# Week 6, Ch 6, Part 1, Problem 1

# Pseudocode to write an algorithm which removes all negative values from a list.

# By: Brendan Albert

# This problem makes more sense to solve using a while loop rather than a for loop because,

# if we try to solve this with a for loop, everytime we find a negative element,

# it will be removed, thus shortening the list’s length, and then our index counter

# will now be at the index of the next element of the list when it increments, before we even

# have a chance to check the value of that element. So if there are any back to back negative

# elements, the second negative will slip through the cracks so to speak.

# The following solution should prevent that from happening.

Declare an index variable counter, and initialize it to zero.

While the counter variable is less than the length of the list:

Check if the value/element at the current index of the list is negative:

If the value at the current index is negative, pop that value out of the list.

Otherwise:

Increment index counter variable by 1.

Once every value has been checked, the list should now contain only positive elements.

# Week 6, Ch 6, Part 1, Problem 2

# Modifies the largest.py program in the text's Section 6.3.

# Mark both the smallest and largest elements of a list.

# By: Brendan Albert

# createList() function header. This function

# prompts a user to enter values to build a list,

# until the sentinel 'q'/'Q' is entered.

# Then the list is returned.

def createList():

# Declare values list variable

values = []

# Read the input values.

userInput = input('Please enter integer values, Q to quit. ')

while userInput.upper() != 'Q':

# Once it is confirmed that the user does not want to quit,

# we cast the input to an int so our largest and smallest

# function calls work correctly.

# If we keep the input as a string,

# we will get incorrect values for our largest

# and smallest elements.

values.append(int(userInput))

userInput = input('Next int value or Q to quit: ')

return values

# largest() function header. Accepts list argument

# and searches it to find the largest value, which is returned.

def largest(userList):

# declare/initialize largest variable

largest = userList[0]

for i in range(1, len(userList)):

if userList[i] > largest:

largest = userList[i]

return largest

# smallest() function header. Accepts a list as an argument.

# Iterates over the list, locates the smallest value and returns it.

def smallest(userList):

# declare/initialize smallest variable

smallest = userList[0]

for i in range(1, len(userList)):

if userList[i] < smallest:

smallest = userList[i]

return smallest

# markestList() function's header. Accepts the user's list.

# Will print each element of the list on a new line

# If the element is the smallest or largest, it will be marked accordingly.

def markedList(userList):

# Declare and initialize our smallest and largest element variables.

# We call their respective functions to get these numbers.

smallestElement = smallest(userList)

largestElement = largest(userList)

# We use a for-loop to cycle through each element of the list argument.

# Each element is printed on a new line.

# However, if the element is the smallest or largest of the list,

# a marker is added next to it before the new line is printed.

for element in userList:

print(element, end='')

if element == smallestElement:

print(" <== smallest value", end='')

if element == largestElement:

print(" <== largest value", end='')

print()

# Declare main() function header.

# Super short and simple.

# When main is called, the markedList() function is seen but

# it is not called yet because the createList() function is its argument,

# so, createList() is called first. createList() gets the list from the

# user, and returns the list. This returned list is then provided as

# the argument for markedList(). markedList then uses this list

# provided as an argument, to call the smallest and largest functions.

# Then the marked list is printed out.

def main():

markedList(createList())

# Entry point of the program, we simply call main()!

main()

