able to use Pythons built in .sort() function, insertion sort is implemented for tasks 2C, 2D and 2G. Insertion sort considers the first element in the ‘list’ sorted then the second element is considered. The advantage is that it only scans as many elements as it requires in order to place the k + 1st element in comparison to selection sort which must scan all remaining elements to locate the k = 1st element. This implies that insertion sort will therefore usually perform half as many comparisons as selection sort. Insertion sort performs well in large sets when dealing with human data that has already been sorted therefore, it will run at O(n) performance. Bubble sort was not used because it’s too simplistic and rarely used in practice.

Remove Duplicates

A simple algorithm was used that removes duplicate items from a list. This algorithm simply creates a temporary list. It checks if an item in the list passed as a parameter is in the temporary list therefore, if the item is not currently in the temporary list it will append the item. When finished the temporary list holds all items with no duplicates.