

What is a Function?

DocumentID : What_is_a_Function.PDF
Author : Michele Harris
Version : 1.0
Date : 07-03-2009

Contents

What is a Function?	2
A Function's Syntax	2
Function Exercise	4

What is a Function?

A Function is the second type of Procedure available in Visual Studio. Again, Procedures are employed to organize a group of related statements to perform a specific task. The difference between a Sub Procedure and a Function is that a function **always** returns a value associated with its name.

The individual statements within a function procedure handle data, compute a calculation, or processing text. Like Sub Procedures, functions can be utilized or called within other events or procedures.

A Function's Syntax

The syntax of a function is as follows.

```
Function FunctionName([arguments]) As Type  
    function statements  
    [Return value]  
End Function
```

FunctionName is, of course, merely the name of the new function.

As *Type* designates what type of value our function will produce. This particular designation is optional in Visual Basic 2008, but it's strongly recommended. If you fail to designate a type, VB defaults to setting it as an object, which during some hefty calculations might significantly slow your program.

The arguments are an optional list of arguments that tell the function how to handle the data. Each argument will need to be declared as a Type. These arguments are enclosed in parentheses, and even if you choose not to use an argument, you'll still need to include the parentheses as a placeholder.

Function Statements are lines of code that perform the processes of the function. Oftentimes variables are declared in the first few statements, while the following statements process those variables to return a value.

Since functions always return a value to their initial procedure (FunctionName), the final statement in a function puts the final calculation equal to that function's name.

Let's look at a function in its most basic form.

```
Function Math () As Integer
    Math = 1 + 2
End Function
```

Here our function name is Math, and the value it will return is an integer. The statement sets our Function's resulting value to the value of our calculation, 1+2. Pretty simple, right? Let's look at another example. In this one, we're going to declare a variable.

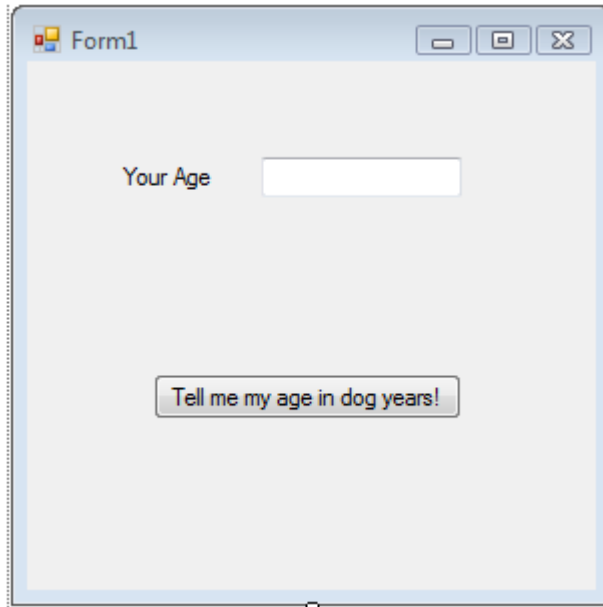
```
Function MinYear () As Long
    Dim Minutes As Long
    Dim Hours As Long
    Dim Days as Long
    Minutes = 60
    Hours = 24
    Days = 365.25
```

```
MinYear = Minutes * Hours * Days  
End Function
```

In this function, we're declaring variables to calculate how many minutes are in a year. The first three lines merely declare our variables: minutes, hours, and days. The next three lines assign values to those respective variables. Minutes is set to 60, as there are sixty minutes in an hour. Hours is set to 24, as there are 24 hours in a day. And of course, 365.25 is set to days, as that's the mean number of days in each year. The final line calculates the multiplication of these three numbers and returns it as a value for our function. Getting the hang of it now? Let's try an exercise.

Function Exercise

Open Visual Studio 2008, and create a new Windows form called Function Example. You should now see a blank Windows form. On this newly created form, you'll want to add 1 button, 1 label, and 1 text box. Change the Text Property of your button to "Tell me my age in dog years!" Change the Text Property of your label to "Your Age." Finally, place your button, label, and text box in the same arrangement displayed below:



Now double click your form to go to the Code Editor. You should see the follow code already in place:

```
Form1.vb* Form1.vb [Design]*
(General) (Declarations)
Public Class Form1
  Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
  End Sub
End Class
```

Place your cursor right after the “End Sub” statement, and hit enter. Type the following code in the lines below:

```
Function DogYears() As Integer
  Dim YourAge As Integer
  YourAge = Val(TextBox1.Text)
  DogYears = YourAge * 7
End Function
```

Here we’re creating a function to determine a person’s age in dog years. The first line below the initial function statement declares a new variable, YourAge, as an integer. The next line sets the

value of YourAge to whatever value the user places in the TextBox at runtime. And the next line merely multiplies the user's supplied age by 7, and returns that as the value for our function.

Now, you'll need to add a little more code so that we can call this function in another event. Go back to the form, and double click the Button. You should be taken to the following screen:

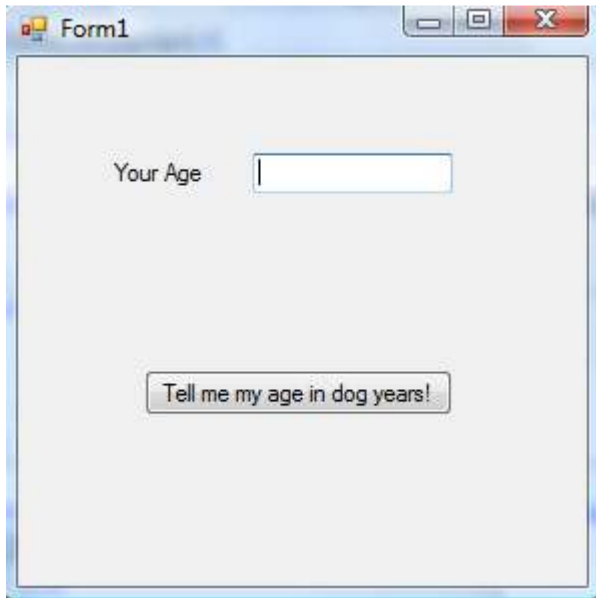
```
Public Class Form1
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
    End Sub
    Function DogYears() As Integer
        Dim YourAge As Integer
        YourAge = Val(TextBox1.Text)
        DogYears = YourAge * 7
    End Function
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
    End Sub
End Class
```

Place your cursor on the line following the Button1_Click statement, and enter the following line of code:

```
MsgBox (DogYears)
```

This line merely tells Visual Studio to launch a new message box displaying the result of our function whenever a user clicks the button. Okay, we're ready to test our new program out!

Press F5 to start debugging. You should shortly see the following screen:



Now enter your age into the Text Box and press the button. You should now be seeing a new message box displaying your age in dog years. Congratulations! You've successfully created and utilized a function in your program!
