【2023 Object-Oriented Programming Homework 2】

1. Please use **Python language** for this homework and make sure it can run correctly.
2. Please provide **ipynb files** to validate your homework.
3. Do not copy the others work definitely. Otherwise, you will fail this class.
4. If you have any question, please send email to TA or drop by Room EC5018.
   However, TA will not help you to debug program.

Turn in your homework:

1. Please compress your homework into zip file.
2. Naming rule: "OOP_HW2.zip".
3. Upload your homework (zip file) to NSYSU Cyber University (網路大學).
4. **Deadline: 2023/03/10 09:00.** You can not get any credit if you do not turn in
   your homework before the deadline.

Homework description:

1. Write a program that prompts the user to input a positive integer n, finds all prime numbers between 1 and n, where prime numbers are positive integers that can only be divided by 1 and themselves, stores these prime numbers in a list, and finally returns this list. The program should interact with the user as follows:

```
Enter a positive integer > 20
prime : [2, 3, 5, 7, 11, 13, 17, 19]
```

2. Write a program that allows the user to input multiple integers separated by commas(,), put all the factors of each integer into a list, print out the list, and then calculate the sum of the factors of each integer. The program should interact with the user as follows:

```
Enter multiple integers, separated by commas > 10,20,30

factor of 10 ：[1, 2, 5, 10]
The sum of the factors of 10 ：18

factor of 20 ：[1, 2, 4, 5, 10, 20]
The sum of the factors of 20 ：42

factor of 30 ：[1, 2, 3, 5, 6, 10, 15, 30]
The sum of the factors of 30 ：72
```

3. Write a program that allows the user to input a string and a substring, and uses a **while loop** to count the number of occurrences of the substring in the string. The program should interact with the user as follows:

```
Enter the string > hello world, hello python.
Enter the substring > hello

The number of times the substring appears in the string is 2 times.
```

4. Use **loop** to write a program to print a Sandglass pattern of star. The program should interact with the user as follows to let the user enter the height of the pattern:

```
Enter the height > 5

* * * * *
 * * * *
  * * *
   * *
    *
    *
   * *
  * * *
 * * * *
* * * * *
```

5. Write a program that prompts the user to input two years, then calculates how many leap years there should be in the area, and finally outputs the result. The conditions for judging a leap year are as follows:

I. If the year is divisible by 4 but not by 100, then it is a leap year.
II. If the year is divisible by 400, it is also a leap year.
III. If the year is neither divisible by 4 nor by 400, then it is not a leap year.

The program should interact with the user as follows:

```
Please enter the start year > 2001
Please enter the end year > 2015

Between 2001 to 2015, there is(are) 3 leap year(s).
```