CSCI 2040: Introduction to Python **2024-2025 Term 1**

Lab Assignment 1

Instructor: Dr. Yim Pan, CHUI Due: 23:59pm on Wednesday, Oct. 2

Notes

1. You are allowed to form a group of two to do this lab assignment.
2. The programming assignments are **graded by Python scripts.** If your script does not pass the tests of our grading scripts, you cannot get full grades.
3. To use our testing scripts, for Linux or MacOS users, please install package `pexpect` by "`pip install pexpect`"; For Windows users, please install package `wexpect` by "`pip install wexpect`" first. Then put your scripts and the testing scripts under the same file folder and run the test scripts. Some online Python coding platforms are also available for use without the need of installation, such as repl.it.
4. Note that the grading scripts will be DIFFERENT from the provided testing scripts. Passing the test scripts we have provided does not guarantee full marks for your solution as our grading scripts will test it against more cases.
5. You may assume that all user inputs are always valid and your program for each question need not include code to validate them unless you are required to do so. You may assume that all the corner cases we have not mentioned in this document will not appear in the hidden grading cases, so you do not have to worry too much about wrong inputs unless you are required to do so.
6. **Please follow closely the format of the sample output and demo file for each question.** Your program should produce **exactly** the same output as the sample (same text, symbols letter case, spacing, number format, etc.). The output for each question should end with a single newline character, which has been provided by the `print()` function by default.

Exercise 1: Finding the Most Frequent Letter (20 marks)

The `input()` function is a build-in function that reads a string from the user via console input. The calling syntax of the function is: `input(['prompt'])`. For example, suppose that `x = input('type a number:')` is executed, it shows the message "`type a number:`" and waits for the user to enter some value and hit the Enter key. If the user types "`one`" followed by Enter, the variable x will store the string "`one`".

Write a Python script named `p1.py` that prompts the user to enter a string. Your script should:

1. **Count the number of alphabetic letters** in the string. (Hint: Use a for-loop and the `isalpha()` method to check if a character is an alphabetic letter.)

2. **Find and print the most frequent alphabetic letter** in the string (we assume case-insensitive here). If there is a tie, print the letter with the lowest order from "a-z".

Hint: You may also want to use and methods which return the Unicode code point (an integer) of a given character and return the character corresponding to a given Unicode code point (an integer), respectively. Below are some sample runs of Python script `p1.py`. **Please remember to check the environment using in your IDE (VSCode/PyCharm) every time before you run any Python script.**

Sample Run 1

```
python p1.py
Enter a string: CSCI2040
The number of alphabetic letters in "CSCI2040" is "4".
The most frequent alphabetic letter in "CSCI2040" is "c".
```

Sample Run 2

```
python p1.py
Enter a string: I Love Python More than java
The number of alphabetic letters in "I Love Python More than java" is "23".
The most frequent alphabetic letter in "I Love Python More than java" is "a".
```

To better assist you in checking the correctness of your code, we have provided additional test code `test_p1.py`. Please put the test code `test_p1.py` in the same **root directory** as your Python script `p1.py`. The same rule also applies to other exercises. Here is a sample of running the test code

Sample Run Test

```
python test_p1.py
Test for input 'CSCI2040' passed!
Test for input 'Python.py' passed!
Test for input 'I Love Python More than java' passed!

3 out of 3 test cases passed.
```

Exercise 2: Triangle **(20 marks)**

Write a program to prompt the user to enter an integer h and print a triangle pattern of height h as shown in the following sample runs. For example, for $h = 5$, the triangle consists of 5 lines (or rows). The bottom line is composed by underscores.

There is a validation requirement in this question: if the user input is not in this specified range: $1 < h \leq 30$, keep printing an error message *"Invalid input for h!"* and prompting the user to input again until a valid value is received.

Sample Run 1

```
python p2.py
Enter h: 2
 /\
/__\
```

Sample Run 2

```
python p2.py
Enter h: 5
    /\
   /  \
  /    \
 /      \
/________\
```

Sample Run 3

```
python p2.py
Enter h: 0
Invalid input for h!
Enter h: -1
Invalid input for h!
Enter h: 1
Invalid input for h!
Enter h: 31
Invalid input for h!
Enter h: 7
      /\
     /  \
    /    \
   /      \
  /        \
 /          \
/____________\
```

Hints: think about how many spaces to print before printing the / symbol for each line. Note also how to correctly print the \ symbol (which has special meaning to the Python interpreter). To better visualize the spaces printed in the output, consider the auxiliary diagrams below denoting a space by ␣ for $h = 4$ and $h = 5$ respectively.

```
12345678
   /\
  /  \
 /    \
/______\

1234567890
     /\
    /  \
   /    \
  /      \
 /        \
/__________\
```

Sample Run Test

```
python test_p2.py
Running Test Case 1 ...
Test case 1: Passed!

Running Test Case 2 ...
Test case 2: Passed!

Running Test Case 3 ...
Test case 3: Passed!

Congrats! You passed all test cases!
```

Exercise 3: Single-Choice String Transformation **(20 marks)**

Write a Python script named `p3.py` that transforms a user-input string according to a set of rules based on the user's choice. The script will then print the transformed string.

1. The script will prompt the user to enter a string `'s'`.
2. The script will then ask the user to choose which transformation to apply from the following options:
   - **Option 1:** Convert all characters in the string to uppercase.
   - **Option 2:** Replace all vowels `'a', 'e', 'i', 'o', 'u'` with the character `'*'`. We also assume case-insensitive here.
   - **Option 3:** Reverse the entire string.
   - **Option 4:** Append `"EVEN"` or `"ODD"` depending on the length of the string.

3. Based on the user's choice, the script will apply the selected transformation to the string `'s'` and print the result.
4. The script should terminate after displaying the transformed string, with a message `"Transformation complete. Goodbye!"`

Below are some sample runs.

Sample Run 1

```
python p3.py
Enter a string: Python
Choose a transformation:
1. Convert to uppercase
2. Replace vowels with '*'
3. Reverse the string
4. Append 'EVEN' or 'ODD'
Enter your choice: 4
Transformed string: PythonEVEN
Transformation complete. Goodbye!
```

Sample Run 2

```
python p3.py
Enter a string: CSCI2040
Choose a transformation:
1. Convert to uppercase
2. Replace vowels with '*'
3. Reverse the string
4. Append 'EVEN' or 'ODD'
Enter your choice: 5
Invalid choice. No transformation applied.
Transformed string: CSCI2040
Transformation complete. Goodbye!
```

Sample Run Test

```
python test_p3.py
Test for input 'Python' passed!
Test for input 'HongKong' passed!
Test for input 'I love Python!' passed!
Test for input 'programming' passed!
Test for input 'CSCI2040' passed!

5 out of 5 test cases passed.
```

Exercise 4: Interest Calculator (40 marks)

Write a program that calculates the compound interest earned by depositing a principal amount $\$P$ in a premium savings account with r percent annual interest rate. The interest is paid n times per year. Assuming that all interest is left in the account, please calculate and print the accrued amount A at the end of each year and the interest I earned within each year for t years in a tabular format.

The program should also generate a summary report after the calculations are complete, showing:

1. Total interest earned over the entire period.
2. Final account balance at the end of t years.

Hint: Use the following formula to compute the accrued amount A at the end of each year:

$$A = P\left(1 + \frac{r}{n}\right)^n \tag{1}$$

where

- P is the principal amount,
- r is the annual interest rate,
- n is the number of compounding periods per unit of time,
- A is the accrued amount (i.e., principal + interest).

Input and Output Requirements:

1. Input from the User:
   - Principal amount (P): A positive float. A valid range should be $0 < P < 100,000$.
   - Annual interest rate (r): A float representing the interest rate percentage (e.g., enter `5` for 5%). A valid range is set to $0 < r < 10$.
   - Number of compounding periods per year (n): Choosing from 1 (yearly), 2 (half-yearly), 4 (quarterly), or 12 (monthly)).
   - Number of years (t): An integer representing the total duration in years. For simplicity, we consider a valid range: $0 < t < 10$. The user will always input data in this valid range.
2. Output Format:
   - Print the year, accrued amount A at the end of each year and the interest I earned within each year.
   - After the yearly breakdown, print the total interest earned and the final balance.
   - **Please refer to the output format in file `demo_p4.py`**. If you use a different output format, you may not pass the testing script and get a full mark.

Sample Run 1

```
python p4.py
Enter the initial principal amount ($P): 10000
Enter the annual interest rate (r%): 5
Enter the number of compounding periods per year (n = 1, 2, 4, 12): 4
Enter the number of years (t): 5
Year |     Amount    | Interest Earned
----------------------------------------
   1 | $    10509.45 | $        509.45
   2 | $    11044.86 | $        535.41
   3 | $    11607.55 | $        562.68
   4 | $    12198.90 | $        591.35
   5 | $    12820.37 | $        621.48
Summary:
Total interest earned: $2820.37
Final account balance: $12820.37
```

Sample Run 2

```
python p4.py
Enter the initial principal amount ($P): 5500
Enter the annual interest rate (r%): 3
Enter the number of compounding periods per year (n = 1, 2, 4, 12): 12
Enter the number of years (t): 7
Year |     Amount    | Interest Earned
----------------------------------------
   1 | $     5667.29 | $        167.29
   2 | $     5839.66 | $        172.38
   3 | $     6017.28 | $        177.62
   4 | $     6200.30 | $        183.02
   5 | $     6388.89 | $        188.59
   6 | $     6583.22 | $        194.32
   7 | $     6783.45 | $        200.23
Summary:
Total interest earned: $1283.45
Final account balance: $6783.45
```

Sample Run Test

```
python test_p4.py
Running Test Case 1 ...
Test case 1: Passed!

Running Test Case 2 ...
Test case 2: Passed!

Running Test Case 3 ...
Test case 3: Passed!

Congrats! You passed all test cases!
```

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 23:59pm on Wednesday, Oct. 2
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!