School of Information Systems
Academic Year 2019/20
Term I

IS111: Introduction to Programming

Lab Exercise 05: Lists

Optional submission deadline: **Sun 22 Sep 2019 @ 2359 hrs**

INSTRUCTIONS

You are expected to read through the accompanying lesson 05 notes before attempting lab exercise 05. You should attempt the questions using **Visual Studio Code**. Each question should be submitted on a different file.

Do not submit Level 1 questions; use the discussion forum for that.

To submit, you should:

1. Name your 3 files as **week5-qn2-1.py**, **week5-qn2-2.py** and **week5-qn3-1.py**.
2. Submit the 3 .py files to **eLearn → Assignments** by the stipulated deadline.

Level 1 Questions (no submission required; use the discussion forum)

1.1 List Intersection

Write a Python function `get_intersection` that takes in two list parameters `list1` and `list2`. The function should then return a list of all the (unique) elements that appear in both lists. If there are no common elements between the two lists, the function should then return an empty list.

You can use the following code snippet to test your code:

```
# the following statements should return True
print(get_intersection([1,2,'a','#',2], [2,3,3,'#']) == ([2,'#'] or
['#',2]))
print(get_intersection([1,2,3],[4,5,'d']) == [])
```

1.2 Square of Odds

Write a Python function `compute_square_of_odds` that takes in a single list parameter `input_list` that contains a list of integers. The function returns a list where all the elements with odd indices are squared.

For example, if `input_list=[2,6,3,-1,7,8]`, then the function should return `[2,36,3,1,7,64]` (because the elements in the list at indices 1, 3 and 5 are squared).

You can use the following code snippet to test your code:

```
# the following statements should return True
print(compute_square_of_odds([2,6,3,-1,7,8]) == [2,36,3,1,7,64])
print(compute_square_of_odds([33]) == [33])
```

Level 2 Questions

2.1 Manhatten Distance

The Manhattan Distance D between two vectors $X = \{x_1, x_2, ..., x_n\}$ and $Y = \{y_1, y_2, ..., y_n\}$ is given by $D = \sum_{i=1}^{n} |x_i - y_i|$. For instance, if $X = \{1, 4, 10, 5\}$ and $Y = \{5, 2, 8, 5\}$, then the Manhattan Distance D is given by:

$$D = |1 - 5| + |4 - 2| + |10 - 8| + |5 - 5| = 8.$$

Note that $|a - b|$ is equivalent to obtaining the absolute (i.e., positive) value of $a - b$.

Write a Python function `compute_manhattan_distance` which takes in two parameters *X* (containing the list of *x* values) and *Y* (containing the list of *y* values), and returns the Manhattan Distance *D*. If the lengths of *X* and *Y* do not match, your function should return `None`. You may assume that *X* and *Y* are lists that contain valid numbers.

You can use the following code snippet to test your code:

```
# the following statements should print True
print(compute_manhattan_distance([1,4,10,5], [5,2,8,5]) == 8)
print(compute_manhattan_distance([1,4,10,5,12,52], [5,2,8,5,18]) ==
None)
```

2.2 Two-Dimensional Array (List of Lists)

Write a Python function `generate_2d` that takes in two positive integer parameters *a* and *b*, and generates a list that contains *a* lists, and with each of these *a* lists containing *b* elements. The *b* elements in each of the *a* lists are numbered from 1 to *b*.

For instance, if `a=3` and `b=4`, then the list that is generated is `[[1,2,3,4], [1,2,3,4],[1,2,3,4]]`.

You can use the following code snippet to test your code:

```
# the following statements should print True
print(generate_2d(3,4) == [[1,2,3,4],[1,2,3,4],[1,2,3,4]])
print(generate_2d(2,5) == [[1,2,3,4,5],[1,2,3,4,5]])
```

3 Brownie Points 🍪

3.1 Sum of Neighbors

Write a Python function `sum_of_neighbors` that takes in a single list parameter `input_list`. The function returns a list with the same number of elements as the original list, such that each integer in the new list is the sum of its neighbors and itself in the original list.

For instance, if `input_list = [10, 20, 30, 40, 50]`, then the function should return the list `[30, 60, 90, 120, 90]`.

You can use the following code snippet to test your code:

```
# the following statements should print True
print(sum_of_neighbors([10,20,30,40,50]) == [30,60,90,120,90])
print(sum_of_neighbors([23]) == [23])
print(sum_of_neighbors([56,-10,25,-32]) == [46,71,-17,-7])
```

💾 11 Sep 2019