

## Lab Questions: Lab Session 9

Deadline: 25.10.2017 11:59pm SGT

Complete all assignments below. For those questions that are marked with an asterisk \*, i.e. **Questions 8 and 9**, create the script/function files as requested. Once you are done with it, submit the file(s) via iNTU. Remember to put plenty of comments for all assignments, as this helps us to better understand your program (which might give you higher marks).

**Important!!! Make sure your scripts work properly, as we give 0 marks otherwise. Please name the scripts according to the requirements, and upload each file separately and not in a Zip file or similar. The submission system closes at the deadline. Hence after that, you will get no marks for your solution.**

1. Create three lists *L1*, *L2* and *L3*, filled with random data (it does not matter what data you actually put in the lists). Open a file named `my_data.txt` using `w` mode and save in it the contents of *L1* (you can open the file with a text editor to see the result). Then, open again the file `my_data.txt` using `w` mode and save in it the contents of *L2* (you can open again the file with a text editor to see the result). Finally, open the file `my_data.txt` using `a` mode and save in it the contents of *L3* (you can open again the file with a text editor to see the result). Note that a list can be converted to a string using the `str()` built-in function.
2. Download the file `transport.txt` from iNTU Learn and open it in a text file editor to check how it is organised (it contains the average daily public transport ridership in Singapore for each year from 1995 to 2016, for MRT, LRT, Bus and Taxi). Write a script `transport.py` that will load the data from that file and print in the console the average daily public transport ridership for the entire period 1995 to 2016 for all four transports: MRT, LRT, Bus and Taxi.

For example, the output of the script can look like this:

```
The average MRT ridership is 1638954.55
The average LRT ridership is 72181.82
The average Bus ridership is 3228545.45
The average Taxi ridership is 671727.27
```

3. What will be the output of the following programs ?

(a)

```
try:
    print("a")
except:
    print("b")
else:
    print("c")
finally:
    print("d")
```

(b)

```
try:
    print("%d",5/0)
except:
    print("b")
else:
    print("c")
finally:
    print("d")
```

4. Write a Python program `copy_no_space.py` that will ask the user to input a source file name and a target file name, and that will copy the content of the source file to the target file, while removing all space characters (space, tab, newline, return, etc.). An error check must be done using `try/catch` in case the source file input by the user does not exist (in which case you can print a message explaining that the source file does not exist).
5. Write a Python program `count_occurences.py` that will ask the user to input a source file name and a character, and that will count how many times the character is present in that file. An error check must be done using `try/catch` in case the source file input by the user does not exist (in which case you can print a message explaining that the source file does not exist).
6. Debug the following programs (you can use any technique that you want for that), it should execute without errors and it should have the functionality described. You can download the Python files on iNTU Learn.
  - (a) We would like to generate 100 random integers in  $[0,10]$  and print them in the console.

debug1.py

```
import random

L = [random.randint(0,10) for i in range(100)]

for i in L
    print("the number is %d",L[i])
```

- (b) We would like to ask the user to input positive integers, and to continue asking him as long as he didn't 10 of them (in which case we quit). We would also like to give him the opportunity to stop the program by simply entering the special number -10. You can assume that the user only inserts integers.

debug2.py

```
count = 0
while count < 10:
    my_int = -1
    while my_int<0:
        my_int = int(input('Insert a positive integer: '))
        if my_int== -10:
            break

    count+=1
    print('You have entered %d positive integers' % (count))
```

- (c) We would like to generate a list of 100 random integers in [0,10] and ask the user which element he would like to see. The program will then print this element in the console. You can assume that the user only inserts integers.

debug3.py

```
import random

L = [random.randint(0,10) for i in range(100)]

my_element = input('which element would you like to print ? ')
print('The element is %d' % (L[my_element]))
```

7. Download the file `temp_pressure.txt` from iNTU Learn. It contains temperature and air pressure measurements: one measurement per line, the first element of each line representing the temperature in degrees Celsius, and the second element representing the air pressure in millibar. Unfortunately, the researcher that generated this file is not very reliable, and the file contains some errors. Write a script `temp_pressure.py` that will read the data from the file, compute and print the overall temperature and air pressure average. Use appropriate `try/catch` block to handle the file errors: if either the temperature or the air pressure measurement contains an error, print in the console the file line where this error is located. Moreover the entire measurement of the erroneous lines should be discarded and not used for the average computations).
8. \* Download from iNTU Learn the file `pokemons.txt` that contains a list of Pokemons, their index (from 1 to 718 included), and their type. The file is sorted according to the Pokemons names. Write a Python script `<YourMatricNo>_Lab9_catching_pokemons.py` that will read the content of this file (the potential non-existence of the file must be handled with an appropriate `try/catch` block) and that will create another file `pokemons_new.txt` that contains the same data, but this time sorted according to the pokemons indexes.
9. \* A student was requested to produce a function named `sum_and_product` that takes as input a matrix of unknown size (implemented as a Numpy array), and that outputs two values: the sum of all the elements of the matrix, and the product of all the elements in the matrix. The student came up with the following code (available on iNTU Learn):

sum\_and\_prod.py

```
import numpy as np

def sum_and_prod(my_mat):
    (rows,columns) = my_mat.shape
    for r in rows
        my_sum = 0
        my_product = 0
        for c in columns
            my_sum += my_mat[r,c]
            my_product *= my_mat[r,c]

    return (my_sum,my_product)
```

His code contains syntax as well as logical errors. Debug this program, in order to ensure that it executes properly and that it actually does what it is supposed to do. Copy the corrected code in a file `<YourMatricNo>_Lab9_sum_and_prod.py`.