COMP1022Q
Introduction to Computing with Excel VBA

More on Variables

David Rossiter and Gibson Lam

Outcomes

- After completing this presentation, you are expected to be able to:
  1. Create and use global variables in VBA
  2. Explain the difference between local and global variables
  3. Ask VBA to stop automatically creating variables

A Quick Reminder of Some Variables We've Seen

```
Dim Pi As Single
```
– storing a number with a decimal place, fair accuracy

```
Dim Weight As Double
```
– storing a number with a decimal place, great accuracy

```
Dim Total As Integer
```
– storing an integer up to ~32xxx

```
Dim Money As Long
```
– storing an integer up to ~2bn

```
Dim Name As String
```
– storing text

```
Dim Comparison As Boolean
```
– storing True/False

- These examples are taken from the notes we've looked at before

Creating Multiple Variables

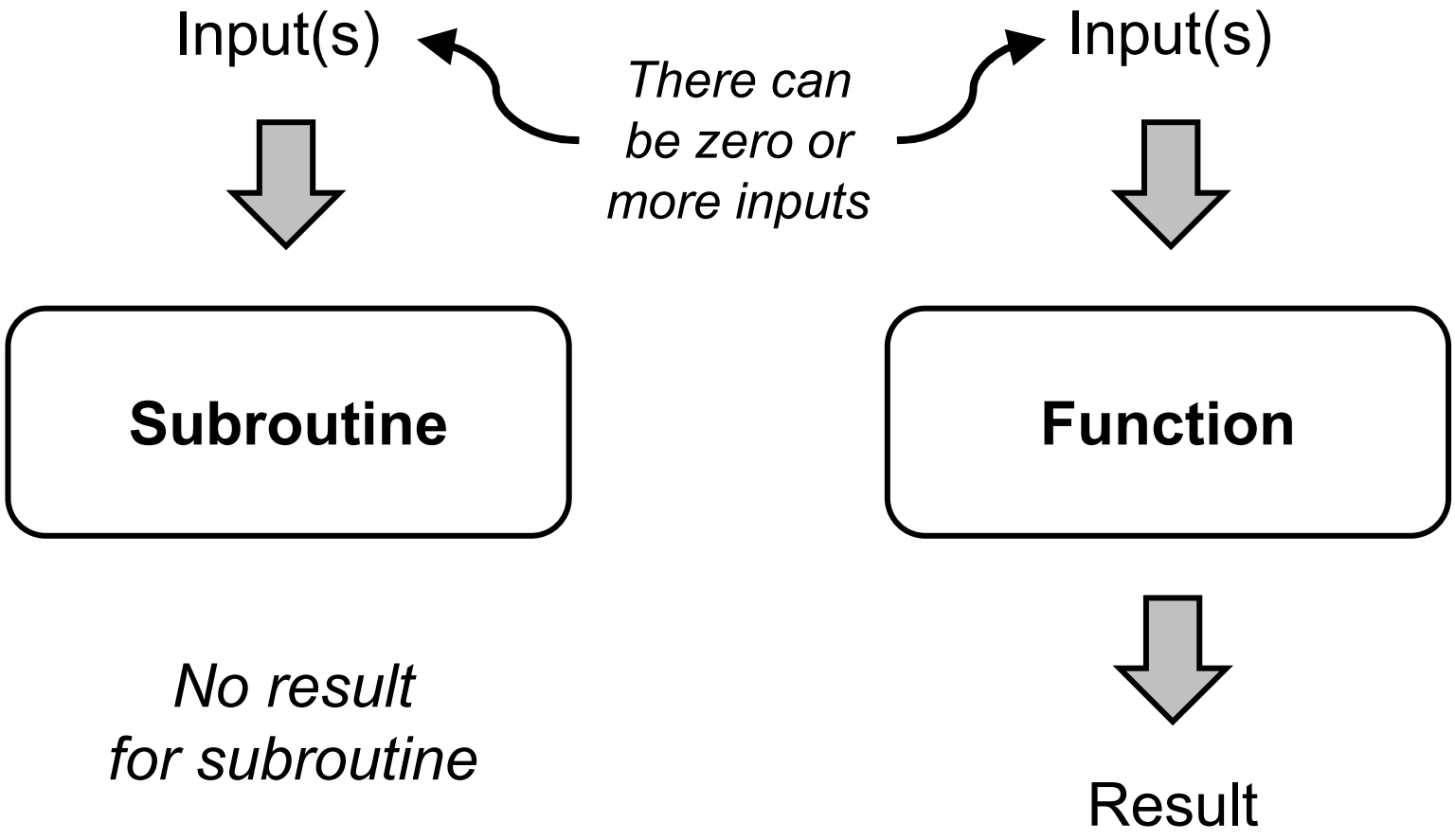
- You can create several variables one by one, like this:

```
Dim Target As Long
Dim OneYear As Long
Dim Total As Long
```

or you can do the same thing in one line, like this:

```
Dim Target As Long, OneYear As Long, Total As Long
```

Reminder – VBA Subroutines and Functions



An Example

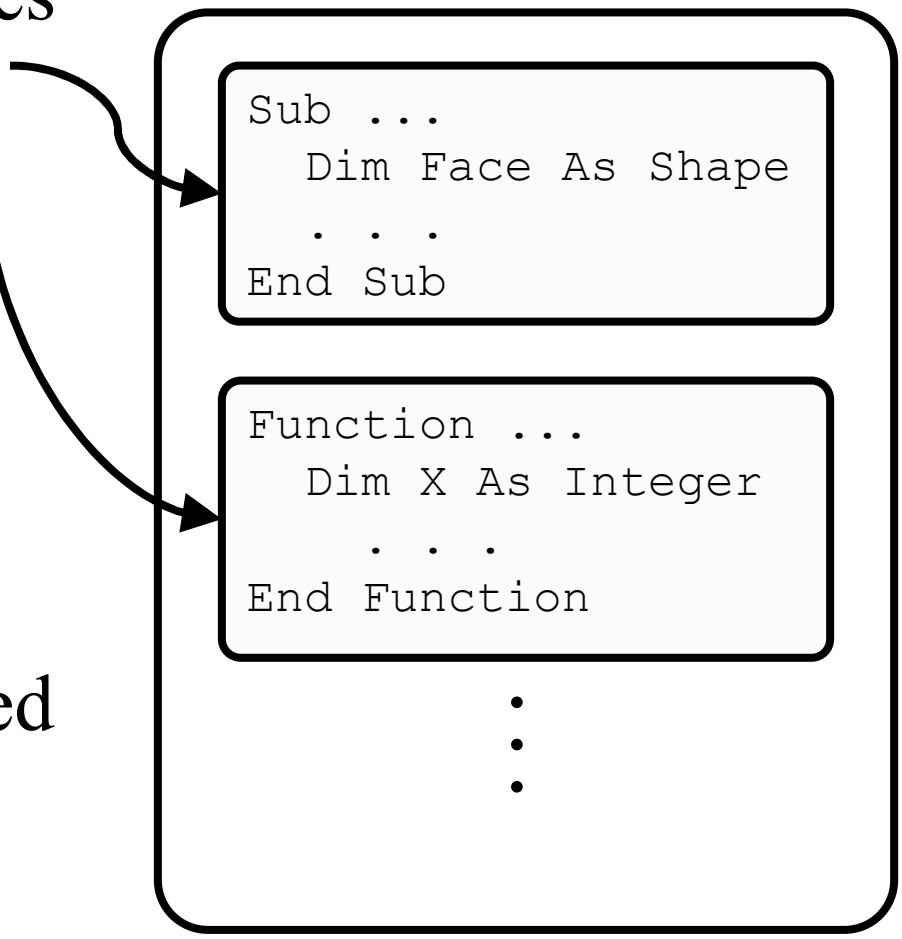
- Here is an example we will use in the next few slides

```
Function CalculateAmount(Quantity, Price)
    Dim SmallestQuantity As Integer
    SmallestQuantity = 100
    If Quantity < SmallestQuantity Then
        Quantity = SmallestQuantity
    End If
    CalculateAmount = Quantity * Price
End Function

Sub BuyAndSell()
  Dim Cost As Single, Income As Single
  MsgBox "I am buying some shares!"
  Cost = CalculateAmount(500, 1)
  MsgBox "I am selling the shares!"
  Income = CalculateAmount(500, 1.1)
  MsgBox "Profit:" & (Income - Cost)
End Sub
```

Local Variables

- *Local variables* are variables created inside a subroutine or inside a function

- You can only use a local variable inside the subroutine/ function where the variable has been created



```
Function CalculateAmount(Quantity, Price)
    Dim SmallestQuantity As Integer
    SmallestQuantity = 100
    If Quantity < SmallestQuantity Then
        Quantity = SmallestQuantity
    End If
    CalculateAmount = Quantity * Price
End Function

Sub BuyAndSell()
  Dim Cost As Single, Income As Single
  MsgBox "I am buying some shares!"
  Cost = CalculateAmount(500, 1)
  MsgBox "I am selling the shares!"
  Income = CalculateAmount(500, 1.1)
  MsgBox "Profit:" & (Income - Cost)
End Sub
```

SmallestQuantity is a local variable

You can use SmallestQuantity in this area

You cannot use SmallestQuantity in this area

```
Function CalculateAmount(Quantity, Price)
    Dim SmallestQuantity As Integer
    SmallestQuantity = 100
    If Quantity < SmallestQuantity Then
        Quantity = SmallestQuantity
    End If
    CalculateAmount = Quantity * Price
End Function

Sub BuyAndSell()
  Dim Cost As Single, Income As Single
  MsgBox "I am buying some shares!"
  Cost = CalculateAmount(500, 1)
  MsgBox "I am selling the shares!"
  Income = CalculateAmount(500, 1.1)
  MsgBox "Profit:" & Income - Cost
End Sub
```

Quantity is also a local variable

You can use Quantity in this area

You cannot use Quantity in this area

```
Function CalculateAmount(Quantity, Price)
    Dim SmallestQuantity As Integer
    SmallestQuantity = 100
    If Quantity < SmallestQuantity Then
        Quantity = SmallestQuantity
    End If
    CalculateAmount = Quantity * Price
End Function

Sub BuyAndSell()
  Dim Cost As Single, Income As Single
  MsgBox "I am buying some shares!"
  Cost = CalculateAmount(500, 1)
  MsgBox "I am selling the shares!"
  Income = CalculateAmount(500, 1.1)
  MsgBox "Profit:" & Income - Cost
End Sub
```

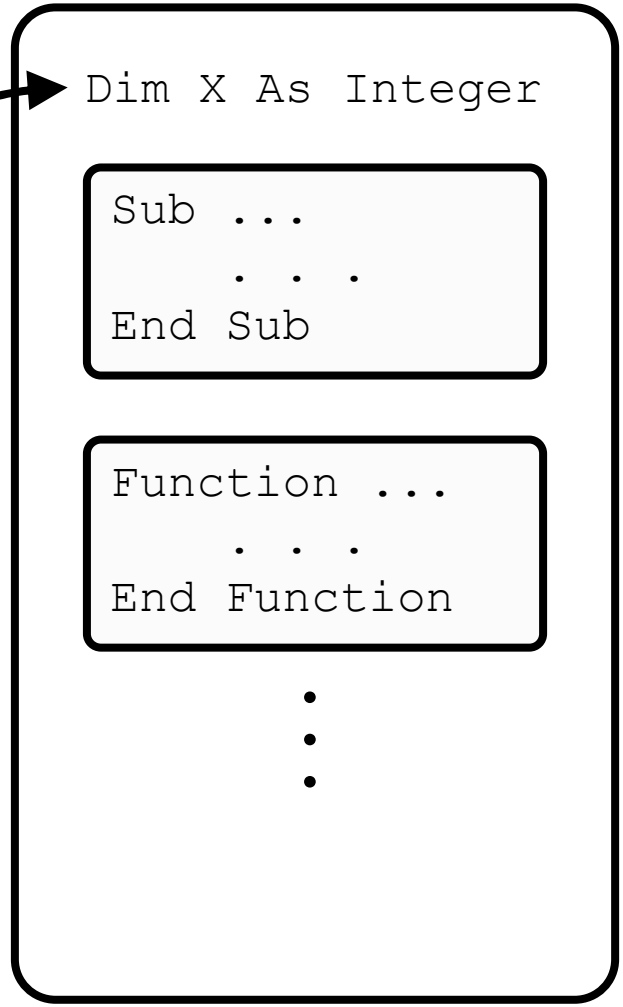
Price is also a local variable

You can use Price in this area

You cannot use Price in this area

Global Variables

- *Global variables* are variables which are created *outside* a subroutine/function
- You can use a global variable anywhere you like
- You can read and change the value of a global variable in a subroutine, or a function



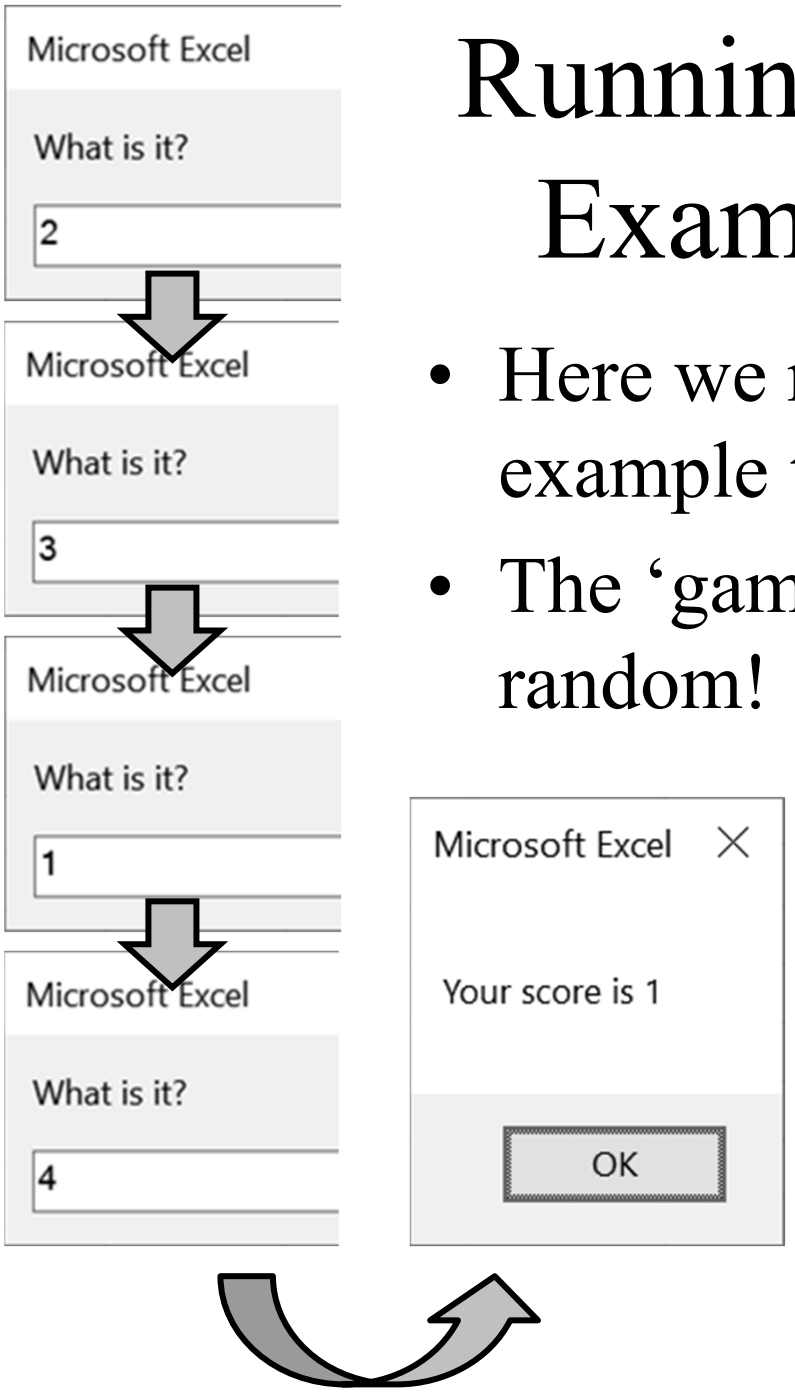
An Example

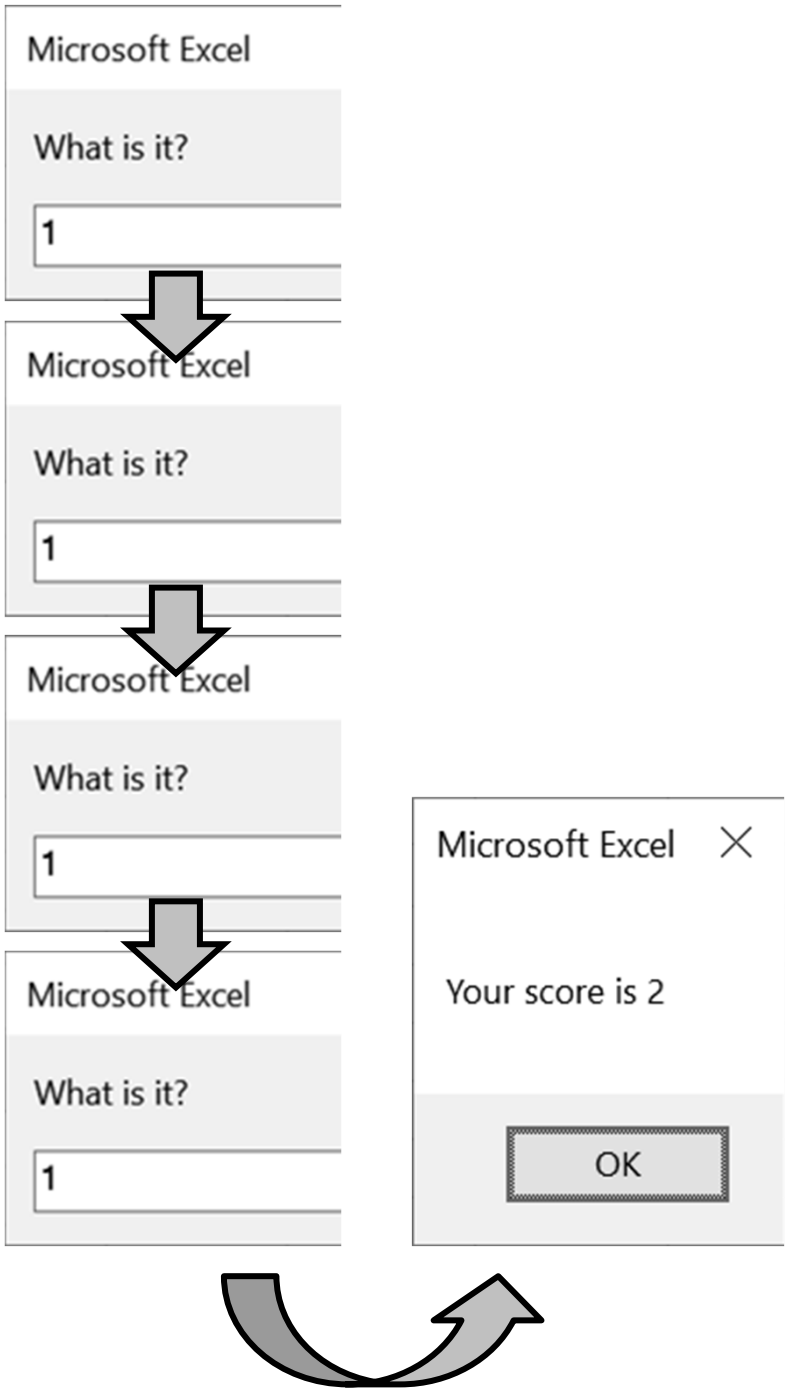
```
Dim Score As Integer

Sub ShowFinalResult()
    MsgBox "Your score is " & Score
End Sub

Sub PlayGame()
    Dim Answer As Integer
    Dim Guess As Integer
    Dim GameNumber As Integer

    Randomize

    Score = 0
    For GameNumber = 1 To 4
        Answer = Int(Rnd() * 4) + 1
        Guess = InputBox("What is it?")
        If Guess = Answer Then
            Score = Score + 1
        End If
    Next GameNumber

    ShowFinalResult
End Sub
```

- This example has a global variable called `Score`
- That means `Score` can be used anywhere
- Here it is used in 2 subroutines
- Use `PlayGame` to start the game

Running the Example

- Here we run the example twice
- The ‘game’ is random!





Using a Local Variable and a Global Variable with the Same Name

- You cannot use the same name for two different variables inside the same subroutine/function
- However, you *can* use the same name for a global variable and a local variable – although this is rather confusing!
- If you have a global variable and a local variable with the same name, inside the subroutine/function **the local variable is used** instead of the global variable
- Let's look at an example

An Example Using Local and Global Variables with the Same Name 1/3

- In this example, a global variable is called `Name` and a local variable is also called `Name`

```
Dim Name As String

Function GetName()
    GetName = InputBox("What is your name?")
End Function

Sub ShowGreeting()
    MsgBox "Hello, " & Name & "."
End Sub

Sub Welcome()
    Dim Name As String
    Name = GetName()
    ShowGreeting
End Sub
```

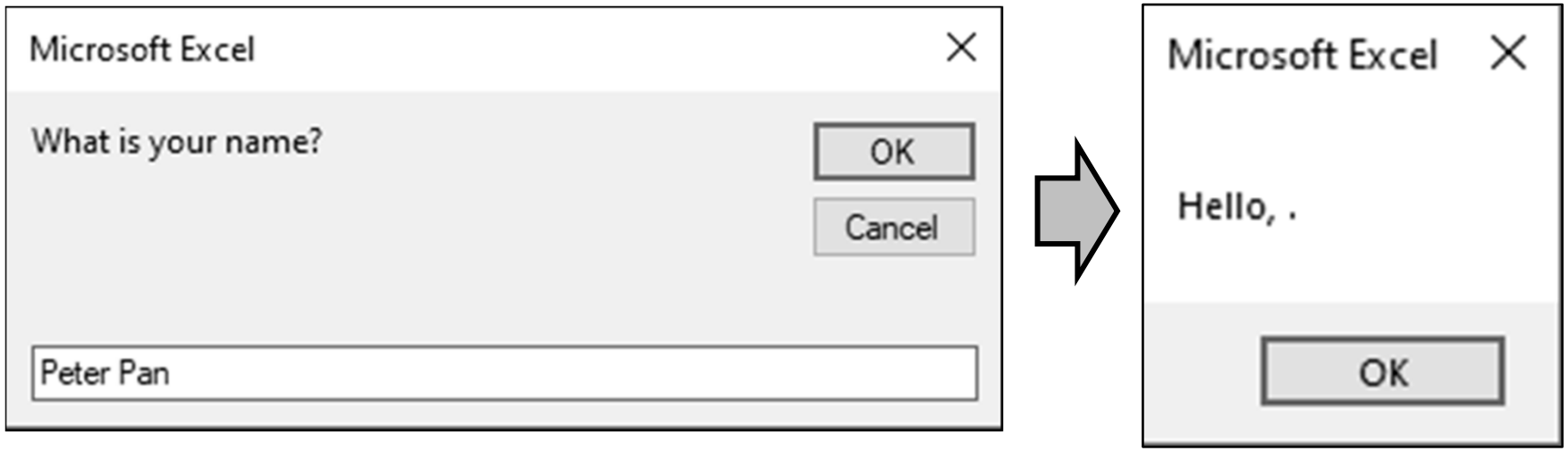
A global variable

A local variable

- This is confusing, please don't do this in your own programs!

An Example Using Local and Global Variables with the Same Name 2/3

- Here is an example of what happens when you run `Welcome()` and then enter 'Peter Pan':



An Example Using Local and Global Variables with the Same Name 3/3

- What happens after you enter 'Peter Pan'?
  - The text 'Peter Pan' is returned to `Welcome()` and then stored in the **local variable** `Name`
- What happens then when `ShowGreeting()` is run?
  - `ShowGreeting()` gets the value of the **global variable** `Name` and then displays a message using a message box

```
Sub Welcome()
    Dim Name As String
    Name = GetName()
    ShowGreeting
End Sub
```

Using the local variable

```
Sub ShowGreeting()
    MsgBox "Hello, " _
    & Name & "."
End Sub
```

Using the global variable

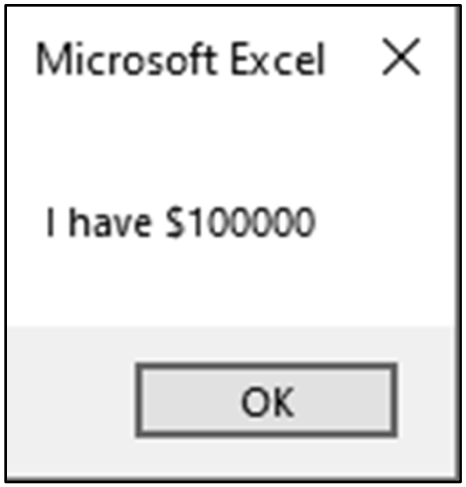
Variable Scope

- *Variable Scope* is a computer science term which basically means ‘where a variable works’
- The scope of a local variable is inside the function/subroutine where it was created
- The scope of a global variable is everywhere

VBA Can Automatically Make Variables

- Usually, when VBA is running some code and finds a variable name it hasn't seen before, **it will automatically create a new variable with that name**, and then carry on with the code
- For example, the variable `MyMoney` is automatically created in the following code:

```
MyMoney = 100000
MsgBox MyMoney
```



There's no Dim here; that means the variable MyMoney is automatically created here

Automatic Variable Creation – Good or Bad?

- Because VBA will make variables for you, you can be lazy and not create any variables before using them
  - i.e. you don't need to use `Dim` in the code at all!
- That sounds great – but sometimes this can lead to bugs, i.e. mistakes in the code, that are difficult to find
- In the following example we show a bug that means the money in a bank account is displayed wrongly

An Example of Automatic Variable Creation

- Let's look at the following code:

```
Sub ShowMoneyInBankAccount()
    MoenyInTheBank = 1000
    MsgBox "I have $" & MoneyInTheBank
End Sub
```

- Can you spot a problem in the code?
  - The variable `MoneyInTheBank` is not spelled correctly in the first line inside the subroutine

Running the Example

- When VBA sees `MoenyInTheBank` it automatically creates a variable with that name, and puts 1000 in it
- Later, the code sees `MoneyInTheBank`, and creates that also, but doesn't put anything in it

```
Sub ShowMoneyInBankAccount()
MoenyInTheBank = 1000
MsgBox "I have $" & _
       MoneyInTheBank
End Sub
```

- So when it shows the content of `MoneyInTheBank`, you can't see anything, because there's nothing in it

Using Option Explicit

- You can avoid this problem by adding the following line at the top of your code:

```
Option Explicit
```

- This line means *all variables must be declared before they are used*
- In other words, you are forcing the programmer (yourself!) to plan better

The Example with Optional Explicit

- Let's use `Option Explicit` on the previous example:

```
Option Explicit
```

The variable MoneyInTheBank must be created before we use it

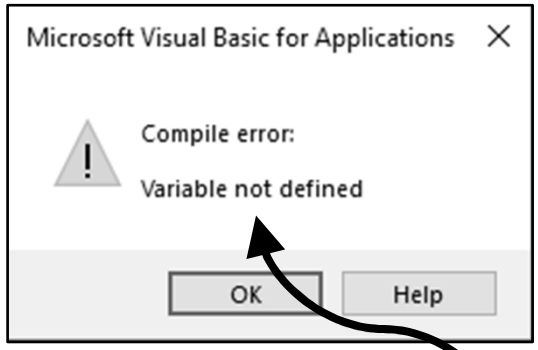
```
Sub ShowMoneyInBankAccount()
    Dim MoneyInTheBank As Integer

    MoenyInTheBank = 1000
    MsgBox "I have $" & MoneyInTheBank
End Sub
```

Spelling mistake

Running Code with Option Explicit

- Let's look at what happens when we run the example

```
' Force all variables to be declared before they are used
Option Explicit

Sub ShowMoneyInBankAccount()
    Dim MoneyInTheBank As Integer

    MoenyInTheBank = 1000
    MsgBox "I have $" & MoneyInTheBank
End Sub
```



Excel saw this and thought it was a variable

Because the variable hasn't been declared, Excel stops running the code and shows an error message

After Fixing the Code …

- After adjusting the code, you run the program again:

```
Option Explicit

Sub ShowMoneyInBankAccount()
    Dim MoneyInTheBank As Integer

    MoneyInTheBank = 1000
    MsgBox "I have $" & _
           MoneyInTheBank
End Sub
```

The problem has been fixed

This time, the program code is right, and the result is correct