

# Lecture 12

## *Learning outcomes:*

- *Excel Visual Basic for Applications (VBA) programming*

# Introduction

- Microsoft Excel is a powerful tool that most of us have used to manage data, perform calculations, and create charts.
- But did you know that you can take Excel to the next level by utilizing Visual Basic for Applications (VBA)? VBA is a programming language integrated into Excel that allows you to automate tasks, build custom functions, and create interactive user interfaces.
- This blog, we'll introduce you to Excel VBA programming for beginners and demonstrate how it can enhance your productivity and efficiency.

# Why VBA?

- Visual Basic for Applications is a programming language developed by Microsoft, primarily used within their Office suite of applications such as Excel, Word, and Access. VBA empowers users to automate tasks, enhance productivity, and extend the functionality of these applications. Whether you are a seasoned developer or someone new to the world of coding, VBA offers an accessible and rewarding path to achieving your goals.
- VBA stands for Visual Basic for Applications. It's a programming language that enables you to control just about everything in Excel. You'll learn how to create Macros that can be run from things like a button on a spreadsheet, the Excel Ribbon - in fact, lots of places. Learning Excel VBA will enable you to do a lot more with the software than you can via the normal spreadsheet view.

# Understanding Visual Basic for Applications (VBA)

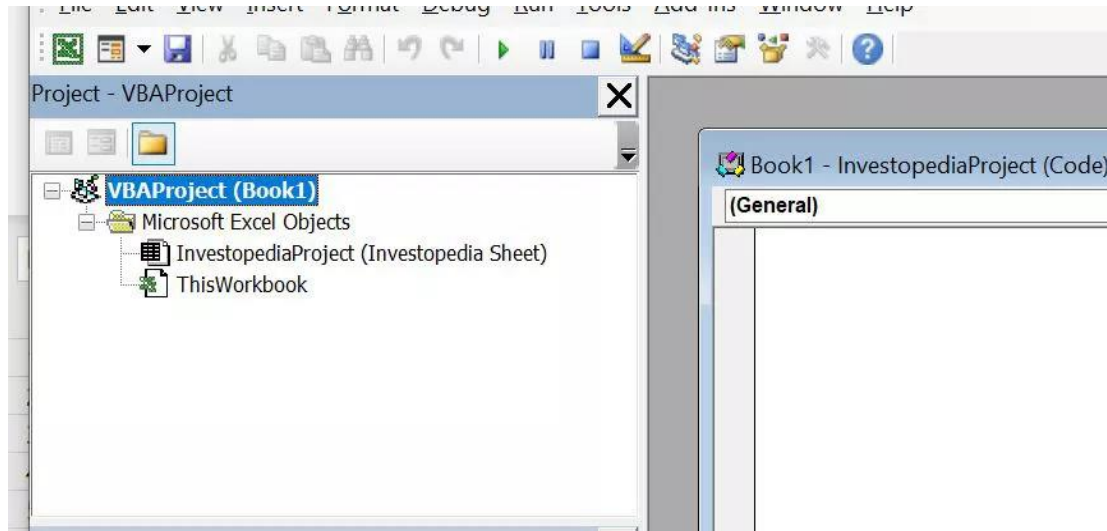
- VBA is an event-driven tool. You can use it to tell the computer to initiate an action or string of actions by typing commands into an editing module to build custom macroinstructions (macros).
- A macro is essentially a sequence of characters that inputs results in another sequence of characters (its output). This accomplishes specific computing tasks. You don't have to purchase the VBA software because VBA is the version of Visual Basic that ships with Microsoft Office.

# VBA in Excel

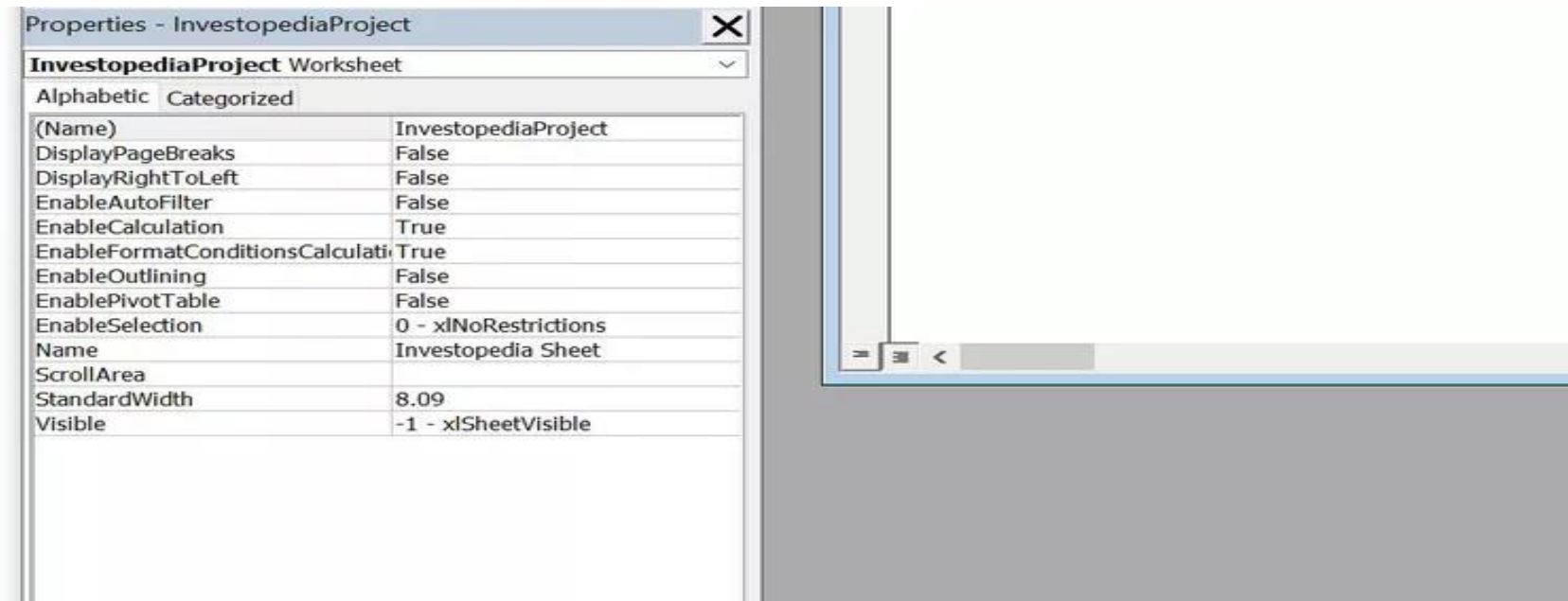
- All Office suite programs share common programming languages and each is capable of integrating VBA code to enhance the program. VBA has been a natural fit with Excel more so than with other Office suite programs because of the repetitive nature of spreadsheets, data analytics, and organizing data.
- The root of the relationship between VBA and Excel is often tied to the use of macros. You can use VBA to run a macro in Excel but you can use it for non-macro activities as well.

# How to Access VBA in Excel

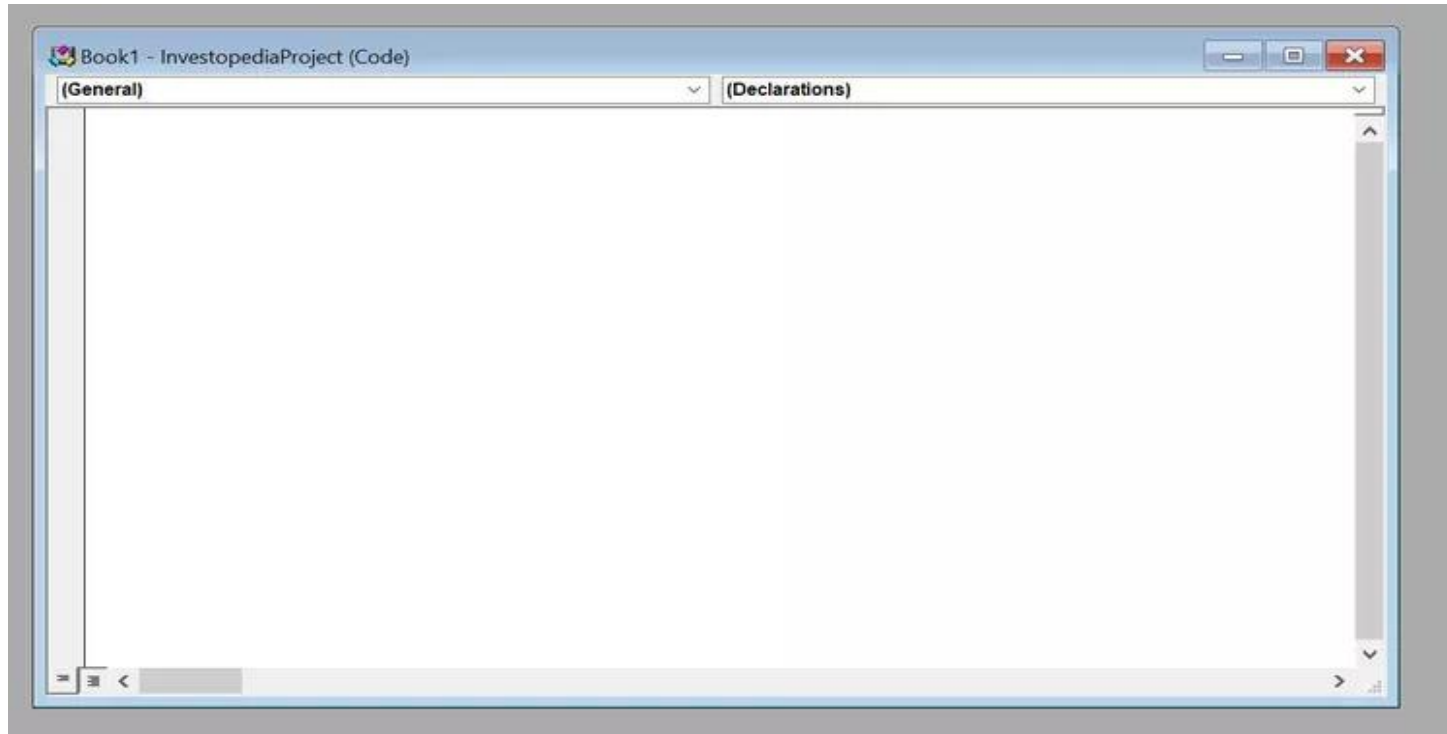
- Simply press Alt + F11 to access VBA in Excel.4 Your existing Excel workbook will remain running but a new window will appear for Microsoft Visual Basic for Applications. The top left of the VBA window will show the current projects. The InvestopediaProject file is ready to receive VBA code in this example:



- The window displays the properties of the selected project at the bottom left. Properties are listed as projects or workbooks are selected. These properties are listed alphabetically by default although they can be sorted by category.

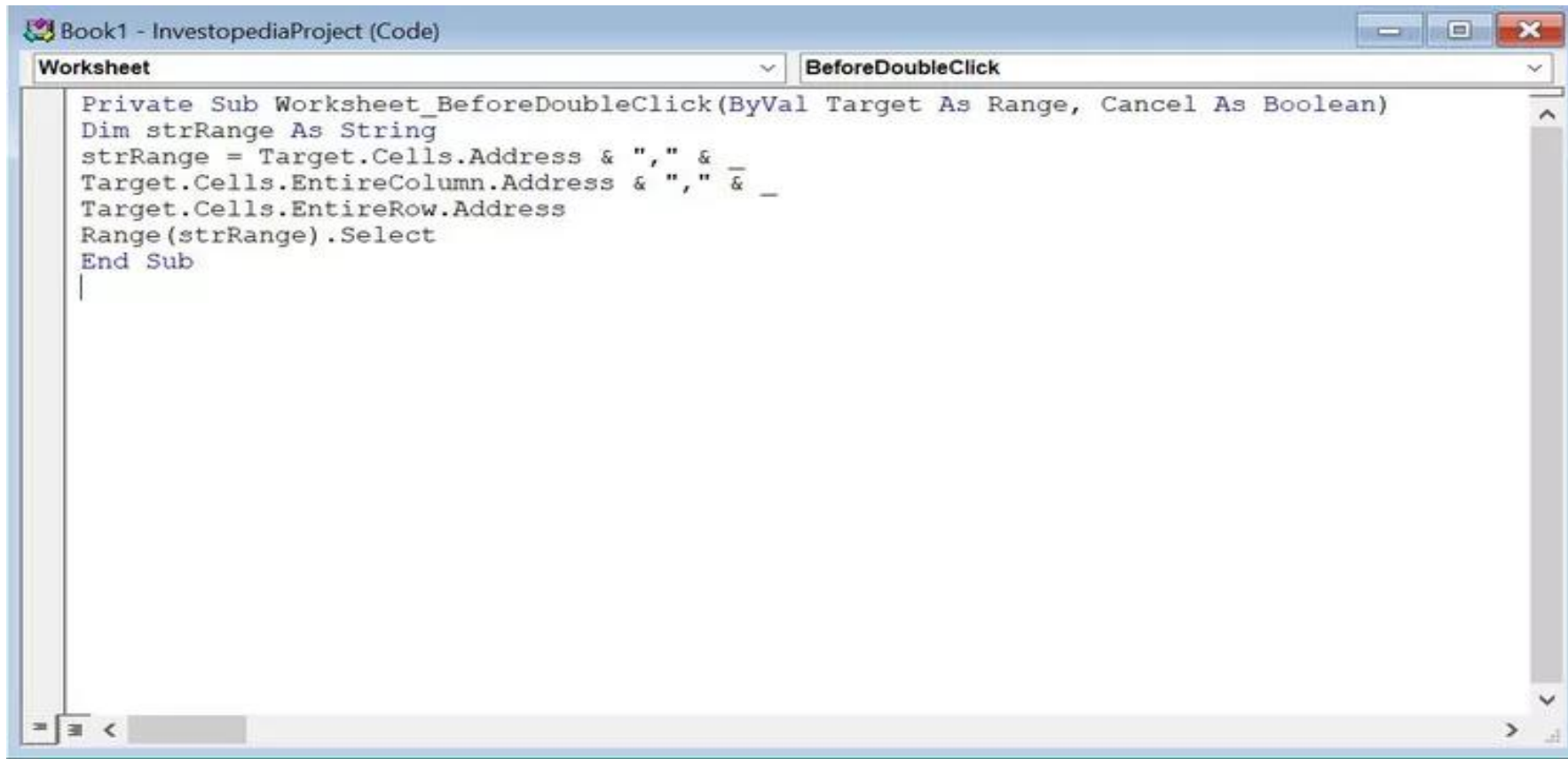


- A new window appears when you double-click on a project on the top left. There's no information in this area but you'll see two dropdowns that say "(General)" and "(Declarations)." VBA code is directly entered into this coding window.



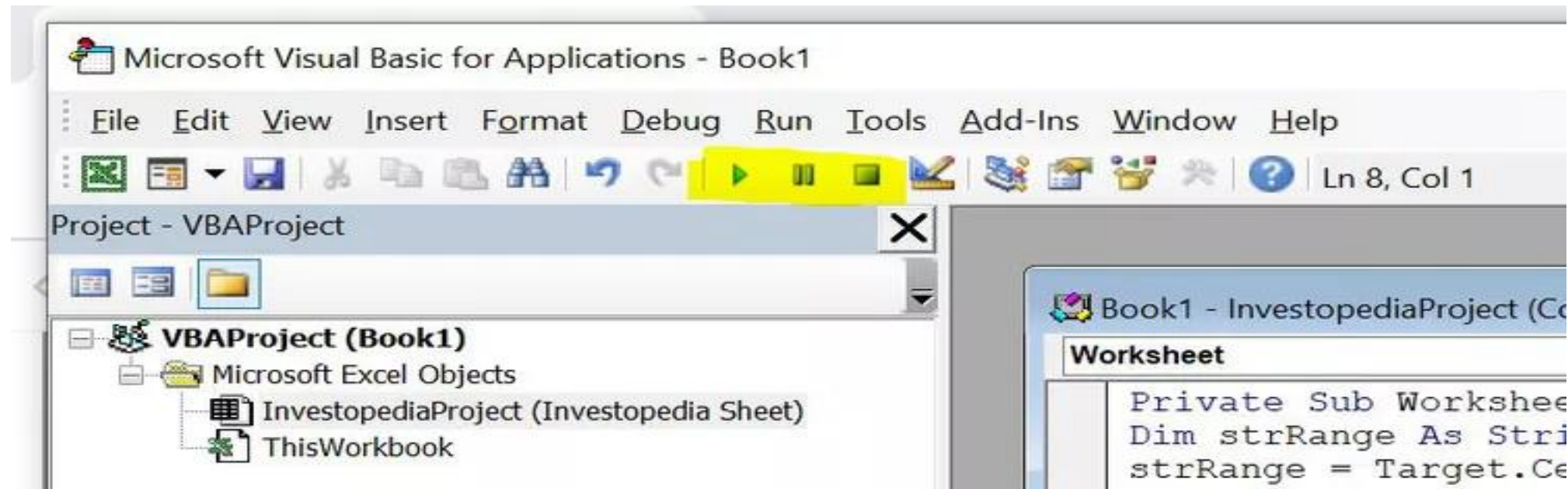


- Here's an example of a VBA code that's been entered:



```
Book1 - InvestopediaProject (Code)
Worksheet BeforeDoubleClick
Private Sub Worksheet_BeforeDoubleClick(ByVal Target As Range, Cancel As Boolean)
    Dim strRange As String
    strRange = Target.Cells.Address & ", " & _
    Target.Cells.EntireColumn.Address & ", " & _
    Target.Cells.EntireRow.Address
    Range(strRange).Select
End Sub
```

- Many important buttons and tools appear on the toolbar. The items highlighted in yellow are the run, break, and reset toggles for the VBA code. The run button executes the code. The break button pauses the activity of the code. The reset stops the execution of the code and brings the process back to the starting position of the code.



# Important VBA Terms

## □ Module

A module is where Excel stores the VBA code. Information regarding the modules within a spreadsheet can be found in Project Explorer, one of the sections of the Visual Basic Editor. All modules can be saved within a modules folder. Modules are sometimes referred to as standard modules.

## □ Objects

Most code is used to manipulate objects in VBA. Objects are items such as workbooks, worksheets, cells, cell ranges, or cell fonts. Objects are often selected or referred to as part of the code when you're coding in VBA. The code can use the "ActiveCell" language to manipulate the object currently selected in the spreadsheet. You can also create a process that executes when a bar chart is edited.

## □ Procedures

The procedure is the part of a computer program that performs a specific task. It's the block of code that starts with a declaration and finishes with an end declaration. There are two types of procedures in VBA. Sub procedures form an action in Excel and begin with the text "Sub." Function procedures carry out calculations and return a value.

## □ Statement

A statement is an instruction that can be broken into two types. First, a declaration statement is used to state something such as defining a constant or a variable value. Second, executable statements designate code that specifies what a certain action is.

## □ Variables

Variables are storage locations for defined items. They hold specific values that may change as VBA scripts are performed. The variable "FirstName" may not contain any value, but it can be assigned the value "Jo" after the user inputs their name. Variables in coding can be different depending on the situation, similar to how variable costs can change over time.

## □ Logical Operators

Logical operators are the functions that allow for greater analytical and processing capabilities. They're bits of code that allow a computer to understand and compare items. VBA can analyze whether the user's name is "Jo." The program can analyze the input and perform a logical evaluation using logical operators such as 'if, then', 'true', and 'false.'

## VBA code

- VBA code consists of procedures (subroutines or functions) that perform specific tasks. A procedure can be initiated by a user or executed automatically when certain events occur. Understanding the structure of a simple VBA subroutine:

```
Sub MyFirstMacro()  
' Your code goes here  
End Sub
```

# **Subroutine (Macro)**

```
Sub HelloWorld()
```

```
    MsgBox "Hello, World!"
```

```
End Sub
```

# Variables and Data Types:

```
Sub ExampleVariables()
```

```
    Dim num As Integer
```

```
    Dim text As String
```

```
    num = 42
```

```
    text = "Excel VBA"
```

```
    MsgBox text & " Number: " & num
```

```
End Sub
```



# Conditional Statements

```
Sub ExampleIf()
```

```
    Dim score As Integer
```

```
    score = 85
```

```
    If score >= 90 Then
```

```
        MsgBox "Grade: A"
```

```
    ElseIf score >= 80 Then
```

```
        MsgBox "Grade: B"
```

```
    Else
```

```
        MsgBox "Grade: C"
```

```
    End If
```

```
End Sub
```

# Loops

```
Sub ExampleLoop()
```

```
    Dim i As Integer
```

```
    For i = 1 To 10
```

```
        Cells(i, 1).Value = "Row " & i
```

```
    Next i
```

```
End Sub
```

## Reading and Writing to Cells:

```
Sub ReadWriteCells()  
    ' Write to a cell  
    Range("A1").Value = "Hello"  
  
    ' Read from a cell  
    MsgBox Range("A1").Value  
End Sub
```

# Working with Ranges

```
Sub ManipulateRange()  
    Dim rng As Range  
    Set rng = Range("A1:A10")  
    rng.Value = "Test"  
End Sub
```

# Interacting with Worksheets

```
Sub WorksheetExample()
```

```
    Dim ws As Worksheet
```

```
    Set ws = ThisWorkbook.Sheets("Sheet1")
```

```
    ws.Range("A1").Value = "VBA Rocks!"
```

```
End Sub
```

# Recording Macros

- One of the easiest ways to get started with VBA is by recording macros. Excel allows you to record your actions and generate VBA code that replicates those actions. To record a macro, go to the “View” tab, click on “Macros,” and then select “Record Macro.”
- Once you finish your actions, stop the recording. Excel will generate VBA code that reflects your actions. This recorded code can be used as a starting point for further customization and automation.

# Automating Repetitive Tasks

- One of the primary reasons for using VBA is to automate repetitive tasks. For instance, if you find yourself frequently formatting data in a specific way or performing calculations on the same set of data, VBA can save you significant time and effort.
- Let's take an example of automating data formatting:

```
Sub FormatData()  
' Select the range of data  
Range("A1:D10").Select  
  
' Apply bold font to the headers  
Selection.Font.Bold = True  
  
' Autofit columns for better visibility  
Columns.AutoFit  
End Sub
```

# Creating Custom Functions

- Excel VBA allows you to create custom functions that can be used in your worksheets just like built-in functions. Custom functions are written using VBA and can take arguments and return values.
- Here's an example of a simple custom function that calculates the area of a rectangle:

```
Function CalculateRectangleArea(Length As Double, Width As Double) As Double  
CalculateRectangleArea = Length * Width  
End Function
```

➤ After defining this function, you can use it in your worksheet:  
=CalculateRectangleArea(5, 10)



- Here is an example of a simple custom function that adds two numbers:

Function AddNumbers(num1 As Double, num2 As Double) As Double

    AddNumbers = num1 + num2

End Function

- After defining this function, you can use it in your worksheet:

= AddNumbers(5, 10)

# Handling User Input

- VBA also enables you to create user interfaces within Excel to gather input from users and perform actions based on that input. You can create custom dialog boxes, input boxes, and message boxes to interact with users.
- For example, let's create an input box to get the user's name and display a personalized greeting:

```
Sub PersonalizedGreeting()
```

```
Dim userName As String
```

```
userName = InputBox("Please enter your name:")
```

```
MsgBox "Hello, " & userName & "! Welcome to our Excel blog!"
```

```
End Sub
```

# Error Handling

- Like any programming language, VBA allows you to implement error handling to gracefully deal with unexpected issues that may arise during code execution. By using error handling, you can display custom error messages and prevent Excel from crashing due to runtime errors.

```
Sub DivideNumbers()  
On Error GoTo ErrorHandler  
Dim num1 As Double, num2 As Double, result As Double  
num1 = InputBox("Enter the first number:")  
num2 = InputBox("Enter the second number:")  
result = num1 / num2  
MsgBox "The result is: " & result  
Exit Sub  
ErrorHandler:  
MsgBox "An error occurred: " & Err.Description  
End Sub
```

```
Sub ErrorHandlingExample()  
    On Error GoTo ErrorHandler  
    Dim result As Double  
    result = 100 / 0 ' Will cause an error  
    Exit Sub  
ErrorHandler:  
    MsgBox "An error occurred: " & Err.Description  
End Sub
```