

CSC401- Assignment 3

Jay Tang

Due Wednesday, April 27, 11:59pm

Reading

Read **Chapter 5.1 and 5.2** in Introduction to Computing using Python: An Application Development Focus, Second Edition by Ljubomir Perković.

Logistics

You need to do this assignment on a computer which has Python 3 installed on it.

[Python 3.10 download page can be found here.](#)

You are encouraged to work with your classmates on the assignments. If you do work with someone on the assignments, please include the name of your collaborators at the top of the file you submit. If you worked alone, please indicate that at the top of your submission. **A submission without collaboration information will not receive credit.**

A submission that includes code which does not run will not get any points for the part unless specifically documented reason of the error.

Submission

Submit the assignment using Assignment 3 folder. Submit only a **single python file** using your name as file name (e.g. Jay_Tang_Assign_3.py).

This assignment is due Wednesday, April 27, 11:59pm. Submissions after the deadline will be automatically rejected by the system.

Assignment

1. Decisions (15pt)

If there is a vote at a meeting, there are several possible outcomes based on the number of yes and no votes (abstains are not counted). If all the votes are yes, then the proposal passes "unanimously", if at least 2/3 of the votes are yes, then the proposal passes with "supermajority", if more than 1/2 of the votes are yes, then the proposal passes by "simple majority", and otherwise it fails. Write a program that asks the user for the number of yes and no votes (two separate prompts), and then prints the outcome of the vote ('unanimous', 'supermajority', 'simple majority', 'fail').

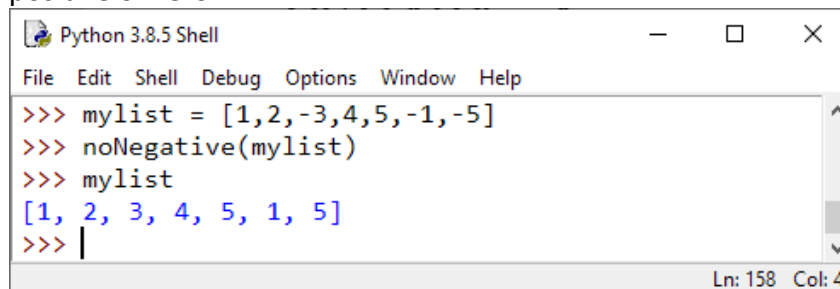
Hint: In a first step compute the total number of votes. Then if, elif, else.

2. Loop Patterns (85pt)

- a. (20pt) Implement function `myMin(lst)` that takes a list of number as a parameter. **Without using the built-in `min()` function**, write your own implementation to find the minimum value of the list and return that value.

Hint: If you want a number larger than or others, you can use `float('inf')`. If you want a number smaller than or others, you can use `float('-inf')`.

- b. (20pt) Implement function `noNegative(lst)` that takes a list of numbers as a parameter. The function changes the list items so that all numbers in the list are positive or zero.



```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
>>> mylist = [1,2,-3,4,5,-1,-5]
>>> noNegative(mylist)
>>> mylist
[1, 2, 3, 4, 5, 1, 5]
>>> |
```

Ln: 158 Col: 4

Hint: Make sure the list you pass to the function is changed after the function call.

- c. (25pt) We have seen a multiplication chart like this in the last assignment.

x	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	2	4	6	8	10	12	14	16	18	20
3	3	6	9	12	15	18	21	24	27	30
4	4	8	12	16	20	24	28	32	36	40
5	5	10	15	20	25	30	35	40	45	50
6	6	12	18	24	30	36	42	48	54	60
7	7	14	21	28	35	42	49	56	63	70
8	8	16	24	32	40	48	56	64	72	80
9	9	18	27	36	45	54	63	72	81	90
10	10	20	30	40	50	60	70	80	90	100

This time implement a function `multChart(row, col)` that takes two parameters: `row` indicating how many rows to print and `col` deciding how many columns to print. First print the line number then followed by multiples.

The following screenshot shows an example of the results. You should also align your numbers so the chart can be easily readable.

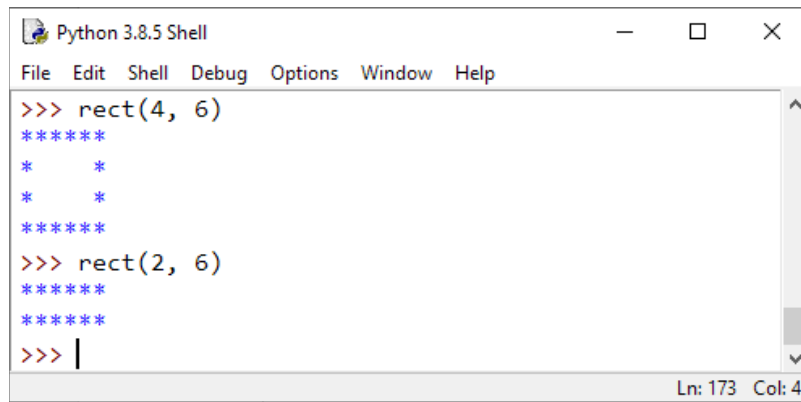
```

Python 3.8.0 Shell
File Edit Shell Debug Options Window Help
>>> multChart(10,10)

10 x 10 Multiplication Chart
x:   1  2  3  4  5  6  7  8  9 10
1:   1  2  3  4  5  6  7  8  9 10
2:   2  4  6  8 10 12 14 16 18 20
3:   3  6  9 12 15 18 21 24 27 30
4:   4  8 12 16 20 24 28 32 36 40
5:   5 10 15 20 25 30 35 40 45 50
6:   6 12 18 24 30 36 42 48 54 60
7:   7 14 21 28 35 42 49 56 63 70
8:   8 16 24 32 40 48 56 64 72 80
9:   9 18 27 36 45 54 63 72 81 90
10:  10 20 30 40 50 60 70 80 90 100
>>>
Ln: 399 Col: 4

```

- d. (20pt) Write a function `rect(n, m)` which creates the boundary of an `n` by `m` rectangle, that is an `n` by `m` (empty) box with asterisks in the first and last columns and rows.



A screenshot of a Python 3.8.5 Shell window. The window has a title bar with the text "Python 3.8.5 Shell" and standard window controls (minimize, maximize, close). Below the title bar is a menu bar with the following items: File, Edit, Shell, Debug, Options, Window, and Help. The main area of the window contains the following text:

```
>>> rect(4, 6)
*****
*       *
*       *
*****
>>> rect(2, 6)
*****
*****
>>> |
```

The text is color-coded: the prompt characters ">>>" are red, the function calls "rect(4, 6)" and "rect(2, 6)" are blue, and the asterisks "*" are black. A vertical scrollbar is visible on the right side of the text area. At the bottom right of the window, the status bar displays "Ln: 173 Col: 4".