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

Lab Assignment 1

Instructor: Yim Pan, CHUI Due: 23:59 on Friday, Sept. 30

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. As **python 2.x** will not be officially updated anymore, **Python 3.x** is the only acceptable version for this assignment. It is necessary for us to use **python 3.x** as it has a lot more up-to-date libraries than **python 2.x** and much more stable.
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 `\\ntsvr6\userapps` and click "Finish". Then open the "S:" drive, open the **Python3** folder, and click the "IDLE" 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. The programming assignments are **graded by Python scripts on MacOS.** If your script does not pass the tests of our grading scripts, you cannot get full grades.
6. The test scripts we provide can be used **on MacOS, Windows and Linux** for your own test. You are able to test the input and output format for an exercise by the corresponding testing script. For example, by running `test_p1.py`, you can test `p1.py` for exercise 1. Note that the grading script will be DIFFERENT from the provided testing script. To use our testing scripts, for Linux or MacOS users, you should install package `pexpect` by "`pip install pexcept`", and run the testing script in the same folder of the script; for Windows users, you should install package `wexpect` by "`pip install wexpect`", and run the testing script in the same folder of the script. If you cannot run the test scripts, one way you can try is to change the line "`child = exp.spawn('python3 p1.py')`" to "`child = exp.spawn('python p1.py')`" in each script.
7. Passing the test scripts we have provided doesn't guarantee full marks for your question as our grade scripts will test for more cases.
8. You may assume that all the corner cases we have not mentioned in this document will not appear in the hidden test cases, so you do not have to worry too much about wrong

[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.

inputs unless you are required to do so.

Exercise 1: Palindrome Number (20 marks)

`input()` is a function that read a string from a user's input. The function can be used as `input(['prompt'])`. For example, after executing `x = input('type a number:')` and reading the user typed "one", the variable `x` will store the string `"one"`.

An integer is a palindrome number when it reads the same backward as forward. For example: 545 is a palindrome number; 120 is not a palindrome number; -545 is not a palindrome number.

Write a script and save it as `p1.py` that will read an interger from the user's input. If the input is not a palindrome number, then the user should be asked again to input a palindrome number. (*Hint:* use `int()` to convert a string to the corresponding integer).

A sample run should be like as follows,

```
Please input a palindrome number: 210
Your input is not a palindrome number!
Please input a palindrome number: 01210
Your input is not a palindrome number!
Please input a palindrome number: -11
Your input is not a palindrome number!
Please input a palindrome number: Nba A B n
Your input is not a palindrome number!
Please input a palindrome number: 2112
2112 is a palindrome number! Program ends.
```

Exercise 2: Shift Cipher (20 marks)

In cryptography, the shift cipher, is one of the simplest and most widely known encryption techniques.

A simple shift cipher encrypts each alphabetic letter in a string by changing `A` to `B`, `B` to `C`, ..., `Y` to `Z`, and `Z` to `A`; `a` to `b`, `b` to `c`, ..., `y` to `z`, and `z` to `a`. Other characters in the string remain unchanged.

Write a script and save it as `p2.py` that will read a string from the user's input, and print the encrypted string by a simple shift cipher. (*Hint:* ASCII code of A is 65, and ASCII code of a is 97).

Below are some samples.

Sample 1:

```
Please input a string: ABcd
The encrypted string is: BCde
```

Sample 2:

```
Please input a string: csci2040
The encrypted string is: dtdj2040
```

Sample 3:

```
Please input a string: I love python!
The encrypted string is: J mpwf qzuipo!
```

Exercise 3: Right-angled Triangle Pattern of A String **(20 marks)**

Using `while` loop or `for` loop to write a script `p3.py` that reads an integer value `n` and a string `str` from the user's input and then produces an `n`-line block, repeatedly. For the `n`-line output, the first line contains `1 str`, the second line contains `2 str`s, and so on until the `n`-th line, which contains n `str`s. Print the pattern as a right-angled triangle.

The script will keep asking the user to input an integer value `n` and a string `s` and then print the corresponding `n`-line block. The script will print the line `Program ends.` and exit if the user inputs a negative number or `0`. For this exercise, you don't need to check whether the user's input is an integer or not and also you don't need to consider that a very long input would screw the format.

A sample run should be as follows:

```
Enter an integer: 4
Enter a string: 1
   1
  11
 111
1111
Enter an integer: 2
Enter a string: ^_^
   ^_^
^_^^_^
Enter an integer: 0
Program ends.
```

Exercise 4: GPA Calculator **(20 marks)**

In order to improve GPA (Grade Point Average), the first thing at hand is to write a script `p4.py` that can calculate GPA quickly. With the script, a user could input the grade followed

by the credit of a course. Immediately after reading the grade for a course, the GPA calculator prints out the current GPA. The user could keep adding the grades for more and more courses until the user inputs `exit`. When the user inputs `exit`, the script will print the line `Program ends.` and exit. The grade and credit are separated by space. If the grade or credit inputted is negative, your script should print `Wrong input!` and continue to wait for new inputs.

Note that the upper bound of grades is 5.0. So any input grade which is larger than 5.0 is also a wrong input.

The GPA MUST be in the format of 2 decimal points. (*Hint:* Use `"0:.2f".format(a)` if you want to output 2 decimal points of `a`).

A sample run should be as follows:

```
3.7 3
Current GPA: 3.70
3.3 2
Current GPA: 3.54
2.5 -2
Wrong input!
5.1 4
Wrong input!
exit
Program ends.
```

Exercise 5: Set Your Password Secretly (20 marks)

Hint: To hide the input in Python, you could simply replace `input()` with `getpass.getpass()`.

```
import getpass
secret_input = getpass.getpass()
```

Write a script `p5.py` to set password secretly. In this exercise, we will implement a simple password setting process.

First, a user will be required to input her own password by "`Please input your password:`". The password should contain at least 8 characters, including at least 1 English letter and 1 digit. For example, "`abc123`' ' is not valid; "`1234567890`' ' is not valid; "`abcdefghi`' ' is not valid; "`csci2040`' ' is a valid password.

If the password does not satisfiy these requirements, the user will be required to input again by "`Invalid input.`" and "`Please try again: `" until the input is valid.

After that, the user will be required to input the same password again by "`Please input again to confirm your password:`". There are 3 chances for the password confirmation. If the input is the same as the password before, the program will print "`Successfully set your password!`" and end. If the input is different from the password before, the user will be required to input again by "`Those passwords did not match.`" and "`Please try again:`"

for the first two chances, and for the third chance, the program will print "`Those passwords did not match. Please set your password next time. Program ends.`" and end.

Below are some samples. (Because the inputs are hidden, when we run the program, we cannot see them. Here, for the purpose of demonstration, the inputs are stated by gray words.)

Sample 1:

```
Please input your password: the input here is csci2040
Please input again to confirm your password: the input here is csci2040
Successfully set your password!
```

Sample 2:

```
Please input your password: the input here is 123456
Invalid input.
Please try again: the input here is abc123
Invalid input.
Please try again: the input here is python2040
Please input again to confirm your password: the input here is python2040
Successfully set your password!
```

Sample 3:

```
Please input your password: the input here is python
Invalid input.
Please try again: the input here is csci2040
Please input again to confirm your password: the input here is csci1040
Those passwords did not match.
Please try again: the input here is csci2010
Those passwords did not match.
Please try again: the input here is csci2040
Successfully set your password!
```

Sample 4:

```
Please input your password: the input here is csci2040
Please input again to confirm your password: the input here is csci204
Those passwords did not match.
Please try again: the input here is csci1234
Those passwords did not match.
Please try again: the input here is csci2440
Those passwords did not match. Please set your password next time. Program ends.
```

Submission rules

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

   `<student-id1>_<student-id2>_lab1.zip`

   For example, `1155012345_1155054321_lab1.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_lab1.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>_lab1` if you are in a two-person group or `<student-id1>_lab1` if not.
   - No later than 23:59 on Friday, Sept. 30
4. Students in the same group would get the same marks. Marks will be deducted if you do not follow the submission rules. Anyone/Any group caught plagiarizing would get 0 score!