CSCI 2040: Introduction to Python **2022-2023 Term 1**

Lab Assignment 4

Instructor: Dr CHUI Yim Pan Due: 23:59 on Wednesday, Nov. 23, 2022

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 laptop to the lab with Anaconda[1] and Pycharm[2] installed. You don't even have to attend the lab session if you know what you are required to do by reading this assignment.
3. Only **Python 3.x** is acceptable. You need to specify your python version as the first line in your script. For example, if your scripts are required to run in **Python 3.6**, the following line should appear **in the first line of your scripts**:

```
#python_version == '3.6'
```

4. For those of you using the Windows PC in SHB 924A (NOT recommended) with your CSDOMAIN account[3], please 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 `\\ntsvr1\userapps` and click "Finish". Then open the "S:" drive, open the **Python3** folder, and click the "IDLE (Python 3.7 64-bit)" shortcut to start doing the lab exercises. You will also receive a paper document and if anything has changed, please be subject to the paper.
5. Your code should only contain specified functions. Please delete all the debug statements (e.g. print) before submission.

Exercise 1 **(30 marks)**

1. We want to get a RPN (Reverse Polish Notation) calculator by reusing the class `CalculatorEngine` as follows. Implement the `RPNCalculator` so that the method `eval(line)` would return the result of the RPN in `line`. (*Hint: read the lecture notes*) **(15 marks)**

   Remark: suppose all inputs are valid(do not need to consider scenarios like $('12**')$ **or** $('123')$

   For example, the expected value of x in the following is 5.

[1]An open data science platform powered by Python. `https://www.continuum.io/downloads`
[2]A powerful Python IDE. `https://www.jetbrains.com/pycharm/download/`
[3]A non-CSE student should ask the TA for a CSDOMAIN account.

```
cal = RPNCalculator ()
x = cal.eval('1␣2␣*␣3␣+')
```

2. We now want to add the modulo division operator, %, to the RPN calculator. Do this by making **changes only to** the class `RPNCalculator`, without modifying the `CalulatorEngine`. Note that you should do modulo in a `try` block, catching the divide by zero error and printing a message: "divide by 0" (exit the program, similar to the implementation of division). **(15 marks)**

```
class Stack(object):
    def __init__(self):
        self.storage = []

    def push (self, newValue):
        self.storage.append( newValue )

    def top(self ):
        return self.storage[len(self.storage) - 1]

    def pop(self ):
        result = self.top()
        self.storage.pop()
        return result

    def isEmpty(self ):
        return len(self.storage) == 0


class CalculatorEngine (object ):
    def __init__(self ):
        self.dataStack = Stack ()

    def pushOperand (self , value ):
        self.dataStack.push(value)

    def currentOperand (self ):
        return self.dataStack.top()

    def performBinary (self , fun):
        right = self.dataStack.pop()
        left = self.dataStack.~~top()~~pop()
        self.dataStack.push( fun(left , right ))
    def doAddition (self ):
        self.performBinary (lambda x, y: x + y)

    def doSubtraction (self ):
        self.performBinary (lambda x, y: x - y)

    def doMultiplication (self ):
        self.performBinary (lambda x, y: x * y)

    def doDivision (self ):
        try:
            self.performBinary (lambda x, y: x / y)
        except ZeroDivisionError :
            print("divide␣by␣0!" )
            exit (1)

    def doTextOp (self , op):
        if (op == '+'): self.doAddition ()
        elif (op == '-'): self.doSubtraction ()
        elif (op == '*'): self.doMultiplication ()
        elif (op == '/'): self.doDivision ()
```

No extra indentation here, i.e. RPNCalcuator should not be an inner class of CalculatorEngine

```
class RPNCalculator ( CalculatorEngine ):
    def __init__(self ):
        # your code here

    def eval(self, line):
        # your code here
```

Save your script for this exercise in `p1.py`

Exercise 2 (20 marks)

Let a be the list of values produced by `range(1, 12)`.

1. Using the function `map` with a `lambda` argument, write an expression that will produce each of the following:

   - A list of square of the corresponding values in the original list; The expected output should be a list as follows, **(5 marks)**

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121]
```

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

  ```
  [3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
  ```

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

2. Write a ***list comprehension*** that will produce each of the following:

   - A list contains values in the original list that are less than or equal to 8; The expected output should be a list as follows, **(5 marks)**

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

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

     ```
     [4, 16, 36, 64, 100]
     ```

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.
Save your script for this exercise in `p2.py`

Exercise 3 **(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, str, n)` in the functional programming paradigm.

This function takes a list of strings `x` , the string we want to find `str` and a number `n` as the inputs. The output or the returned value is the total number of occurrences 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, str, n):
    # your code here
```

Hint:

- You could 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.
Save your script for this exercise in `p3.py`

Exercise 4 (30 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[4]).

   ```
   # 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.

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

```
# ``ugly'' code
file = open('some_file_name')
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"[5].

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

```python
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.

Submission rules

1. Please name the `functions` and script 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.

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

2. For each group, please pack all your script files as a single archive named as

   `<student-id1>_<student-id2>_lab4.zip`

   For example, `1155012345_1155054321_lab4.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_lab4.zip`.

3. Upload the zip file to your blackboard (`https://blackboard.cuhk.edu.hk`),

   - Only one member of each group needs to upload the archive file.
   - Subject of your file should be `<student-id1>_<student-id2>_lab4` if you are in a two-person group or `<student-id1>_lab4` if not.
   - No later than 23:59 on Wednesday, Nov. 23, 2022

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!