DAV 5400 Module 2 Assignment (30 Points)

This assignment will extend the work you started in Module 1 with the **M1_Data40.txt** data set to encompass list comprehensions, some additional Python data structures (e.g., **dict** objects), and user-defined functions.

Start by reading the **M1_Data40.txt** file and **re-creating the six distinct lists of hospital patient attributes** (i.e., Diagnosis, Age, Gender, Type of chest pain, blood pressure reading, cholesterol level) we worked with last week. Make sure you convert the content of the Age, Blood Pressure, and Cholesterol lists to integer values. You should be able to do this via a simple cut + paste of your code from last week. Then, complete the following six tasks:

List Comprehensions

1. **(5 Points)** Your first task is to find the **list index** values of each hospital patient having a "**blood pressure**" reading **that is greater than 140**. However, you are required to do this **<u>using a list comprehension</u>** 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.

2. **(5 Points)** Your second task is to find the "**pain type**" value for each hospital patient having a "**blood pressure**" value **that is greater than 140 <u>using a list comprehension</u>** 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.

3. **(5 Points)** Your third task is to find the index value for each hospital patient having a "**gender**" value of "**FEMALE**" and a "**pain type**" value that ***<u>is not</u>*** "**ASYMPTOMATIC <u>using a list comprehension</u>**. 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 = [[2112, 'YYZ', 81], [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)
--------
```

4. **(5 Points)** For your fourth task, implement this same logic **<u>using a list comprehension.</u>** 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

5. **(5 Points)** Your fifth task is to create a user defined function that accepts as input one of the six hospital patient 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 word 'KEY' 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 ('KEY0', 'KEY1', 'KEY2'). If we apply this concept to the first four elements of our "type of chest pain" list we'd have a dict object that looks like this:

First four elements of the type of chest pain list:

['TYPICAL_ANGINA', 'ASYMPTOMATIC', 'ASYMPTOMATIC', 'NON_ANGINAL']

New dict Object:

{'KEY0' : 'TYPICAL_ANGINA', 'KEY1': 'ASYMPTOMATIC', 'KEY2': 'ASYMPTOMATIC', 'KEY3':''NON_ANGINAL'}

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

6. **(5 Points)** Now that you have your reusable user-defined function, use it to create two new **dict** objects containing the first twelve elements of the '**Age**' and '**Diagnosis**' lists, respectively. Then, use these dict objects to tell us the Diagnosis and Age of the hospital patient identified by the 'KEY10' key value.

Be sure to <u>**include some commentary within formatted Markdown cells**</u> explaining your approach to solving each of the individual problems. Save your work to a Jupyter Notebook and submit it via the Module 2 Assignment page within Canvas. Be sure to save your Notebook using the nomenclature we've been using, i.e., **first initial_last name_M2_assn**" (e.g., J_Smith_M2_assn_).