# Tutorial 8 Worksheet

COMP1117B Computer Programming I 2018-2019

**Tutorial objectives:**

- To practice strings and lists in Python.

**Tutorial 8 Exercises Deadline: April 2, 2019 11:20**

## 1. Demo: Print the list of integers in descending order

Write a program to read a list of integers separated by "`,`", then print the list of integers in descending order.

We can first create a main function to define the flow of the program.

```
def main():
    numbers = readNumbers()
    ordered = sortList(numbers)
    printList(ordered)

main()
```

To read numbers in a comma separated list, we can make use of the `split()` function to split the input string into a list of numbers. However, since the list of values after splitting are of the type String, we have to convert each of them to integer before returning.

```
def readNumbers():
    s = input()
    numbers = []
    for numStr in s.split(','):
        numbers.append(int(numStr))

    return numbers
```

Next, to sort a list of numbers, we can make use of the `sorted()` function. Since we need to sort in descending order, we have to specify "`reverse=True`".

```
def sortList(numbers):
    ordered = sorted(numbers, reverse=True)
    return ordered
```

Finally we can print the list in comma separated form. This can be done using the `join()` function of

String.

```
def printList(myList):
    print(myList)
```

You can test the program now to see if it works.

An alternative method is to make use of the `sort()` function of a list to sort the list, then use the `reverse()` function to reverse the order.

```
def sortList(numbers):
    numbers.sort()
    numbers.reverse()
    return numbers
```

Are there any differences for the two methods? Let's compare the two methods.

```
def sortList(numbers):
    ordered = sorted(numbers, reverse=True)
    return ordered
```

```
def sortList(numbers):
    numbers.sort()
    numbers.reverse()
    return numbers
```

You can see that `sorted()` will return the sorted list, while `list.sort()` will sort the list in-place. That means in the second method, the original list will be modified.

To confirm such observation,you can try both version of sorting and print the original list afterwards.

**Exercise 8.1 Determine a sorted list**

Consider the following program.

```
def main():
    numbers = readNumbers()
    if isSortedList(numbers):
        print('Yes')
    else:
        print('No')

main()
```

Complete the function `readNumbers()` that reads and returns a list of integers until a zero; and the function `isSortedList()` that accepts a list of numbers and return True if the list of numbers is sorted, or False otherwise.

| Test case | Input | Output |
|---|---|---|
| Yes | 1<br>2<br>3<br>0 | Yes |
| No | 1<br>3<br>2<br>0 | No |

**Exercise 8.2 Character frequency**

Read an input string from user. Report a list of unique characters in the string (case sensitive) and the number of times the character appearing in the string, in the same order as the first time that character appearing in the string. You may consider to use one list to store the characters and one list to store the corresponding frequency.

| Test case | Input | Output |
|---|---|---|
| Hello | Hello | H 1<br>e 1<br>l 2<br>o 1 |
| aaaaa | aaaaa | a 5 |

**Exercise 8.3 Selection Sort**

Read an input string from user and store it as a list of characters. You may assume that the user would only input a string with 'a'-'z' characters. Print the original list. Then implement your own selection sort to sort this list and print the result.

| Test case | Input | Output |
|---|---|---|
| hello | hello | ['h', 'e', 'l', 'l', 'o']<br>['e', 'h', 'l', 'l', 'o'] |
| aaaaa | aaaaa | ['a', 'a', 'a', 'a', 'a']<br>['a', 'a', 'a', 'a', 'a'] |

**Exercise 8.4 Read data from a file**

Read data from a file "testfile.txt". The file contains some numbers (one number per row):

```
10
20
30
40
50
60
70
```

Read the numbers in this file and output the average of these numbers.

| **Test case** | **Output** |
|---|---|
| 1 | `40.0` |