**Week 5 In-Class Exercises (Extra)**

**(Lists)**

**Note:** These exercises are for you to do using Notepad++ and Anaconda Prompt.

**Q1: Reverse Words [ \*\* ]**

Write a function called reverse\_words() that takes in a piece of text that consists of multiple words. The function **returns** a string that is a transformed version of the text where each word is reversed.   
  
You can assume that the words in the given text are separated by spaces, i.e., there is a single space between two adjacent words.

For example, reverse\_words("I study at SMU") returns "I yduts ta UMS". reverse\_words("Python is a programming language.") returns "nohtyP si a gnimmargorp .egaugnal". If the input text is an empty string, the function returns an empty string.

**Q2: List of Tuples [ \*\*\* ]**

Take a look at the starting code movies.py that’s given to you. Inside the file, you are given a list of tuples, where each tuple represents the title, genre and duration (in minutes) of a movie.

Inside another file called movies\_utility.py, implement the following functions:

* get\_average\_duration(): This function takes in a list of tuples as described above. The function returns the average duration of all the movies in the list. If the list is empty, the function returns 0.0.
* get\_num\_movies\_of\_genre(): This function takes in a list of tuples as described above together with a string that indicates a genre. The function returns the number of movies belonging to that genre in the give list. If the list is empty, the function returns 0.
* get\_title\_of\_longest\_movie(): This function takes in a list of tuples as described above and returns the title of the movie whose duration is the longest among all the movies in the list. If the list is empty, the function returns an empty string.
* get\_movies\_with\_keyword(): This function takes in a list of tuples as described above together with a string that represents a keyword. The function returns a new list of tuples where each tuple still represents a movie. The returned list contains those movies in the original list whose titles have the specified keyword as a substring. If the original list is empty, the function returns an empty list.

**Q3: Average Height [ \*\*\* ]**

Write a function called compute\_avg\_height. The function takes in a single parameter, which is a string that consists of some people’s heights. An example of such a string is "Jonathan Li:1.75m, Lim, Josephine : 1.59m , George Khoo: 1.7 m". We can see that for each person, the person’s name is given, followed by a colon. After that there may be zero, one or more spaces, followed by the person height. The person’s height is followed by the letter 'm', but there might be zero, one or more spaces before 'm' and after 'm'. The function compute\_avg\_height should return the average height of all the people listed in the string.

You can assume that the input string is always well formatted as described above. To be more specific, you can assume the following:

* Digits and the symbol '.' appear only as part of a person’s height.
* A person’s name doesn’t contain the symbol ':'.
* However, a person’s name may be expressed as '<surname>, <given name>', such as 'Lim, Josephine'.
* The string parameter contains at least one person’s height.