**Objective:**

The objective of this lab exercise is to give you practice in programming with one of Python’s most widely used “container” data types -- the *List* (commonly called an “Array” in most other programming languages).  More specifically you will demonstrate how to:

1. Declare *list* objects
2. Access a *list* for storing (i.e., writing) into a cell (a.k.a., element or component) and retrieving (i.e., reading) a value from a *list* cell/element/component
3. Iterate through a *list* looking for specific values using a **for**…**in** loop repetition construct
4. Pass an entire *list* argument to a function
5. Process a *list* parameter received by a function

**Discussion:**

For this exercise you will be simulating a Weather Station responsible for recording hourly temperatures and reporting on certain characteristics (e.g., average temperature) from your recordings.  You will also be expanding on your experience in creating functions and passing them more complex parameters (i.e., complete list arguments – sometimes empty and sometimes full).

**Specifications:**

**DDI&T** a Python program to input, store, and process hourly temperatures for each hour of the day (i.e., 24 temperatures).  Your program should be divided logically into the following parts:

1. In function **main**() declare an empty List container:

         HourlyTemperatures = []

1. Pass the empty HourlyTemperatures list to a function,

GetTemperatures(HourlyTemperatures)

         This function must interactively prompt for and input temperatures for each of the 24 hours in a day (0 through 23).  For each temperature that is input, verify that its value lies in the range of minus 50 degrees and plus 130 degrees (i.e., validate your input values).  If any value is outside the acceptable range, ask the user to re-enter the value until it is within this range before going on to the next temperature (HINT:  use a nested loop (e.g., Python’s equivalent to a do-while loop) that exits only when a value in the specified range has been entered).  Store each validated temperature in its corresponding element of the list container passed as a parameter to the function (Hint:  Look at the ‘append(…)’ function that can be called on a list container).

1. Next pass the filled list container to a function,

ComputeAverageTemp(HourlyTemperatures)

         This function computes the average temperature of all the temperatures in the list and returns this average to the calling function.

1. Finally, pass the filled list container and the computed average temperature to another function,

DisplayTemperatures(HourlyTemperatures, AverageTemp)

         which displays the values of your temperature list in a columnar format followed by the values for the high temperature, low temperature, and average temperature for the day.  **NOTE:  If you want to create separate function(s) to find and return the high and low temperature values, then feel free to do so!**

The resulting output should look something like this:

Hour                          Temperature

00:00                                42

01:00                                42

. . . . .                                . . . // Your program output must include all of these too!

22:00                                46

23:00                                48

High Temperature:          68

Low Temperature:           42

Average Temperature:    57**.**4

5.      Since you have several created functions to perform each of the major steps of your solution, your **main**(): function should be quite simple.  The pseudo code for main() might look something like this:

                  main()

                       #declare any necessary variable(s) and HourlyTemperatures list

**while** user-wants-to-process-another-days’-worth-of-temperatures

call GetTemperatures(HourlyTemperatures)

                                 call ComputeAverageTemp(HourlyTemperatures)

                                 call DisplayTemperatures(HourlyTemperatures, AverageTemperature)

                                 ask if user want to process another days’ worth of temperatures

6.      **Test your program with** **at least two (2) different sets of temperatures**.  Make sure you enter values to adequately test your temperature-validation code (e.g., temperatures below –50 and above +130 degrees).

**Deliverable(s):**

Your deliverable should be a Word document with screenshots showing the sample code you have created, and discuss the issues that you had for this project related to AWS and/or Python IDE and how you solved them.

**Turn in the properly documented source listing of your program and complete outputs from at least TWO (2) test “runs”.**  Additionally, turn in screen captures of some of your interactive inputs demonstrating that your program properly detects invalid inputs and prompts the user to re-enter the temperature(s).