Name: Robert McKinney

**Week 6 Assignment Part 1 and Part 2**

I’m sorry the font for the code is so small – it wrapped in several places in this document.

**Part I – python code**

# Robert McKinney

# Week Six Assignment Part 1

def find\_longest\_word(wordList):

"""

Function to take a list as a parameter, and return the longest

item in the list

"""

print("The list of words entered is:", '\n', wordList)

longest\_word = "" # Declare empty variable to hold longest word

for word in wordList: # Iterate through each word in wordList parameter

if len(word) > len(longest\_word): # Use len function and if statement for comparison

longest\_word = word # Assign longest iterated word to previously declared variable

return longest\_word

# Get user input

user\_input = input('Please enter a few words separated by a space, and I will '

'find the longest: ')

# Use split function to separate string from user into a list

words = user\_input.split()

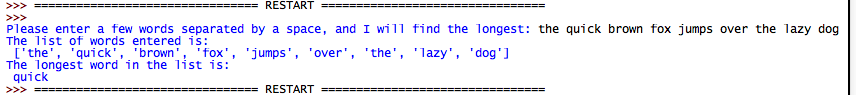
# Call function, pass the list, and assign returned value to variable

final\_word = find\_longest\_word(words)

# Display longest word to user

print("The longest word in the list is:", '\n', final\_word)

**Part I – screen shot of sample run**

****

**Part II – python code**

# Robert McKinney

# Week Six Assignment Part 2

all\_num = [] # Create empty list for all numbers

all\_pos\_num = [] # Create empty list for all positive numbers

all\_non\_pos\_num = [] # Create empty list for all negative numbers

# Get number from user

number = eval(input("Enter a number (-9999 to quit): ")) # Priming read, outside of loop

while number != -9999: # -9999 is the sentinel to stop loop

all\_num.append(number) # Insert all numbers into all\_num list

if number > 0:

all\_pos\_num.append(number) # Insert all positive numbers into all\_pos\_num list

elif number <= 0:

all\_non\_pos\_num.append(number) # Insert all non-positive numbers into all\_non\_pos\_num list

number = eval(input("Enter a number (-9999 to quit): ")) # Ask user to enter another number

def allNumAvg(numList):

"""Take a list of all numbers and calculate the average, then return average"""

total\_sum = 0 # Create empty variable to hold sum

total\_count = 0 # Create empty variable to hold count

for num in numList: # Iterate through each number in the list

total\_sum += num # Add each number to the sum variable

total\_count += 1 # Add one to count variable each loop

total\_average = total\_sum/total\_count # Calculate average

return total\_average # Return calculated average

def posNumAvg(numList): # Follows the same structure as "allNumAvg" function

"""Take a list of positive numbers and calculate the average, then return average"""

pos\_sum = 0

pos\_num\_count = 0

for num in numList:

pos\_sum += num

pos\_num\_count += 1

pos\_average = pos\_sum/pos\_num\_count

return pos\_average

def nonPosAvg(numList): # Follows the same structure as "allNumAvg" function

"""Take a list of negative numbers and calculate the average, then return average"""

non\_pos\_sum = 0

non\_pos\_num\_count = 0

for num in numList:

non\_pos\_sum += num

non\_pos\_num\_count += 1

non\_pos\_average = non\_pos\_sum/non\_pos\_num\_count

return non\_pos\_average

def average\_dictionary(total, pos\_total, neg\_total):

"""Take returned averages as parameters and create dictionary with those values"""

avg\_dictionary = {'AvgAllNum': total, 'AvgPositive': pos\_total, 'AvgNonPos': neg\_total}

return avg\_dictionary

# Call allNumAvg function and pass all\_num as parameter, saving returned value in variable

total\_average = allNumAvg(all\_num)

# Call posNumAvg function and pass all\_pos\_num as parameter, saving returned value in variable

pos\_average = posNumAvg(all\_pos\_num)

# Call nonPosAvg function and pass all\_non\_pos\_num as parameter, saving returned value in variable

non\_pos\_average = nonPosAvg(all\_non\_pos\_num)

# Print the list containing all numbers

print("The list of all numbers entered is: ", '\n', all\_num)

# Call the average\_dictionary function and pass three parameters, printing the returned item

print("The dictionary with averages is: ", '\n',

average\_dictionary(total\_average, pos\_average, non\_pos\_average))

**Part II – screen shot of sample run**

