

CSC401- Assignment 4

Jay Tang

Due Wednesday, May 4, 11:59pm

Reading

Read **Chapter 5 and 6.4** 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

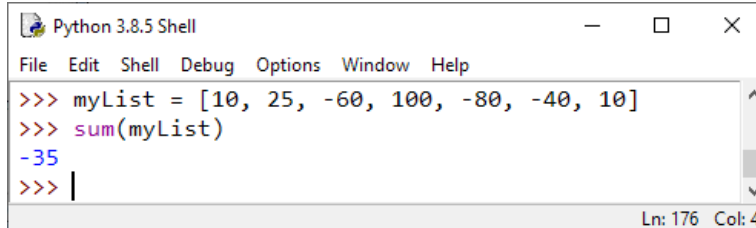
Submit the assignment using Assignment 4 folder. Submit only a **single python file** using your name as file name (e.g. Jay_Tang_Assign_4.py).

This assignment is due Wednesday, May 4, 11:59pm. Submissions after the deadline will be automatically rejected by the system.

Assignment

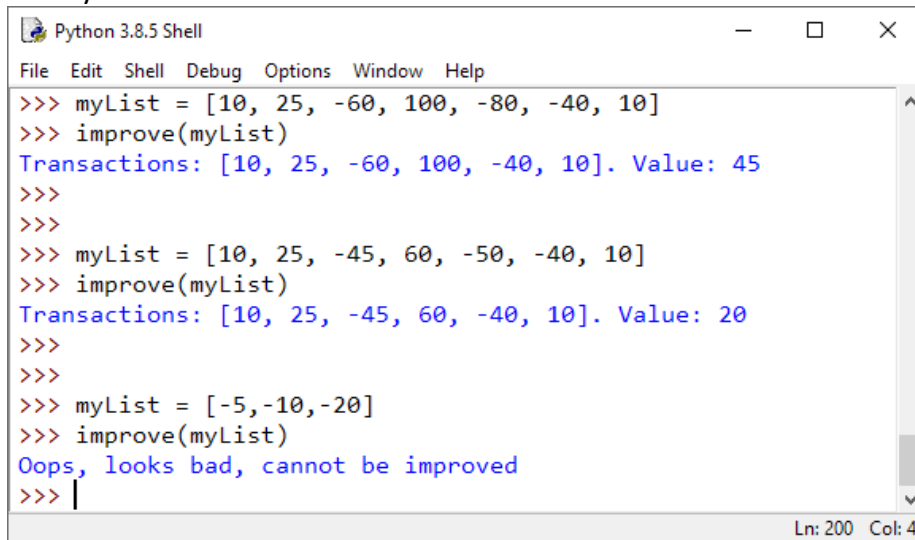
1. (30pt) while loop

You've been keeping track of your planned bank account transactions, and it doesn't look so good, e.g. you had the following transactions in the list `lst`. Positive values are incoming money, negative values are money spent.



```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
>>> myList = [10, 25, -60, 100, -80, -40, 10]
>>> sum(myList)
-35
>>> |
```

As you see that left you \$35 short. Your strategy to deal with this is to remove the largest amount spent to improve the situation. In this case removing the -\$80 you were planning to spend bring you to a positive budget. In general, you may have to remove more than one of the negative numbers, but you'll always remove them in order of value, that is largest negative transactions get eliminated first, until you reach a positive budget. Write a program `improve(lst)` which implements that strategy and prints the remaining transactions as well as the overall resulting budget. If this is not possible (if you're only spending money), the program should warn you about that.

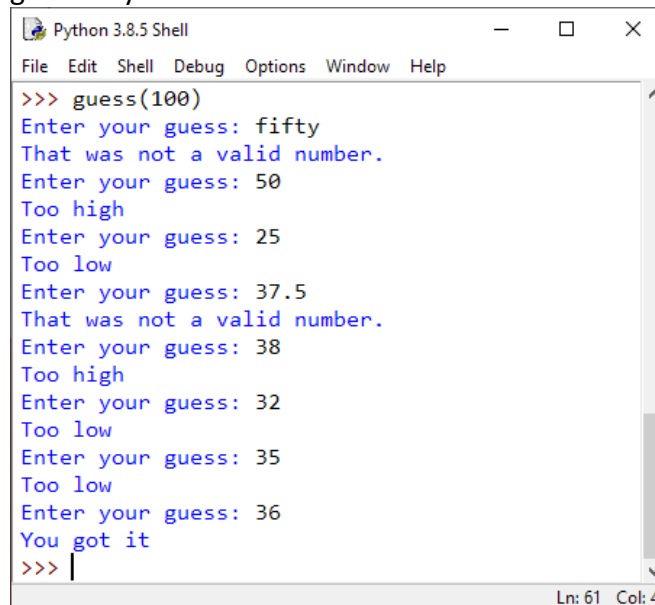


```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
>>> myList = [10, 25, -60, 100, -80, -40, 10]
>>> improve(myList)
Transactions: [10, 25, -60, 100, -40, 10]. Value: 45
>>>
>>>
>>> myList = [10, 25, -45, 60, -50, -40, 10]
>>> improve(myList)
Transactions: [10, 25, -45, 60, -40, 10]. Value: 20
>>>
>>>
>>> myList = [-5, -10, -20]
>>> improve(myList)
Oops, looks bad, cannot be improved
>>> |
```

2. (70pt) Infinite loop/Loop-and-a-half and exceptions

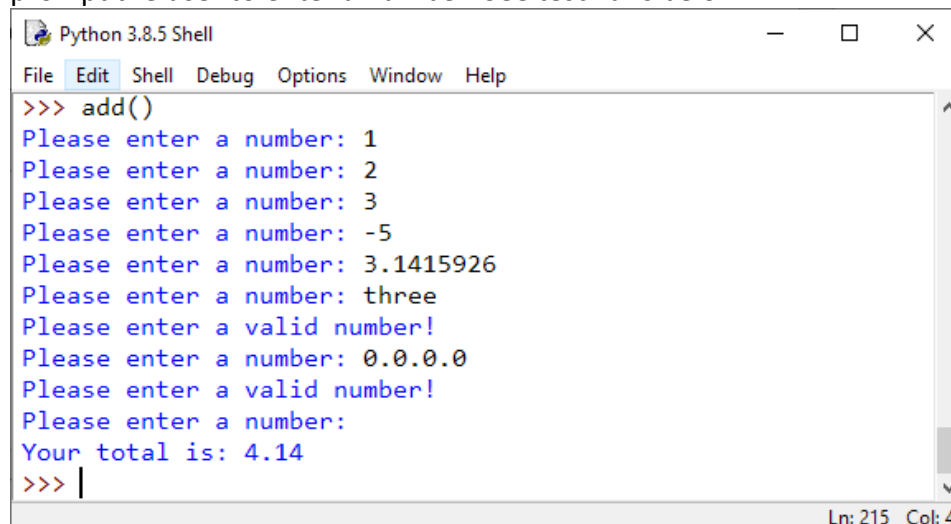
- (30pt) Implement a function `guess(n)` that takes an integer as a parameter and implements a simple, interactive guessing game. The function should start by choosing a random number in the range from 1 up to and including `n`. The

function should then repeatedly ask the user to enter to guess the chosen number. When the user guesses correctly, the function should print a message to congratulate the user and terminate. Each time the user guesses incorrectly, the function should indicate whether the guess was too high or too low. If the user types something other than an integer, the function should recover gracefully.



```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
>>> guess(100)
Enter your guess: fifty
That was not a valid number.
Enter your guess: 50
Too high
Enter your guess: 25
Too low
Enter your guess: 37.5
That was not a valid number.
Enter your guess: 38
Too high
Enter your guess: 32
Too low
Enter your guess: 35
Too low
Enter your guess: 36
You got it
>>> |
```

- b. (40pt) Write a program `add()` that keeps prompting the user for numbers until they simply hit return. At that point, the program should return the sum of all numbers entered by the user. If user enters something other than a number, prompt the user to enter a number. See test-runs below.



```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
>>> add()
Please enter a number: 1
Please enter a number: 2
Please enter a number: 3
Please enter a number: -5
Please enter a number: 3.1415926
Please enter a number: three
Please enter a valid number!
Please enter a number: 0.0.0.0
Please enter a valid number!
Please enter a number:
Your total is: 4.14
>>> |
```

Hint: use try/except for wrong user inputs.