Introduction to Programming in MatDeck

MatDeck allows the integration of text editing, script language, the ability to generate GUIs, flowcharts, virtual instrumentation, data visualization, programing, and parallel processing on multiple computers all to be comprehensively done within its documents. Expressing programming ideas in MatDeck takes usually less lines of code compared to others. The main objectives of this example are: illustrate enough MatDeck script that it is easy to do most common data manipulations, analyzing, and comparing, and provide a firm knowledge foundation so that learning more advanced MatDeck techniques is possible.

We presume the reader will be looking for information about why and how to get started with MatDeck. Fortunately an experienced programmer in any programming language (whatever it may be) can pick up MatDeck script very easy. More important, It's also easy for beginners to use and learn programming in MatDeck. We will illustrate several examples about basic flow control in MatDeck.

MatDeck if...else statement

In MatDeck, programming is done within functions, all variables used within a function are used as local variables. Functions can modify global variables defined in the document. User made functions can have one or more arguments. The code can be written in Canvas, or in Text Mode, which is preferred in this example. We switch to Text Mode by ctrl+i.

The `if` statement evaluates the test expression inside the parenthesis. If the test expression is evaluated to true (nonzero), statement(s) inside the body of `if` is executed. If the test expression is evaluated to false (0), statement(s) inside the body of the `if` statement are skipped from execution, or the statement inside the body of else is executed.

Let us illustrate a simple program in MatDeck used to check whether an integer entered by the user is odd or even.

```
1  // Program to check whether an integer entered by user is odd or even
2  program()
3  {
4    print("Enter number")
5     // Console input
6    number := getc(1)
7     // Even if remainder is zero, divided by two
8    if(remainder(number, 2) == 0)
9    {
10     print("Even")
11   }
12   else
13   {
14     print("Odd")
15   }
16   return(0)
17    // End of program
18 }
```

The program must be executed. In the next line we call the above user made function. Because we use console input and output here, we have to evaluate the document using the Build And Run Exe option.

```
19 program()
```

MatDeck For Loop

Loops are used in programming to repeat a block of code until a specific condition is met. There are two loops in MatDeck programming: for and while. The initialization statement used for the for loop is executed only once. Then, the test expression is evaluated. If the test expression is false, the for loop is terminated. But if the test expression is true, codes inside the body of the for loop is executed and the update expression is updated. This process repeats until the test expression is false. The for loop is commonly used when the number of iterations is known.

Let us illustrate the use of a for loop in MatDeck by writing a program which calculates the sum of the first n natural numbers, where n is given by user.

```
20  example()
21  {
22     // Sum of first n natural numbers
23     // User enters n
24    print("Enter a positive integer")
25    n := getc(1)
26     // initialize result, as global variable with respect to for(;;)
27    result := 0
28     // for(initialization; test expression; update expression)
29    for(i := 1; i <= n; i = i + 1)
30    {
31      result += i
32    }
33    print("Sum is")
34    print(result)
35    return(0)
36     // End of program
37  }
```

The program must be executed. In the next line we call the above user made function. Because we use console input and output here, we have to evaluate the document using the Build And Run Exe option.

```
38  example()
```