COMP1022Q
Introduction to Computing with Excel VBA

Functions and Subroutines in VBA

David Rossiter, Oz Lam and Gibson Lam

This Presentation

- This presentation looks at VBA *functions* and *subroutines*
- Both are used to group together VBA code so that the code can be used any number of times
  - For example, a macro is a subroutine that can be used many times
- A VBA function runs some code and returns a value after finishing the code, e.g. `InStr()`
- A VBA subroutine runs some code and does not return anything, e.g. `MsgBox()`

VBA Functions

- In VBA, there are many functions you can use
- For example, you have seen `InputBox()` and `InStr()` before
- Usually, we pass some *parameters* to a function and the function returns a result
- For example,

```
position = InStr(1, "Hello! I am Dave!", "Dave")
```

- This function uses 3 input parameters and returns a number (in this case, it returns 13)

Making Our Own Function

- We can make our own function to do whatever we like
- To define a new function, we do this:

Put your function name here

```
Function FuncName ( name(s) of input parameters )

      . . .the main code of the function goes here . . .

End Function
```

- After defining the function, we can use it, i.e:

```
    Result = FuncName(1, 2, 3)
```

Example of Making Our Own Function

- In VBA, a function always returns something
- Here is an example:

```
Function SquareSize(SideLength)

    SquareSize = SideLength * SideLength

End Function
```

Return a value by putting the value in a variable that has the same name as the function

- This function takes the length of the side of a square and returns the size (i.e. area) of the square
- For example, `SquareSize(10.6)` returns 112.36

Specifying the Input Type

- If you want to, you can specify the type of the inputs, like this:

You need this word when you want to define your input type (it may cause automatic conversions)

```
Function SquareSize(ByVal SideLength As Integer)

    SquareSize = SideLength * SideLength

End Function
```

- The input is automatically converted to an integer
- For example, with the above function `SquareSize(10.6)` returns 121
  - It doesn't return 112.36, because 10.6 is first converted to 11 and stored in `SideLength` inside the function

Specifying the Return Type

- You can also specify the returning result type, like this:

```
Function SquareSize(SideLength) As Integer

     SquareSize = SideLength * SideLength

End Function
```

- The result is automatically converted to an integer before it is returned
- For example, using the above function `SquareSize(10.6)` returns 112, not 112.36

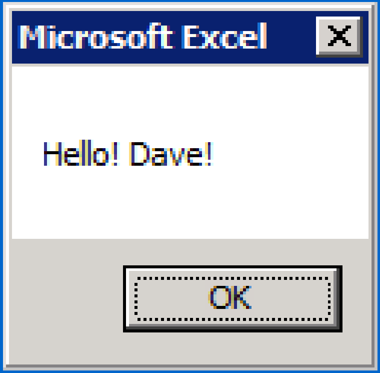
VBA Subroutines

- A *subroutine* is very similar to a function but it does not return anything, i.e. `MsgBox()`
- You create subroutine using `Sub`, like this:

```
Sub SayHello(Name)
    MsgBox "Hello! " & Name & "!"
End Sub
```

- Here is an example of using the above subroutine:

```
SayHello "Dave"
```



Hello! Dave!

Using Functions/Subroutines

- You may have noticed that when you use functions and subroutines sometimes you need to use brackets and sometimes you don't
- Here are three (separate) examples:

```
MsgBox "It's a sunny day!"

Current_time = Now

Age = InputBox("What is your age?")
```

`Now` *returns the current time*

When You Have to Use Brackets

- You need to use brackets, i.e. `()`, when you run a function that has one or more input parameters **and** you are going to use the result returned by the function
- For example:

```
Result = InStr( 1 , "the fat cat sat" , "at" )
```

  - Use `()` to enclose the parameters
  - Parameters are separated using commas
  - In this example, we put the result in a variable

When You Don’t Have to Use Brackets

- You don’t need brackets, i.e. `()`, if there is no parameters
- For example, both of these work:

```
myFirstRandomNumber = Rnd()

mySecondRandomNumber = Rnd
```

`Rnd` *returns a random number in the range 0 to 0.99999*

- A subroutine does not return anything, which means you never need to use brackets for a subroutine
- For example, you don’t need `()` when you call `MsgBox`:

```
MsgBox "It's a great day!"
```

Private/Public Functions and Subroutines

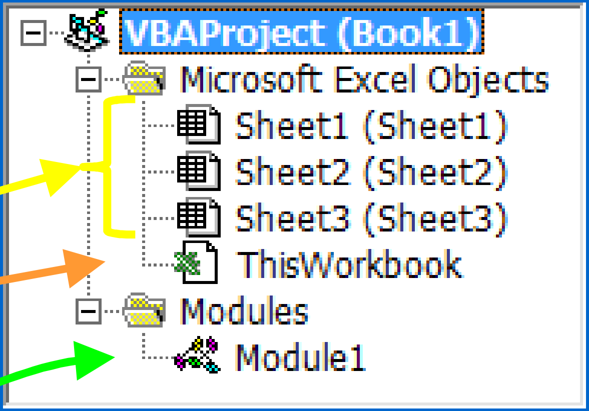
- When you create an event handler (which is a VBA subroutine) in a worksheet or a workbook you will see the word `Private` used before the subroutine, e.g.:

```
Private Sub Worksheet_Open()
        . . .
End Sub
```

- You can put `Private` or `Public` before the definition of your function/subroutine
- They control the way that other code can access your function/subroutine within the Excel file

Places to Create Functions and Subroutines

- There are many places where you can define functions/subroutines in VBA
- For example, you can define functions/subroutines in one of the worksheets, the workbook or a module in the Excel file



- If you define a **private** function/subroutine in one of these places, you can use them **only in the place where you have created it**
- If you define a **public** function/subroutine in one of these places, you can use them **from anywhere**

Private or Public?

- The idea of using private and public functions/ subroutines is an advanced topic in computer programming
- When you create your own functions/subroutines you can simply ignore the use of `Private` and `Public`, i.e.:

`Private` *or* `Public` *are not used here*

```
Function MyFunc(...)
        ...
End Function
```

- When `Private` or `Public` are not specified VBA assumes you are creating a public function/subroutine