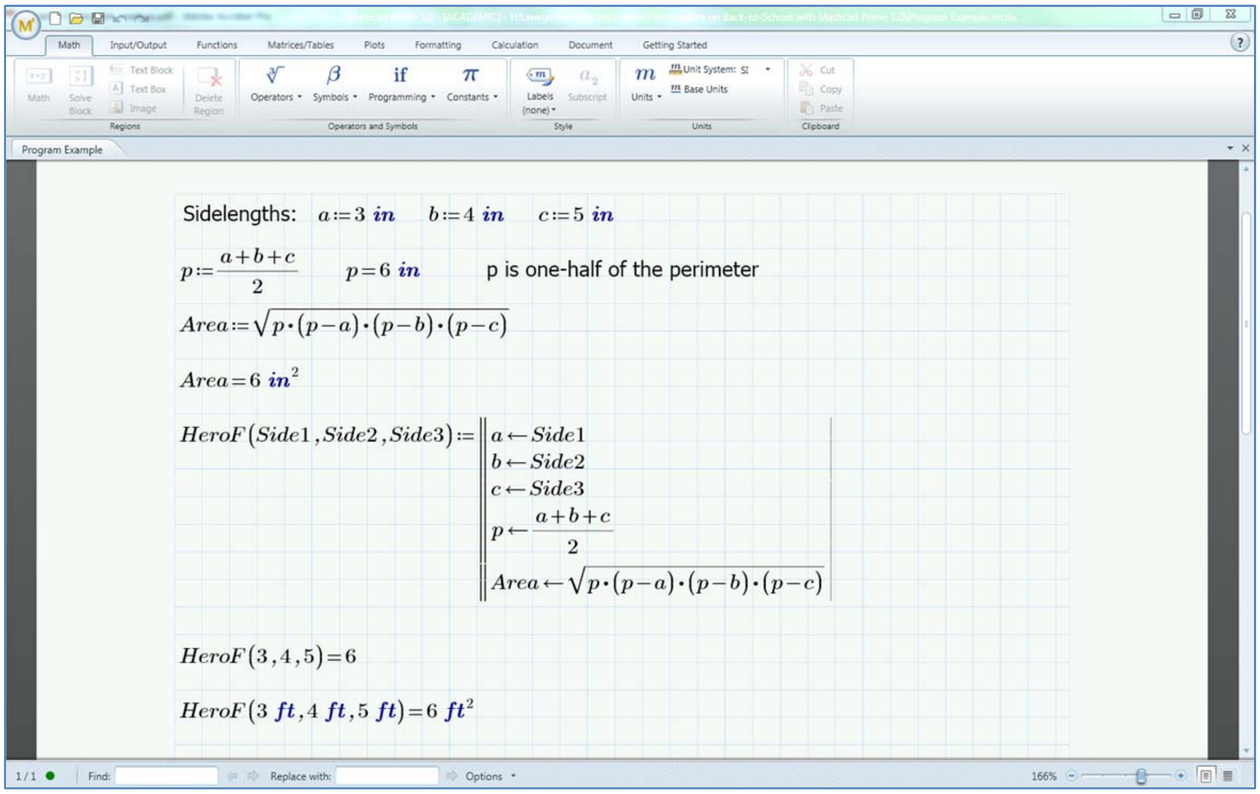
PTC®

Programming with Mathcad Prime™



Sidelengths: $a:=3\ in$ $b:=4\ in$ $c:=5\ in$

$p:=\frac{a+b+c}{2}$ $p=6\ in$ p is one-half of the perimeter

$Area:=\sqrt{p\cdot(p-a)\cdot(p-b)\cdot(p-c)}$

$Area=6\ in^2$

$$HeroF(Side1, Side2, Side3):=\left\| \begin{array}{l} a\leftarrow Side1 \\ b\leftarrow Side2 \\ c\leftarrow Side3 \\ p\leftarrow \frac{a+b+c}{2} \\ Area\leftarrow\sqrt{p\cdot(p-a)\cdot(p-b)\cdot(p-c)} \end{array}\right.$$

$HeroF(3,4,5)=6$

$HeroF(3\ ft, 4\ ft, 5\ ft)=6\ ft^2$

PTC Academic Program

Learn. Create. Collaborate. Succeed.

PTC®

Written By Chris Hartmann, Anji Seberino & Roger Yeh

Conditions of Use

Acknowledgments

The authors gratefully acknowledge the support of the Mathcad Business Unit and the PTC Academic Program.

Questions or Corrections

Please direct inquiries or questions regarding the contents of this tutorial to the Mathcad Academic Program at MathcadEducation@ptc.com. Suggestions for improvement or further development will be gladly accepted.

PTC®

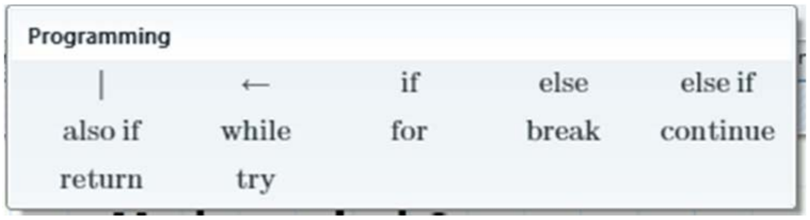
Programming with Mathcad Prime

Mathcad Prime is an environment for **Computational Thinking** – an approach to calculation, data analysis and problem solving that uses the capabilities of a computer to construct better solutions. When used appropriately computational thinking:

- Helps to illustrate the solution to a problem more clearly
- Produces meaningful answers to a challenging problems more quickly
- Off-loads some of the complexities of a problem to a computer in ways that enable better human insight into the nature of a problem in order to further the solution process

Mathcad Prime possesses a simple and easy to use Programming capability that allows users to write multistep functions directly on a worksheet alongside equations, tables, graphs, and text.

Mathcad Prime's Programming Commands



Using Mathcad's programming capabilities in math and science coursework can help students understand concepts more deeply. It also helps introduces them to some computer programming basics.

In this tutorial we will focus on two different ways to use Mathcad's Programming capabilities to **Go Beyond the Basics** in math and science courses:

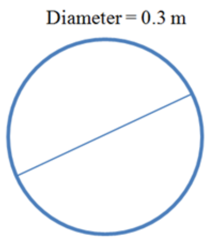
1. Formalizing Mathematical Relationships
2. Modeling Mathematical Processes

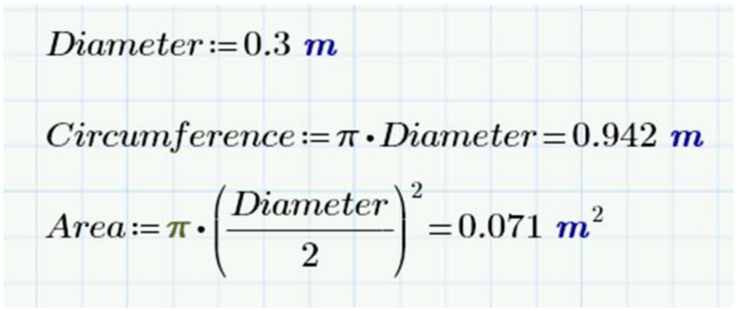
Each example will introduce a small number of Mathcad's Programming capabilities in order to help you to build up your proficiency in writing Functions on a Mathcad worksheet.

Formalizing Mathematical Relationships

When working on a Mathcad Prime worksheet, the expression of a formula is equivalent to writing a one-line computer program. I will illustrate this idea by showing two different ways to define the mathematical relationship between a diameter of a circle and the circumference and area measurements of the circle.

In a Mathcad worksheet it would be typical to use formulas to calculate the Area and Circumference of a circle using assignment statements such as those shown below.





$$Diameter := 0.3\ m$$

$$Circumference := \pi \cdot Diameter = 0.942\ m$$

$$Area := \pi \cdot \left(\frac{Diameter}{2}\right)^2 = 0.071\ m^2$$

Writing a computer program on a Mathcad worksheet is a lot like defining and evaluating formulas like those above. Once defined, the formulas above instruct Mathcad to perform all of the steps in the definition and return a result. ***Similarly, Mathcad programs report the last result calculated or assigned in the program.*** Returning this result is a lot like reporting the final result after following order of operations in a numerical expression.

We can use Mathcad's Programming tools to create the formulas above as Mathcad programs. Although we do not need to write programs to perform these calculations, we will use this simple example to introduce the Local Assignment (←) operator and the Program Structure (|) symbol on Mathcad's Programming Toolbar.

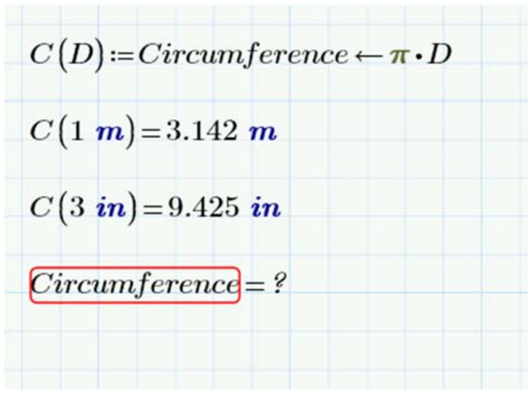
Local assignment in simple Mathcad programs

When writing computer programs it is good programming practice to reduce the memory requirements of a program by defining variables that only exist while the program is running. Similarly, in a Mathcad worksheet it is good practice to carefully manage the variable definitions one uses. Since variable definitions are live everywhere down and to the right of the initial assignment in a Mathcad Worksheet, Local Assignment in programs reduces the total number of live variable definitions.

In the box below, a simple one-line Mathcad program is used to calculate the circumference of a circle using the measure of a diameter. This example demonstrates how the Local Definition operator works in a Mathcad worksheet.

Local Definition

In the image at right the formula for circumference of a circle is defined in a Mathcad Program using Mathcad Prime's Local Definition operator ▯←▯. In this case the use of the Local Definition helps to clarify the meaning of the program named *C*. *C* accepts a single parameter, the measurement of a diameter of the circle. During program execution *Circumference* is calculated using the formula π*D. *C* returns the value of *Circumference*, the last value calculated. But, since it is defined locally, the value of *Circumference* is undefined outside of the program definition. Therefore, *Circumference* is undefined on the worksheet and returns an error when evaluated.



$C(D) := Circumference \leftarrow \pi \cdot D$

$C(1\ m) = 3.142\ m$

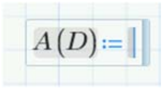
$C(3\ in) = 9.425\ in$

$\boxed{Circumference} = ?$

Follow these steps to create a Mathcad program that calculates the area of a circle given its Diameter as an input parameter:

Step 1: Type the following keystrokes to give the program a name and a parameter:

A(D: You should see this on the worksheet:



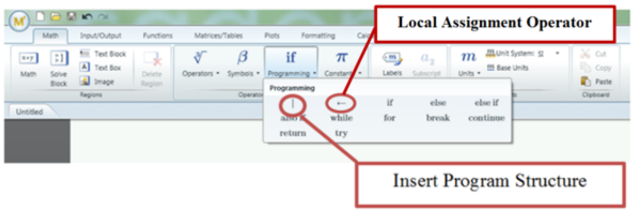
$A(D) :=$ ▯

Step 2: Mathcad shows that it is waiting for the program to be defined on the right side of the assignment statement by leaving the blue cursor in the placeholder. Follow these steps to insert the first line of the program:

Left-click to select the **Math** ribbon

Left-click on the **Programming** icon

Left-click the Insert Program Structure icon on the Programming menu

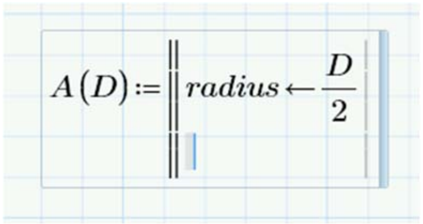


Type: ***radius*** in the placeholder

Left-click on the Programming icon on the Math ribbon

Left-click on the Local Assignment Operator icon on the Programming menu

Type: **D/2** in the placeholder. Hit the **Right Arrow** key once to advance the cursor and then hit the **ENTER** key. A new Program line will appear as shown.



$A(D) := \left\| \begin{array}{l} radius \leftarrow \dfrac{D}{2} \\ ▯ \end{array} \right.$

Step 3: Follow these instructions to add the second line of the program:

Type **Area** in the placeholder

Left-Click on the Local Assignment operator icon on the Programming toolbar

Left-Click on the Constants icon on Mathcad's Math ribbon

Left-Click on the symbol for **π** to insert the value for this constant

Type ***radius^2** to finish the definition of the formula for Area and then hit the **ENTER** key

Left-Click outside of the Math region to complete the definition of the program

Using a Mathcad Program

Once defined, a Mathcad program can be called anywhere below the program definition in the worksheet using an evaluation statement. ***A*** is a program to find the area of a circle. The program has two steps: (1) calculate the radius of the circle and (2) calculate the area using the formula π*radius^2.

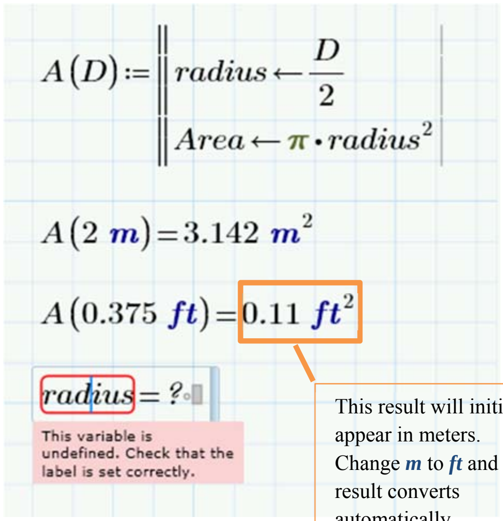
We can call (or evaluate the) the program on the worksheet by typing:

A(2m=

Or

A(0.375ft=

If we attempt to evaluate *radius* outside of the function, we receive an error message since it is a locally defined variable.



$$A(D) := \left\| \begin{array}{l} radius \leftarrow \frac{D}{2} \\ Area \leftarrow \pi \cdot radius^2 \end{array} \right.$$

$$A(2\ m) = 3.142\ m^2$$

$$A(0.375\ ft) = \boxed{0.11\ ft^2}$$

$$radius = ?$$

This variable is undefined. Check that the label is set correctly.

This result will initially appear in meters. Change ***m*** to ***ft*** and the result converts automatically.

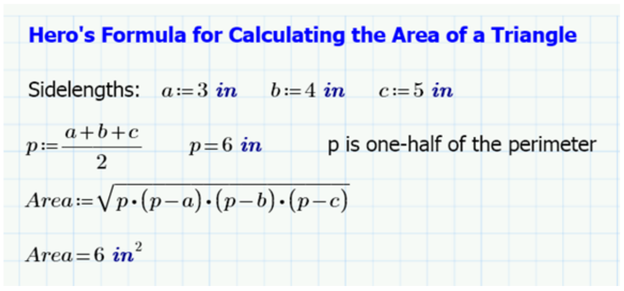
Notes about Program A (Area of a Circle)

A is a simple Mathcad program that can also be written as a single line statement, rather than a multiline program. We have used this example to introduce two important Mathcad Programming tools: new Program structure (|) and Local Assignment (←). The new Program structure icon allows for multi-line function definitions. The Local Assignment operator enables variable names to be created and used within a Mathcad function. These local variables do not exist outside of the program definition on the worksheet.

The presentation of ***A*** in this tutorial highlights the ways that Mathcad's programming capabilities allow for both clear communication and efficient notation. In this program the use of local variables for ***radius*** and ***Area*** in the function definition clearly identifies the purpose of the program. The use of ***A*** as the name of the function makes it easy to call the function as often as necessary within a worksheet.

Programming Practice: Formalizing Mathematical Relationships

Hero's Formula can be used to calculate the area of a triangle when you know the lengths of all three sides of the triangle. The image below shows how to use Hero's Formula in a Mathcad worksheet.



Hero's Formula for Calculating the Area of a Triangle

Sidelengths: $a := 3\ in$ $b := 4\ in$ $c := 5\ in$

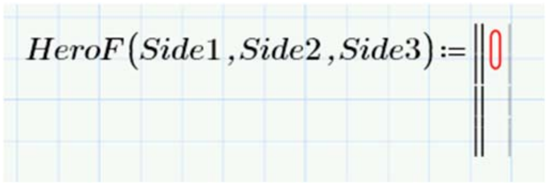
$p := \frac{a+b+c}{2}$ $p = 6\ in$ p is one-half of the perimeter

$Area := \sqrt{p \cdot (p-a) \cdot (p-b) \cdot (p-c)}$

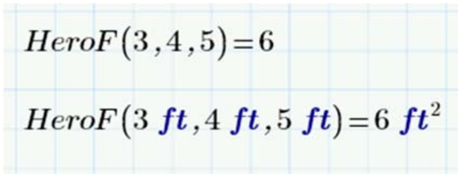
$Area = 6\ in^2$

Square Root Symbol
The "\" backslash key is the hotkey for the square root symbol in Mathcad.

Task: Use Mathcad's Programming capabilities to create a function called HeroF that takes the lengths of the 3 sides of a triangle as input parameters as shown at left below. The program should return the calculation of the area of the triangle as shown in the two examples at right.



$HeroF(Side1, Side2, Side3) :=$



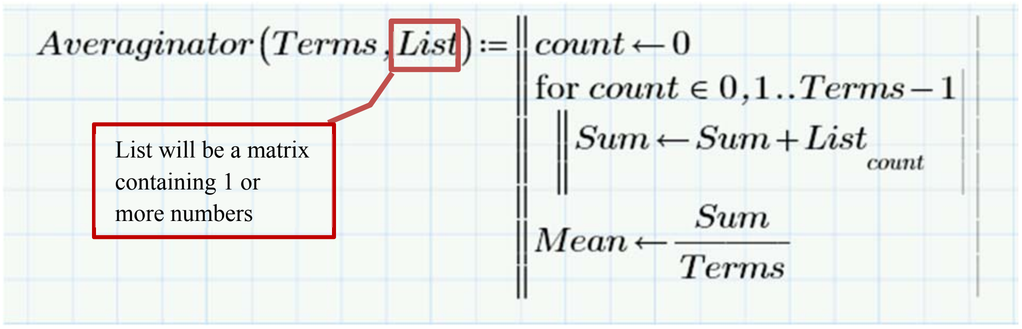
$HeroF(3,4,5) = 6$

$HeroF(3\ ft, 4\ ft, 5\ ft) = 6\ ft^2$

Modeling a Mathematical Process

Many mathematical concepts can be modeled in multiple ways, one of which is through iteration (repeating the same step over and over). For instance, the concept of the mathematical mean (a.k.a. average) of a set of numbers is often informally defined as “add up all the numbers in a list and then divide by the number of terms in the list.” Such a definition can be programmed quite literally using Mathcad’s Programming capabilities. I will use this example to demonstrate how to create a looping structure in a Mathcad function.

In this part of the tutorial I will introduce the steps necessary to create the following function:



```
Averaginator(Terms, List) := | count ← 0
                             | for count ∈ 0,1..Terms−1
                             |   | Sum ← Sum + List_count
                             | Mean ← Sum / Terms
```

Averaginator is a function that follows the informal definition of a mathematical average in the above paragraph. One goal of this part of this tutorial is to show how a program can be used to mimic a process in a step-by-step manner.

Step 1: Define the Program Name and Parameters List

Type **Averaginator(Terms,List:** to create the left side of the assignment statement for the program

Left-Click on the **Math** ribbon and then **Left-Click** on the **Programming** icon

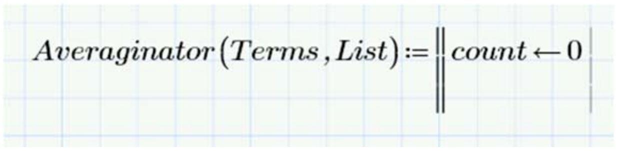
Left-Click on **|** to insert a new **Program Structure** in the function definition

Step 2: Define a local variable ***count*** to control the execution of the loop

Type ***count*** in the placeholder on the first line of the program

Left-click on the **Programming** menu and then **left-click** on ← to insert a **Local Assignment** symbol

Type **0** and hit **Enter** to add a new line to the function definition



```
Averaginator(Terms, List) := | count ← 0
```

In this example we will count from 0 because count will be used to control the execution of the loop and as a matrix subscript to access the terms in the list of values to be averaged.

Step 3: Define a local variable to keep track of the ***sum*** of the terms in List

Type ***sum*** in the placeholder on the second line of the program

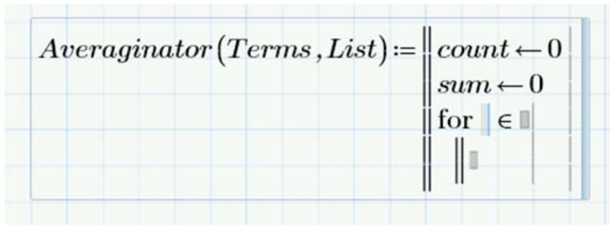
Left-click on the **Programming** menu and then **left-click** on **←** to insert a **Local Assignment** symbol

Type **0** and hit **Enter** to add a new line to the function definition

Step 4: Create a **for** loop in the program to calculate the average of the terms in List

With the blue cursor in the placeholder on the third line **Left-Click** on the **Programming** icon

Left-Click on **for** in the **Programming** toolbar. Mathcad will insert a template for defining a **for** loop


$Averaginator(Terms, List) :=$ $count \leftarrow 0$, $sum \leftarrow 0$, for $\in$

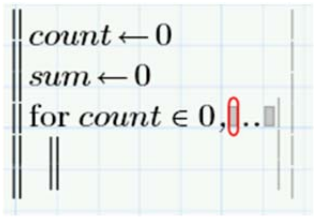
*The **for** loop template has placeholders for the name of the counter variable, the range of values for the counter, and the body of the loop. The body can be one or more lines.*

Step 5: Complete the values in the template for the **for** loop

In the first placeholder type ***count***, the name given to our local variable for controlling the loop

Create a range variable: In the second placeholder type **0,**

Mathcad will insert a template for a **range variable** as shown below:

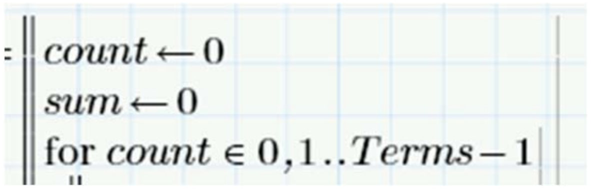

$count \leftarrow 0$, $sum \leftarrow 0$, for $count \in 0, \,..$

In the first empty placeholder for the **range variable** type **1**

In the second placeholder for the **range variable** type ***Terms* - 1**

Left-Click in the placeholder for the body of the for loop

Important Note: The list of terms to be averaged will be passed to the program as an nx1 matrix, so *count* will also be used as a matrix subscript to refer to each element in the matrix. By default, Mathcad defines the first position in an array as position 0.

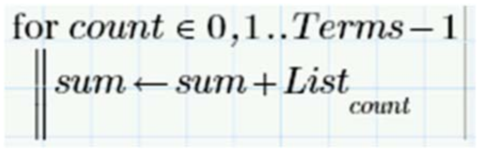

$count \leftarrow 0$, $sum \leftarrow 0$, for $count \in 0, 1 .. Terms - 1$

Step 6: Enter the expression to sum the terms in the List matrix as a line in the body of the **for** loop

Type ***sum*** in the placeholder of the body of the **for** loop

Insert a **Local Assignment** operator

Type **sum+List[count** to complete the definition of sum



$$\begin{array}{l} \text{for } count \in 0,1..Terms-1 \\ \quad \| \; sum \leftarrow sum + List_{count} \end{array}$$

Step 7: Calculate the average (mean) as the sum divided by the number of terms

Use the **right arrow** key to move the blue cursor to the right until it extends to the full height of the **for** loop template. You should only need to hit it twice. If necessary, you can place the cursor outside of the **for** loop with your mouse.

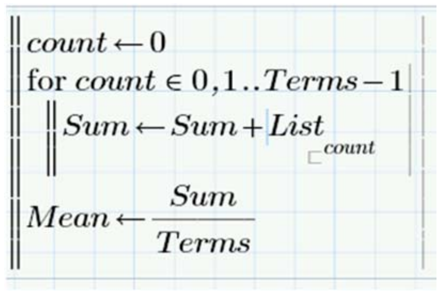
Hit the **Enter** key to create a new line in the function definition

In the placeholder type **Mean**

Insert a Local Assignment Operator

Type **Sum/Terms** in the open placeholder

Click in the worksheet outside the Averaginator region



$$\begin{array}{l} count \leftarrow 0 \\ \text{for } count \in 0,1..Terms-1 \\ \quad \| \; Sum \leftarrow Sum + List_{count} \\ Mean \leftarrow \dfrac{Sum}{Terms} \end{array}$$

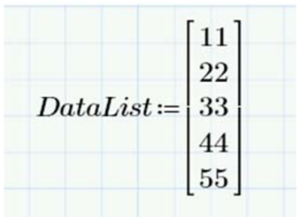
Step 8: In the main body of the Mathcad worksheet define a variable containing a 5x1 matrix

Type **DataList:**

Left-Click on the **Matrices/Tables** tab

Left-Click on the **Insert Matrix** icon and drag select a **5x1** matrix

Type 5 numerical values into the matrix placeholders

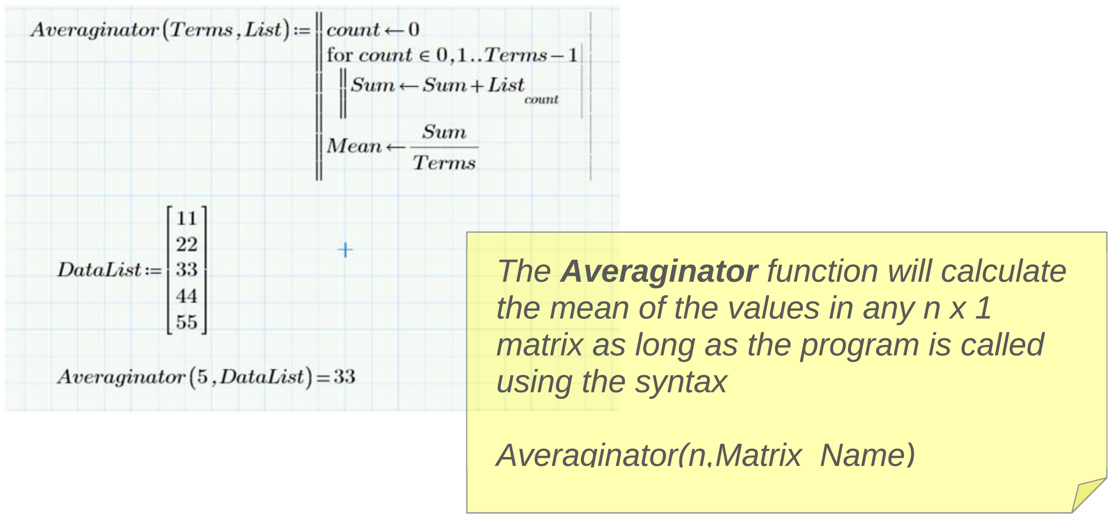


$$DataList := \begin{bmatrix} 11 \\ 22 \\ 33 \\ 44 \\ 55 \end{bmatrix}$$

Step 9: Test the Averaginator Program

Beneath the definition of the **Averaginator** function and the **DataList** variable type:

Averaginator(5,DataList)=



```
Averaginator(Terms, List) := | count ← 0
                             | for count ∈ 0,1..Terms−1
                             |   | Sum ← Sum + List_count
                             | Mean ← Sum / Terms
```

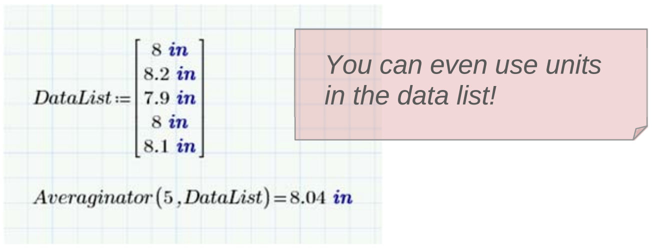
$DataList := \begin{bmatrix} 11 \\ 22 \\ 33 \\ 44 \\ 55 \end{bmatrix}$

$Averaginator(5, DataList) = 33$

*The **Averaginator** function will calculate the mean of the values in any n x 1 matrix as long as the program is called using the syntax*

Averaginator(n,Matrix Name)

Try the Averaginator Program using different values in the *DataList*!



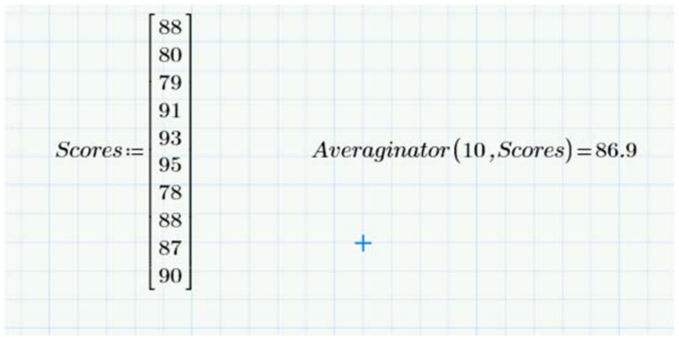
$DataList := \begin{bmatrix} 8\ in \\ 8.2\ in \\ 7.9\ in \\ 8\ in \\ 8.1\ in \end{bmatrix}$

$Averaginator(5, DataList) = 8.04\ in$

You can even use units in the data list!

Try the Averaginator Program using a different size data set!

(Review Step 8 if you forget how to create a Matrix variable)



$Scores := \begin{bmatrix} 88 \\ 80 \\ 79 \\ 91 \\ 93 \\ 95 \\ 78 \\ 88 \\ 87 \\ 90 \end{bmatrix}$

$Averaginator(10, Scores) = 86.9$

Programming Practice: Modeling a Mathematical Process

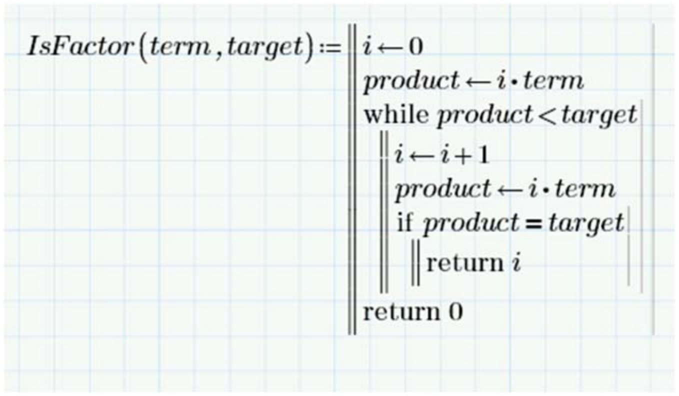
Cooper's Is it a factor? Algorithm

While I was writing this tutorial, I took a break to help my 10-year-old son learn how to identify if a given number is a factor of a target number. As he described his process to me, I realized that it would be relatively easy to recreate his thinking process as a Mathcad program. Here is the gist of his strategy in English and as a Mathcad program:

1. Starting with the given number itself, work your way through the multiples of the given number until you reach a number that is greater than or equal to the target number.

2. If you reach the exact target number, write down the number of multiples because that is another factor of the target number

3. If you exceed the target number, then the given number is not a factor of the target number



```
IsFactor(term, target) := || i ← 0
                          || product ← i · term
                          || while product < target
                          ||     i ← i + 1
                          ||     product ← i · term
                          ||     if product = target
                          ||         return i
                          || return 0
```

<u>Task Notes</u>:

1. A **while** loop is a conditional loop. It is created in a similar way as you create a **for** loop. In the ***IsFactor*** program above the **while** loop runs until ***product*** is greater than or equal to ***target***.
2. Inside the **while** loop an **if** statement is used to determine if the product is currently equal to the ***target*** value. You need to use the **if** command from the Mathcad Prime programming palette to insert an **if** statement into a program.
3. The **return** command is a special command that ends the execution of the program and returns the value given in the return statement. In this case we use return to return the value of the other factor or the value 0 if the given ***term*** is not a factor of the ***target***.

<u>Task:</u> Create a Mathcad program that determines if a given number is a factor of a target number. You may recreate the program above for practice, or write your own program from scratch.

Summary

1. In this tutorial you have learned some basic Mathcad Programming capabilities. Mathcad makes it easy to include computer programs in a worksheet alongside text, numerical calculations, tables, and graphs.
2. The following skills were emphasized in this tutorial:
   - Defining a Mathcad program to perform a mathematical calculation or process
   - Using local variable assignment to create self-documenting programs
3. The following Mathcad Programming tools were introduced in a tutorial:
   - Program Structure "|"
   - Local Assignment "←"
   - ***For*** loop
4. The following Mathcad Programming tools were shown in a demonstration:
   - **While** loop
   - **if** statement
   - **return** statement
5. To learn more about writing programs using Mathcad Prime see the **Programming Tutorial** that is available from the Getting Started tab on the Mathcad Prime ribbon.

Great job completing this tutorial!