ENGG1330 Computer Programming

Tutorial 4

Objectives:

1. Write programs with flow control statement (loop)
2. Gain hands-on experience on nested for-loop

Tutorial participation (1%):

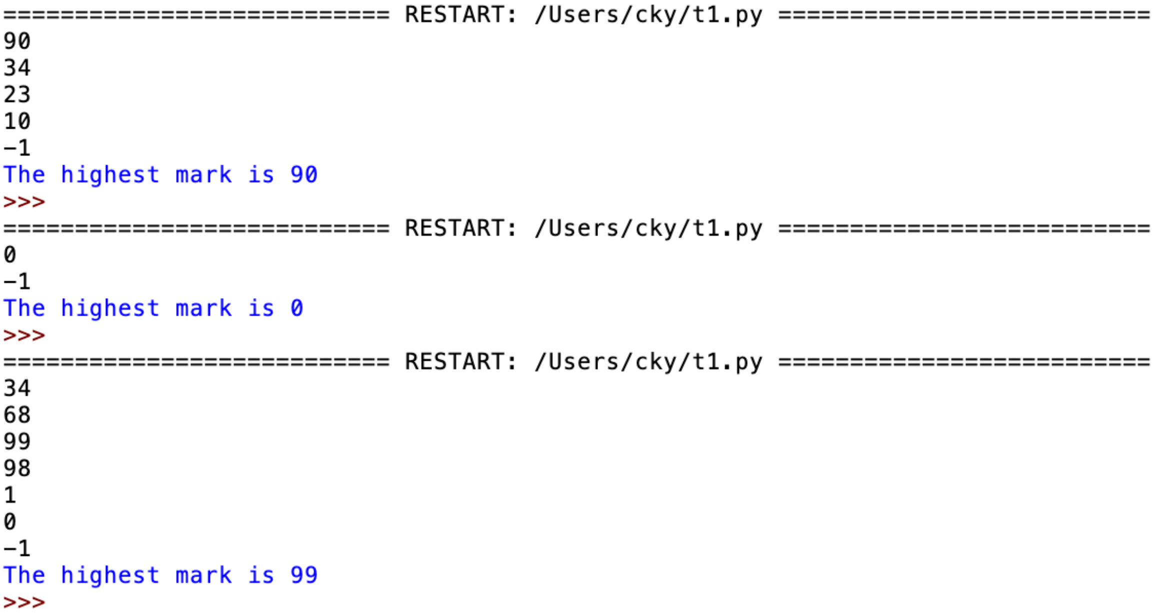
- t4_vpl_1
- Submission period: **within your OWN tutorial period**

Tutorial/take-home exercises (2%):

- Remaining problems in the worksheet
- Submission deadline: **noon, 1-MAR-2023 (Wednesday)**

t4_vpl_1. Write a program to determine the highest mark of a quiz in a class, assuming that the marks are integers in the range of [0, 100]. The program prompts the user to input the marks until input equals -1, indicating the end of entry. When the user inputs -1, the program outputs the highest mark. If there is no entry, the highest mark is 0.

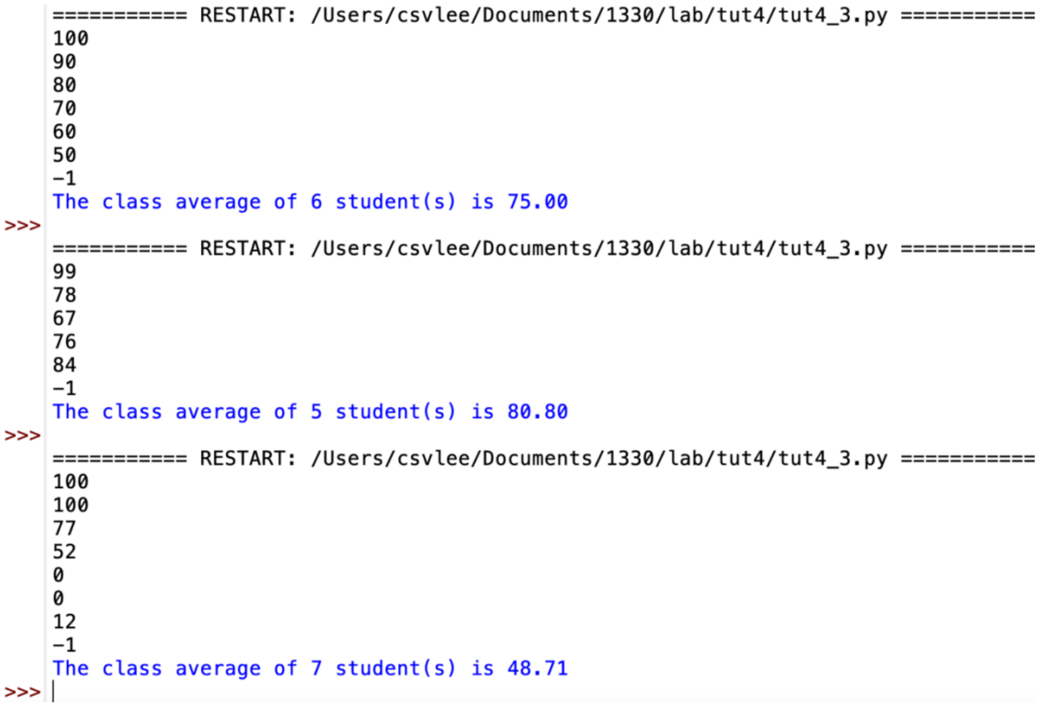
Sample cases and screenshots



```
========================== RESTART: /Users/cky/t1.py ==========================
90
34
23
10
-1
The highest mark is 90
>>>
========================== RESTART: /Users/cky/t1.py ==========================
0
-1
The highest mark is 0
>>>
========================== RESTART: /Users/cky/t1.py ==========================
34
68
99
98
1
0
-1
The highest mark is 99
>>>
```

t4_vpl_2. Modify the program in t4_vpl_1 to print the number of students and the class average with 2 places of decimal.

Sample cases and screenshots

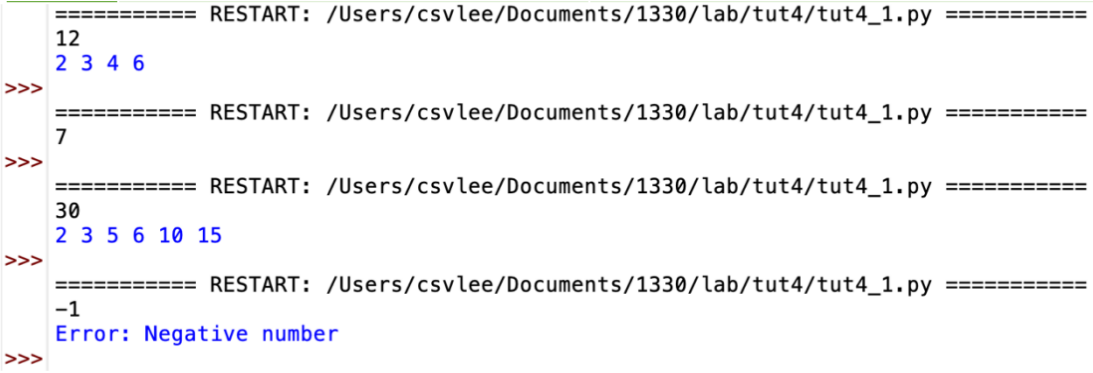


```
=========== RESTART: /Users/csvlee/Documents/1330/lab/tut4/tut4_3.py ===========
100
90
80
70
60
50
-1
The class average of 6 student(s) is 75.00
>>> 
=========== RESTART: /Users/csvlee/Documents/1330/lab/tut4/tut4_3.py ===========
99
78
67
76
84
-1
The class average of 5 student(s) is 80.80
>>> 
=========== RESTART: /Users/csvlee/Documents/1330/lab/tut4/tut4_3.py ===========
100
100
77
52
0
0
12
-1
The class average of 7 student(s) is 48.71
>>> |
```

t4_vpl_3. Write a program to read an integer *n* and print out all the factors of *n* where the factors are larger than 1 and less than *n*. If *n* is negative, print the error message: “`Error: Negative number`”.

Sample cases and screenshots

Case	Input	Output
1	12	2 3 4 6
2	7	
3	30	2 3 5 6 10 15
4	-1	Error: Negative number



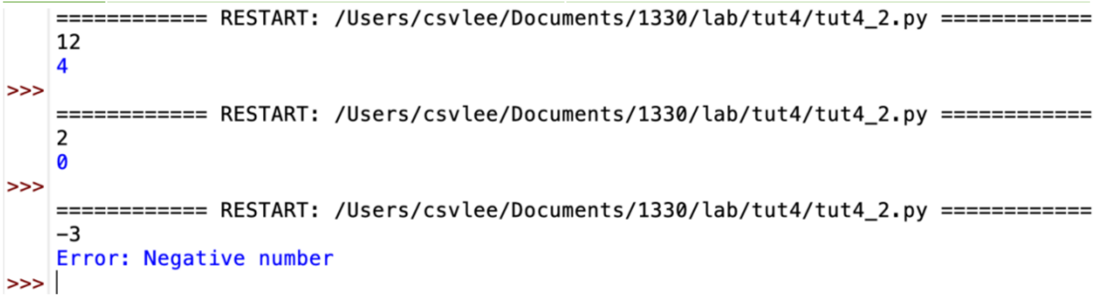
```
=========== RESTART: /Users/csvlee/Documents/1330/lab/tut4/tut4_1.py ===========
12
2 3 4 6
>>> 
=========== RESTART: /Users/csvlee/Documents/1330/lab/tut4/tut4_1.py ===========
7
>>> 
=========== RESTART: /Users/csvlee/Documents/1330/lab/tut4/tut4_1.py ===========
30
2 3 5 6 10 15
>>> 
=========== RESTART: /Users/csvlee/Documents/1330/lab/tut4/tut4_1.py ===========
-1
Error: Negative number
>>> 
```

Hint: if *k* is a factor of *n*, the remainder of *n*/*k* should be zero. Which operator in Python gives the remainder of a division?

t4_vpl_4. Modify the program in t4_vpl_3 to print the number of factors for the input number.

Sample cases and screenshots

Case	Input	Output
1	12	4
2	2	0
3	-3	Error: Negative number



```
============ RESTART: /Users/csvlee/Documents/1330/lab/tut4/tut4_2.py ============
12
4
>>> 
============ RESTART: /Users/csvlee/Documents/1330/lab/tut4/tut4_2.py ============
2
0
>>> 
============ RESTART: /Users/csvlee/Documents/1330/lab/tut4/tut4_2.py ============
-3
Error: Negative number
>>> |
```

t4_vpl_5. One way to approximate π is to use Leibniz formula which has a form:

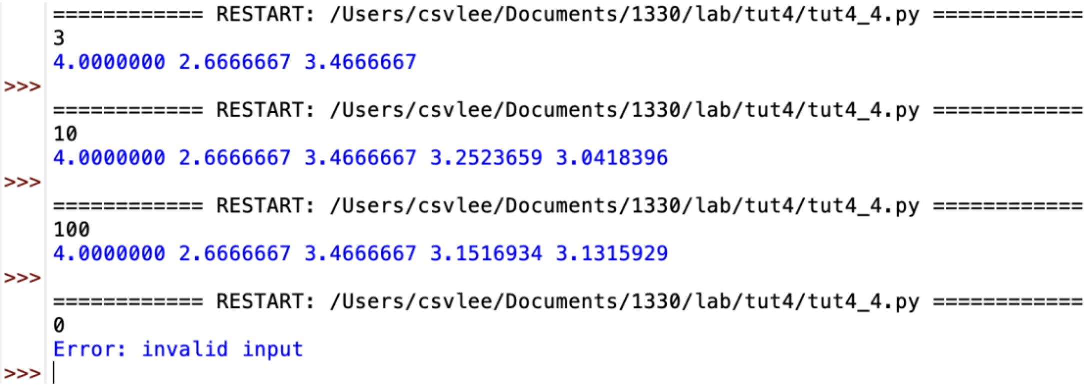
$$1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \cdots = \frac{\pi}{4} = \sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1}$$

Write a program to estimate π by evaluating this formula up to kth term, where k is an integer inputted by user. The program should print the first 3, $(k-1)$ and kth approximations to 7 decimal places. If k is non-positive, print the error message: “Error: invalid input”.

Note: this series has a very slow convergence. It would take millions of iterations (terms) to get π to 7 correct decimal places!

Sample cases and screenshots

Case	Input	Output
1	3	4.0000000 2.6666667 3.4666667
2	10	4.0000000 2.6666667 3.4666667 3.2523659 3.0418396
3	100	4.0000000 2.6666667 3.4666667 3.1516934 3.1315929
4	0	Error: invalid input



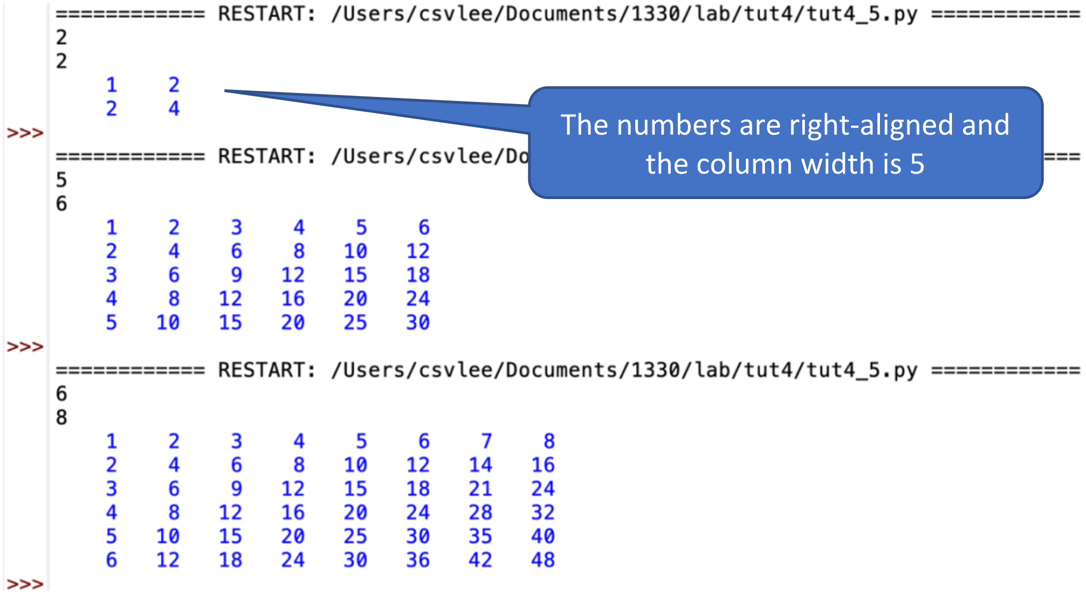
```
============ RESTART: /Users/csvlee/Documents/1330/lab/tut4/tut4_4.py ============
3
4.0000000 2.6666667 3.4666667
>>> 
============ RESTART: /Users/csvlee/Documents/1330/lab/tut4/tut4_4.py ============
10
4.0000000 2.6666667 3.4666667 3.2523659 3.0418396
>>> 
============ RESTART: /Users/csvlee/Documents/1330/lab/tut4/tut4_4.py ============
100
4.0000000 2.6666667 3.4666667 3.1516934 3.1315929
>>> 
============ RESTART: /Users/csvlee/Documents/1330/lab/tut4/tut4_4.py ============
0
Error: invalid input
>>> |
```

Hint: kth approximation is the sum of first k terms

t4_vpl_6. Use nested for-loop to generate a *m*-row x *n*-column ‘multiplication table’, where *m* and *n* are two positive integers inputted by user, assuming that the first input is *m* and the second input is *n*. The column width of the table is 5-character.

Sample cases and screenshots

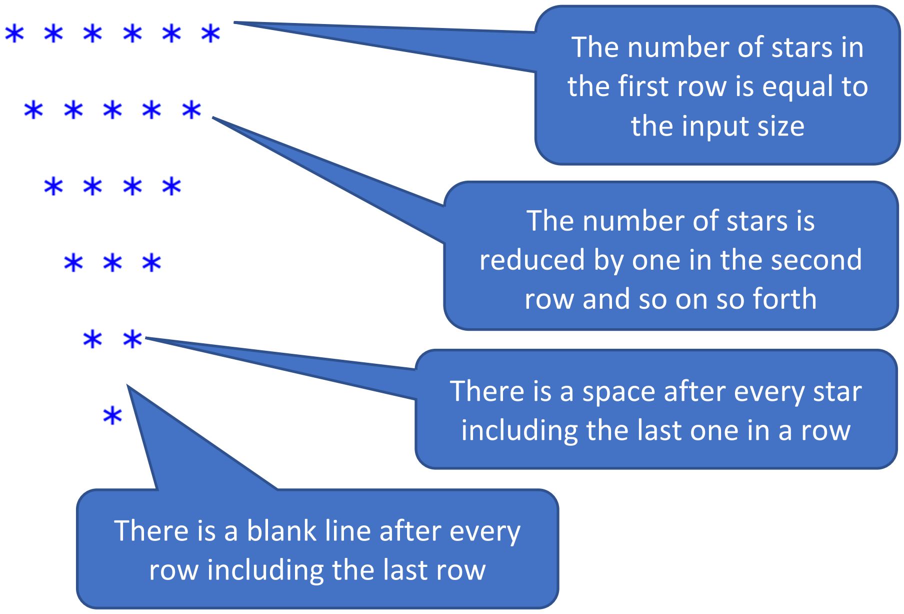
Case	Input	Output							
1	2	1	2						
	2	2	4						
2	5	1	2	3	4	5	6		
	6	2	4	6	8	10	12		
		3	6	9	12	15	18		
		4	8	12	16	20	24		
		5	10	15	20	25	30		
3	6	1	2	3	4	5	6	7	8
	8	2	4	6	8	10	12	14	16
		3	6	9	12	15	18	21	24
		4	8	12	16	20	24	28	32
		5	10	15	20	25	30	35	40
		6	12	18	24	30	36	42	48



```
============ RESTART: /Users/csvlee/Documents/1330/lab/tut4/tut4_5.py ============
2
2
    1    2
    2    4
>>>
============ RESTART: /Users/csvlee/Do                                        ===
5
6
    1    2    3    4    5    6
    2    4    6    8   10   12
    3    6    9   12   15   18
    4    8   12   16   20   24
    5   10   15   20   25   30
>>>
============ RESTART: /Users/csvlee/Documents/1330/lab/tut4/tut4_5.py ============
6
8
    1    2    3    4    5    6    7    8
    2    4    6    8   10   12   14   16
    3    6    9   12   15   18   21   24
    4    8   12   16   20   24   28   32
    5   10   15   20   25   30   35   40
    6   12   18   24   30   36   42   48
>>>
```

The numbers are right-aligned and the column width is 5

t4_vpl_7. Use nested for-loop to print an ice-cream cone pattern. Ask the user to input a positive integer to denote the size of ice-cream cone. For example, the following diagram shows the pattern when the size is 6.



```
* * * * * * 

 * * * * * 

  * * * * 

   * * * 

    * * 

     * 

```

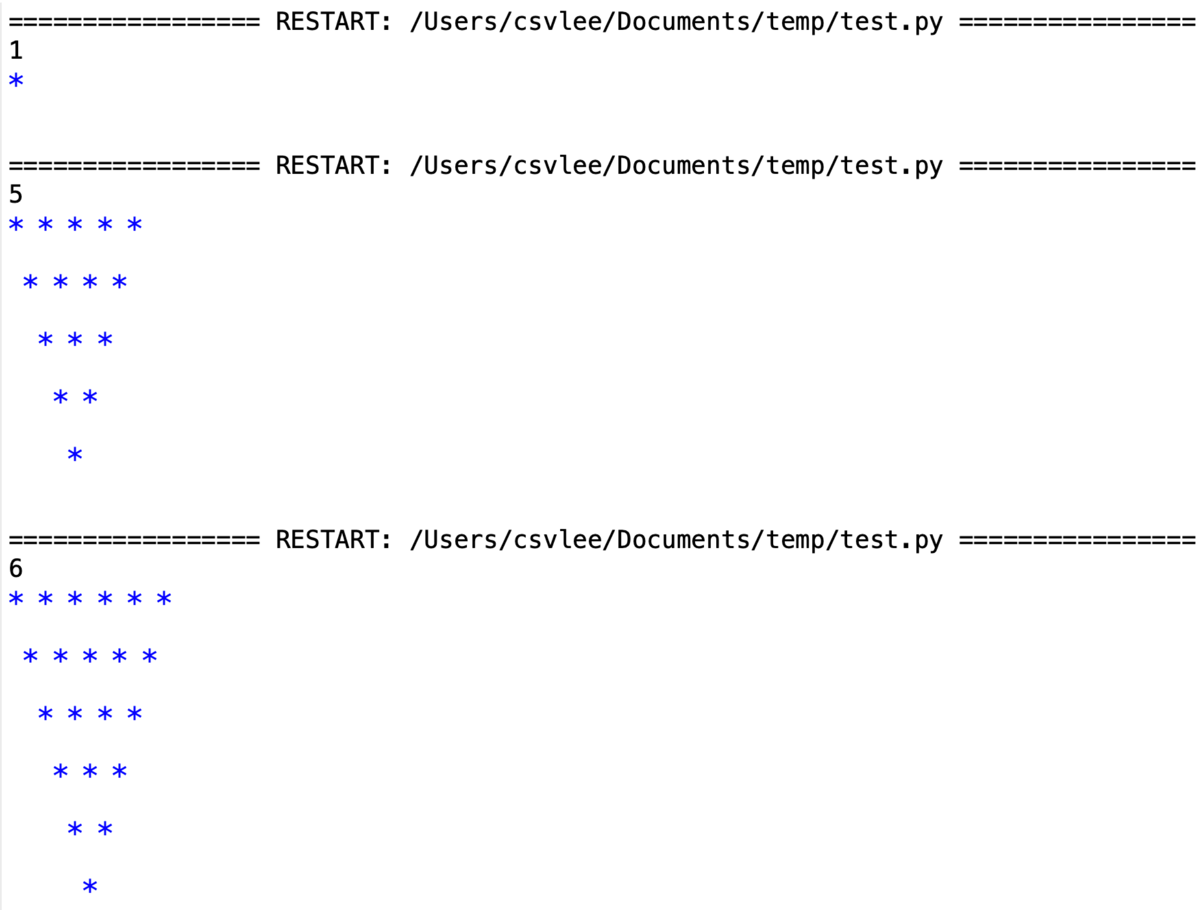
The number of stars in the first row is equal to the input size

The number of stars is reduced by one in the second row and so on so forth

There is a space after every star including the last one in a row

There is a blank line after every row including the last row

Sample cases and screenshots



```
================= RESTART: /Users/csvlee/Documents/temp/test.py ================
1
* 


================= RESTART: /Users/csvlee/Documents/temp/test.py ================
5
* * * * * 

 * * * * 

  * * * 

   * * 

    * 


================= RESTART: /Users/csvlee/Documents/temp/test.py ================
6
* * * * * * 

 * * * * * 

  * * * * 

   * * * 

    * * 

     * 
```