

Unit 3: Conditions, if and while

Skill Builder 2: Flag Waving

In this lesson, you will learn about *counters* and *accumulators* by calculating the number and the total of numbers entered and determine the average of the numbers.

Objectives:

- Counter statement
- Accumulator statement
- While loop with a flag
- Calculating an average (mean)

You will write a program that will count and add up a set of numbers. You do not know in advance how many numbers there will be. This lesson introduces the concept of a counter statement, an accumulator statement, a 'flag' value to end a loop, and the calculation of the average of the entered numbers.

Teacher Tip: Python shortcuts:

c=c+1 can be written c+=1 (but not c++)
t=t+n can be written t+=n
count=total=0 is allowed as well as
count, total = 0,0

1. Begin a new Python file and name it **count_ttl_avg.py**.

Make two variables and set them both equal to zero:

count = 0
total = 0

count keeps track of how many numbers are entered and

total keeps a running total of the numbers that have been entered.

A third variable, **num**, stores each number entered one at a time.

Rather than setting it equal to 0, use an **input** statement to get the first number from the user:

num=float(input("Enter a number: "))

2. Now start a **while** loop that will end when a certain number is entered. Some options here are 0, -999, or -1. Keep in mind that this unique number cannot be part of the set of numbers you are trying to process. We will use -999 as our 'flag' value to signal that we have finished entering numbers.

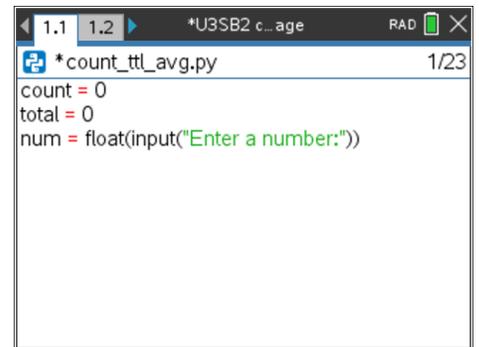
Select the **while** structure from **menu > Built-ins > Control**.

The 'does not equal' sign (**!=**) is on **ctrl+=**. We use it because we want the loop to *continue* as long as **num** *does not equal* -999.

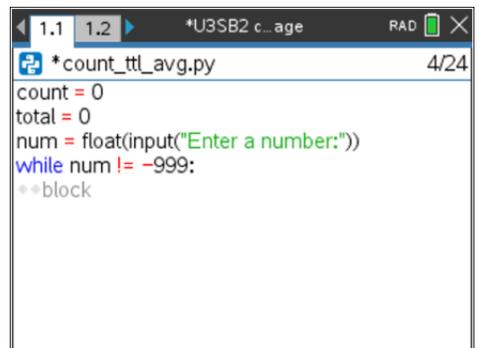
Remember to leave the colon (:) at the end of the line!

Next we will fill in the while **block**...

Teacher Tip: Flags (or sentinels), counters, and accumulators are common programming terms.



```
*count_ttl_avg.py 1/23
count = 0
total = 0
num = float(input("Enter a number:"))
```



```
*count_ttl_avg.py 4/24
count = 0
total = 0
num = float(input("Enter a number:"))
while num != -999:
  +=block
```



3. The loop block will:
 - a) Count the number of numbers entered.
 - b) Add them up.
 - c) Then, ask for another number.

Counting is done with the statement

count = count + 1

which adds 1 to the variable **count** each time it is processed.

Adding them up, or totaling, is done with the statement:

total = total + num

Asking for another number is an **input** statement like the first one

num=float(input("Enter a number (-999 to end):")) but here we add a message telling how to end the loop.

Teacher Tip: The screen above has a smaller font to make the long **input** line fit on the screen. To change the Python font size use **menu > Actions > Settings**. The only setting available is the font size. This font size is used by all Python Editors and Shells in the TI-Nspire document and is saved with the document.

```

1.1 *U3SB2 c...age RAD 8/9
*count_ttl_avg.py
count = 0
total = 0
num = float(input("Enter a number:"))
while num != -999:
    count=count+1
    total=total+num
    num = float(input("Enter a number (-999 to end:"))
  
```

4. We have finished with the loop. Skip a line or two and erase (backspace) the indent spaces.

Now is the time to process the inputted numbers. We kept track of the **count** and the **total** so we can calculate the average.

Try it yourself before looking at the next step...

```

1.1 1.2 *U3SB2 c...age RAD 6/18
*count_ttl_avg.py
count = 0
total = 0
num = float(input("Enter a number:"))
while num != -999:
count=count+1
total=total+num
num = float(input("Enter a number (-999 to end:"))
  
```

5. The statement

avg = total / count

calculates the average and stores the result in the variable **avg**.

Ponder this: Why do we use / here and not // as in the last lesson?

Finally, write the print statement(s) that report the count, total and average to the user. You may wish to use more than one print statement. Our code is shown in the next step...

```

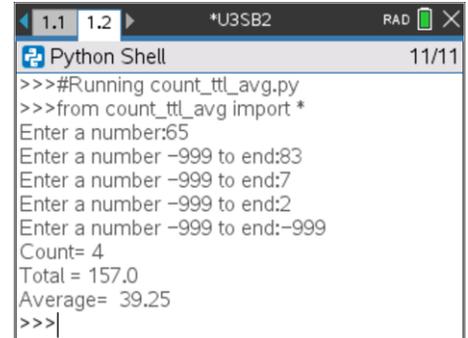
1.1 *U3SB2 c...age RAD 10/11
*count_ttl_avg.py
count = 0
total = 0
num = float(input("Enter a number:"))
while num != -999:
    count=count+1
    total=total+num
    num = float(input("Enter a number (-999 to en
avg = total / count
  
```



Teacher Tip: *Why do we use / here and not // as in the last lesson?*
 / is floating point division and results in decimal answers.
 // is integer division and the results are integers only.

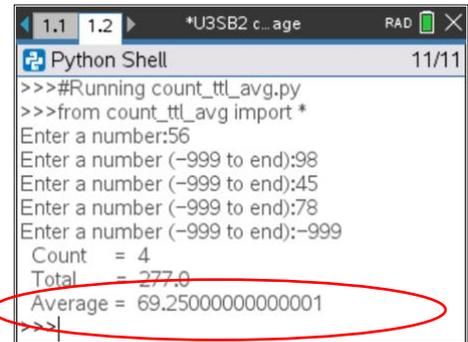
6. We use three print statements to report the results:

```
print(" Count   = ", count)
print(" Total   = ", total)
print(" Average = ", avg)
```



Special Note:

Sometimes you might notice the average has a lot of trailing zeros followed by a 1. Python's method of storing and computing with binary floating point numbers then converting them to decimal approximations can lead to stray decimals in some calculations. This anomaly is common with Python on virtually any computing platform and has nothing to do with TI-Nspire.



Challenge: As part of the input routines, display the number of numbers entered so far. The input function is not like the print statement where you can print text and variables together, separating them with commas. The input function can only display the (string prompt).

What happens if -999 is entered first (before any other numbers)?

Answer: *You will see the 'Divide by zero' error message.*