CompSci 101 Lab 6 — More on Python `list` Objects

Topics covered:

- List methods, e.g. `sort()`, `index()`, `reverse()`, `append()`, `pop()`, `insert()`
- Slicing Lists
- Updating the elements in a list
- The inbuilt functions which can be used with lists, e.g. `sum()`, `min()`, `max()`

Programming Exercises

IMPORTANT: for each of your programs you need to add a docstring at the top of the program. The docstring should contain your name, your username, date and a short description of the program.

Notes

1. The skeleton code for each of the Lab 6 programs is provided. Each program requires you to complete a function. As you complete each function you can test that the function is correct by pasting the whole function (including the header) into CodeRunner3 and pressing the CHECK button.
2. After the description of each program there is a textbox with some example code and beside it a textbox containing the expected output for that code.

Question 1 Program 1
Complete the `have_same_start_total(list1, list2)` function which takes two lists of numbers as parameters. The function returns `True` if the first four elements of both lists have the same total (i.e., add up to the same number), `False` otherwise.

Note: if either of the parameter lists has less than four elements, the function returns `False`.

```
numbers1 = [1, 4, 6, 2, 9, 8]
numbers2 = [4, 0, 8, 1, 9, 7, 1]
print(have_same_start_total(numbers1, numbers2))

numbers1 = [1, 4, 5, 2, 9, 8]
numbers2 = [4, 0, 8, 1, 9, 7, 1]
print(have_same_start_total(numbers1, numbers2))
```

```
True
False
```

Question 2 Program 2

Complete the `swap_elements(numbers_list, number1, number2)` function which is passed three parameters: a list of numbers and two integer values. If both integer values (the second and third parameters) are elements of the list of numbers, the function swaps the two values. If both integer values are NOT elements of the list of numbers the function does nothing.

Notes

- The function does not return a result, it makes changes to the parameter list.
- If there is more than one occurrence of the value to be changed then the value closest to the start of the list of numbers is changed.

```
numbers = [9, 0, 2, 5, 6, 4, 5]
print(numbers)
swap_elements(numbers, 2, 4)
print(numbers)
print()

list1 = [9, 0, 9, 5, 6, 6, 5]
swap_elements(list1, 9, 6)
print(list1)
```

```
[9, 0, 2, 5, 6, 4, 5]
[9, 0, 4, 5, 6, 2, 5]

[6, 0, 9, 5, 9, 6, 5]
```

Question 3 Program 3

Complete the `get_sorted_increased_decreased(numbers_list)` function which is passed a list of whole numbers as a parameter. The function returns a new list with elements which are the numbers from the parameter list increased or decreased tenfold, i.e. if the parameter list element is less than 100 then it is multiplied by 10, otherwise it is divided by 10 using the `"//"` operator. The list which is returned is **sorted from largest to smallest**.

Notes

- If the parameter list is empty, the function returns an empty list.
- You MUST use the `append()` method to add numbers to the end of the list.

```
numbers_list = [31, 636, 2042, 40, 447]
print(numbers_list)
numbers_list2 = get_sorted_increased_decreased(numbers_list)
print(numbers_list2)
print()
print(get_sorted_increased_decreased([11, 4559, 241, 12, 924]))
```

```
[31, 636, 2042, 40, 447]
[400, 310, 204, 63, 44]

[455, 120, 110, 92, 24
```

Question 4 Program 4

Complete the `increase_decrease(numbers_list)` function which is passed a list of whole numbers as a parameter. The function makes changes to the elements of the parameter list by increasing or decreasing them tenfold: if the parameter list element is less than 100 then it is multiplied by 10, otherwise it is divided by 10 using the `"//"` operator.

Note: the function does not return a result, it makes changes to the parameter list.

```
numbers_list = [31, 636, 2042, 40, 447]
print(numbers_list)
increase_decrease(numbers_list)
print(numbers_list)
print()

numbers_list = [11, 4559, 241, 99, 100]
increase_decrease(numbers_list)
print(numbers_list)
```

```
[31, 636, 2042, 40, 447]
[310, 63, 204, 400, 44]

[110, 455, 24, 990, 10]
```

Question 5 Program 5

Define the `get_shortest_length(word_list)` function which is passed a list of strings as a parameter. The function returns an integer, the length of the shortest word in the parameter list.

Note: if the parameter is the empty list, the function returns 0

```
word_list = ['fish', 'barrel', 'like', 'shooting', 'sand', 'bank']
shortest_length = get_shortest_length(word_list)
print(shortest_length)
print()

word_list = ['cat', 'the', 'a', 'bag', 'let', 'out', 'can']
print(get_shortest_length(word_list))
print()

word_list = ['it', 'the', 'the', 'the', 'out', 'big', 'I']
print(get_shortest_length(word_list)
```

```
4
1
1
```

Question 6 Program 6

Firstly, copy the `get_shortest_length(word_list)` function from Program 5 into the Program 6 file.

Complete the `remove_short_words(word_list)` function which is passed a list of strings as a parameter. The function first makes a call to the `get_shortest_length()` function developed in the previous question to find the length of the shortest word in the list. Then the function uses the `pop()` method to remove all the words in the parameter list which have the shortest length.

Notes

- The function does not return a result, it makes changes to the parameter list.
- You can assume that the get_shortest_length() function is included in the testing code and you do **not** need to include this function in your answer.

```
word_list = ['fish', 'barrel', 'like', 'shooting', 'sand', 'bank']
remove_short_words(word_list)
print(word_list)
print()
word_list = ['cat', 'the', 'the', 'bag', 'let', 'out', 'can']
remove_short_words(word_list)
print(word_list)
print()
word_list = ['it', 'the', 'the', 'the', 'out', 'big', 'if']
remove_short_words(word_list)
print(word_list)
```

```
['barrel', 'shooting']

[]

['the', 'the', 'the', 'out', 'big']
```

Question 7 Program 7

Complete the `insert_word(word_list, word_to_insert, before_this_word)` function which is passed three parameters:

- a list of strings,
- a word to be inserted into the list of words,
- the word before which the word is to be inserted.

If the word to be inserted is not already in the list of words, the function inserts the word into the correct position in the list of words.

Notes

- The function does not return a result, it makes changes to the parameter list.
- If the word to be inserted is already in the list of words then the function does nothing.
- If the third parameter word is not in the list of words then the function does nothing.

```
word_list = ['fish', 'barrel', 'like', 'shooting', 'sand', 'bank']
insert_word(word_list, "water", "bank")
print(word_list)
print()
word_list = ['cat', 'the', 'bag', 'let', 'out', 'can']
insert_word(word_list, "dog", "let")
print(word_list)
```

```
['fish', 'barrel', 'like', 'shooting', 'sand', 'water', 'bank']
['cat', 'the', 'bag', 'dog', 'let', 'out', 'can']
```