CSCI 2040 A/B: Introduction to Python **2018-2019 Term 1**

Lab Assignment 5

Instructor: John C.S. Lui and S.H. Or Due: 17:00 on Nov. 23rd, 2018

Notes

1. You are allowed to form *a group of two* to do this lab assignment.
2. You are strongly recommended to bring your own labtop to the lab with Anaconda and Pycharm installed.
3. **Python 2.x** and **Python 3.x** are both acceptable. But you need to specify the python version in requirements.txt. For example, if your scripts are required to run in Python 2.7, the following line should appear in requirements.txt:

   ```
   python_version == '2.7'
   ```

4. (Not recommended.) Use the Windows PC in SHB 924 with your CSDOMAIN account. Login and open "Computer" on the desktop to check if an "S:" drive is there. If not, then you need to click "Map network drive", use "S:" for the drive letter, fill in the path `\\ntsvr6\userapps` and click "Finish". Then open the "S:" drive, find the **Python2** folder, and click the "IDLE" shortcut to start doing the lab exercises.
5. Your code should only contain specified functions or classes. Please delete all the debug statements (e.g. print) before submission.

In this lab assignment, you will practice functional programming, visualization, and get a sense of Pythonic way of coding.

Exercise 1 **(20 marks)**

Let a be the list of values produced by `range(1, 11)`.

(a) Using the function `map` with a `lambda` argument, write an expression that will produce each of the following: **(5 marks)**

- A list of square of the corresponding values in the original list; The expected output should be a list as follows,

  ```
  [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
  ```

- A list where each element is smaller by one than the corresponding element in the original list; The expected output should be a list as follows, **(5 marks)**

  ```
  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
  ```

Note that you should use `lambda` arguments in this part.

(b) Write a ***list comprehension*** that will produce each of the following:

- A list contains values in the original list that are larger than or equal to 6; The expected output should be a list as follows, **(5 marks)**

```
[6, 7, 8, 9, 10]
```

- A list contains values that are the 3 power of even values in the original list. The expected output should be a list as follows, **(5 marks)**

  ```
  [8, 64, 216, 512, 1000]
  ```

Note that in this exercise, you are required to write only one line of code for each expression. We will manually check the correctness of your answer.
Write your script for this exercise in `p1.py`

Exercise 2 (20 marks)

`x` is a list of strings. For example,

```
x = ['python is cool',
'pythom is a large heavy-bodied snake',
'The python course is worse taking']
```

In the example, there is totally 1 occurrence of the word 'python' (case sensitive) in the strings whose length are more than 20 in the list `x`.

In order to count the number of occurrences of a certain word `str` in the strings which are long enough, use `filter` and `reduce` and write a function `word_count(x, n, str)` in the functional programming paradigm.

This function takes a list of strings `x` , a number `n` and the string we want to find `str` as the inputs. The output or the returned value is the total number of occurences of the string `str` in the strings whose length is more than `n`(>n) in the list `x`. The function should be in the following format:

```
def word_count(x, n, str):
    # your code here
```

Hint:

- You cound use `filter` to filter out the string whose length are not long enough and use `reduce` to do the summation.
- `str.count(sub)` return the number of occurrences of substring `sub` in string `str`.

Note that you are required to use `filter`, `reduce` to finish your code. No use of these will result in the deduction of your grade for this exercise.
Write your script for this exercise in `p2.py`

Exercise 3 (40 marks)

Visualization is widely used for data analysis. In this exercise, you will use Python scripts to draw 5 kinds of figures. All the script for this exercise should be in a single file `p3.py`. The package matplotlib is useful for this exercise, which can be imported as following[1]:

[1] https://matplotlib.org/

```
import matplotlib.pyplot as plt
```

Note: your grade for this exercise depends on the readability of your figures. All the plots should be clear to read, and all the plots must contain x-label and y-label.

Histogram (10 marks)

Histogram[2] is a way to visualize the distribution of continuous variables.

- ⌨ Write script in `p3.py` to plot a histogram for the random numbers in `random_numbers` generated by the following scripts. The x-axis is the value of random numbers, and the y-axis is the probability density. *Hint:* You could use `hist()` in `matplotlib`.
- ✍ Save your figure as `histogram.png`. *Hint:* You could use `savefig()` in `matplotlib`.

```
import numpy as np
random_numbers = np.random.normal(0, 1, 1000)
```

Pie chart (10 marks)

Pie chart[3] is a way to visualize the distribution of categorical data.

- ⌨ Write script in `p3.py` to plot a pie chart for the number of students of **3 selected colleges** in CUHK in 2017. You can select any 3 colleges in CUHK. The data for the number of students can be found in `http://www.iso.cuhk.edu.hk/images/publication/facts-and-figures/2017/html5/english/#10`. The categories in the pie chart are the colleges in CUHK, and there should be label text for each category on the figure. *Hint:* You may use `pie()` in `matplotlib`.
- ✍ Save your figure as `pie.png`. *Hint:* You could use `savefig()` in `matplotlib`.

Bar chart (10 marks)

Bar chart is also suitable to visualize categorical variables.

- ⌨ Write script in `p3.py` to plot a bar chart for the number of students of **all the 9 colleges** in CUHK in 2017. There should be label text for each colleges on the figure. *Hint:* We could use `bar()` in `matplotlib`.
- ✍ Save your figure as `bar.png`. *Hint:* You could use `savefig()` in `matplotlib`.

Scatter plot and line chart (10 marks)

Scatter plot[4] and line chart[5] are common ways to visualize the relationship between two continuous variables. Suppose we have two lists of numbers `x_list` and `y_list` generated by the following scripts.

[2] `https://en.wikipedia.org/wiki/Histogram`

[3] `https://en.wikipedia.org/wiki/Pie_chart`

[4] `https://en.wikipedia.org/wiki/Scatter_plot`

[5] `https://en.wikipedia.org/wiki/Line_chart`

```
import numpy as np
x_list = np.linspace(0, 1, 100)
y_list = x_list + np.random.rand(100)
```

⌨ Write script in `p3.py` to draw a scatter plot of `x_list` (x-axis) and `y_list` (y-axis). Draw a line chart for the function $y = x$ in the same figure.
Hint: You may use `plot()` and `scatter()` in `matplotlib`, and you could set the `alpha` blending value to make the dots more transparent.

✍ Save your figure as `scatter_line.png`. *Hint:* You could use `savefig()` in `matplotlib`.

Exercise 4 (20 marks)

Mastering a programming language is not only about the syntax, but also requires one to know the programming style. In this exercise, you will get a sense of the Pythonic way of programming. In a nutshell, a Pythonic way of programming is to utilize Python's features that are designed to make a programmer's life easier. Here are some examples:

1. **Creating list of lists** (using list comprehension).
   Suppose you want a 2-dimensional array that is a list of 4 empty lists. Since Python does not have declaration for a 2-dimensional array, you need to construct it from lists. The wrong way is to append the same list for 4 times (Why it is wrong[6]).

   ```
   # wrong code
   list = []
   list_of_lists = []
   for i in range(4):
       list_of_lists.append(list)
   ```

   The ugly code runs a explicit for-loop.

   ```
   # correct but ``ugly'' code
   list_of_lists = []
   for i in range(4):
       list_of_lists.append([])
   ```

   The Pythonic code has only one line that utilizes list comprehension.

   ```
   # Pythonic code
   list_of_lists = [[] for _ in range(4)]
   ```

2. **Open a file, reading a file**
   Suppose you need to process the contents in a file, line by line. The following is the ugly code, and may forget reading a new line in the `while`-loop or forget closing the file.

   ```
   # ``ugly'' code
   file = open('some_file_name')
   ```

[6] http://cryptroix.com/2016/10/25/python-call-object/

```
line = f.readline()
while line:
    # do something with the line
    line = f.readline() # you may forget this
file.close() # you may forget this
```

In a Pythonic way, we use `with` which automatically close the file after usage, and we do a `for`-loop directly over the file.

```
# Pythonic code
with open('some_file_name') as file:
    for line in file:
        # do something with the line
```

3. **Chained comparison**

```
# ``ugly'' code
if 0 <= x and x <= 100:
    x = x + 1

# Pythonic code
if 0 <= x <= 100:
    x += 1
```

4. **Conditional operator**

```
# ``ugly'' code
if  and x <= 100:
    y = x + 1
else:
    y = x - 1

# Pythonic code
y = x+1 if x <= 100 else x-1
```

5. **Multiple assignment**

```
# ``ugly'' code
x = 0
y = 1

# Pythonic code
x, y = 1, 2
```

More examples can be found in many online posts by searching "Pythonic"[7].

⌨ In this exercise, you need to write a function named `get_average_grades` in `p4.py` that takes the name of the grading file as the input, and returns a list of the average grades for each lab assignments of the Python course. A prototype of your function can be

[7] `https://medium.com/the-andela-way/idiomatic-python-coding-the-smart-way-cc560fa5f1d6`

```
def get_average_grades(filename='grades.csv')
    return average_grades_list
```

By default, the grades are recorded in an input file named `grades.csv`, in the same folder of your scripts. Each line in this file records the grades of a student in the past lab assignments, which are separated by commas (that is called "CSV" file). For example, we have 3 students and 4 lab assignments, and the `grades.csv` has the following contents:

```
60,61,62.5,-1
-1,70,75,73
80,-1,87.5,-1
```

Here, if a student does not submit a lab assignments, his grade is recorded as -1. For example, the student for the first row has grades 60, 61 and 62.5 for the first three lab assignments respectively, and the "-1" indicates that this student does not submit the fourth lab assignment.

The average grade for a lab assignment is the average grade of all the students who submit this lab assignment. For the above example, the average grade for lab assignment 1,2,3,4 are 70, 65.5, 75, 73. **The output is a list of the average grades for each lab assignments (each number should be float which will be compared by the sample answer)**. The return value of `get_average_grades` for the above example should be a Python list:

```
[70, 65.5, 75, 73]
```

Your scripts should not contain any one of the above mentioned 5 kinds of "ugly" code. Your marks will be deducted by 4 for each kind of "ugly" code in your scripts. Your scripts can be in any style that does not contain the above mentioned "ugly" code, you are NOT necessarily required to use the Pythonic code.

Exercise 5 (Optional, Bonus 20 marks)

In this exercise, we will learn how to build a C extension for Python[8].

Python scripts can be slow, especially when you have many `for`-loops[9]. But this should not be a big problem, because only a small portion of your scripts are critical to the running time of your program. One way to boost the performance is to rewrite these critical parts using "faster" programming languages such as C. In fact, many famous packages such as `numpy` and `scipy` are writen in C (or C++), which can be called in Python.

⌨ We note that we repeatedly take average of a list in Exercise 4. Therefore, in this exercise, you will write a C extension function `nn_ave()` insides a module named `myutil`, that takes a Python list as input and returns the average of all non-negative value in this list. Your function should be called as the following example illustrate.

⌨ Then, rewrite your function `get_average_grades` for Exercise 4 in a new file `p5.py` using your self-created new function `myutil.nn_ave()`

[8] `https://docs.python.org/3/extending/building.html`

[9] `https://medium.freecodecamp.org/if-you-have-slow-loops-in-python-you-can-fix-it-until-you-cant-3a39e03b6f35`

```
import myutil
list = [1, 2, 3, 4]
average = myutil.nn_ave(list) # the variable 'average' should take a value of 2.5
```

Hint: we list the following two possible ways for your reference

- The first way (writing C code): 1. you can copy the codes and follow this tutorial [10] to get started. 2. After that, see this blog [11] to know how to process a Python list in C. In particular, the function `PySequence_Fast` can read a Python list to a C object. The function `PySequence_Fast_GET_ITEM` can get a specific item in a Python list.
- The second way (writing Cython and convert to C): Cython is a superset of the Python language, which allows you to add typing information and class attributes that can then be translated to C code and to C-Extensions for Python. You can watch some online tutorials[12].

Submission rules

1. Please name the ***functions***, *script files* and *data files* with the **exact** names specified in this assignment and test all your scripts. Any script that has any *wrong name or syntax error will not be marked.*
2. For each group, please pack **all your *.py script files* and *.png figures*** as a single archive named as

   `<student-id1>_<student-id2>_lab5.zip`

   (if you do Exercise 5, please put the *setup.py*, *.c* or *.cpp* and all other necessary files into this archive, so that we can compile and then run your `p5.py`) For example, if your student ids are 1155012345 and 1155054321, then the file name should be `1155012345_1155054321_lab5.zip`, i.e., just replace `<student-id1>` and `<student-id2>` with your own student IDs. If you are doing the assignment alone, just leave `<student-id2>` empty, e.g, `1155012345_lab5.zip`.
3. Send the zip file to `cuhkcsci2040@gmail.com`,
   - Each group only needs to send *one Email*
   - *Subject of your Email* should be `<student-id1>_<student-id2>_lab5` if you are in a two-person group or `<student-id1>_lab5` if not.
   - No later than 17:00 on Nov. 23rd, 2018
4. Students in the same group would get the same marks. Marks will be `deducted` if you do not follow the submission rules. Anyone/Anygroup who is caught plagiarizing would get 0 score!
5. If you have any question regarding **Lab Assignment 5**, please email to liuxt@cse.cuhk.edu.hk or yli@cse.cuhk.edu.hk.

[10] `https://tutorialedge.net/python/python-c-extensions-tutorial/`

[11] `https://medium.com/@joshua.massover/python-c-extension-example-cef86ffab4ed`

[12] `https://pythonprogramming.net/introduction-and-basics-cython-tutorial/`