Data 301 Data Analytics Microsoft Execl VBA

Dr. Irene Vrbik

University of British Columbia Okanagan
irene.vrbik@ubc.ca

Term 1, 2018

What is VBA?

Excel Visual Basic for Applications (VBA)
Visual Basic for Applications (VBA) is a programming language developed by Microsoft

What is VBA?

Excel Visual Basic for Applications (VBA)
Visual Basic for Applications (VBA) is a programming language developed by Microsoft

It allows users to build their own functions, automate tasks in Microsoft Office, and develop customized code.

What is VBA?

Excel Visual Basic for Applications (VBA)
Visual Basic for Applications (VBA) is a programming language developed by Microsoft

It allows 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.

What is VBA?

Excel Visual Basic for Applications (VBA)
Visual Basic for Applications (VBA) is a programming language developed by Microsoft

It allows 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.

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.

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.

- We can use VBA to automate tedious processes in Excel, eg. formatting a month report

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.

- We can use VBA to automate tedious processes in Excel, eg. formatting a month report

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

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.

- We can use VBA to automate tedious processes in Excel, eg. formatting a month report

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.

Macros

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

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

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

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.

Developer Tab

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

This tab is disabled by default.

Developer Tab

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

This tab is disabled by default.

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

Developer Tab

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

This tab is disabled by default.

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

For a Mac, go to **Excel**, **Preferences**, **View**. Under the *In Ribbon, Show* heading, select the checkbox marked “Developer Tab"

Recording a Macro

To record a macro, go to the **View** tab and select the "Record Macro" button

Recording a Macro

To record a macro, go to the **View** tab and select the "Record Macro" button

Recording a Macro

To record a macro, go to the **View** tab and select the "Record Macro" button

Macro names cannot contain spaces or begin with a number.

Recording a Macro

To record a macro, go to the **View** tab and select the "Record Macro" button

Macro names cannot contain spaces or begin with a number.

It is recommended to use Ctrl +Shift +<Key> (PC)/Option +Cmd +<Key> (Mac) for a Shortcut key so that you do not override built-in shortcuts.

Recording a Macro

To record a macro, go to the **View** tab and select the "Record Macro" button

Macro names cannot contain spaces or begin with a number.

It is recommended to use Ctrl +Shift +<Key> (PC)/Option +Cmd +<Key> (Mac) for a Shortcut key so that you do not override built-in shortcuts.Macs will give you a warning when you attempt to override an existing shortcut

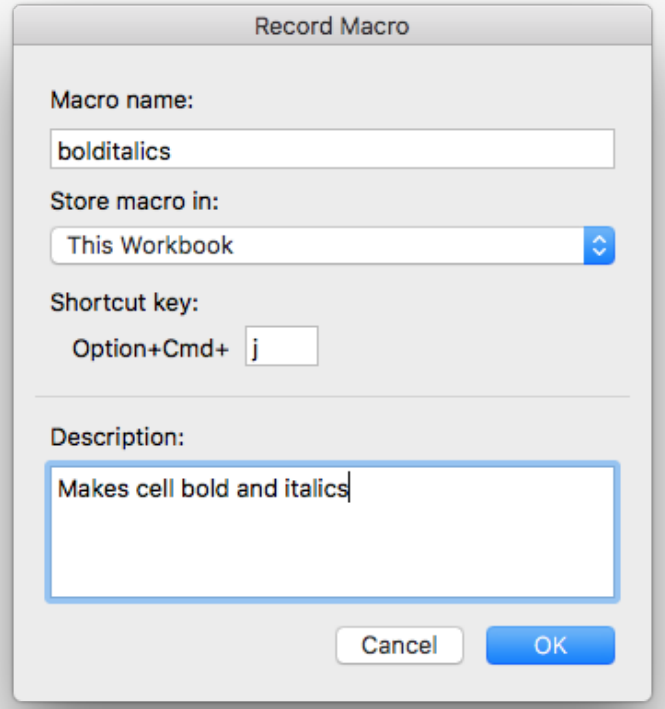
Recording a Macro

To record a macro, go to the **View** tab and select the "Record Macro" button Macro names cannot contain spaces or begin with a number.

It is recommended to use Ctrl +Shift +<Key> (PC)/Option +Cmd +<Key> (Mac) for a Shortcut key so that you do not override built-in shortcuts.Macs will give you a warning when you attempt to override an existing shortcut

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

As a simple example, we could create a macro that bold and italicizes a cell.



Excel will record your actions until you select **Stop Recording**.

- Note: Cursor movement is not captured.

Recording Macros

Make sure we don't move out of the active cell once we start to record our macro.

By default, macros are created using absolute references

- To use Relative references, we need turn on the "Use Relative References" button, before we record.

While the Macro Recorder makes creating macros very easy, it has it's limitations:

- ☹☹ There is no way (that I can find) to do this on a Mac ☹☹
- eg. cannot handle "loops" (more on loops later)
- generates more code than is necessary (which can slow down your process).

Using a Macro

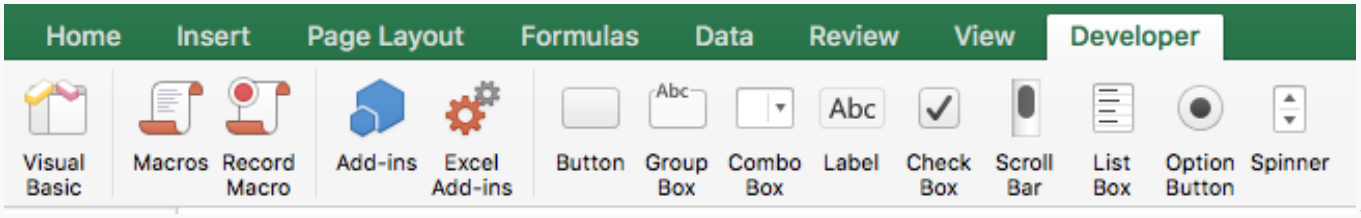
There are a number of ways you can use your macro:

1. With the shortcut key if defined
2. Under **Macros**, Select **View Macros** then pick the macro and **Run**.
3. Assign a macro to a button or on the toolbar

Macro Buttons

To assign a macro button:

1. Select the **Developer** tab and click



2. To prompt the Assign Macro popup window, click where you would like this button to appear on your worksheet.
   - Note: If you have already inserted a button (eg. **Insert** "Shapes"), you can right-click on it, and select **Assign Macro**.
3. Assign a macro to the button by selecting one from the list of existing macros or creating a brand new one and click OK.

Macro Toolbars

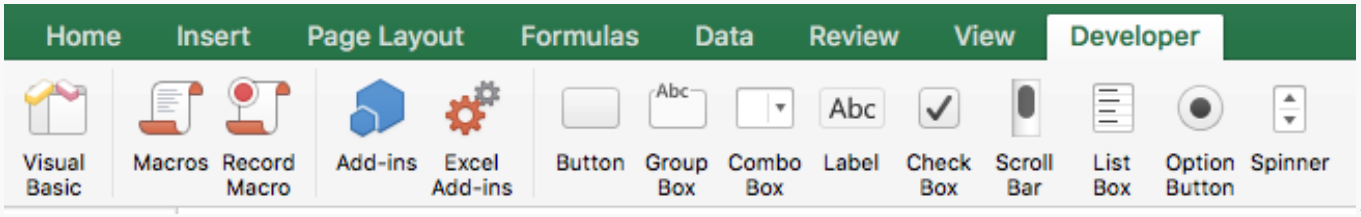
To add a macro to The Quick Access Toolbar, select:

1. PC **File** → **Options** → *Quick Access Toolbar*.
   Mac **Excel** → **Preferences** and click *Ribbon and Toolbar* → *Quick Access Toolbar* header
2. In the "Choose commands" drop-down menu, select **Macros**
3. Select the macro and add to your toolbar (click this ›)
4. Modify the button with a unique symbol (eg. ☺).
   - ☹☹ Feature not available on a mac? ☹☹

Macro Buttons

You can assign multiple macros to a single button! To assign multiple macro button:

1. Select the **Developer** tab and click



2. To prompt the Assign Macro popup window, click where you would like this button to appear on your worksheet.
3. Select "New" and the VBA editor will open. Type the name of the macros you wish to assign in the body of the Sub.

```
Sub ButtonX_Click()
    bolditalics
    undercenter
End Sub
```

Try It: Macros

Question

Create a macro called `MyFormat` that does the following tasks:

1. Use a shortcut of Ctrl + Shift + b (PC)/Opt + Cmd + b (Mac).
2. Bolds the cell and makes the font Courier 20
3. Sets the cell background to orange.
4. Centers the text in the cell.
5. Add it to a button in the shape of a star that says OC20

Try-out your macro using 1) the shortcut key, 2) the macro dialog, and 3) the macro button, and

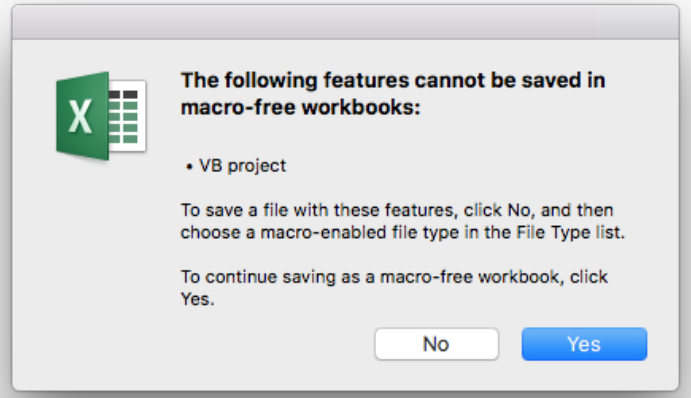
Saving macros

In order to save the workbook with the macros we've created while it was open, we need use a "macro-enabled" file format.

Saving macros

In order to save the workbook with the macros we've created while it was open, we need use a "macro-enabled" file format.

When you go to save a workbook containing macros, you will receive the following popup



- 'Yes' will save it as as macro-enabled workbook (*.xlsm file)
- 'No' will save it as as macro-free workbook (*.xlsx file)

Saving macros

Saving a macros to a given workbook will only allow you to use it with that file.

Saving macros

Saving a macros to a given workbook will only allow you to use it with that file.

We could alternatively save it to a *Personal Workbook* allowing them to be used in multiple workbooks.

Saving macros

Saving a macros to a given workbook will only allow you to use it with that file.

We could alternatively save it to a *Personal Workbook* allowing them to be used in multiple workbooks.

Your Personal Workbook is a hidden workbook saved under the name **personal.xlsb**

Saving macros

Saving a macros to a given workbook will only allow you to use it with that file.

We could alternatively save it to a *Personal Workbook* allowing them to be used in multiple workbooks.

Your Personal Workbook is a hidden workbook saved under the name **personal.xlsb**

This file loads every time you open up excel

Saving macros

Saving a macros to a given workbook will only allow you to use it with that file.

We could alternatively save it to a *Personal Workbook* allowing them to be used in multiple workbooks.

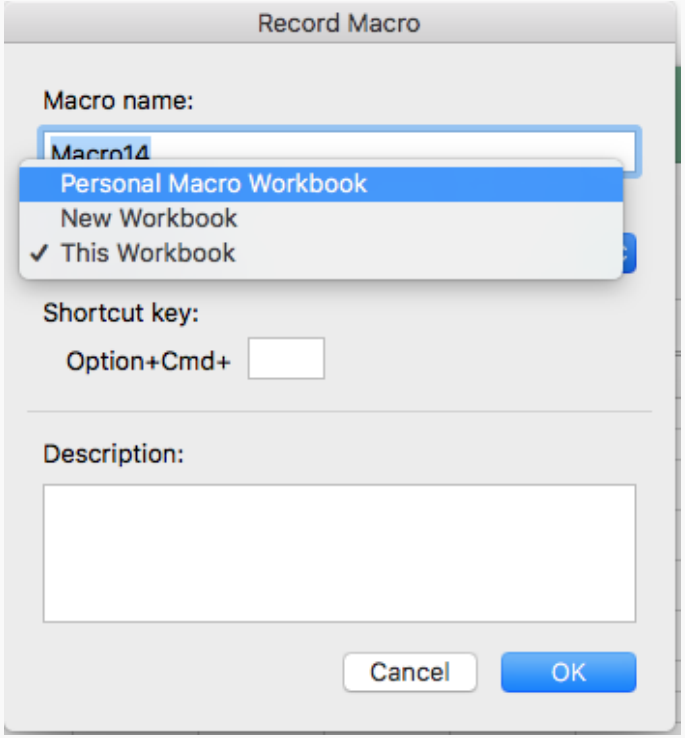
Your Personal Workbook is a hidden workbook saved under the name **personal.xlsb**

This file loads every time you open up excel

Hence saving a macro to **personal.xlsb** will allow you to use that macro on any workbook that you open on your computer

Saving macros

To save it to your personal workbook, select the "Personal Macro Workbook" option in the drop-down menu for "Store Macro in"



Macro Security

Since macros can execute any code, they have been a target for virus writers.

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.

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.

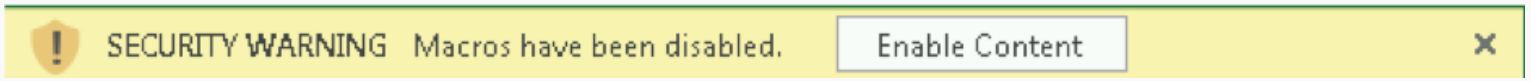
Macro Security

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 to *Disable all macros with notification*. This prevents macros from running but displays a warning allowing you to enable them.

Macro Security Settings

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

One of the biggest issues with macros is security

Macro Security Settings

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

One of the biggest issues with macros is security

Make sure you are only using macros from a trusted source.

Macro Security Settings

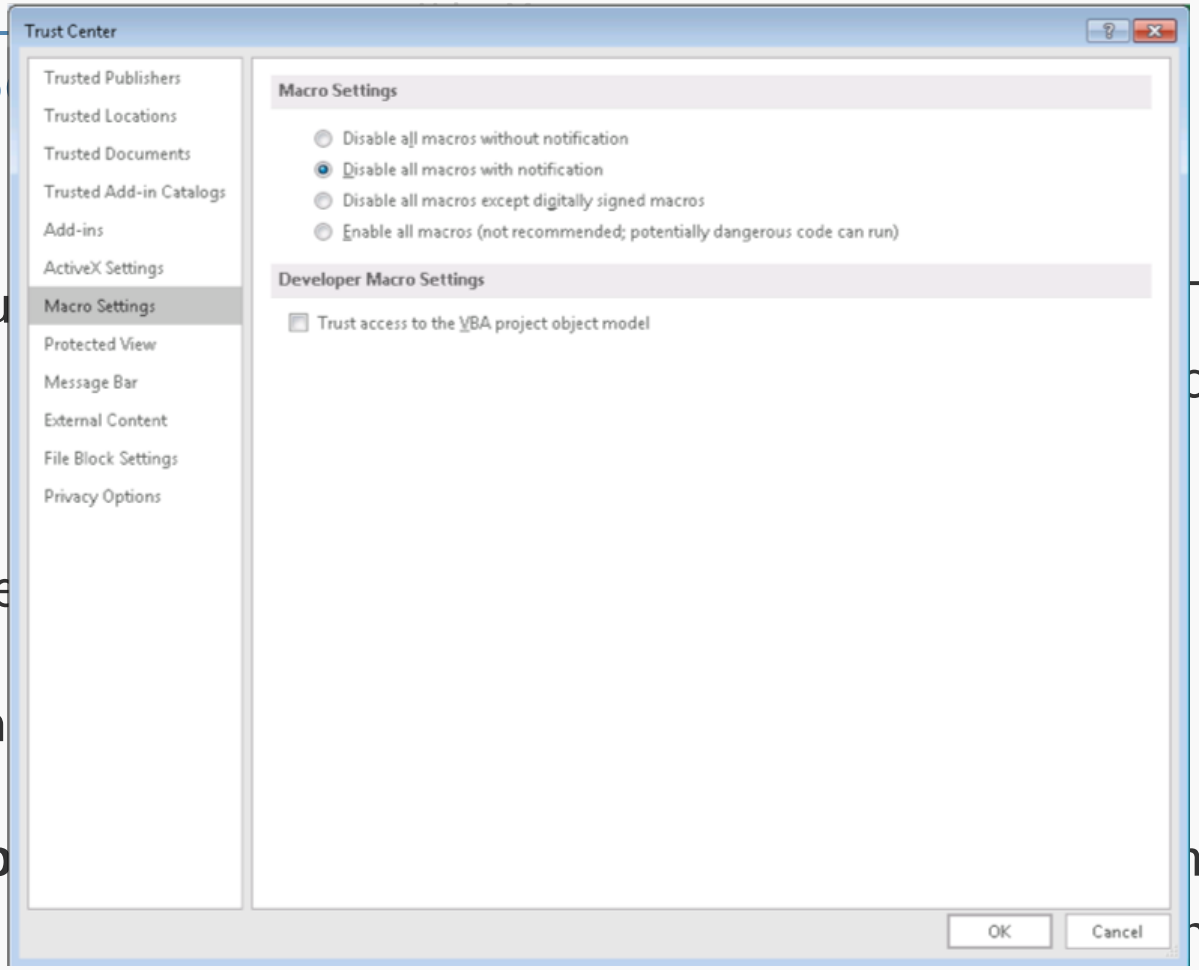
The default security is to *Disable all macros with notification*. This prevents macros from running but displays a warning allowing you to enable them.

One of the biggest issues with macros is security

Make sure you are only using macros from a trusted source.

On the **Developer** tab, click "Macro Security" located below the "Record Macro" button. Select a desired option in Macro Settings category, under the *Macro Settings* header (PC)/ or in **Excel** → **Preference** "Security & Privacy"

Macro Security S



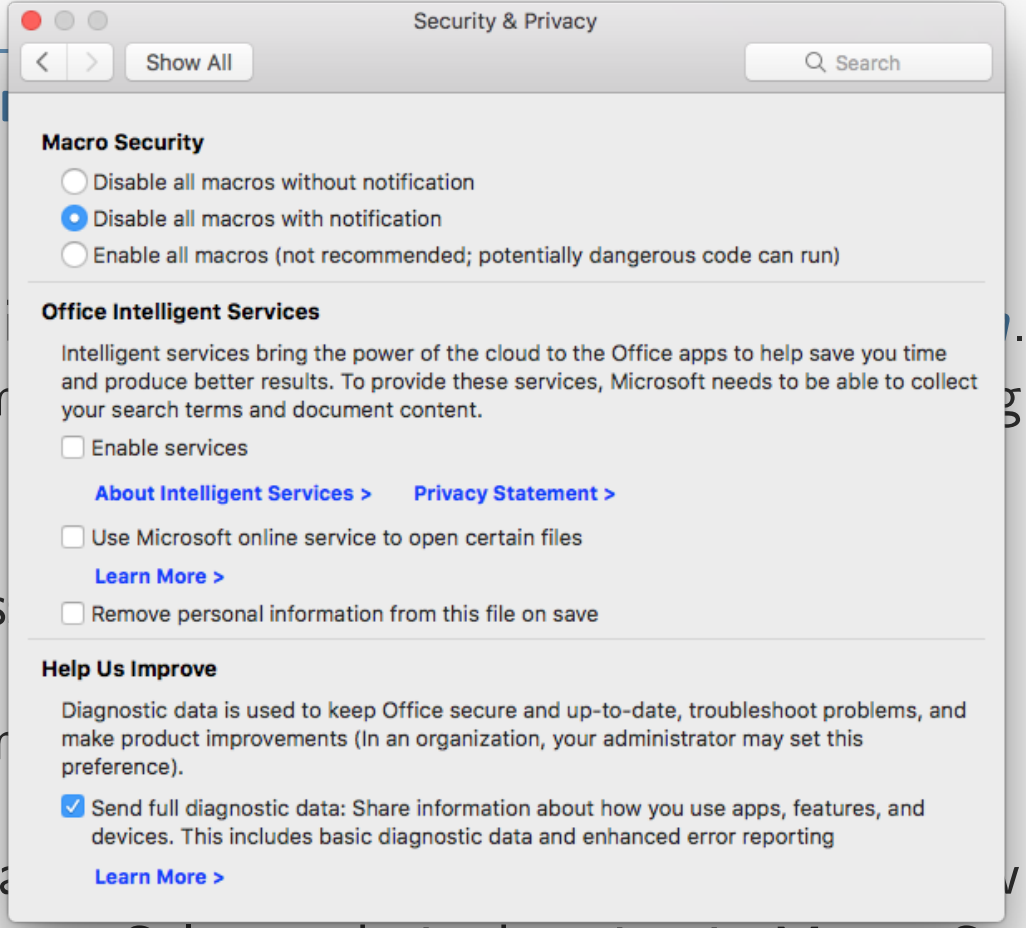
The default secu This prevents macros ou to enable them.

One of the bigge

Make sure you a

On the **Develop** ne "Record Macro" ngs category, under the *Macro Settings* header (PC)/ or in **Excel** → **Preference** "Security & Privacy"

Macro Security Setti



The default security . This prevents macros from g you to enable them.

One of the biggest is

Make sure you are or

On the **Developer** ta v the "Record Macro" button. Select a desired option in Macro Settings category, under the *Macro Settings* header (PC)/ or in **Excel** → **Preference** "Security & Privacy"

Macros: Implementation

Macros are converted to Visual Basic code.

Macros: Implementation

Macros are converted to Visual Basic code.

You can edit macro code and create your own code.

Macros: Implementation

Macros are converted to Visual Basic code.

You can edit macro code and create your own code.

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

Macros: Implementation

Macros are converted to Visual Basic code.

You can edit macro code and create your own code.

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

The code will then open in you *Visual Basic Editor (VBE)*

Macros: Implementation

Macros are converted to Visual Basic code.

You can edit macro code and create your own code.

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

The code will then open in you *Visual Basic Editor (VBE)*

It is best practice to break long tasks into, smaller relevant macros rather than one big macro.

Macros: Implementation

Macros are converted to Visual Basic code.

You can edit macro code and create your own code.

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

The code will then open in you *Visual Basic Editor (VBE)*

It is best practice to break long tasks into, smaller relevant macros rather than one big macro.

Macros can be written for and any other Office application that supports VBA.

Macros: Implementation

Macros are converted to Visual Basic code.

You can edit macro code and create your own code.

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

The code will then open in you *Visual Basic Editor (VBE)*

It is best practice to break long tasks into, smaller relevant macros rather than one big macro.

Macros can be written for and any other Office application that supports VBA.

Visual Basic Editor

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

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.

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 (PC)/ Opt +F11 (Mac).

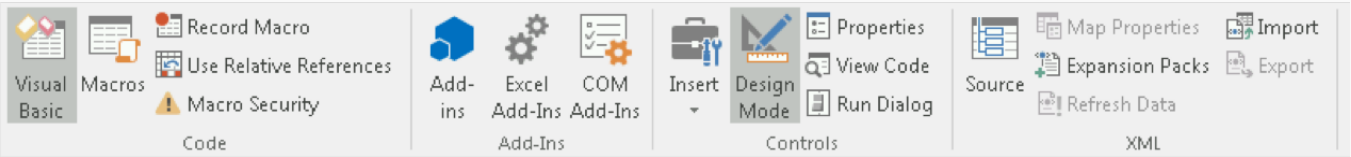
Visual Basic Editor

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

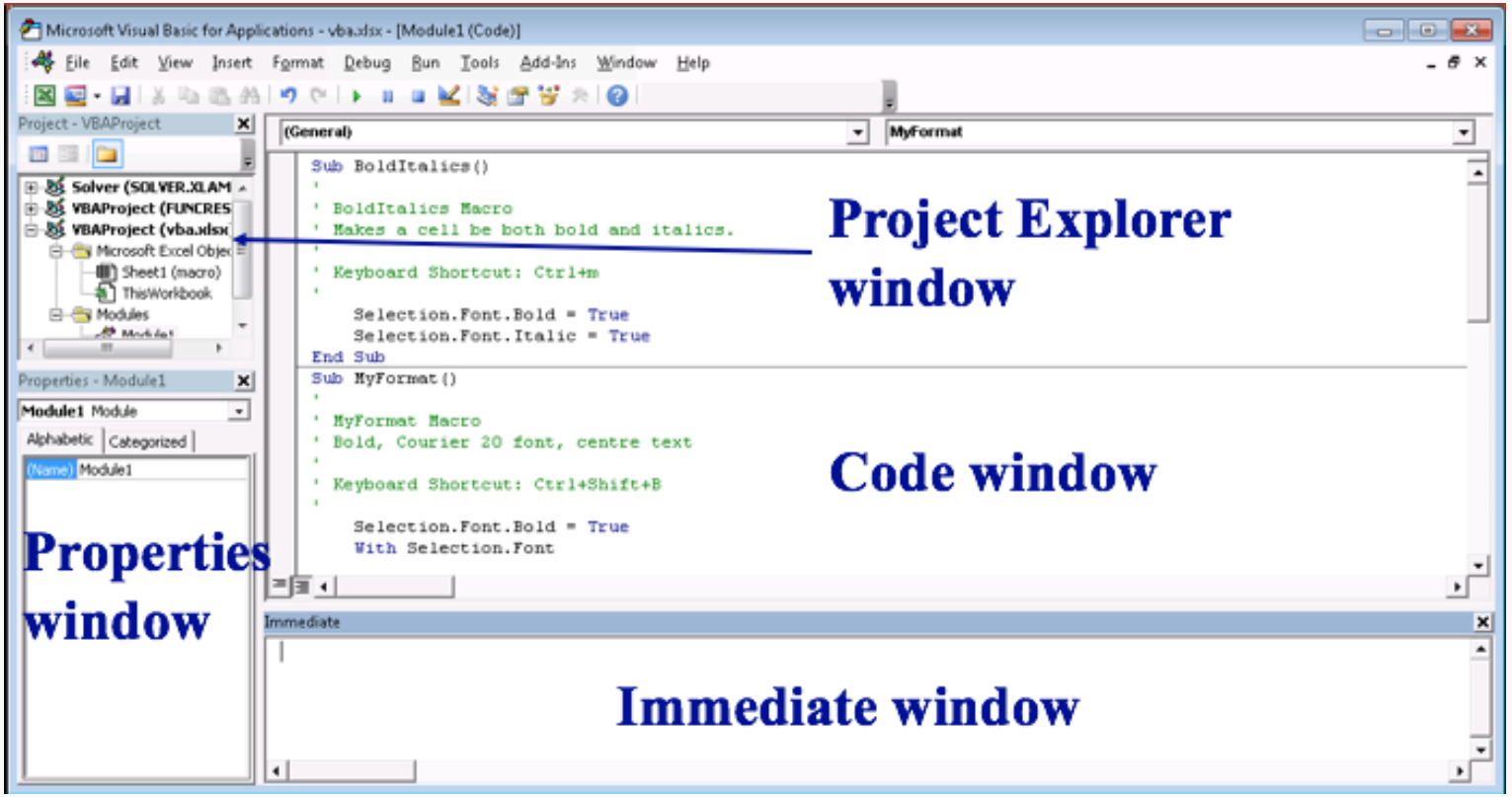
Users can create and edit macros as well as other Visual Basic code with the editor.

To open the VBE, under **Developer** tab → **Visual Basic** or Alt +F11 (PC)/ Opt +F11 (Mac).

There is also a quick button for this in the left hand side of the Developer tab:

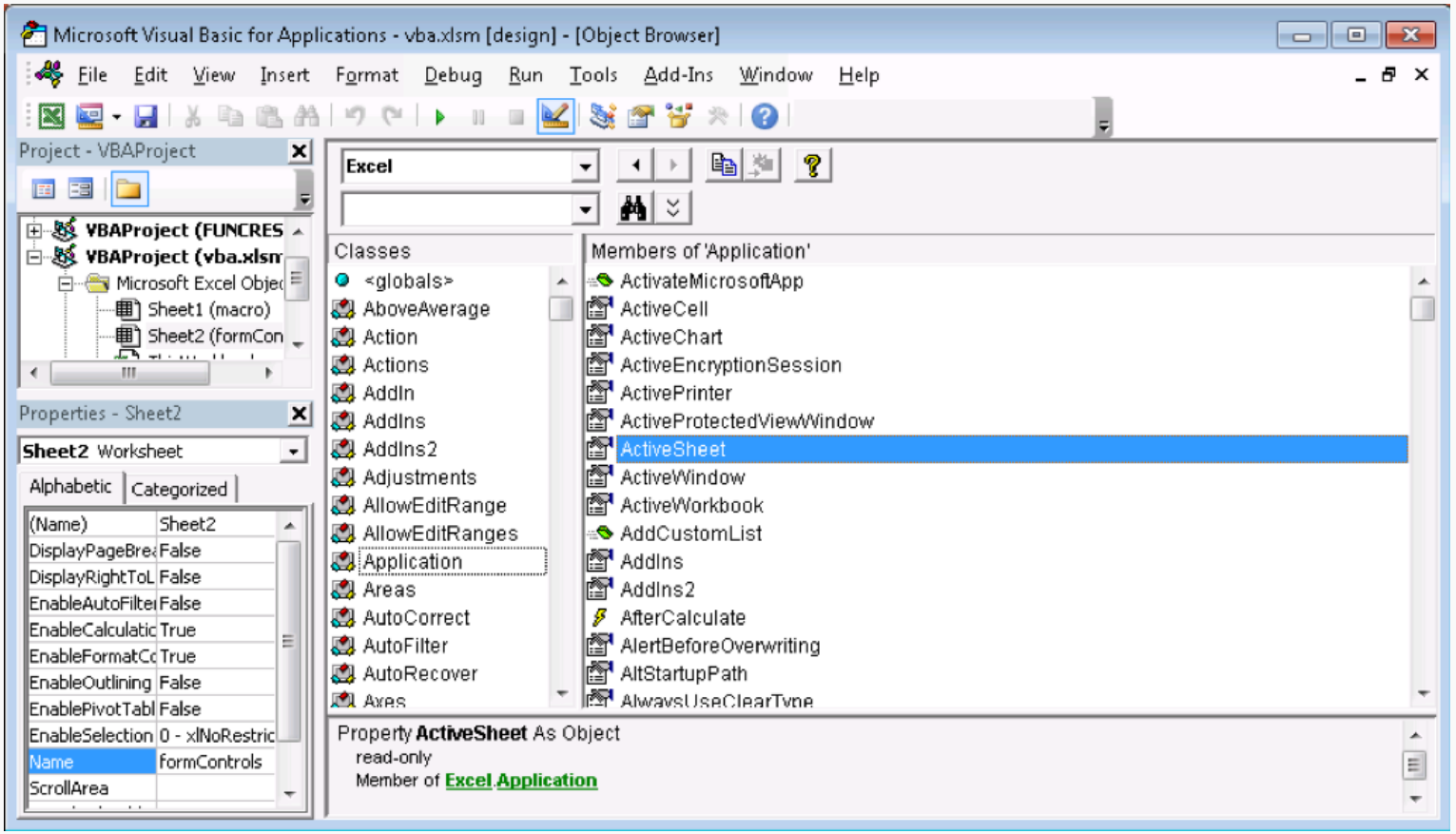


Visual Basic Edior



Object Browser

Object browser allows for exploring objects and methods (the application programming interface (API)) of Excel VBA. Open with F2 or using **View** → **Object Browser** from the menu



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.

Immediate

```
? "Hello world!"
Hello world!
? Range("A2").Value
 1
Range("A2").Value = 10
? Range("A2").Value
 10
```

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 a 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 colour values are numbers from 0 to 255.

Introduction to Programing

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.

Introduction to Programing

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.

Introduction to Programing

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.

Introduction to Programing

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.

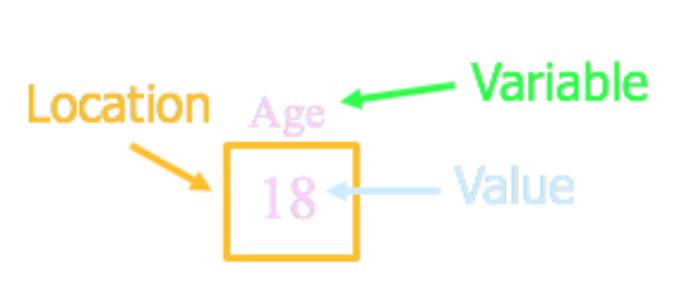
There are numerous languages that programmers can use to specify instructions. Each language has its different features, benefits, and usefulness.

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

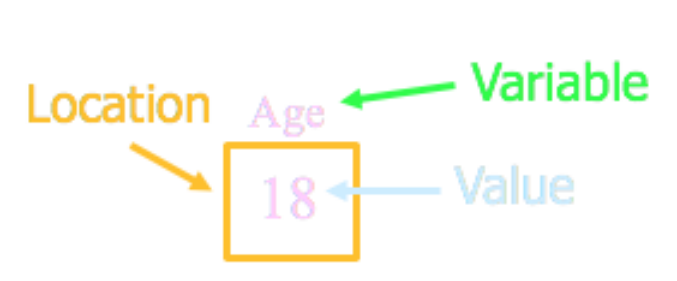
A *variable* is a name that refers to a location that stores a data value.



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

Variables

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

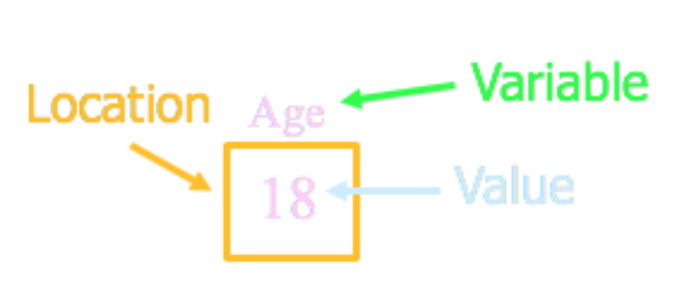


IMPORTANT: The *value* at a location can change using initialization or assignment.

Assignment using an sets the value of a variable.

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.

Assignment using an sets the value of a variable.

Example:

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

Excel Variables

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

Excel Variables

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

Variables increase code efficiency and readability.

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)

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`

The `Dim` keyword *declares* the variable (named `num`) as an integer.

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`

The `Dim` keyword *declares* the variable (named `num`) as an integer.

Excel Variables

There are number of different declaration keywords we can use (eg. Const, Public, Private, Public Const, Set . . .)

Excel Variables

There are number of different declaration keywords we can use (eg. Const, Public, Private, Public Const, Set . . .)

There are also different variable types (eg. Boolean, Single, Double, String, Variant, . . .)

Excel Variables

There are number of different declaration keywords we can use (eg. Const, Public, Private, Public Const, Set . . .)

There are also different variable types (eg. Boolean, Single, Double, String, Variant, . . .)

Example:

- ```
  Dim Class_List(1 To 150) As String
  Class_List(1) = "Irene Vrbik"
  ```

The `(1 To 150)` tells Excel that `Class_List` in an array of 150 String variables.
  ```

Excel Variables

There are number of different declaration keywords we can use (eg. Const, Public, Private, Public Const, Set . . .)

There are also different variable types (eg. Boolean, Single, Double, String, Variant, . . .)

Example:

- ```
  Dim Class_List(1 To 150) As String
  Class_List(1) = "Irene Vrbik"
  ```

The `(1 To 150)` tells Excel that `Class_List` in an array of 150 String variables.

See Part 2 of this (click me) tutorial for more on this.
  ```

Although VBA does not require us to declare variables, it is good coding practice to do so.

Although VBA does not require us to declare variables, it is good coding practice to do so.

By default, all undeclared variable in Excel will have the Variant type (which can hold Dates, Floating Point Numbers or Strings of Characters)

Although VBA does not require us to declare variables, it is good coding practice to do so.

By default, all undeclared variable in Excel will have the Variant type (which can hold Dates, Floating Point Numbers or Strings of Characters)

By default, arrays are indexed starting from 0.

Although VBA does not require us to declare variables, it is good coding practice to do so.

By default, all undeclared variable in Excel will have the Variant type (which can hold Dates, Floating Point Numbers or Strings of Characters)

By default, arrays are indexed starting from 0.

Hence `Dim Class_List(150) As String` would be recognized as an array of 150 variables, which are indexed from 0 to 149.

Collections

Collections are variables that store multiple data items.

Collections

Collections are variables that store multiple data items.

Data items can either be indexed (selected) by name or number.

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).

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).

Another example is 'Rows' which is a collection object containing all the rows of a Worksheet.

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

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 F) 5 G) 6

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 F) 5 G) 6

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 F) 5 G) 6

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 F) 5 G) 6

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 F) 5 G) 6

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* F) 5 G) 6

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 F) 5 G) 6

Decisions

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

Decisions

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

Logical operators: `AND, OR, NOT`

Decisions

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

Logical operators: `AND, OR, NOT`

Example decision syntax:

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

Question: Decisions

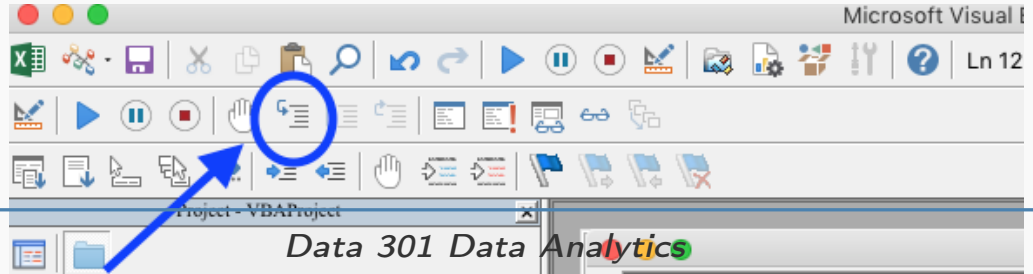
Question

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

To step through you code on a PC press F8
On a Mac Press this button:



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

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

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

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

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

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

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.

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
```

MsgBox function in VBA displays a message in a window and waits for click on a button.

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
```

MsgBox function in VBA displays a message in a window and waits for click on a button.

Debug.Print is telling VBA to print that information in the Immediate Window.

Loops and Iteration

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

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*.

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. Example:

```
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

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

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

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

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

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

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

Try it: loops

Homework:

Create a method called `TryFor` that prints the numbers 1 to 20.

Challenging variants:

a) Print the numbers from 10 down to 1.

b) 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.

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.

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).

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).

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
```

NOTICE: The value returned must be assigned to a variable with the same name as the Function (this variable does not need to be declared)

Sub

A Sub procedure (also know as a subroutine) is set of commands to perform a certain task.

Sub

A Sub procedure (also know as a subroutine) is set of commands to perform a certain task.

Unlike a UDF, a Sub procedure does not return a result, nor can they be typed directly into a Worksheet in Excel

Sub

A Sub procedure (also know as a subroutine) is set of commands to perform a certain task.

Unlike a UDF, a Sub procedure does not return a result, nor can they be typed directly into a Worksheet in Excel

- eg. the function SUM() returns a result of summing a group of numbers
- the Sub might carry out actions like formatting a set of cells

Most of the macros you write in VBA are Sub procedures.

Sub

A Sub procedure (also know as a subroutine) is set of commands to perform a certain task.

Unlike a UDF, a Sub procedure does not return a result, nor can they be typed directly into a Worksheet in Excel

- eg. the function SUM() returns a result of summing a group of numbers
- the Sub might carry out actions like formatting a set of cells

Most of the macros you write in VBA are Sub procedures.

Both VBA Sub procedures and UDF procedures can take arguments but they are not essential

UDF Example: Sum Cells by Background Colour

```
' 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
```

Click here for more on colour indexing.

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 objects, classes, methods, and properties. This organization makes it easier to understand and construct large programs.

Advanced: Object-Oriented Programming

Object-oriented programming structures code as objects, 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.

Advanced: Object-Oriented Programming

Object-oriented programming structures code as objects, 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).

Advanced: Object-Oriented Programming

Object-oriented programming structures code as objects, 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.

Advanced: Object-Oriented Programming

Object-oriented programming structures code as objects, 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.

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.

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.

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 Objects

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

Example:

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

Excel Objects

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

Example:

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

Object Methods Methods perform an action.

Example:

- `Worksheets("macro").Activate`

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

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

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

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

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

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

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.

Forms and Input Controls

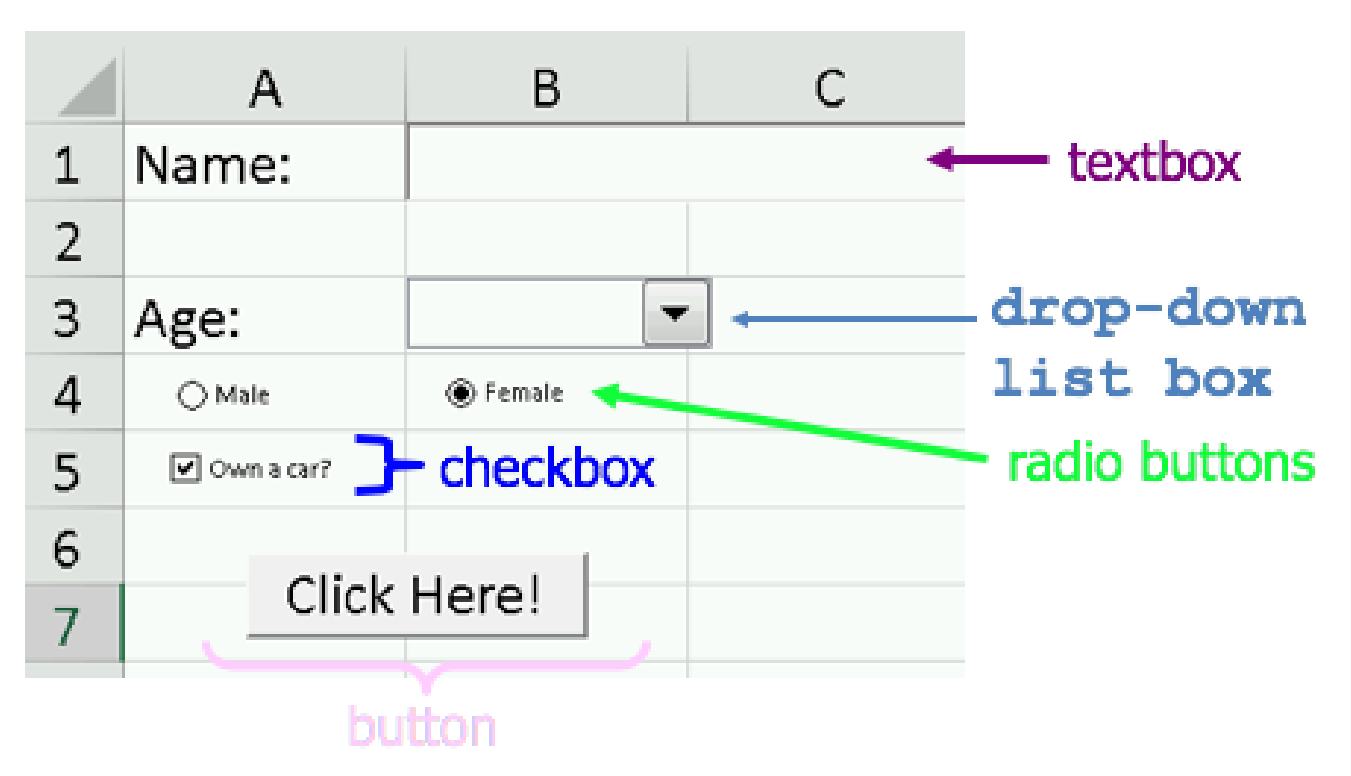
Excel allows the creation of forms with controls for a better interface.

There are two types of controls in Excel:

1. Form controls – default
2. 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

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.

Conclusion

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

Macros record a set of actions so they can be easily executed again. Be aware of security risks when using macros.

Conclusion

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

Macros 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.

Conclusion

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

Macros 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

Objectives (2)

- Define: variable, value, location
- 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

Questions