"Introduction to Programming" - Lab 6

Exercise Sheet – Practice Session (our last practice session)

This exercise sheet examines the concepts covered in recent lectures (as well as content we may cover this week).

Setup:

1. Note: anywhere I write <text> (text enclosed in a pair of angle brackets), I want you to change this text to the text I'm looking for (making sure to remove the angle brackets), so for example when I say:
    a. # <todays date>, you write:
    b. # Tuesday 12-10-2021
2. Create a new Folder called "Programming_Folder", if it does not exist. This will be your course Project folder.
3. Download from Canvas the zip file " Lab<number>.zip", unzip and place the contents in the "Programming_Folder" folder.
4. Rename Lab<number> to the number of this Lab
5. The unzipped folder contains, two Python files:
    a. my_functions.py
        i. my_functions.py - is where you will place all the functions created for this lab
    b. main.py
        i. main.py - is the primary file you will call when testing your code
        ii. all of the functions created in 'my_functions.py', will be called/invoked from your 'main.py' file
    c. In these files, I give one example where I create a function called 'print_function()' in my_functions.py and I call it from main.py
6. You should understand all of the code in these two files, if not ask me and I will explain
7. Make sure you add your name and student number to both 'main.py' and my_functions.py'
8. Once complete, and prior to the submission deadline, please upload your updated version of 'functions.py' to Canvas. Do not upload your version of 'main.py', as I will test the code in your 'my_functions.py' file, with my own code.

Exercises:

1. I want you to rewrite the *seasons()* function from Lab 4. Within this function, define a list of the seasons, and select the season based on the input value where 1=Winter, 2=Spring, 3=Summer, and 4=Autumn. Remember list indexing begins at 0. All other error message handling, param names, etc., in the original function, must be kept.

2. This question examines string slicing. Write a function, *slice_reverse(input_value)*, which determine if the input_value is a palindrome i.e. reads the same backwards as forwards. The program should return *True* or *False* (booleans) depending on the input. Do not use the Python reverse() function. Examples:

   slice_reverse(12321) -> True (This is the boolean True and not the string "True")

   slice_reverse([1, 2, 3, 2, 1]) -> True

   slice_reverse(“rotavator”) -> True

   slice_reverse(("r","o","t","a","v","a","t","o","r”)) -> True

   slice_reverse(“ ”) ->True (a string space)

   slice_reverse(“abcdba”) -> False

3. I want you to create a function called *add_to_list(value, list)*, which will return a sorted list. This function will add *value* to the *list* if the *list* does not already contain the *value.* For now, you can assume the *list* param is already sorted. You can use the python function “sort()” to sort your returning list. Sort() will not allow you to mix ints, floats and strings. In your function set an appropriate default value for the *list* param. Examples:

   add_to_list(5, [1,3,7,9]) -> [1,3,5,7,9]

   add_to_list("c", ["a","b","d","e"]) -> ["a","b","c","d","e"]

   add_to_list(5, [1,5,7,9]) -> [1,5,7,9]

   add_to_list(5) -> [5]

   add_to_list(5, 5) -> “Incorrect value defined for param list”

   add_to_list(5, ["a","d","e"]) -> “sort() does not like this mixture of elements”

4. I now want you to create a function called *add_to_list_no_sort(value, list)*, which will return a sorted list. This function will add *value* to the *list* if the *list* does not already contain the *value.* For now, you can assume the *list* param is already sorted. In your function set an appropriate default value for the *list* param. In this function, you cannot use the python function “sort()” to sort your returning list. But you can now mix ints, floats and strings. If mixing ints, floats and strings, use ASCII values for strings when comparing.

   You can make 3 assumptions:

1. As we have not covered Loops in great detail, you can assume the max length of *list* is 4 elements
2. The input list is already sorted
3. The input list consists of only type of value, i.e., all ints, all string, etc.

Examples:

add_to_list_no_sort (5, [1,3,7,9]) -> [1,3,5,7,9]
add_to_list_no_sort ("c", ["a","b","d","e"]) -> ["a","b","c","d","e"]
add_to_list_no_sort (5, [1,5,7,9]) -> [1,5,7,9]
add_to_list_no_sort (5) -> [5]
add_to_list_no_sort (5, ["a","b","d","e"]) -> [5, "a","b","d","e"]
add_to_list_no_sort (5, 5) -> "Incorrect value defined for param list"

Error Checking:

1. There is no error checking in this lab, but...
    a. I want you to try and break your code :)
    b. Change the inputs to the wrong type, the wrong value, and document what happens as a comment in your code.
    c. Try the break challenge for each function you have created. For now you only need to try to break each function once, and then fix for this once.