# Week 7, Examples 1

# 1.py

# In 8.py of the last lecture we saw a faster way (1.5n comparisons) to find
# min and max simultaneously. We will code that in Python in this module.

# Remember that finding min and max simultaneously in a single function
# using the simple
# method requires 2n comparisons (n for min, n for max) because we have to
# traverse
# the whole n element list.

# By pairing elements (smaller, larger) we compare only what we need to
# compare (i.e.,
# compare min with smaller, compare max with larger
# and thus cut down on part of the work. But in all, since we need to
# traverse the whole list,
# it will still be need 1.5n comparisons.

```python
def even(n):
    if ((n%2)==0):  # tells if n is even or odd
        return(1)
    else:
        return(0)

def order (a,b):
    #returns the same pair as (smaller, larger), so
    increasing order
    if (a < b):
        return(a,b)
    else:
        return(b,a)

def minandmax(list):
    # returns (min,max) of the whole list

    n = len(list)

    if (even(n)):
        min,max = order(list[0], list[1])
        index = 2  #we start with a pair (min,max), n is even

    else:
        min,max = list[0],list[0]
        index = 1  # both min and max refer to the first item in
```

---

Computer Science Courses - http://courses.cs.purdue.edu/
list, n is odd

while (index < len(list)):
    smaller, larger = order(list[index], list[index+1])

    if smaller < min:
        min = smaller

    if larger > max:
        max = larger

    index = index + 2

return(min, max)

def main():

    evenlist = [ -1,7,12,99,-18,73,14,12]  #8 elements

    oddlist = [6,22,-100,27,202,-101,81,11,0]  #9 elements

    min, max = minandmax(evenlist)

    print(" evenlist: min = ",min," max = ",max)

    print(" ")

    min, max = minandmax(oddlist)

    print(" oddlist: min = ",min," max = ",max)

main()

#____________________________________________________________
#2.py

# A for-loop is a DEFINITE loop (you know the stop and start when it begins)

# Let's use our unusual on-the-fly mean and variance calculation from last week. We will
# insert it in loops. To begin with, a for loop that accepts data from keyboard.
def wait():
    x = input()
    print(" ")

def main():

    n = eval(input("Please enter the number of data points you want to work with: "))
    print("\n\n")
    print("____________________for loop
           ____________________")
    for i in range(n):
        newdata = float(input("Enter a number: "))
        if (i == 0):
            mean = newdata
            var = 0
            print(" The value of loop-index i is ", i)
        else:
            mean = ((n-1)/n)*mean + newdata/n
            var = ((n-1)/n)*var + ((newdata - mean)*(newdata-mean))/(n-1)
            # The above variance formula divides by n instead of (n-1)
            # and so is "biased"
            print(" The value of loop-index i is ", i)

        print("____________________end for loop
               ____________________")

    wait()

    print(" number of data points input = ", n)
    var = (var*n)/(n-1)  # correction to get "unbiased" variance
    print(" mean: {0:0.4f}".format(mean))
    print(" variance :{0:0.4f}".format(var))

main()

#____________________________________________________________
#3.py
# Now, without any specific computation let's look at how a WHILE-LOOP works
# We'll call functions to execute simple loops, so we understand

def whileloop_sum(n):
```
sum = 0

i = 0           # initialize the loop index before entering the while-loop

while (i <= n): # loop will be entered ONLY if the condition is true; tested on each pass
    print("while-loop index i = ",i)
    sum = sum + i
    i = i + 1               # DO NOT FORGET TO INCREMENT WHILE LOOP INDEX!

return(sum)

def forloop_sum(n):
    sum = 0

    for i in range(n+1):   # notice for the "definite loop" settings are automatic
        print("for-loop index i = ",i)
        sum = sum + i               # for-loop automatically increments the for-loop index i

    return(sum)

def badwhileloop_sum(n):
    sum = 0

    i = 0                   # initialize the loop index before entering the while-loop

    while (i <= n):         # loop will be entered ONLY if the condition is true; tested on each pass
        print("while-loop index i = ",i)
        sum = sum + i

#        i = i + 1           # suppose we forget to increment loop index (it's commented out)

# now the while-loop condition becomes true forever and it will never exit
    # run forever
```
def wait():
    x = input()
    print("_______________________________________________ ")

def main():

    wait()
    answer1 = whileloop_sum(10)
    print(" (First ) while-loop sum is ",answer1)
    wait()
    answer2 = forloop_sum(10)
    print(" (Second) for-loop sum is ",answer2)
    wait()
    answer3 = forloop_sum(-2)  # note we are calling for-loop with endpoint n = -1
    # observe how it tests if index is in range before entering
    print(" (Third) for-loop sum is ",answer3)
    wait()

    print("Look inside the while loop in the function call")
    print("to see why the Fourth loop runs forever (infinite loop)")
    answer4 = badwhileloop_sum(10)  # note we are calling for-loop with endpoint n = -1
    # observe how it tests if index is in range before entering
# the function will loop forever and not return, so
# kill the program with a CTRL-C

print(" Bad while-loop sum is ",answer4)

main()

#____________________________________________________________
#4.py
# Example of an INDEFINITE loop --- the user tells you when to stop looping
# Let's use our unusual on-the-fly mean and variance calculation from last week.

def wait():
    x = input()
    print(" ")

def main():

    done = 0    #using 0 to mean FALSE, so not done is TRUE
    n = 0       # counts data

    print("____________________while loop ______________________________________")

    while (not done):

        if (n == 0):
            newdata = eval(input(" Enter a number: "))
            n = 1
            mean = newdata
            var = 0
            print(" The value of loop-index i is ",n)

        else:
            newdata = eval(input(" Enter another number: "))
            n = n + 1
            mean = ((n-1)/n)*mean + newdata/n
            var = ((n-1)/n)*var + ((newdata - mean)*(newdata-mean))/(n-1)
            # The above variance formula divides by n instead of (n-1)
            # and so is "biased"
print("The value of loop-index i is ", n)

# Now check if the user wants to input more data

resp = str(input("Another data item? (Y/N, y/n, yes/no, etc): 

if ((resp[0] != "y") and (resp[0] !="Y")): 

    print(" done")

done = 1

    #did not get "y" or "Y", so loop is done

print("____________________end while loop 
__________________________")

wait() 

print(" number of data points input = ", n)

if ( n > 1):

    var = (var*n)/(n-1) 

    #correction to get "unbiased"

variance 

    print(" mean: {0:0.4f}".format(mean))
    print(" variance :{0:0.4f}".format(var))

main()

#____________________________________________________________
#5.py
# Example of a SENTINEL loop --- the user inputs a "sentinel" to tell you when to
# stop looping

# A "sentinel" is a special value (a negative number, if all your data is positive;
# or a number that you are sure will not occur in your data, such as
# 99999999)

# Let's use our unusual on-the-fly mean and variance calculation from last week.

def wait():

    x = input()

    print(" ")

def main():

    done = 0 

    #using 0 to mean FALSE, so not done is TRUE

    n = 0 

    #counts data items
newdata = eval(input(" Enter a number (or a negative number to quit): "))

if (newdata < 0):
    done = 1

print("____________________while loop
____________________")

while (not done):

    if (n == 0):
        mean = newdata
        var = 0
        # update the count
        n = n + 1
        print(" The value of loop-index i is ",n)
    else:
        newdata = eval(input(" Enter another number (or a negative
number to quit): "))
        if (newdata < 0):
            done = 1
        else:
            n = n + 1       #just read in a new data item
            mean = ((n-1)/n)*mean + newdata/n
            var = ((n-1)/n)*var + ((newdata -
mean)*(newdata-mean))/(n-1)
            # The above variance formula divides by n instead of (n-1)
            # and so is "biased"
            print(" The value of loop-index i is ",n)

    print("____________________end while loop
____________________")

wait()

print(" number of data points input = ",n)
if (n > 0) :
    var = (var*n)/(n-1)       #correction to get "unbiased"
    variance
    print(" mean: {0:0.4f}".format(mean))
    print(" variance :{0:0.4f}".format(var))

main()
#6.py

# Using numbers as sentinels can cause problems if the sentinel can also be real data.
# For example, if your data includes negative numbers, and your sentinel is negative.

# We can use a special character string such as the empty string " " as a sentinel

# Let's use our unusual on-the-fly mean and variance calculation from last week.

def wait():
    x = input()
    print(" ")

def main():

    done = 0  #using 0 to mean FALSE, so not done is TRUE
    n = 0  #counts data items

    newdata = input(" Enter a number (or hit Return key to quit): ")

    if (str(newdata) == ""):  #got an empty string
        done = 1

    print("________________while loop
___________________________")

    while (not done):

        if (n == 0):
            mean = float(newdata)
            var = 0
            # update the count
            n = n + 1
            print(" The value of loop-index i is ",n)
        else:

            newdata = input(" Enter another number (or hit Return key to quit): ")
            print("string = ",str(newdata))
if (str(newdata) == ""):  
done = 1                      # got an empty string 
else: 
    newdata = float(newdata)  
n = n + 1                      # just read in a new data item  
mean = ((n-1)/n)*mean + newdata/n   
var = ((n-1)/n)*var + ((newdata - 
mean)*(newdata-mean))/(n-1)   
    # The above variance formula divides by n instead of (n-1) 
    # and so is "biased"
    
print(" The value of loop-index i is ",n) 

print("________________________end while loop
________________________________") 

wait() 

print(" number of data points input = ",n) 
if (n > 0) : 
    var = (var*n)/(n-1)     # correction to get "unbiased" variance 
    print(" mean: {0:0.4f}".format(mean)) 
    print(" variance :{0:0.4f}".format(var)) 

main()

#___________________________________________________________________________
__
save the first set of numbers below in file "even.txt" and the second set of numbers in file "odd.txt" 
( that is, files with an even number and an odd number of entries, 
 respectively) 
The following program will use these files. You can change the file names if you like.

7 
-22 
1 
18 
-16 
12 
74 
-11 
55 
99
def even(n):
    if ((n%2)==0):  # tells if n is even or odd
        return(1)
    else:
        return(0)

def order (a,b):  # returns the same pair as (smaller, larger), so increasing order
    if (a < b):
        return(a,b)
    else:
        return(b,a)

def minandmax(list):  # returns (min,max) of the whole list

    n = len(list)

    if (even(n)):
        min,max = order(list[0], list[1])
        index = 2  #we start with a pair (min,max), n is even
    else:
        min,max = list[0],list[0]
        index = 1  # both min and max refer to the first item in list, n is odd

    while (index < len(list)):
smaller, larger = order(list[index], list[index+1])

if (smaller < min):
    min = smaller

if (larger > max):
    max = larger

index = index + 2

return (min, max)

def main():

    fname = input("Datafile for list with an even number of items? ")
    infile = open(fname,"r")

    evenlist = [ ]

    for line in infile:
        evenlist.append(float(line))

    print(evenlist)

    min, max = minandmax(evenlist)

    print(" evenlist: min = ", min, ", max = ", max)

    print(" ")

    infile.close()

    fname = input("Datafile for list with an odd number of items? ")
    infile = open(fname,"r")

    oddlist = [ ]

    for line in infile:
        oddlist.append(float(line))

    print(oddlist)

    min, max = minandmax(oddlist)

    print(" oddlist: min = ", min, " max = ", max)

    infile.close()