Gold Hw8 Problem 3: TT Securities, Incorporated

Copied from: https://www.cs.hmc.edu/twiki/bin/view/CS5/TTSecuritiesGold on 3/29/2017

[25 points; individual or pair]

Filename: `hw8pr3.py`

Using your own loops...

Note for this problem: Do ***not*** use the built-in functions `sum`, `min`, or `max` here—instead, as you work through the different menu options, write helper functions that handle the data appropriately. If you *really* want to use those functions, feel free to write versions of your own (also, we worked through examples in class, in fact - or will do so on Thu 10/27)

Starter code

In class, we looked at an example user-interaction loop. Here is a variation on that code—this may be a good place to start with this problem. We've named the primary function `main()` in order to set things up similar to how you'll write the larger version described below:

```
#
# example user-interaction looping program
#   (a variant of the one done in class)
#

def menu():
    """ a function that simply prints the menu """
    print()
    print("(0) Continue!")
    print("(1) Enter a new list")
    print("(2) Predict the next element")
    print("(9) Break! (quit)")
    print()

def predict(L):
    """ predict ignores its input and returns
        what the next element should have been
    """
    return 42

def find_min(L):
    """ find min uses a loop to return the minimum of L
        input L: a nonempty list of numbers
        output:  the smallest value in L
    """
```

```
    result = L[0]
    for x in L:
        if x < result:
            result = x
    return result

def find_min_loc(L):
    """ find min loc uses a loop to return the minimum of L
              and the location (index or day) of that minimum
        input L: a nonempty list of numbers
        outputs:  the smallest value in L, its location (index)
    """
    minval = L[0]
    minloc = 0

    for i in list(range(len(L))):
        if L[i] < minval:  # a smaller one was found!
            minval = L[i]
            minloc = i

    return minval, minloc

def main():
    """ the main user-interaction loop """
    secret_value = 4.2

    L = [30,10,20]  # an initial list

    while True:     # the user-interaction loop
        print("\n\nThe list is", L)
        menu()
        uc = input( "Choose an option: " )

        # "clean and check" the user's input
        #
        try:
            uc = int(uc)   # make into an int!
        except:
            print("I didn't understand your input! Continuing...")
            continue

        # run the appropriate menu option
        #
        if uc == 9:    # we want to quit
            break      # leaves the while loop altogether

        elif uc == 0:  # we want to continue...
            continue   # goes back to the top of the while loop

        elif uc == 1:  # we want to enter a new list
            newL = input("Enter a new list: ")    # enter _something_

            # "clean and check" the user's input
            #
            try:
                newL = eval(newL)   # eval runs Python's interpreter! Note: Danger!
                if type(newL) != type([]):
                    print("That didn't seem like a list. Not changing L.")
                else:
                    L = newL  # here, things were OK, so let's set our list, L
            except:
                print("I didn't understand your input. Not changing L.")

        elif uc == 2:        # predict and add the next element
            n = predict(L)   # get the next element from the predict function
            print("The next element is", n)
            print("Adding it to your list...")
            L = L + [n]      # and add it to the list

        elif uc == 3:  # unannounced menu option!
            pass       # this is the "nop" (do-nothing) statement in Python

        elif uc == 4:  # unannounced menu option (slightly more interesting...)
            m = find_min(L)
            print("The minimum value in L is", m)
```

```
        elif uc == 5:  # another unannounced menu option (even more interesting...)
            minval, minloc = find_min_loc(L)
            print("The minimum value in L is", minval, "at day #", minloc)

        else:
            print(uc, " ?      That's not on the menu!")

    print()
    print("See you yesterday!")
```

The TTS problem ...

For this problem, you will implement a (text-based) menu of options for analyzing a list of stock prices (or any list of floating-point values). This provides an opportunity to use loops in a variety of ways, as well as experience handling user input.

The top-level function to write for this problem is called `main()`. Note that it takes no inputs. Instead, it should offer the user a menu with these choices:

```
(0) Input a new list
(1) Print the current list
(2) Find the average price
(3) Find the standard deviation
(4) Find the min and its day
(5) Find the max and its day
(6) Your TT investment plan
(9) Quit

Enter your choice:
```

Feel free to change the wording or text, but please keep the functionality of these choices intact. If you'd like to add additional menu options of your own design, please use different values for them (see extra credit, below).

Once the menu is presented, the program should wait for the user's input. (You may assume that the user will type an integer as input.) The function should then

- print a warning message if the integer is not a valid menu option
- quit if the user inputs `9`
- allow the user to input a new list of stock prices, if the user selects choice `0`
- print a table of days and prices, with labels, if the user selects choice `1`
- compute the appropriate statistics about the list for choices `2-6`

For any option except `9`, the function should reprompt the user with the menu after each choice.

Many of the pieces are straightforward, but here are a couple of pointers on two of them:

formatting neatly...

This is optional, but if you'd like to print neatly-formatted columns, the following approach will help. Try pasting the at the Python prompt, just to get the hang of it). The formatting-string indicates *how* things are formatted, and the inputs to `.format` pass *what* values will get formatted that way.

Here, the printed values are in blue:

```
In [3]: print("{0: >4} : $ {1: >6}".format("Day","Price"))
 Day : $  Price

In [4]: print("{0: >4} : $ {1: >6}".format(3,42))
   3 : $     42

In [5]: print("{0: >4} : $ {1: >6}".format(11,27042))
  11 : $  27042
```

Briefly, the *formatting string*, namely `"{0: >4} : $ {1: >6}"` is saying three things:

- print the format-input #`0` right-justified in a space with a width of 3"
- then, print a space, a colon, a space, a dollar sign, and a space (all as expected)
- then, print the format-input #`1` right-justified in a space with a width of 4

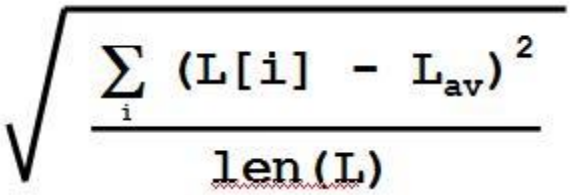
Notice that the `.format( input0, input1 )` after the formatting string provides the inputs needed!

Experiment with this to get it to look the way you'd like. And -- printing without worrying about the format is totally fine, too!

- **The time-travel strategy**: For menu option 6, you will want to find the best day on which to buy and sell the stock in question in order to maximize the profit earned. However, the ***sell day must be greater***

than or equal to the buy day. You may want to adapt the example function `diff` from class in order to find this maximum profit.

- **Calculating the standard deviation**: Please use this formula to calculate the standard deviation of the stock. Note that L_{av} is the average (mean) value of the elements of the list `L`. Also, it's OK to assume that `L` will always be non-empty.



Helper functions

You **need to** write a helper function for each of the menu options. You can always return more than one value from a function. For example

```
def f(x):
    return x, (x+4)
```

This function can be called as follows:

```
a, b = f(38)
```

where `a` will hold the value `38` and `b` will hold the value `42`

Cool!

We include an example run that illustrates one possible interface for your `main()` function a little further down on this page...

Extra credit: creative menu options (up to +5 pts)

If you'd like, you may add other menu options (under different numeric labels) that process the list in some other way of your design -- it can be serious or not. Either way, your options will be graded on what they do and how smoothly they interact with the user.

Remember that the CS 5 graders get lonely -- conversational programs are welcome!

Example run of the basic TTS program

Here is an example run -- you do not need to follow this format exactly (though you may), but it's meant to show an example of each menu possibility:

```
In [1]:  main()
(0) Input a new list
(1) Print the current list
(2) Find the average price
(3) Find the standard deviation
(4) Find the min and its day
(5) Find the max and its day
(6) Your TT investment plan
(9) Quit

Enter your choice: 0
Enter a new list of prices: [20,10,30]
(0) Input a new list
(1) Print the current list
(2) Find the average price
(3) Find the standard deviation
(4) Find the min and its day
(5) Find the max and its day
(6) Your TT investment plan
(9) Quit

Enter your choice: 1

Day  Price
---  -----
  0  20.00
  1  10.00
  2  30.00


(0) Input a new list
(1) Print the current list
(2) Find the average price
(3) Find the standard deviation
(4) Find the min and its day
```

```
(5) Find the max and its day
(6) Your TT investment plan
(9) Quit

Enter your choice: 2

The average price is 20.0


(0) Input a new list
(1) Print the current list
(2) Find the average price
(3) Find the standard deviation
(4) Find the min and its day
(5) Find the max and its day
(6) Your TT investment plan
(9) Quit

Enter your choice: 3

The st. deviation is 8.16496580928


(0) Input a new list
(1) Print the current list
(2) Find the average price
(3) Find the standard deviation
(4) Find the min and its day
(5) Find the max and its day
(6) Your TT investment plan
(9) Quit

Enter your choice: 4

The min is 10 on day 1


(0) Input a new list
(1) Print the current list
(2) Find the average price
(3) Find the standard deviation
(4) Find the min and its day
(5) Find the max and its day
(6) Your TT investment plan
(9) Quit

Enter your choice: 5
```

```
The max is 30 on day 2


(0) Input a new list
(1) Print the current list
(2) Find the average price
(3) Find the standard deviation
(4) Find the min and its day
(5) Find the max and its day
(6) Your TT investment plan
(9) Quit

Enter your choice: 6
Your TTS investment strategy is to

  Buy on day 1  at price 10
 Sell on day 2  at price 30

 For a total profit of 20


(0) Input a new list
(1) Print the current list
(2) Find the average price
(3) Find the standard deviation
(4) Find the min and its day
(5) Find the max and its day
(6) Your TT investment plan
(9) Quit

Enter your choice: 7


The choice 7 is not an option.
Try again



(0) Input a new list
(1) Print the current list
(2) Find the average price
(3) Find the standard deviation
(4) Find the min and its day
(5) Find the max and its day
(6) Your TT investment plan
(9) Quit
```

```
Enter your choice: 9

See you yesterday!
```