DATA 301
Introduction to Data Analytics
Microsoft Excel VBA

Dr. Ramon Lawrence
University of British Columbia Okanagan
ramon.lawrence@ubc.ca

Why Microsoft Excel Visual Basic for Applications?

Microsoft Excel VBA allows for automating tasks in Excel and provides a full programming environment for data analysis.

Excel VBA is commonly used in high finance and frequency trading applications for creating and validating financial models.

Using Excel VBA will be our first practice with programming and allow us to explore fundamental programming concepts of commands, variables, decisions, repetition, objects, and events.

Excel Visual Basic for Applications (VBA)

Visual Basic for Applications (VBA) is a programming language allowing users to build their own functions, automate tasks in Microsoft Office, and develop customized code.

The language has been part of almost all versions of Office for over 20 years.

VBA allows for expanding the capabilities of Excel and adding user-interface elements (buttons, lists) to your spreadsheet.

Macros

A ***macro*** is a recorded set of actions that is saved so that they can be easily executed again.

If you do the same set of actions *repetitively*, then creating a macro allows for doing all those actions with one command.

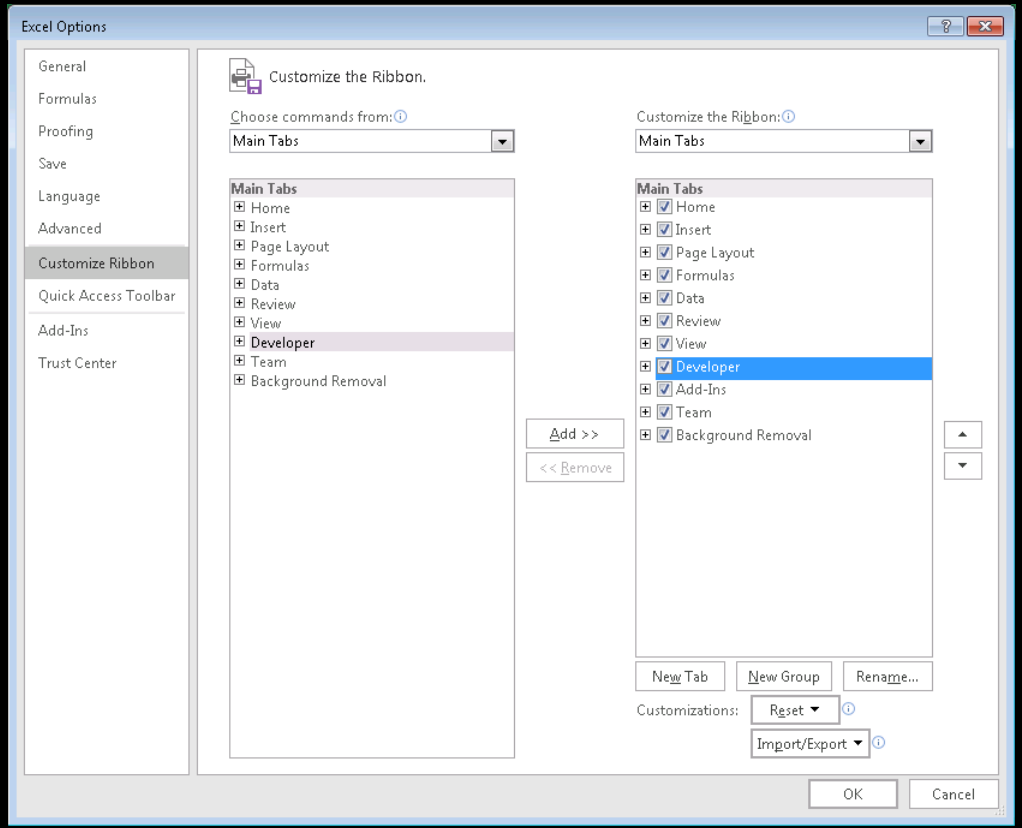
Macros are accessible under the `View` tab in the `Macros` group or the `Developer` tab.

Macros are converted into VBA programs.

Developer Tab

The ***Developer*** tab contains icons for performing VBA and macro development.

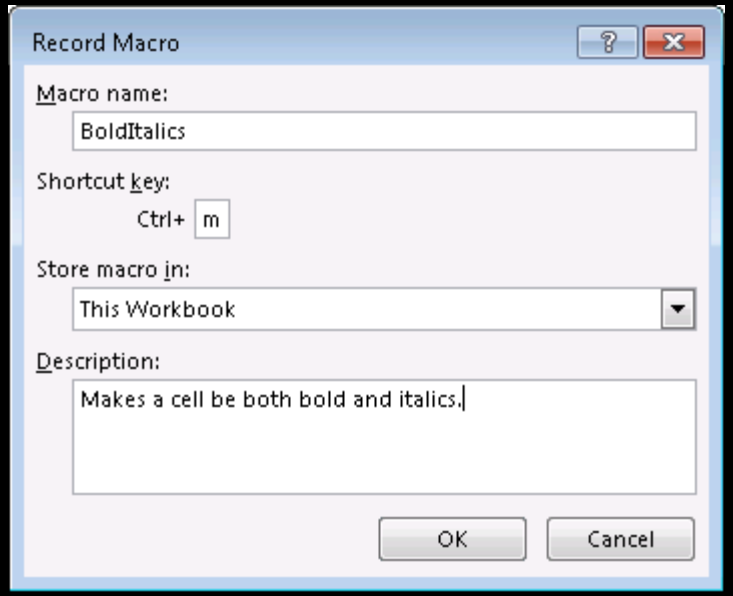
To add the `Development` tab, go to `File`, `Options`, `Customize Ribbon` and make sure it is checked beside `Developer`.



Recording a Macro

To record a macro, under `View` select, `Macros` -> `Record Macro`.

- Excel will record your actions until you select `Stop Recording`.
  - Note: Cursor movement is not captured.



Macro names cannot contain spaces or begin with a number.

It is recommended to use `Ctrl+Shift+`*`Key`* for a Shortcut key so that you do not override built-in shortcuts.

Macros can be created in a given workbook or a Personal Workbook allowing them to be used in multiple workbooks.

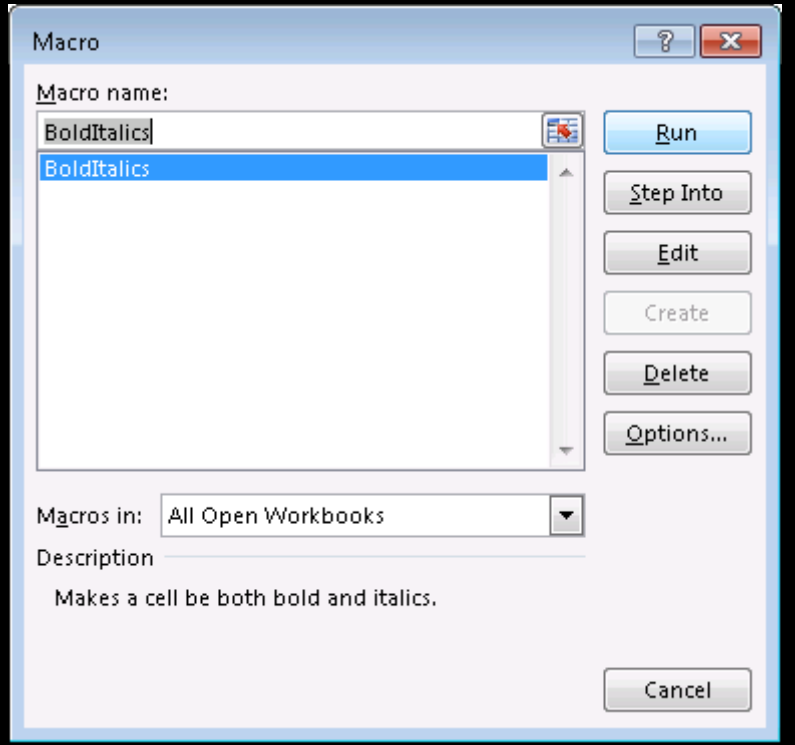
Using a Macro

Use a macro in the following ways:

1) With the shortcut key if defined

2) Assign a macro to a button or on the toolbar

3) Under `Macros`, Select `View Macros` then pick the macro and `Run`.



Macros Question

Question: Select a **TRUE** statement.

A) A macro can be created without assigning it a shortcut key.

B) A macro will record cursor movements.

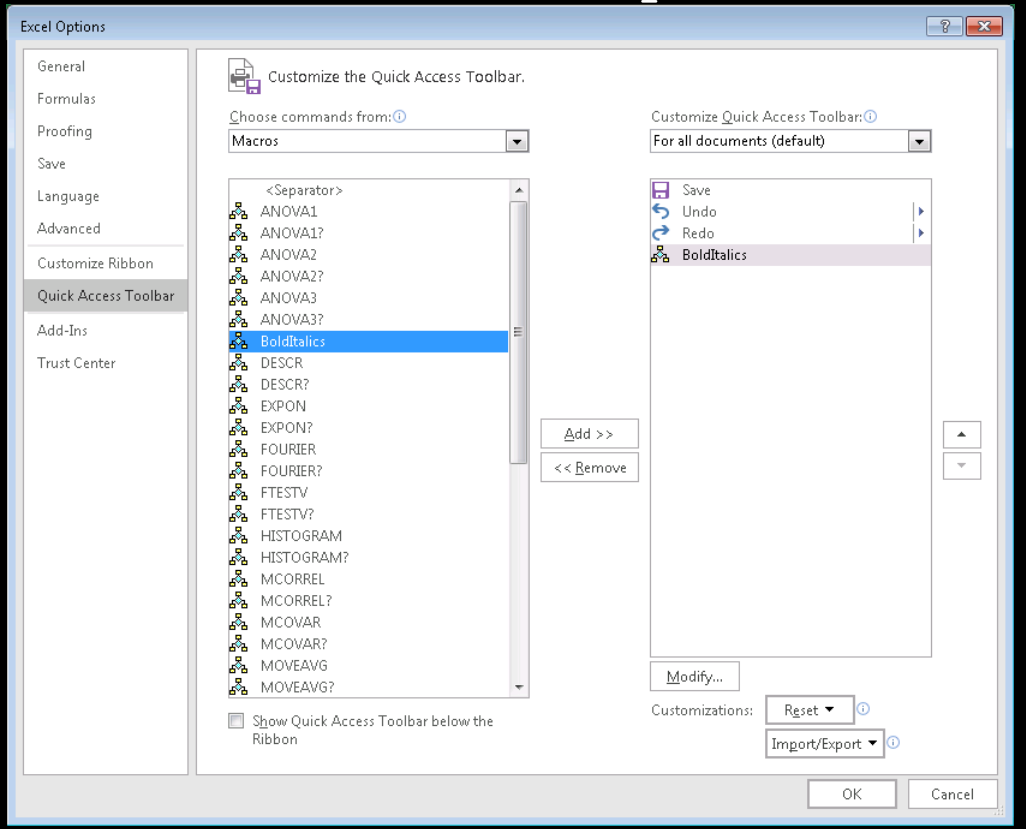
C) Macros can be created in an individual workbook or in a personal macro workbook so they can be used in multiple workbooks.

D) A macro can have only one command it executes.

Adding Macro to Quick Access Toolbar

Add a macro to The Quick Access Toolbar under `File`, `Options`, `Quick Access Toolbar`.

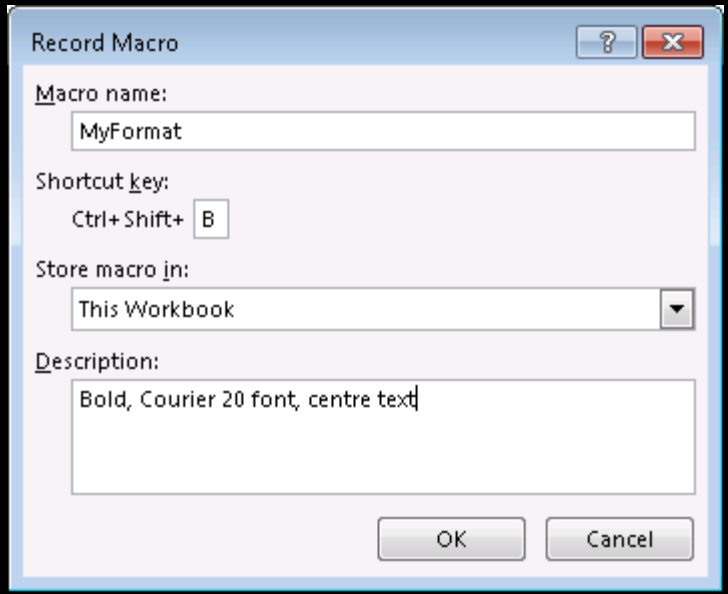
macro icon



Try it: Macros

Question: Create a macro that does the following tasks:

- Bolds the cell and makes the font Courier 20.
- Sets the cell background to orange.
- Centers the text in the cell.
- Use a shortcut of `Ctrl+Shift+b`.
- Add it to the Quick Access Toolbar.

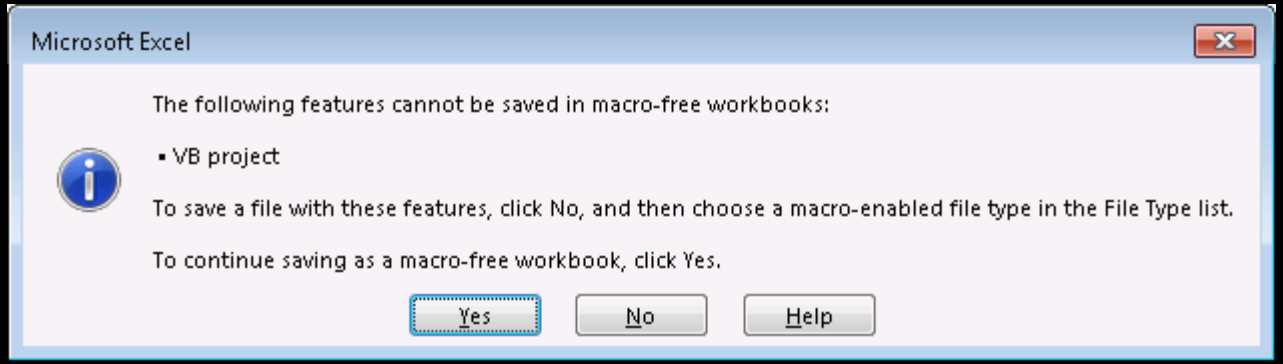


Try-out your macro using the shortcut key, toolbar, and from the macro dialog.

Saving Workbook with Macros

Excel now forces workbooks with macros to be saved in Excel Macro-Enabled Workbook (*.xlsm) format.

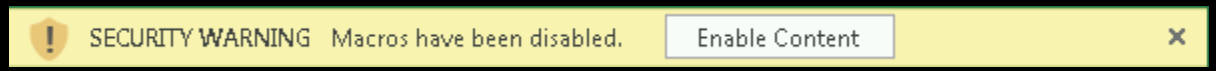
Saving a workbook with macros in regular format gives this error:



Macro Security

Since macros can execute any code, they have been a target for virus writers. Understanding the source of the Excel spreadsheet that contains macros is important when deciding to run them or not.

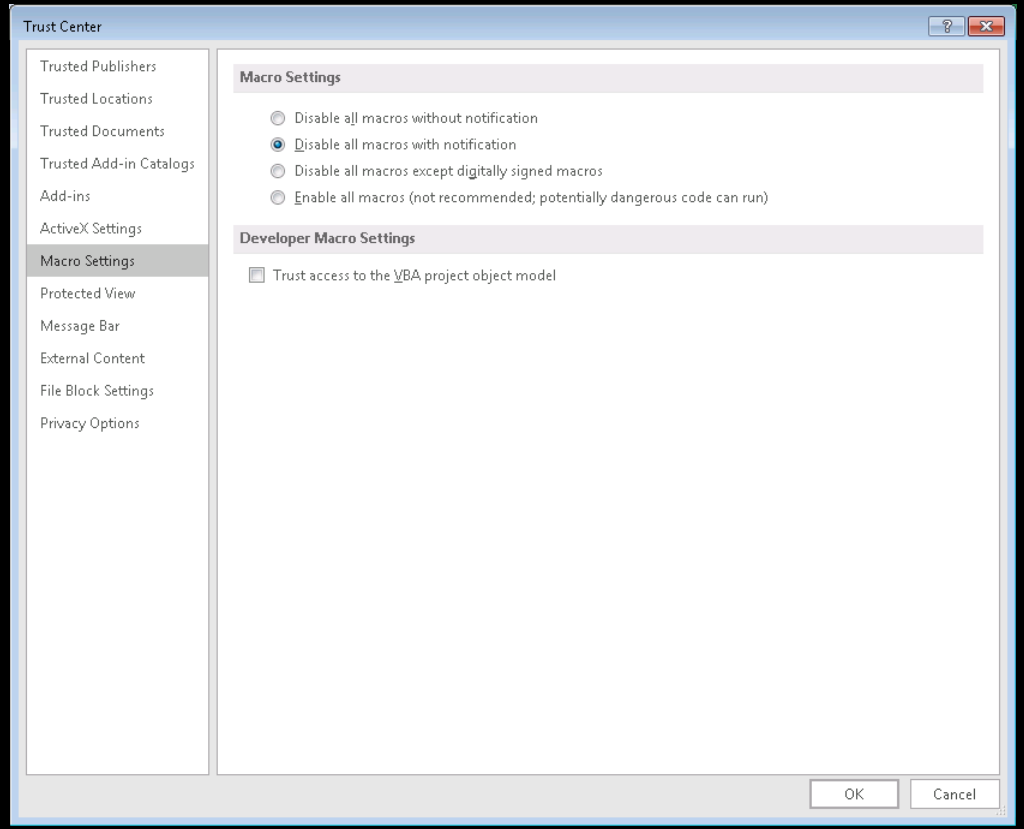
Excel has ***macro security settings*** to allow you to enable or disable running macros. Spreadsheets with macros often will generate a warning when opening them:



Macro Security Settings

The default security is **Disable all macros with notification** that prevents macros from running but displays a warning allowing you to enable them.

One of the biggest issues with macros is security and making sure you are only using macros from a trusted source.

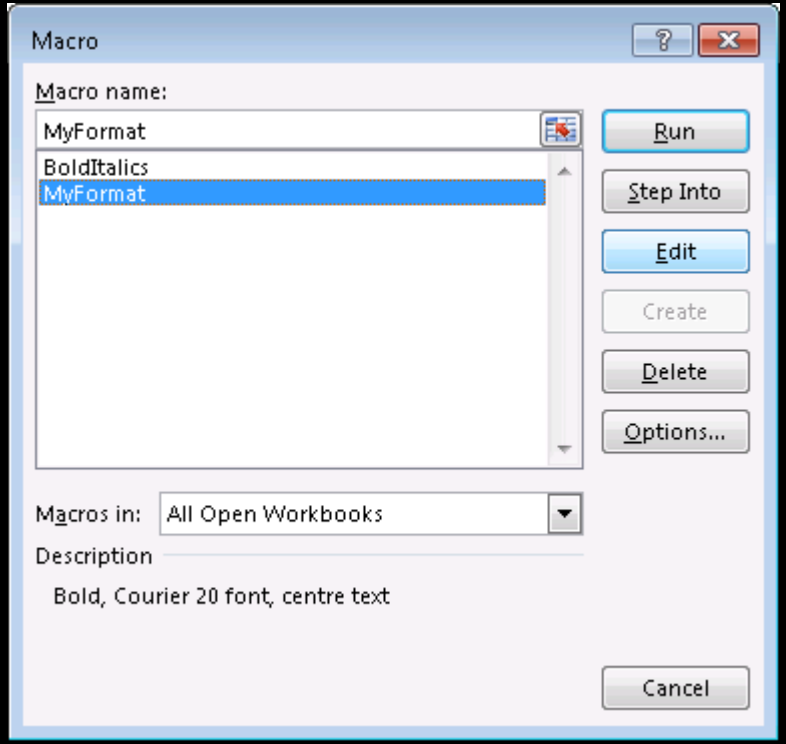


Macros: Implementation

Macros are converted to Visual Basic code.

Can edit macro code and create your own code.

Under the `Developer` tab, select `Macros` then `Edit` macro to modify the code.



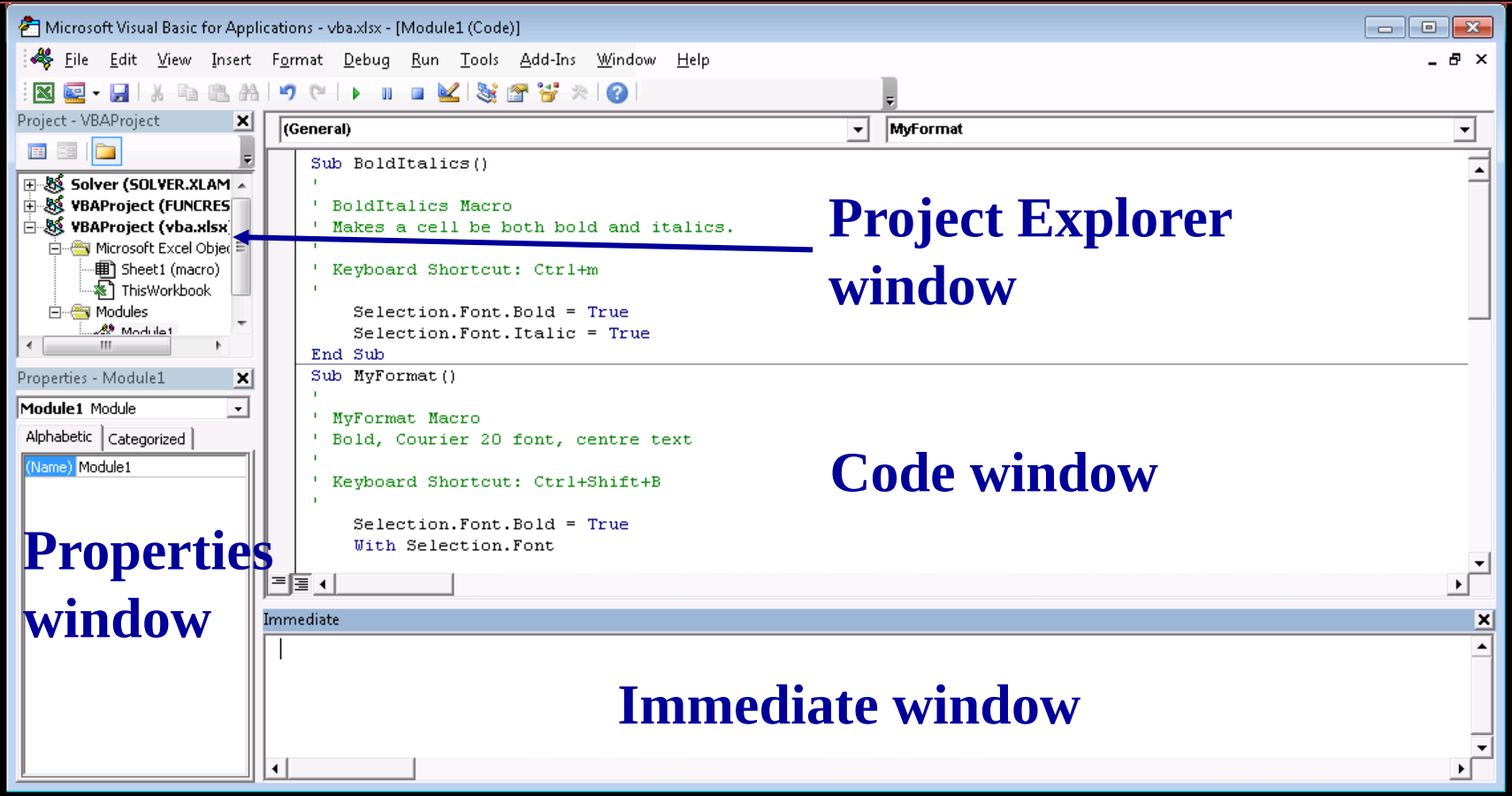
Visual Basic Editor

Visual Basic Editor (VBE) allows editing visual basic code and is a complete integrated development environment (IDE).

Users can create and edit macros as well as other Visual Basic code with the editor.

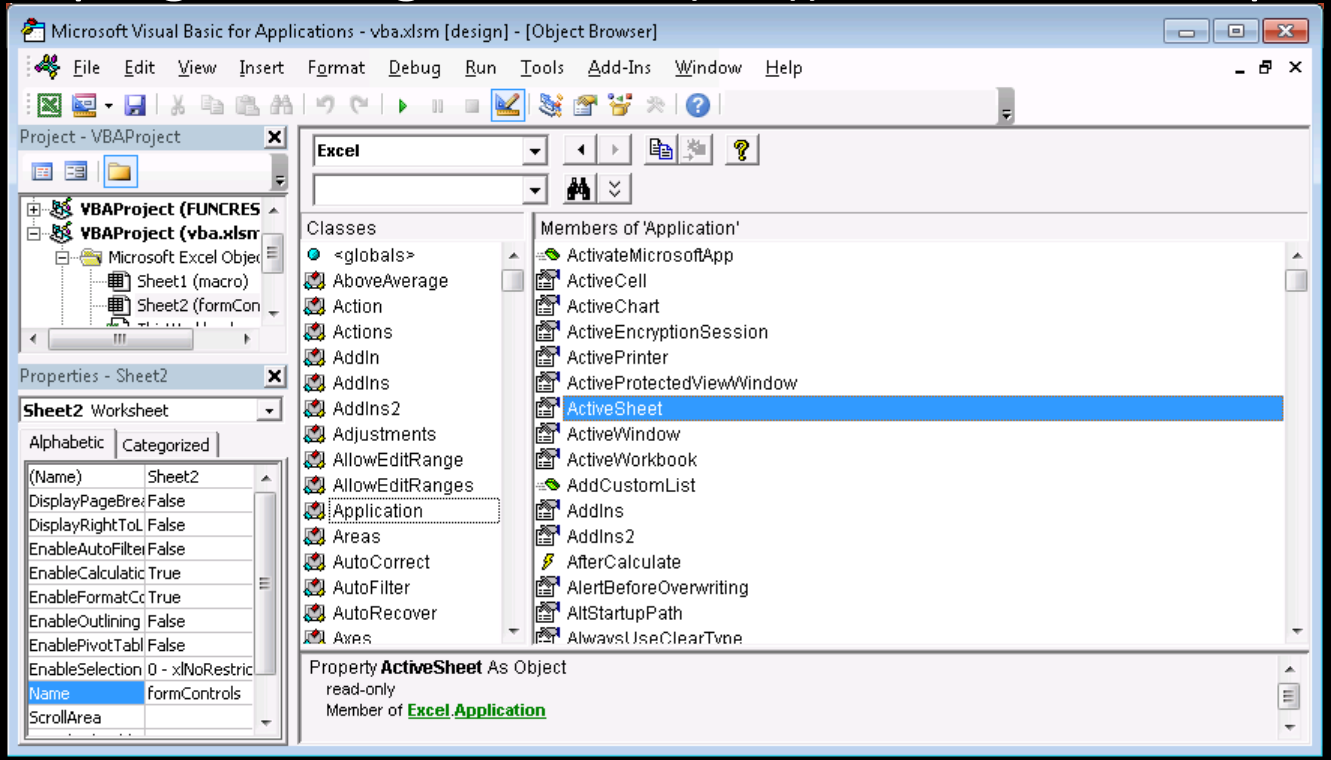
To open the VBE, under `Developer` tab -> `Visual Basic` or `Alt+F11`.

Visual Basic Editor Screenshot



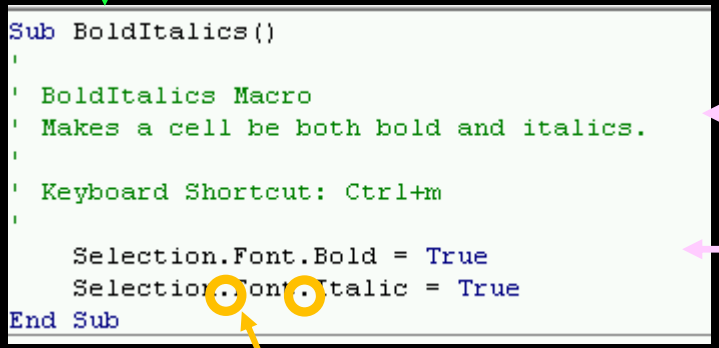
Object Browser

Object browser allows for exploring objects and methods (the application programming interface (API)) of Excel VBA. Open with `F2`.



Macro Code in Visual Basic Editor

Subroutine with name and no arguments



```
Sub BoldItalics()
'
' BoldItalics Macro
' Makes a cell be both bold and italics.
'
' Keyboard Shortcut: Ctrl+m
'
    Selection.Font.Bold = True
    Selection.Font.Italic = True
End Sub
```

Comments start with '

Every statement is on its own line.

Dot notation to separate "items" (objects, methods, properties).

WITH Statement in Visual Basic Code

```
Sub MyFormat()
'
' MyFormat Macro
' Bold, Courier 20 font, centre text
'
' Keyboard Shortcut: Ctrl+Shift+B
'
    Selection.Font.Bold = True
    With Selection.Font
        .Name = "Courier New"
        .Size = 20
        .Strikethrough = False
        .Superscript = False
        .Subscript = False
        .OutlineFont = False
        .Shadow = False
        .Underline = xlUnderlineStyleNone
        .ThemeColor = xlThemeColorLight1
        .TintAndShade = 0
        .ThemeFont = xlThemeFontNone
    End With
    With Selection.Interior
        .Pattern = xlSolid
        .PatternColorIndex = xlAutomatic
        .ThemeColor = xlThemeColorAccent2
        .TintAndShade = 0
        .PatternTintAndShade = 0
    End With
```

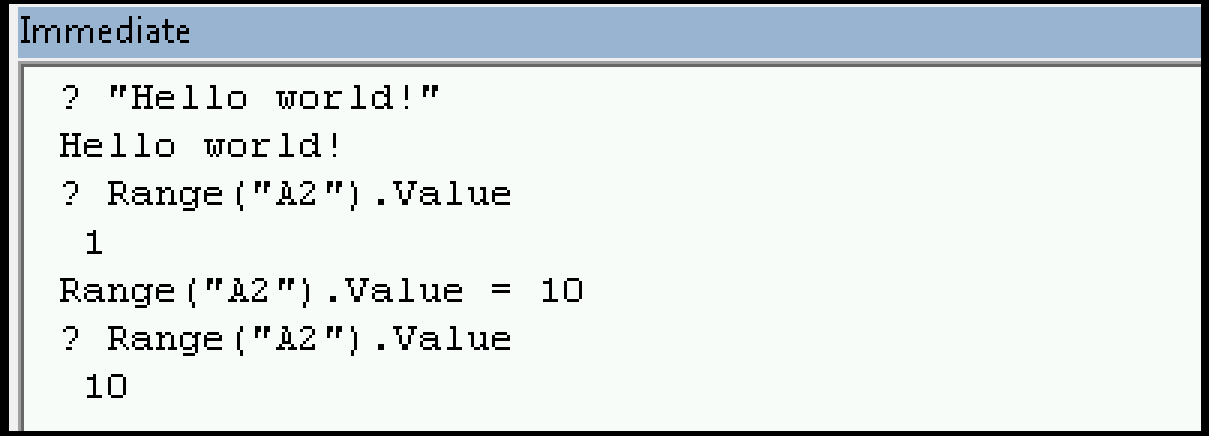
`WITH` syntax simplifies typing same object many times.

These lines all apply to `Selection.Font`.

Visual Basic Editor: Immediate Window

The Immediate window allows entering of single line commands.

- Use `PRINT` or `?`
- In code, use `Debug.Print` to print to immediate window.



Try it: Immediate Window

Question: Try do these actions using the immediate window:

- 1) Print `"Hey There!"`
- 2) Calculate the answer of `765 * 39`.
- 3) Select a cell then call the macro `RedItalics`.
- 4) Change the value of cell `B4` to `"DATA"`.
- 5) Change the value of cell `A6` to `100`.

Challenge Try it: Create Macro in VBE

Question: Copy the `MyFormat` macro and edit to produce a new macro called `RedUnderline` that:

- Underlines the text in the cell.
- Makes the cell background red.
- If the cell was bold or italics before, resets to not have bold and italics.

Hints:

- Underline property in Excel is `Font.Underline` and can set to constant `xlUnderlineStyleSingle`.
- Can change background color with `Interior.Color` and set to `RGB(`*`redValue`*`, `*`greenValue`*`, `*`blueValue`*`)` where the color values are numbers from 0 to 255.

Introduction to Programming

An ***algorithm*** is a precise sequence of steps to produce a result. A ***program*** is an encoding of an algorithm in a ***language*** to solve a particular problem.

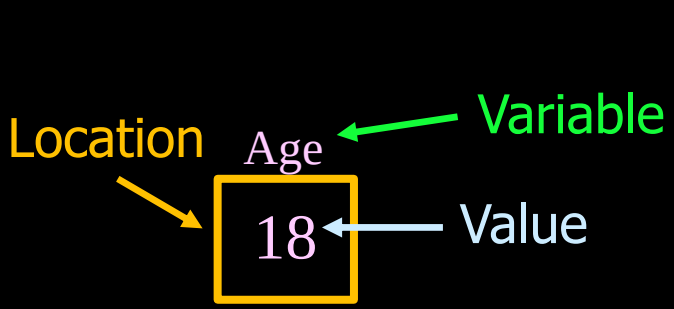
There are numerous languages that programmers can use to specify instructions. Each language has its different features, benefits, and usefulness.

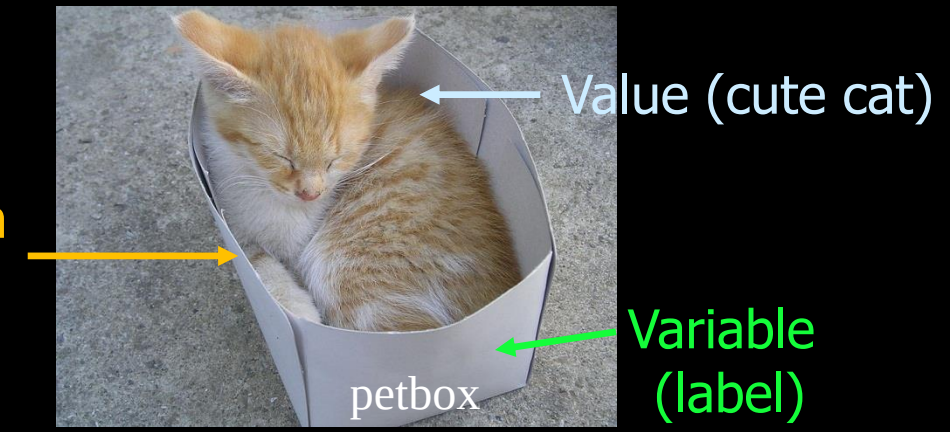
- We will start with Excel VBA but also study Python and R.

The goal is to understand fundamental programming concepts that apply to all languages.

Variables

A ***variable*** is a name that refers to a location that stores a data value.





IMPORTANT: The *value* at a location can change using initialization or assignment.

Variable Assignment

Assignment using an **=** sets the value of a variable.

Example:

```
num = 10
num = Range("A1").Value
num = 20
```

Excel Variables

Every variable in Excel has a ***name*** and a ***data type***.

- Variables increase code efficiency and readability.
- Data types: `Boolean`, `Currency`, `Date`, `Double`, `Integer`, `Long`, `Object`, `String`, `Variant` (any type)

Example:

```
Dim num As Integer
```

Collections

Collections are variables that store multiple data items. Data items can either be indexed (selected) by name or number.

Example:

```
Worksheets("macro")
Worksheets(2)
```

`Worksheets` is a collection as there may be multiple worksheets in the workbook. Select one by name or number (starting with 1).

Variables Question

Question: How many of the following statements are **TRUE**?

1) A variable name cannot change during a program.
2) A variable value cannot change during a program.
3) A collection is a variable that can store multiple data items.
4) A value in a collection can be retrieved by name or by index starting from 0.
5) In Excel, variables are declared using DIM.
6) In Excel, variables are declared with a data type.

A) 0 **B)** 1 **C)** 2 **D)** 3 **E)** 4

Decisions

Decisions allow the program to perform different actions in certain conditions.

- Logical operators: `AND`, `OR`, `NOT`

Excel decision syntax:

```
If condition Then
   statement
End If
```

```
If condition Then
   statement
Else
   statement
End If
```

Question: Decisions

Question: What is the output of the following code?

```
Sub TryIf()
    Dim num As Integer
    num = 20
    If num > 10 Then
        Debug.Print num * 5
    Else
        Debug.Print num * 2
    End If
End Sub
```

A) 100 **B)** 40 **C)** 20 **D)** error – no output

Try it: Decisions

Question: Create a method called `EchoDecision` that asks user a Yes and No question and outputs a message either "`Yes`" or "`No`" depending on what they chose.

```
Sub EchoDecision()
    Dim answer As Integer
    answer = MsgBox("Pick Yes or No!", vbYesNo)
    ' answer will either be vbYes or vbNo
    ' Use debug.print to output "Yes" or "No"
End Sub
```

Loops and Iteration

A ***loop*** repeats a set of statements multiple times until some condition is satisfied.

- Each time a loop is executed is called an ***iteration***.
- A `for` loop repeats statements a given number of times.

Excel `for` loop syntax:

```
Dim i as Integer
For i = 1 To 5
    Debug.Print i
Next i
```

Question: Loops

Question: How many numbers are printed with this loop?

```
Sub TestFor()
    Dim i As Integer

    For i = 0 To 10
        Debug.Print i
    Next i
End Sub
```

A) 11 **B)** 10 **C)** 0 **D)** error – no output

Try it: Loops

Question: Create a method called `TryFor` that prints the numbers 1 to 20. Challenging variants:

- Print the numbers from 10 down to 1.
- Print only the even numbers from 1 to 10.

User-Defined Functions (UDFs)

A ***user-defined function*** is your own Excel function that can be used in formulas like built-in functions.

A UDF must return a number, string, array, or Boolean.

A UDF cannot change the Excel environment including the current cells or other cells (e.g. change formatting).

UDF Example

UDF `doubleIt` will double the input argument.

```
' UDF expect a number to double
Function doubleIt(num As Integer)
    ' To return a value, assign the value to the method name
    doubleIt = num * 2
End Function
```

UDF Example – Sum Cells by Background Color

```
' Sums all the cells with the same color
Function SumColor(RangeToSum As Range, ColorID As Integer) As Long
  Dim ColorCell As Range
  Dim result As Long

  ' Loop through each cell in the range.
  For Each ColorCell In RangeToSum
    If ColorCell.Interior.ColorIndex = ColorID Then
        result = result + ColorCell.Value
    End If
  Next ColorCell

  SumColor = result
End Function
```

Try it: UDF

Question: Create a UDF called `CountNum` that will return a count of the number of digits (0 to 9) in a string.

Advanced: Object-Oriented Programming

Object-oriented programming structures code as object, classes, methods, and properties. This organization makes it easier to understand and construct large programs.

An ***object*** is an instance of a class that has its own properties and methods that define what the object is and what it can do.

A ***class*** is a generic template (blueprint) for creating an object. All objects of a class have the same methods and properties (although the property values can be different).

A ***property*** is an attribute or feature of an object.

A ***method*** is a set of statements that performs an action.

Excel Objects

Excel structures everything as a hierarchy of objects, and commands are done by running a method of an object.

An object may contain other objects as well as methods and properties. A dot " . " is used as a separator between objects and subobjects, methods, and properties.

Examples:

- Top-level object: `Application`
- `Workbook` – individual Excel file
- `Worksheet` - sheet in a workbook

```
Application.ActiveWorkbook.Worksheets("macro").Range("A1").Value
```

Excel Range Object

The `Range` object selects a cell or group of cells.

Example:

```
Worksheets("Sheet1")
   .Range("A1:C3").Font.Italic = True
```

Excel Object Methods

Methods perform an action.

Example:

```
Worksheets("macro").Activate
```

Object-Oriented Question

Question: How many of the following statements are **TRUE**?

1) A method can have no parameters.

2) Two objects of the same class have the same properties.

3) Two objects of the same class may have different values for their properties.

4) `Workbook` is the top-level object in Excel.

A) 0 **B)** 1 **C)** 2 **D)** 3 **E)** 4

Try it: Excel Objects

Question: Using the Immediate window try to perform the following actions with methods on Excel objects:

1) Switch the active worksheet to form.
2) Switch the active cell to macro sheet A4.
3) Use `msgbox` to display value in current cell (`ActiveCell`).

Forms and Input Controls

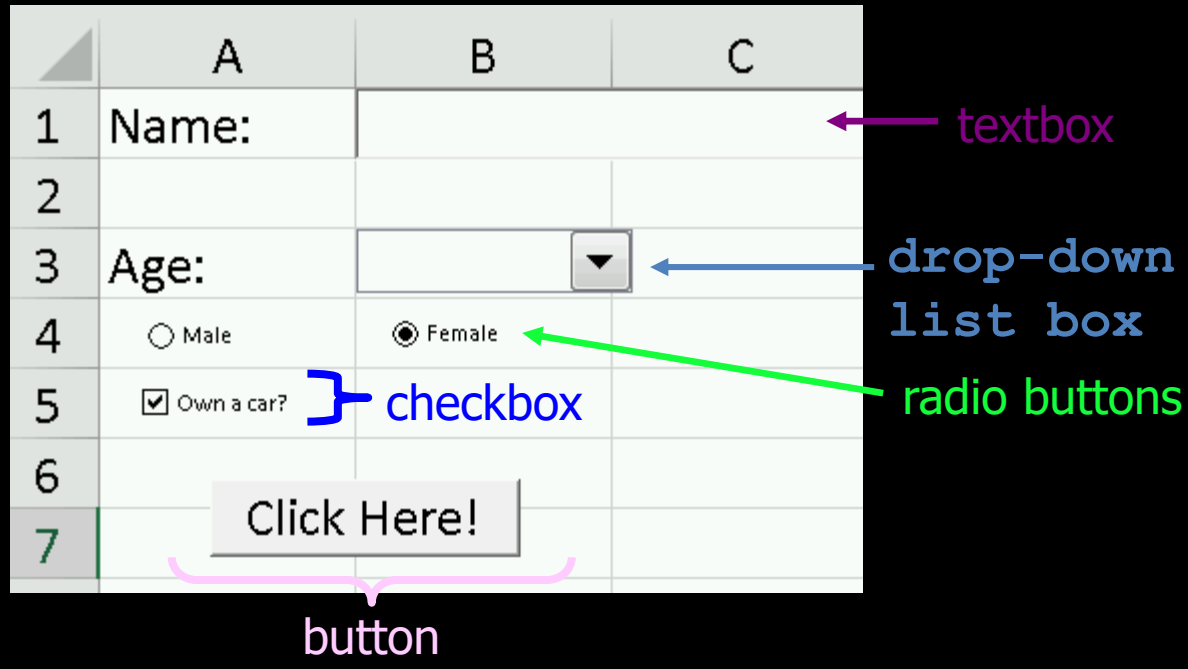
Excel allows the creation of forms with controls for a better interface.

Two types of controls in Excel:

- Form controls – default
- ActiveX controls – allow more flexibility and customization

Controls can be inserted from the Developer tab. Select `Insert`, pick control, and then click and drag the size and shape of the control on the spreadsheet.

Input Controls



Events

An ***event*** is a notification to your program that something has occurred.

Events in Excel:

- add a worksheet
- double-click on a cell
- change a cell value
- calculating a formula
- click on a button (can execute a macro)

Worksheet-level events on a particular worksheet and workbook level events for entire file.

Conclusion

Microsoft Excel VBA allows for automating tasks in Excel and provides a full programming environment for data analysis.

Macro record a set of actions so they can be easily executed again.

- Be aware of security risks when using macros.

The ***Visual Basic Editor (VBE)*** is a complete integrated development environment for editing macros, ***user-defined functions***, and adding forms and controls that dynamically respond to events.

Excel VBA uses ***object-oriented programming*** that structures code as object, classes, methods, and properties. A developer can control and automate everything with Excel using VBA.

Objectives

- List some reasons to use Excel VBA
- Define macro and explain the benefit of using macros
- Be able to record and execute a macro
- Explain the security issues with macros and how Excel deals with them
- List and explain the use of the four main windows of the Visual Basic Editor
- Explain the role of the object browser
- Explain and use the `WITH` statement syntax
- Be able to write simple macros using the VBE
- Define: algorithm, program, language
- Define: object-oriented programming, object, class, property, method
- Understand and use dot-notation
- Use the Range object to select a group of cells
- Define: variable, value, location

Objectives (2)

- Create and use Excel variables
- Explain how a collection is different from a typical variable
- Use `If/Then/Else` syntax to make decisions
- Use `For` loop for repetition
- Create user-defined functions and use them in formulas
- Define: event
- List some typical user interface controls
- Understand that Excel allows for forms and controls to be added to a worksheet which respond to events