

# User-defined Functions (UDFs)

In this lecture I will be introducing the programming language **Visual Basic for Applications (VBA)** and explaining one particular use of it: creating UDFs.

VBA is a powerful programming language which allows you to write your own programs in very much the same way as other programming languages (C, Fortran, Java, Pascal etc.)

In this module we are going to learn how to use VBA in combination with Excel, although VBA can also be used independently from Excel.

In this module we will learn how to write **UDFs** and **Macros or Subroutines** (see Part II).

## What is a UDF?

Just like a built-in function, a UDF is a pre-defined formula which can be executed in the same way. The difference is that you design the definition using VBA.

In this way you can define functions which do not exist as built-in functions, but which you may need to use frequently.

## When and why do you use a UDF?

You use a UDF for the same reason as a built-in function, namely to make calculations (operations) which are repeated more efficiently.

Before writing a UDF make sure that it does not already exist as built-in Excel function!

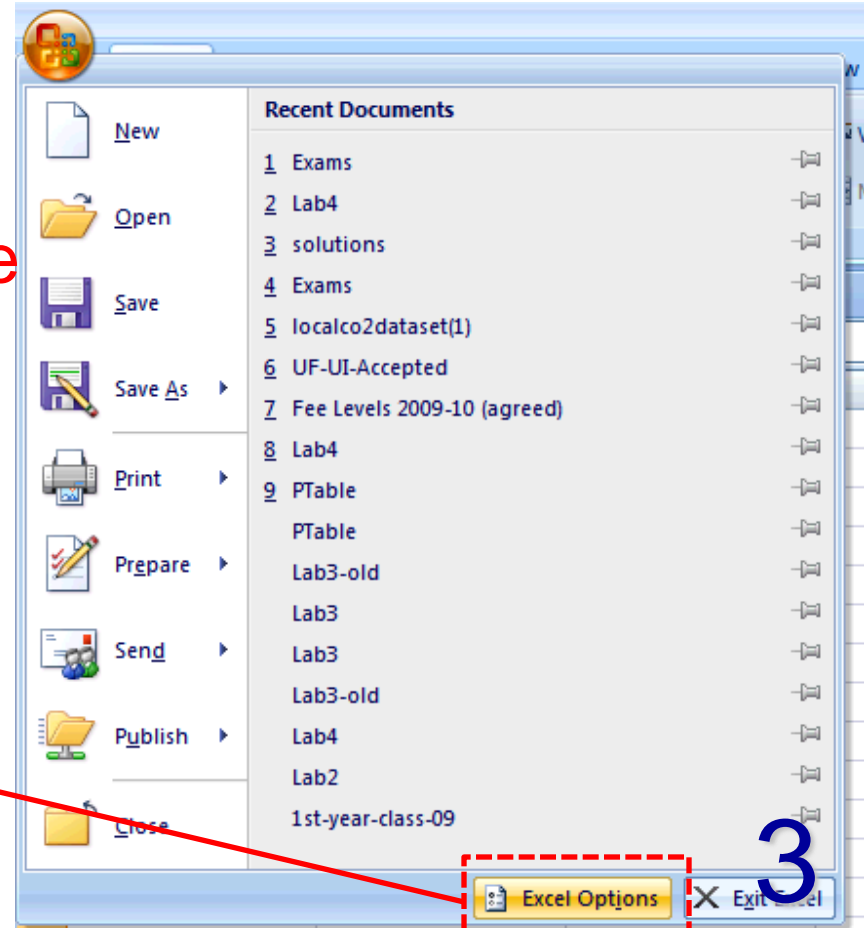
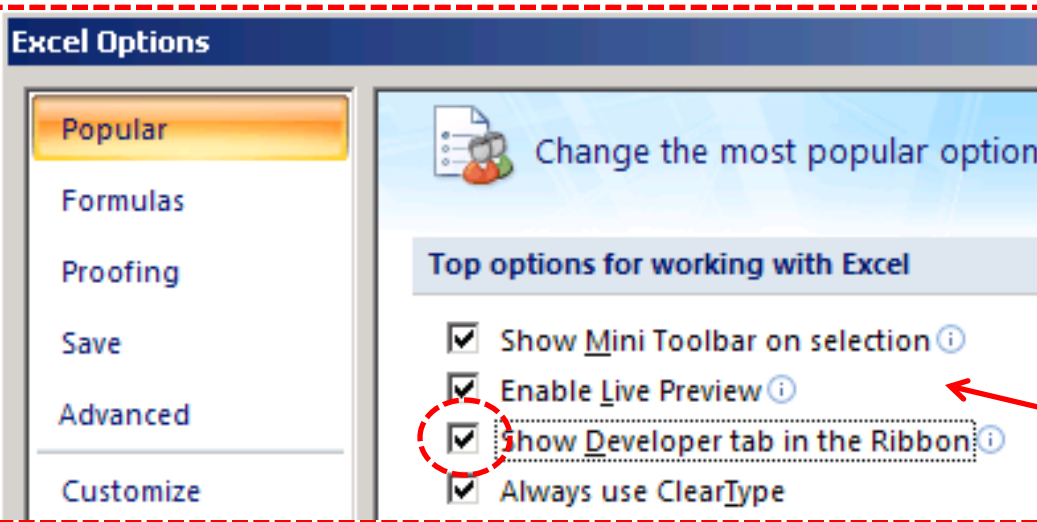
Before we start writing any UDFs we need to learn how to access the **Visual Basic Editor** (VBE).

The Visual Basic Editor is not directly visible from the Excel Worksheet. Microsoft hides the VBA tools by default !

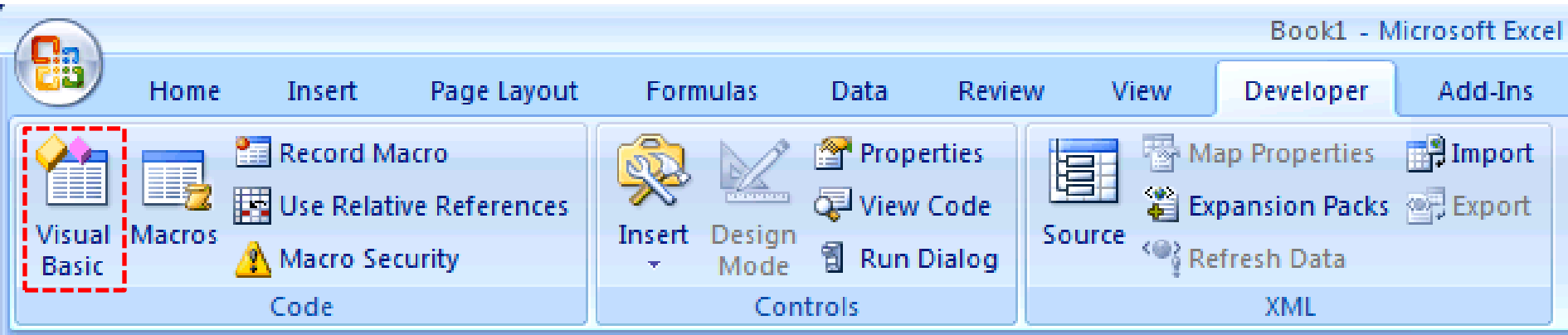
This can be changed by selecting the Office button 

Then selecting **“Excel Options”**:

In the **“popular options”** category, select **“Show Developer Tab in the Ribbon”** and click OK.

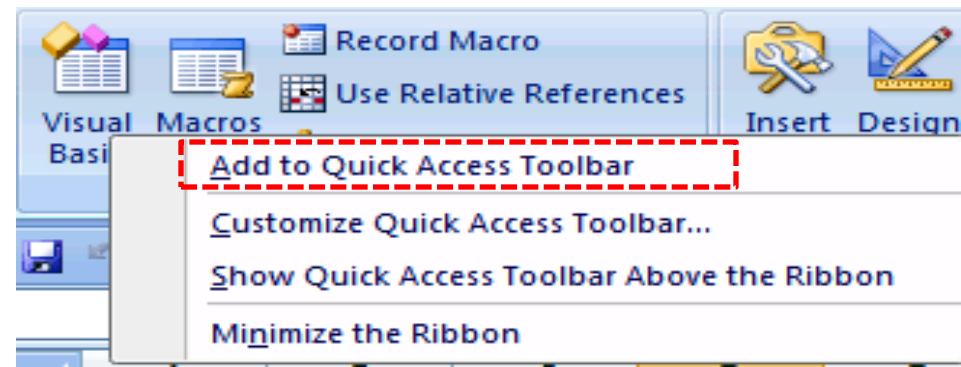


Once you have done this, an extra Tab is going to appear on your Excel window. The Tab is called “**Developer**” and it contains several menus



It includes the “Visual Basic” button that takes you directly to the VBA editor.

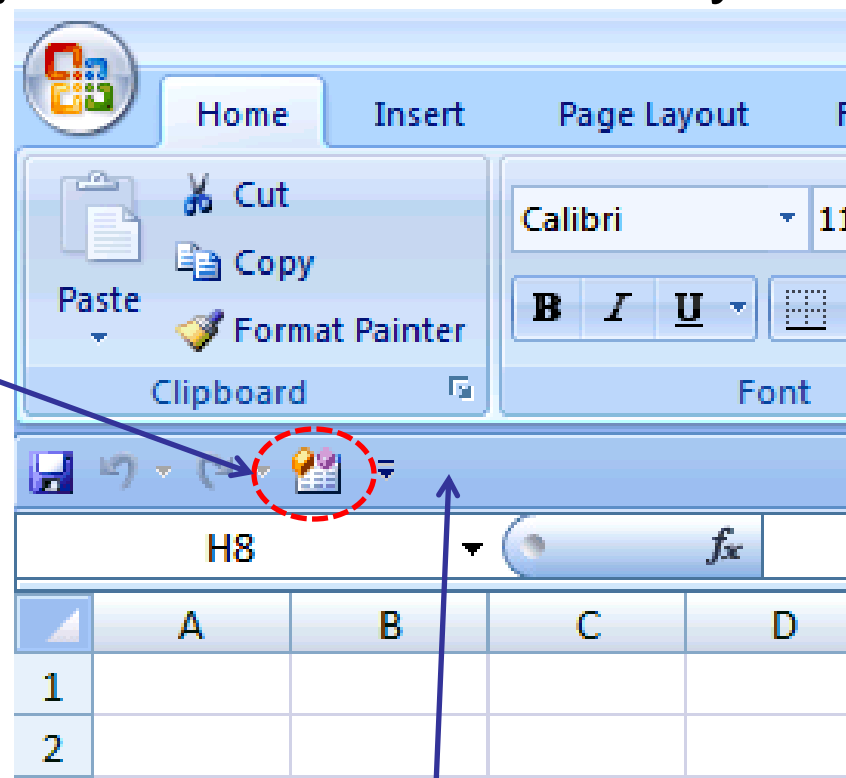
In the future you can access VBA even more quickly if you now right click on the Visual Basic button and select “**add to quick access toolbar**”



When you do this a new button is added to the “quick access toolbar” which you can just click on whenever you want to access the VBA editor:

Visual Basic Editor

Finally, there is one last way of accessing the VBA editor: by using the keyboard shortcut: **Alt + F11**



Quick access toolbar

The anatomy of the VBA editor is like most other applications. It is equipped with a menu and a toolbar at the top of the window and has four different subwindows:

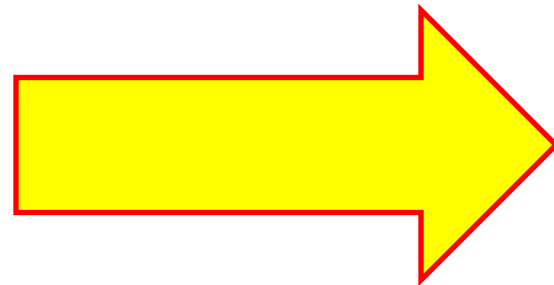
The **Project Explorer** displays the hierarchical structure of projects.

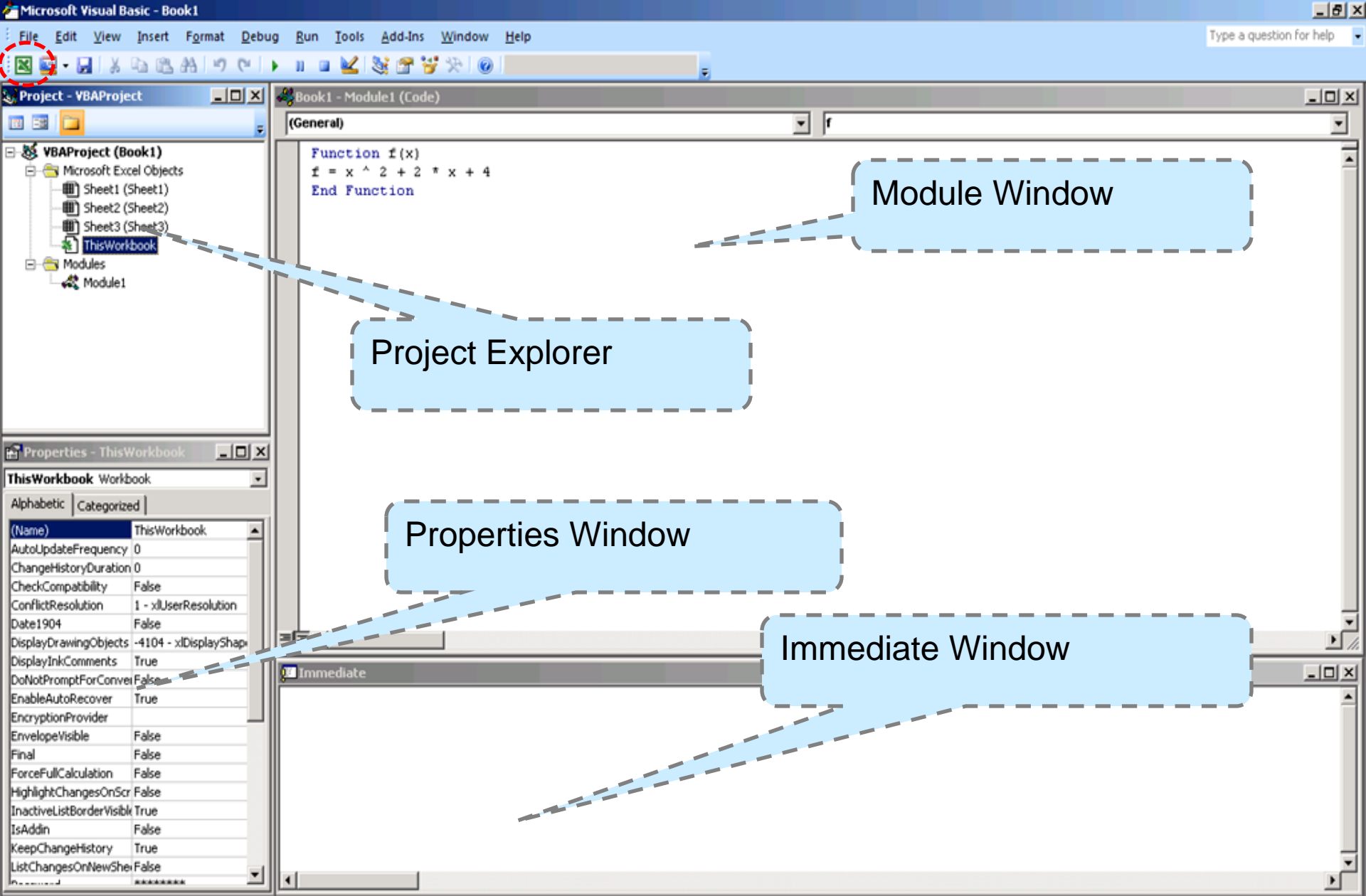
The **Properties Window** displays the properties of the projects.


The **Module Window** contains the VBA-code of your project. This is the most important window and you will write your UDF codes here.

The **Immediate Window** displays compiling messages (for example, error messages).

And this is how it looks...





The Excel Icon  on the top left corner brings you back to the Excel WS!

The Module Window might not be visible when you open VBE. VBE menu bar: **Insert → Module**

The Immediate Window is made visible by VBE menu bar: **View → Immediate Window**

You can also return to the Excel window by: **Alt + F11**

## The user defined function's syntax:

**Function** name [(arguments) [**As** type] ] [**As** type]

[statements]

name = expression

[**Exit Function**]

[statements]

[name = expression]

**End Function**



**Function** name [(arguments) [**As** type] ] [**As** type]

[statements]

name = expression

[**Exit Function**]

[statements]

[name = expression]

**End Function**

Function's input

Variable type of function's output

Variable type of function's input

Valid VBA commands

Function's name

an arithmetic expression assigned to the function name, which will be returned

- Everything in **bold** has to be typed exactly as above.
- Everything in squared brackets [...] is optional.

- Each statement has to begin in a new line.
- In case the statement is longer than the line you can split it by typing “\_” (i.e. space and underscore). You can not split VBA commands this way!
- A program (function) is read from top to bottom, that is each line is executed before the next (this is called a **sequential structure**). This order can be change by **using branching and loop structures** which we will study in part II.
- When **End Function** or **Exit Function** is reached the calculation terminates and the value last assigned to the function s name is returned.
- An assignment is done by an equation, which has to be read from the right to the left, i.e. the value on the right hand side of the equation is assigned to the name on the left hand side.

- Examples:

a) Function F(x)

$$F = 2 * x + 5$$

End Function

- You can now use this function on an Excel WS in the same way as you use a built-in function, e.g. “=F(5)” → 15

b) Function FF(x)

$$h = 2 * x$$

$$FF = h + 5$$

End Function

- The variable **h** only exists temporarily inside the function FF.

- Note: F(x) is the same function as FF(x)

c) Function G(x,y,z)

$$G = y * x + z$$

End Function

- As for built-in functions you can have more than one input variable (argument).

“=G(1,2,1)” → 3

d) Function Q(a,b,c,x)

' quadratic equation

$$Q = a * x^2 + b * x + c$$

End Function

- You can add **comments** to enhance the readability. VBA does not execute text following a single quote.

“=Q(2,3,10,2)” → 24

e) Function S(x, y, z)

S = 2 \* Application.WorksheetFunction.SUM(x, y, z)

End Function

- You can use Excel built-in functions inside UDF by  
Application.WorksheetFunction.FunctionName, e.g.

FunctionName = SUM. “=S(1,2,3)” → 12

f) Function sq(x)

sq = 2\*Sqr(x)

End Function

Sqr is the squared root function in VBA

Called SIGN in Excel

Called ATAN in Excel

Most Excel built-in functions are also built-in in VBA, but may have slightly different names: for example, the excel function SQRT is called Sqr in VBA!

Other VBA functions are: Abs, Atn, Cos, Exp, Fix, Int, Sgn, Log, Mod, Rnd, Sin, Tan (For a list with explanations use the help function and search for “Math Functions“.

## Comments on the **names** of UDF

- The first character in the name has to be a letter.
- The names are not case sensitive.
- Names are not allowed to contain spaces, @, \$, #,... Or be identical to VBA commands.

## A few comments on **debugging**


Inevitably you will make some mistakes either just typos or structural ones and you need some strategy to eliminate them.

- Some mistakes block the entire WS, e.g. suppose you type:

```
Function Err(x)
```

```
    Err = 2 * Sqr    (Brackets missing in Sqr, should be Sqr(x))
```

```
End Function
```

- If you call this function on the WS you will get an error message → LC on OK → the mistake will be highlighted in the VBE → Correct and unlock with “Reset” ≡ 

## Declaration of the variable type

- Recall: **Function** name [(arguments) [**As** type] ] [**As** type]
- The first type refers to the **variable type of the arguments** and the second type to the **variable type of the function**.

Declaring the variable type means that you can tell your VBA code whether the input or output of your function is going to be a **number**, **text**, a **date**, a **logical value** etc.

Your functions will also work without declaring the variable type (we just saw many examples of this) so why is it useful to declare the type?

Declaring the type avoids that different types of data get mixed up. You can trace systematically mistakes in long programs and it saves the computer memory space (your programmes will run faster!)

When you do not declare the type, then VBA assumes that your variables of the **“variant”** type by default.

In VBA there are seven different types of variables: Integer, Single, Double, String, Boolean, Variant and Date

The **Integer**, **Single** and **Double** types always apply to numbers. Integer characterizes variables that take **integer** number values. Single and Double are used for variables that take **real number values**. I will explain the difference between these two types next week!

Example: If we take one of the functions we defined before, we can rewrite it as

```
Function F(x As Single) As Single
```

$$F = 2 * x + 5$$

```
End Function
```

The **String** type is used for variables that take only text values.

Example: The function below takes a piece of text and returns the same text in upper case

```
Function U(x As String) As String
```

```
    U= Application.WorksheetFunction.Upper(x)
```

```
End Function
```

The **Boolean** type is used for variables that take only the logical values **TRUE** or **FALSE**.

The **Date** type is used for variables that take only **date or time values**.

The **Variant** type is used for variables that take values of different kinds, for example a variable that is sometimes a number and sometimes a string.