Text

Description automatically generated

In this screenshot, we have items 1-1, 1-2, and 1-3, demonstrating how to create, add more data, and change an element of data in terms of a list.

For 1-1, we begin with a string that is then formed into a list using the .split() method, creating a new list.

For 1-2, we use a user input of a string, and concatenate it to the end of our list, demonstrating how to add data.

For 1-3, we again use a user string input, (using the same function as the previous task) but this time use the indexability of lists to replace a single element of a list with that user input.

Text

Description automatically generated

In this screenshot, we demonstrate tasks 1-4, 1-5, and 1-6, showing how to remove data, index the list, and use functions to process lists.

For 1-4 we use the .remove() method, along with a list index to tell the computer to remove a specific element from our list.

For 1-5 we simply use a print() function and the indexable feature of lists to print a given index. I was considering user a user integer input here, but decided that adding a new function for user integer input between 0 and the number of elements in the list simply wasn’t worth the space it required.

For 1-6 we use two self-made functions called list\_mult() and is\_string(). The former multiplies all integers in the list, and the latter checks whether or not the value of an element in the list is a string. For more detail on these functions, please see the end of this document, where I discuss the functions themselves, rather than their applications such as here.

Diagram

Description automatically generated with medium confidence

Here we have tasks 4, 7, and a number of duplicate tuple tasks for 2.

Tasks 4 (Complicated List Comprehension) and 7 (Using some methods to accomplish a task with lists) I rolled together, as I felt they fit nicely as two sides of the same task. I used a repeating series of if statements and try features while iterating through the list’s index, to try to give a nice, yet complicated list comprehension.

Next, I began the process of repeating most step 1’s with tuples, and varying degrees of similarity to their list-based counterparts, starting by introducing a string to be turned into a tuple, much like the string turned into a list for 1-1

Text

Description automatically generated

Now we continue our mimicry with tuples! This screenshot displays 2-1, 2-2, 2-3, 2-4, and 2-5.

Most of these steps are very accurate to their list friends, with the exceptions of 2-3, and 2-4 where we begin to run into difficulty with the immutability of tuples. To ‘change data’ in a tuple (which is technically impossible) we have to convert that tuple into a list, change the list, and then form a new tuple carrying the same data as that list, in order to create the illusion of a changed tuple. This becomes an issue both with replacing elements as well as removing them completely.

Graphical user interface, text

Description automatically generated

Lastly, I rolled together 2-6 and 3, in order to use a function to perform a similar task as another function that uses lists as arguments rather than tuples. This function, much like the function used in 1-6 checks whether or not a given element in a data structure (a tuple in this case) is a string, or an integer. For more information on this function, please see the last page of this document.

Text

Description automatically generated with low confidence

This screenshot shows all of the data-related functions in the program (excluding the user input functions which have little bearing on this particular project.)

The top and bottom functions are relatively similar, performing the same process for both lists and tuples. However, the tuple function requires use of a temporary list, as again, tuples cannot be edited directly, while the function for lists can simply make changes in the list itself.