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

Lab Assignment 4

Instructor: John C.S. Lui and S.H. Or Due: 17:00 on Nov. 2nd, 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 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. **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.

Exercise 1 (20 marks)

(a) 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*) **(10 marks)**

Remark: suppose all inputs are valid(do not need to consider scenarios like ('1 2 * *') or ('1 2 3').

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

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

(b) 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 `CalcEngine`. Note that you should do modulo in a `try` block, catching the divide by zero error and printing a message: "divide by 0". **(10 marks)**

[1] An open data science platform powered by Python.

[2] A powerful Python IDE.

```
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()
        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()

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

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

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

Exercise 2 (30 marks)

Read File. Write a program that accepts two arguments: a string and the other string (file name),

- Here we **assume the input string falls into only two cases: a single word or a phrase.** We will use examples to illustrate why we emphasize this assumption. Use `print` to print all lines in the file that contain that string for all the two cases mentioned above. Here the matching is **not** case sensitive. There should be **no** extra newlines between the printed lines. If there is no line matching, your program should print `No match!` **(24 marks)**
- If the number of arguments is less than required, your program should print `Not enough arguments!` **(4 marks)**
- Handle the file nonexistence issue by catching exceptions using `try` statement. The program should print `File does not exist!` and then exits if the file doesn't exist. **(2 marks)**

For example, there is a file named "test1.txt" that has the following content:

```
I heard the echo, from the valleys and the heart
Open to the lonely soul of sickle harvesting
Repeat outrightly, but also repeat the well-being of
Eventually swaying in the desert oasis

I believe I am
Born as the bright summer flowers
Do not withered undefeated fiery demon rule
Heart rate and breathing to bear the load of the cumbersome
Bored
```

If we type `python p2.py heart test1.txt` in the command prompt, your program should print:

```
I heard the echo, from the valleys and the heart
Heart rate and breathing to bear the load of the cumbersome
```

If we type `python p2.py summer flowers test1.txt` in the command prompt, your program should print:

```
Born as the bright summer flowers
```

If we type `python p2.py summer flow test1.txt`, or `python p2.py hello test1.txt` in the command prompt, your program should print:

```
No match!
```

This case is different from the previous one because we cannot find a **word** in test1.txt that is exactly the same with "flow". "flow" is a part of word "flowers".
If we type `python p2.py heart` or `python p2.py test1.txt` in the command prompt, your program should print:

```
Not enough arguments!
```

If we type `python p2.py hello test.txt` in the command prompt, here "test.txt" does not exist in the current directory, then your program should print:

```
File does not exist!
```

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

Exercise 3 (25 marks)

Write File. A **wildcard character** is a single character used to represent a number of characters or an empty string. It is often used in file searches so the full name need not be typed. **Here we just consider two type of wildcard character: the asterisk character (`*`, also called "star") and the question mark `?`.** When specifying file names or paths, `*` matches zero or more characters (e.g., `doc*` matches `doc` and `document` but not `dodo`), `?` matches zero or exactly one character (e.g., `do?` matches `doc` and `do` but not `dodo`).
Write a program that read an input file, and then prints **the number of alphabet characters, words, and lines in file to an output file**.

- The command line should be like "**python p3.py input_filename output_filename**". But the "**input_filename**" can contains the wildcard character `*` and `?`.
- If the input filename doesn't contain any wildcard character, and input file exists, then the content of output file should contain the statistics of the input file as follows: **(4 marks)**

`Number of characters:` XXX.
`Number of words:` YYY.
`Number of lines:` ZZZ.

- If the input filename contains the wildcard character, but there is no matching file, your program should print `No matching!` If several matching files are found, then you should put the collected information for all the matched file(s) into the output file. Suppose two matching files are found. Then the content of output file should be **(16 marks)**

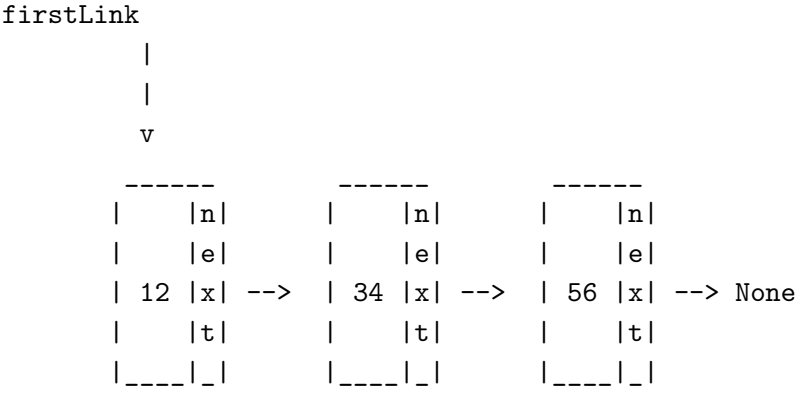
`Name of file:` XXX1.
`Number of characters:` YYY1.
`Number of words:` ZZZ1.
`Number of lines:` KKK1.
`Name of file:` XXX2.
`Number of characters:` YYY2.
`Number of words:` ZZZ2.
`Number of lines:` KKK2.

- Handle the input file non-existence issue by catching exceptions using `try` statement. Suppose your input filename is `test2.txt` and there is no such file in current directory, the error message should be `Opening file test2.txt failed!` Only the message need to be printed, and program should exit after that. **(5 marks)**

Write your script for this exercise in `p3.py`

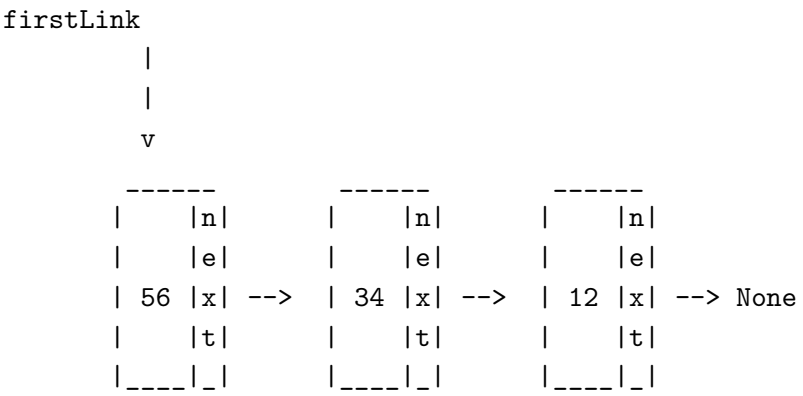
Exercise 4 (25 marks)

Linked List. A linked list could be developed using a pair of classes. The class node is used to store an individual **linked node**:



Complete the implementation of the `LinkedList` by providing methods for `test()`, `remove()` and `len()` (length).

- `test()`: see if test value is in link list. Return true if the value is in the linked list, otherwise return false. **(5 marks)**
- `remove()`: remove value from linked list. Return true if the value is in the linked list and the *first* node contains this value is removed, otherwise return false. **(5 marks)**
- `len()`: return the length of the linked list. The length is the number of valid nodes this list contains. **(5 marks)**
- `reverse()`: return the reverse linked list (implementing it by modifying the links between nodes instead of creating new nodes). For example, for the linked list mentioned above, the reverse link list is:**(6 marks)**



- `Lprint()`: print the linked list. For example, for the linked list mentioned above, the output is: **(4 marks)**

```
Current linked list: 12-->34-->56-->none.
```

```
class Node(object):
    def __init__(self, v, n):
        self.value = v
        self.next = n

class LinkedList(object):
    def __init__(self):
        self.firstLink = None

    def add (self, newElement):
        self.firstLink = Node(newElement, self.firstLink)

    def test(self, testValue):
        # your code here

    def remove(self, testValue):
        # your code here

    def len(self): # return size of linked list
        # your code here

    def reverse(self): # return reverse linked list
        # your code here

    def Lprint(self):
        # your code here
```

If you use python3 and type the following sequences to test your program,

```
l = LinkedList ()
print('list length %d '% (l.len()))
```

```
l.add( 15 )
print('list length %d '% (l.len()))
l.add( 20 )
print('list length %d '% (l.len()))
print(l.test( 15 ))
print(l.test( 30 ))
l.add( 30 )
print('list length %d '% (l.len()))
l.reverse()
l.Lprint()
print('list length %d '% (l.len()))
l.remove( 15 )
print('list length %d '% (l.len()))
l.remove ( 30 )
print('list length %d '% (l.len()))
l.remove ( 20 )
print('list length %d '% (l.len()))
```

should produce the following output

```
list length 0
list length 1
list length 2
True
False
list length 3
Current linked list: 15-->20-->30-->none
list length 2
list length 1
list length 0
```

For python2 users, the print sentences in above test cases should be modified accordingly. For example, change

```
print('list length %d '% (l.len()))
```

as

```
print 'list length %d '% (l.len())
```

To ensure the correctness of your program, you should create more test cases. You can also use the testing script to check the correctness of you code.When submit the script file, **DO NOT** put the test code along with the class implementation.

Write your script for this exercise in `p4.py`

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 *script files* (e.g. p1.py, p2.py) 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. 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>_lab4` if you are in a two-person group or `<student-id1>_lab4` if not.
   - No later than 17:00 on Nov. 2nd, 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 4**, please email to **yiwang@cse.cuhk.edu.hk**.