DAV 5400 Spring 2019 Week 3 Assignment (30 Points)

This assignment will extend the work you started last week with the **cars-sample35.txt** data set to encompass list comprehensions, some additional Python data structures (e.g., **dict** objects), and user-defined functions.

Start by reading the **cars-sample35.txt** file and re-creating the seven distinct lists of automobile attributes (i.e., Price, Maintenance Cost, Number of Doors, Number of Passengers, Luggage Capacity, Safety Rating, Classification of Vehicle) we worked with last week. You should be able to do this via a simple cut + paste of your code from last week.

List Comprehensions

Your first task is to find the list index values of each automobile having a price rating of "med". However, you are required to do this **using a list comprehension** instead of a basic **for** or **while** loop. The list comprehension should create a new list containing your result. Be sure to print your results to the screen.

Your second task is to find the "number of passengers" value for each auto having a "price" value of "med" **using a list comprehension** instead of a basic **for** or **while** loop. The list comprehension should create a new list containing your findings. Be sure to print your results to the screen.

Your third task is to find the index value for each automobile having a price value of "high" and a maintenance value that ***is not*** "low" **using a list comprehension**. The list comprehension should create a new list containing your findings. Be sure to print your results to the screen.

Nested List Comprehensions

Consider the following list of lists:

nlist = [[1, 2, 3], [‘A’, ‘B’, ‘C’], [4, 5], [‘D’, ‘E’]]

If we wanted to extract each individual element of the component lists contained within **nlist** and add them to a new list, we could use a nested **for** loop similar to this:

```
--------
flist = []

for x in nlist:

   for y in x:

        flist.append(y)

print(flist)

--------
```

For your fourth task, implement this same logic **using a list comprehension.** Apply your list comprehension to the **nlist** list of lists shown above. Be sure to print your newly created list to the screen.

(** NOTE: The assignment continues on the next page **)

Write a Function That Converts a List to a Dict Object

Your fifth task is to create a user defined function that accepts as input one of the seven automobile attribute lists as well as an integer value and returns a Python **dict** object. Remember: a Python **dict** object is comprised of 'key/value' pairs. The integer value accepted as a parameter by your function will represent the exact number of items you are to use from the list for purposes of creating the new **dict** object. So for example, if your function was defined as:

```
def makedict(mylist, x):
        newdict = {}
        # your code goes here
return(newdict)
```

---

you would take the first **x** items from mylist to create the new **dict** object. The key values you assign to the elements of the **dict** object should all be comprised of a combination of the letter 'A' and the alphanumeric representation of the index value corresponding to the location of the item in the mylist object. So if you had 3 **dict** elements in total, the key values for those elements would be ('A0', 'A1', 'A2'). If we apply this concept to the first 5 elements of our "Luggage" list we'd have a dict object that looks like this:

First five elements of luggage list:

['med', 'small', 'big', 'big', 'med']

New dict Object:

{'A0' : 'med', 'A1': 'small', 'A2': 'big', 'A3':'big', 'A4':'med'}

Be sure to check that the integer value passed into your function does not exceed the length of the list parameter!

Using Your New Function

Now that you have your reusable user-defined function, use it to create two new **dict** objects containing the first seven elements of the 'Price' and 'Luggage' lists, respectively. Then, use these dict objects to tell us the price and luggage capacity of the auto identified by the 'A4' key value.

Be sure to include some commentary explaining your approach to solving each of the individual problems. Save your work to a Jupyter Notebook and upload it to your online DAV5400 GitHub directory. Be sure to save your Notebook using the nomenclature we've been using, i.e., "**first initial_last name_W3_assn**" (e.g., J_Smith_W3_assn_).