

Visual Basic for Applications



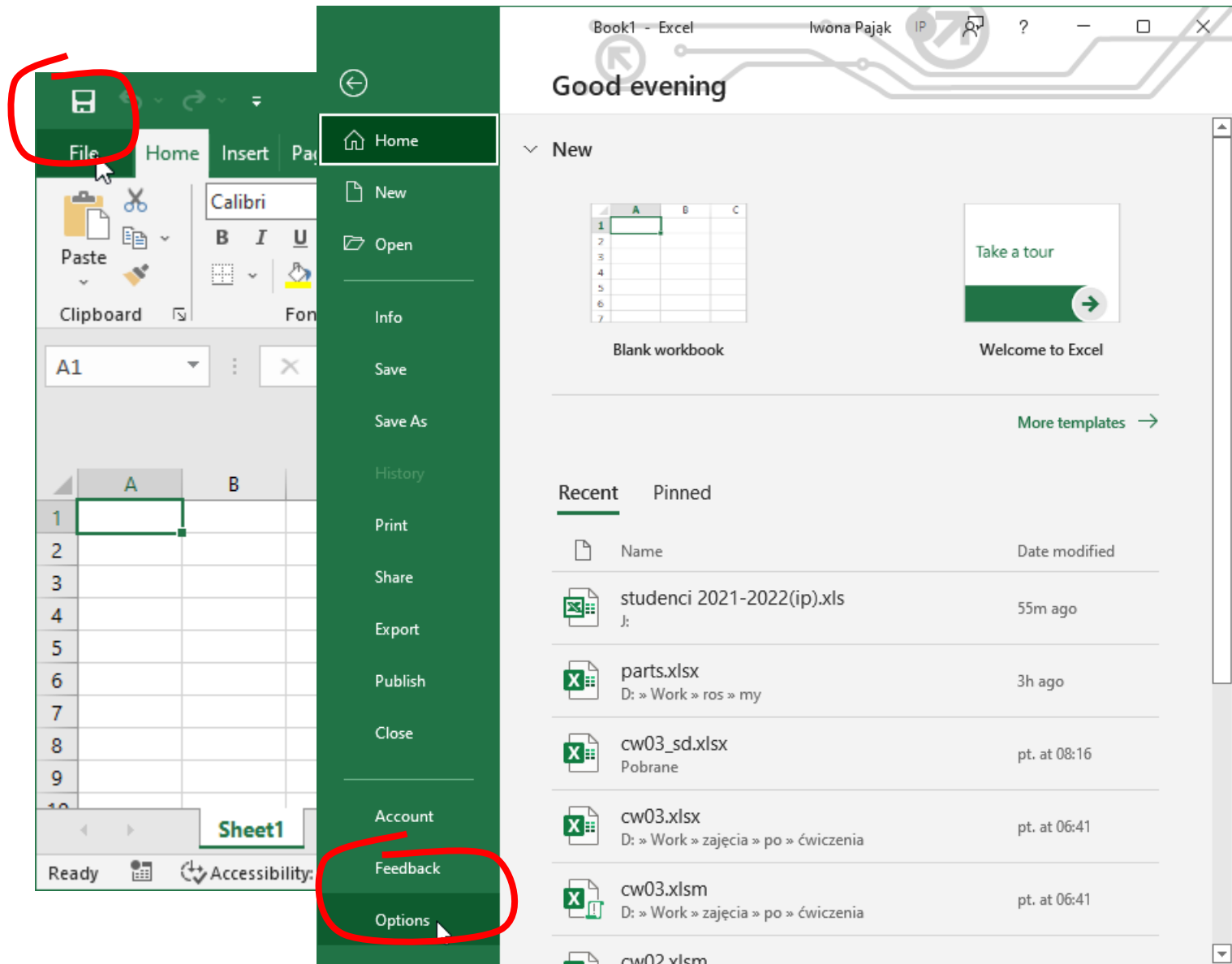
Introduction to programming
VBA Excel

Materials
<http://staff.uz.zgora.pl/ipajak>
<http://staff.uz.zgora.pl/gpajak>

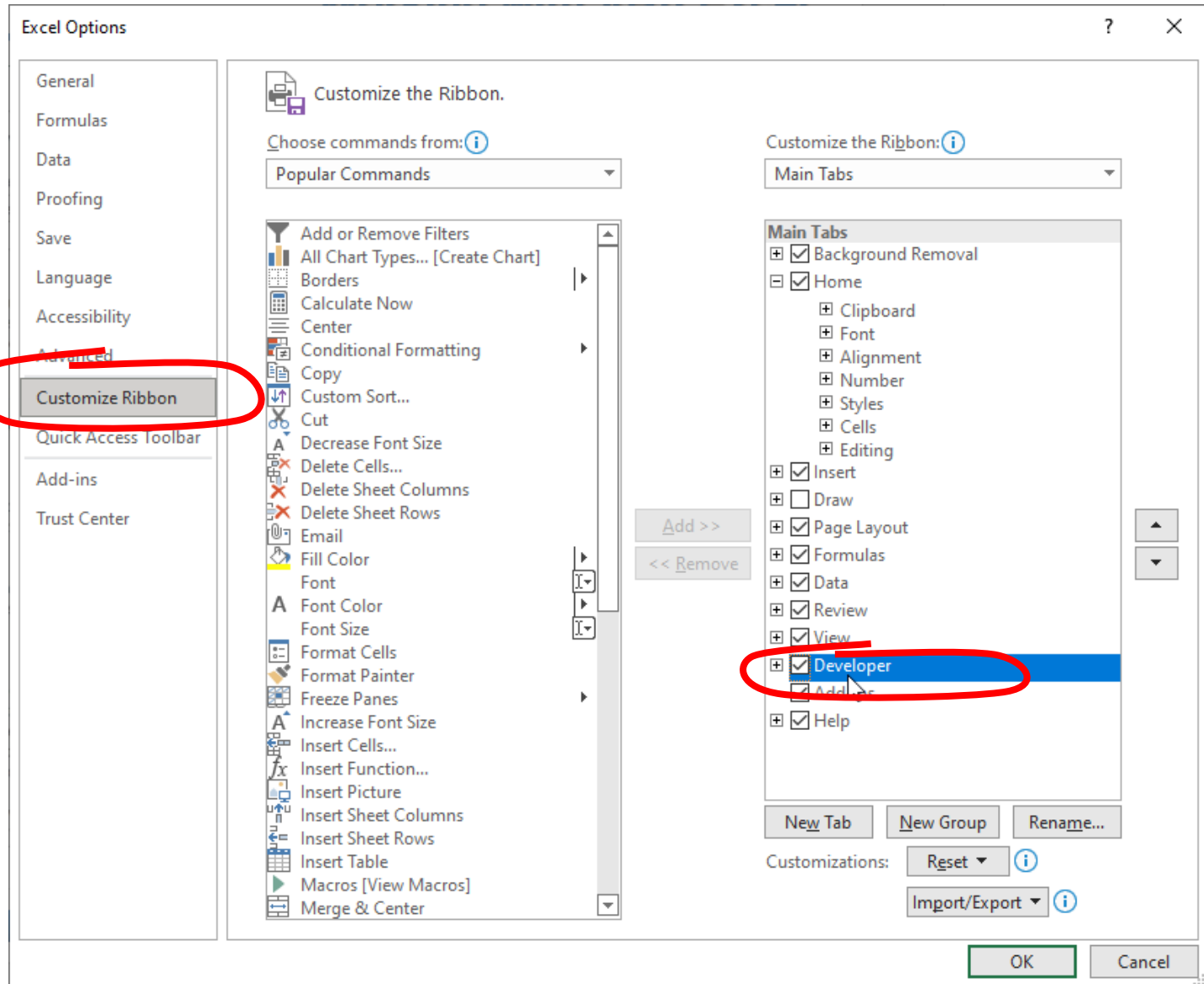


Developer Ribbon

Working with VBA Excel



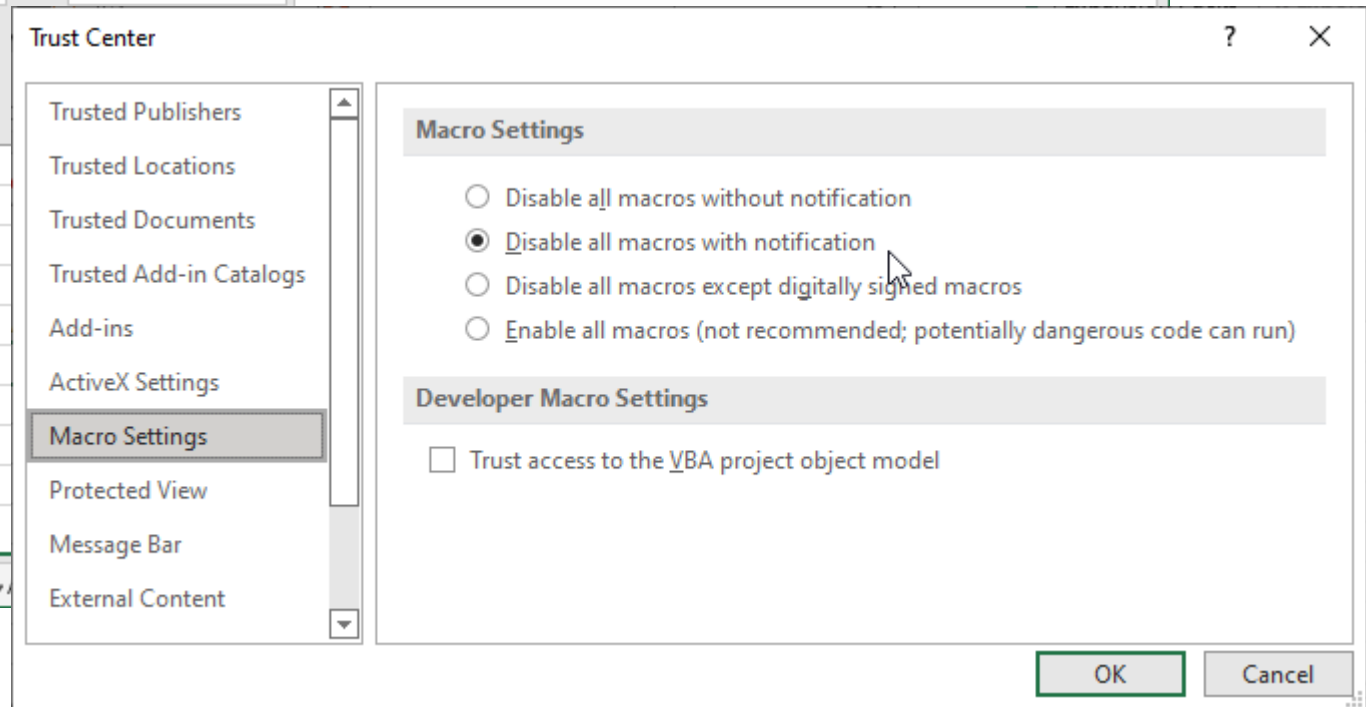
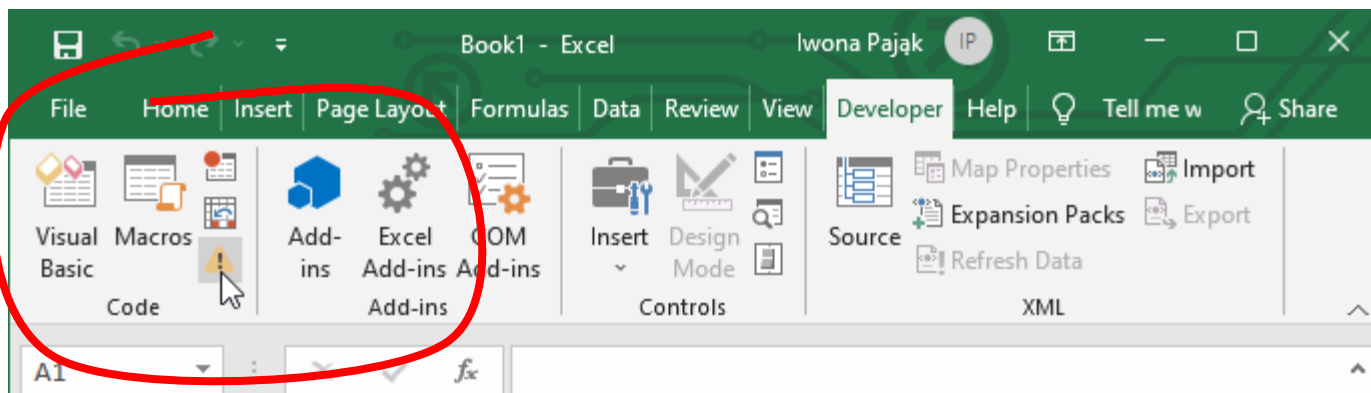
Working with VBA Excel





Safety

Working with VBA Excel



Macro List1

First macro – macro List1

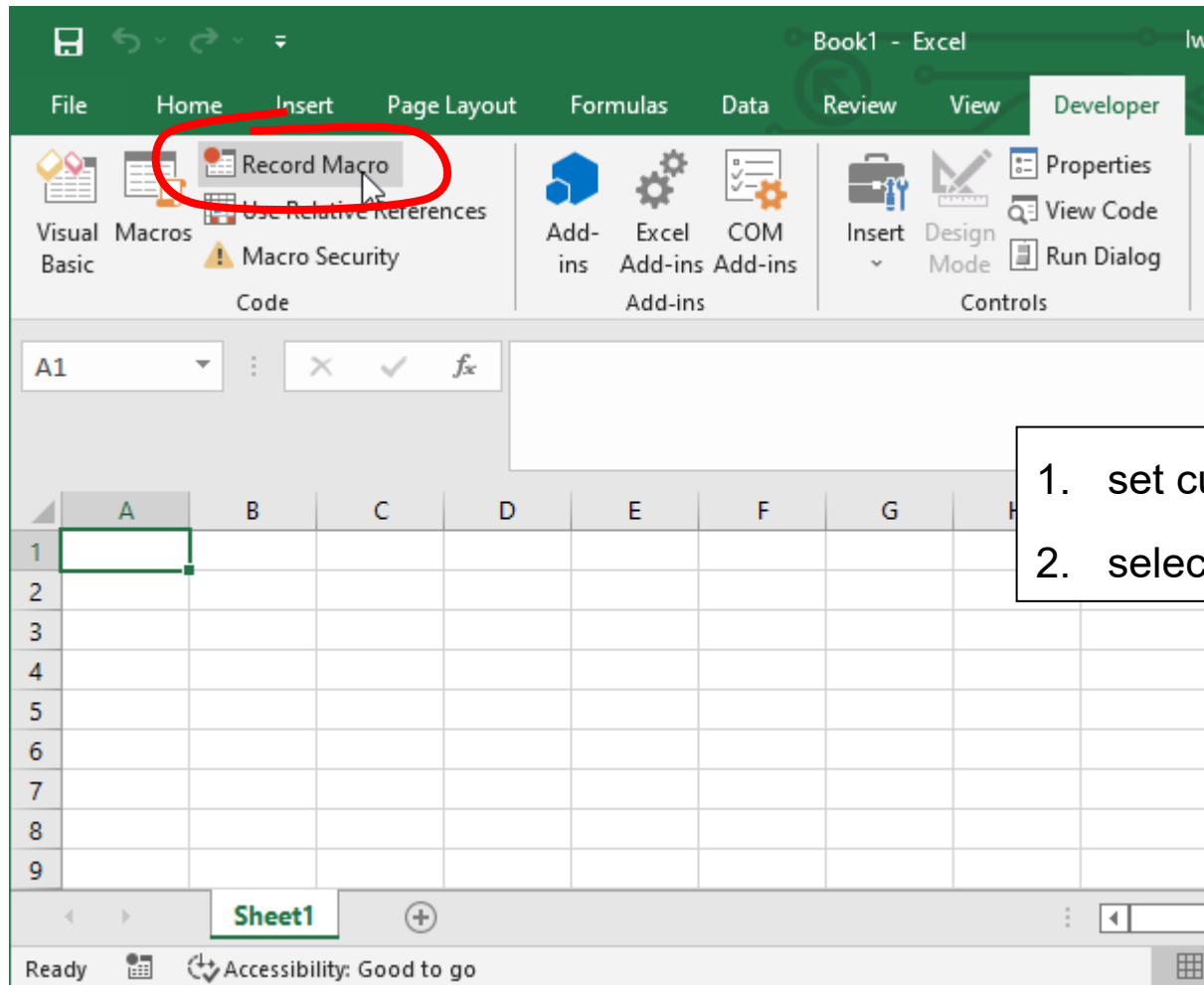
Assumptions

macro writes down:

- digit 1 in cell A1
- digit 2 in cell A2
- digit 3 in cell A3
- digit 4 in cell A4

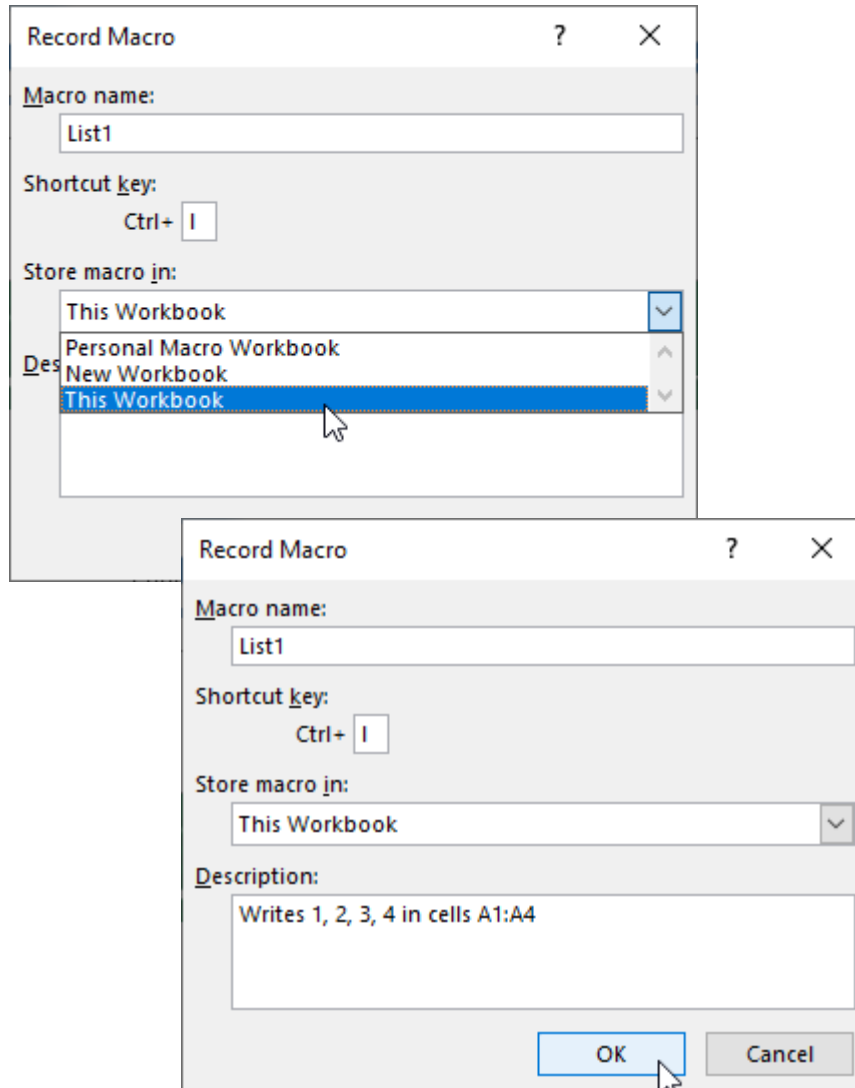
	A	B
1	1	
2	2	
3	3	
4	4	
5		
6		

Recording macro – macro List1



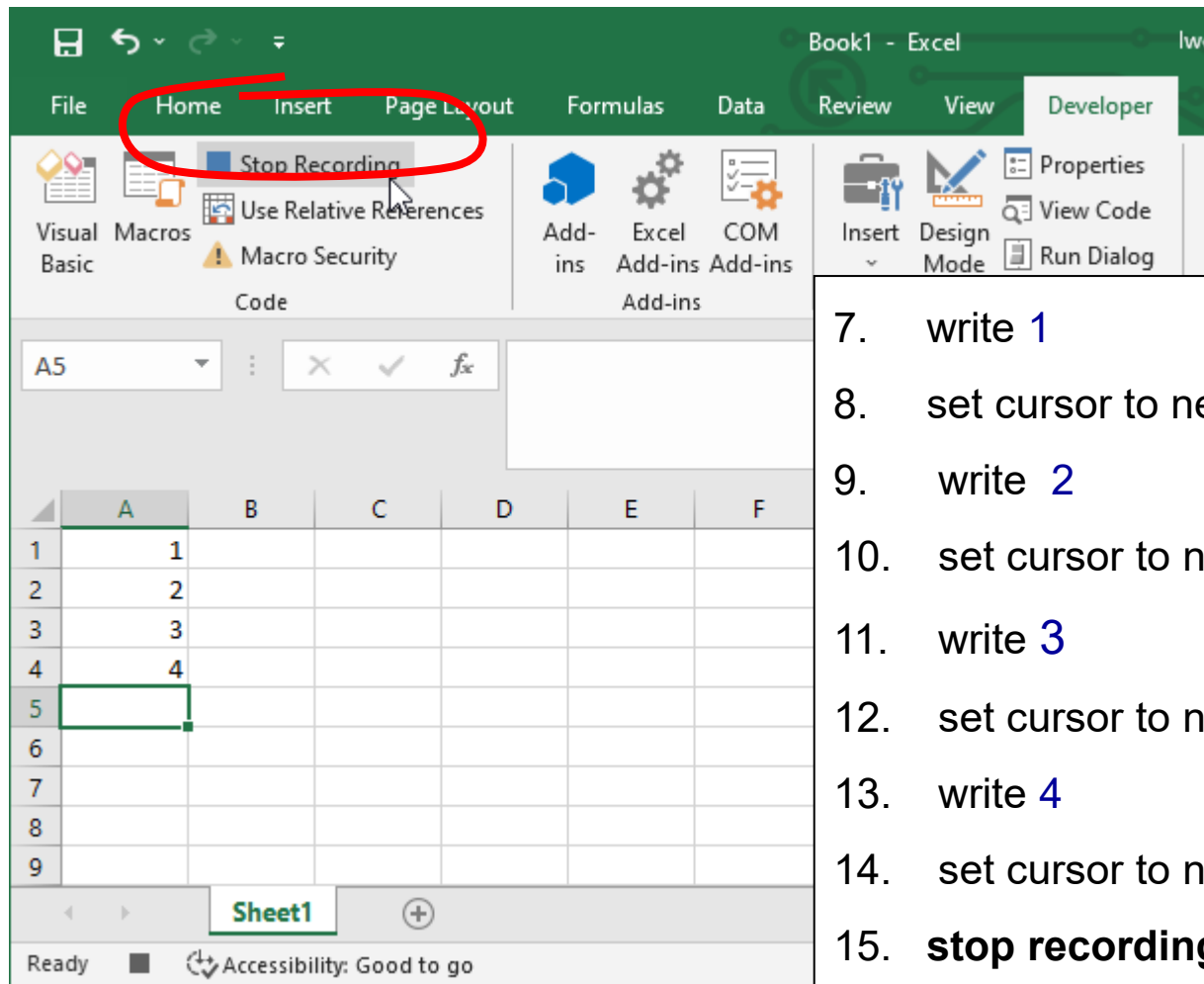
1. set cursor in cell A1
2. select **Record Macro**

Recording macro – macro List1



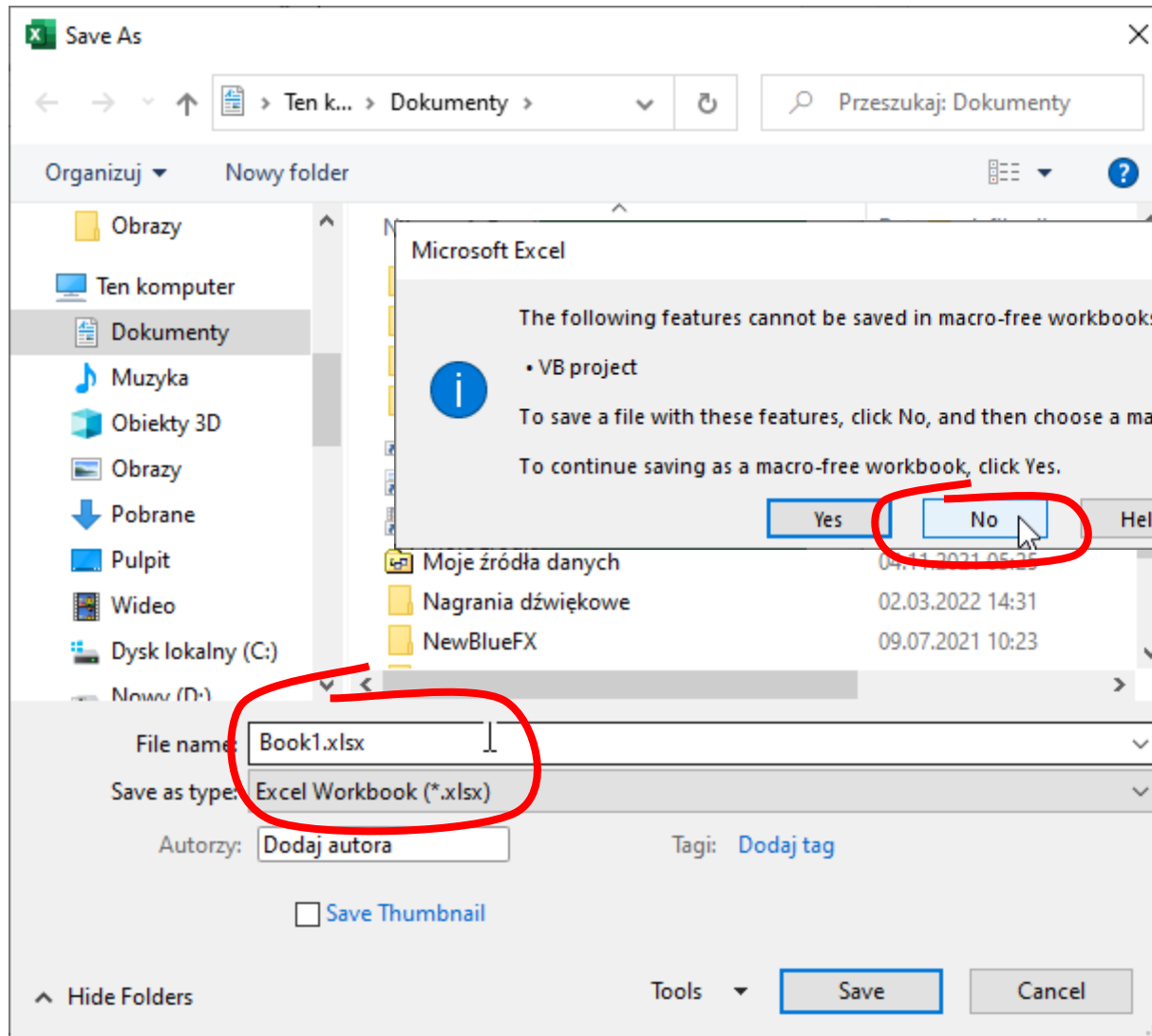
3. set macro name
here: List1
4. optionally set shortcut key
here: Ctrl+I
5. set where macro will be stored
here: This workbook
6. optionally add description
here: Writes ...

Recording macro – macro List1

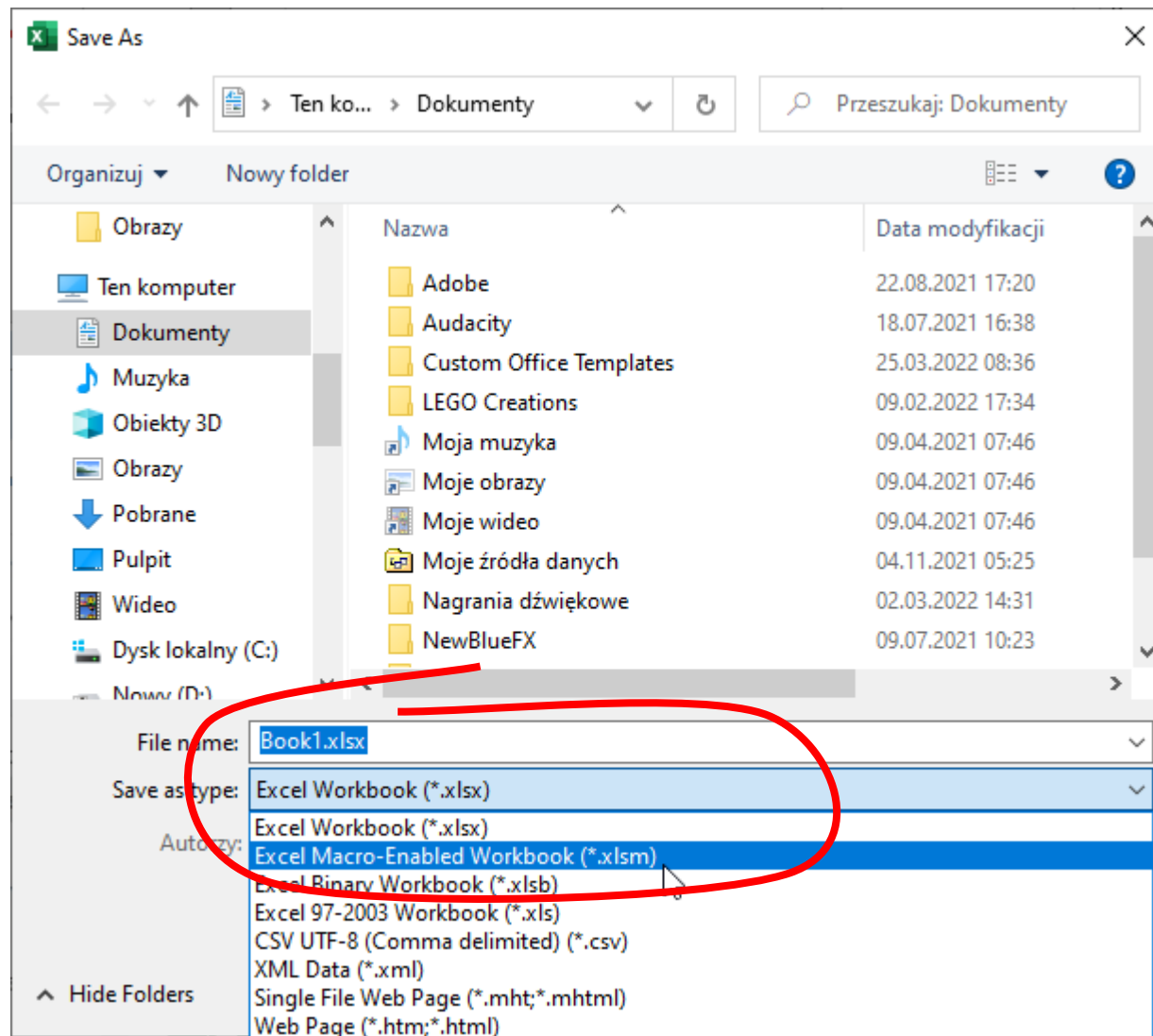


7. write 1
8. set cursor to next cell
9. write 2
10. set cursor to next cell
11. write 3
12. set cursor to next cell
13. write 4
14. set cursor to next cell
15. **stop recording**

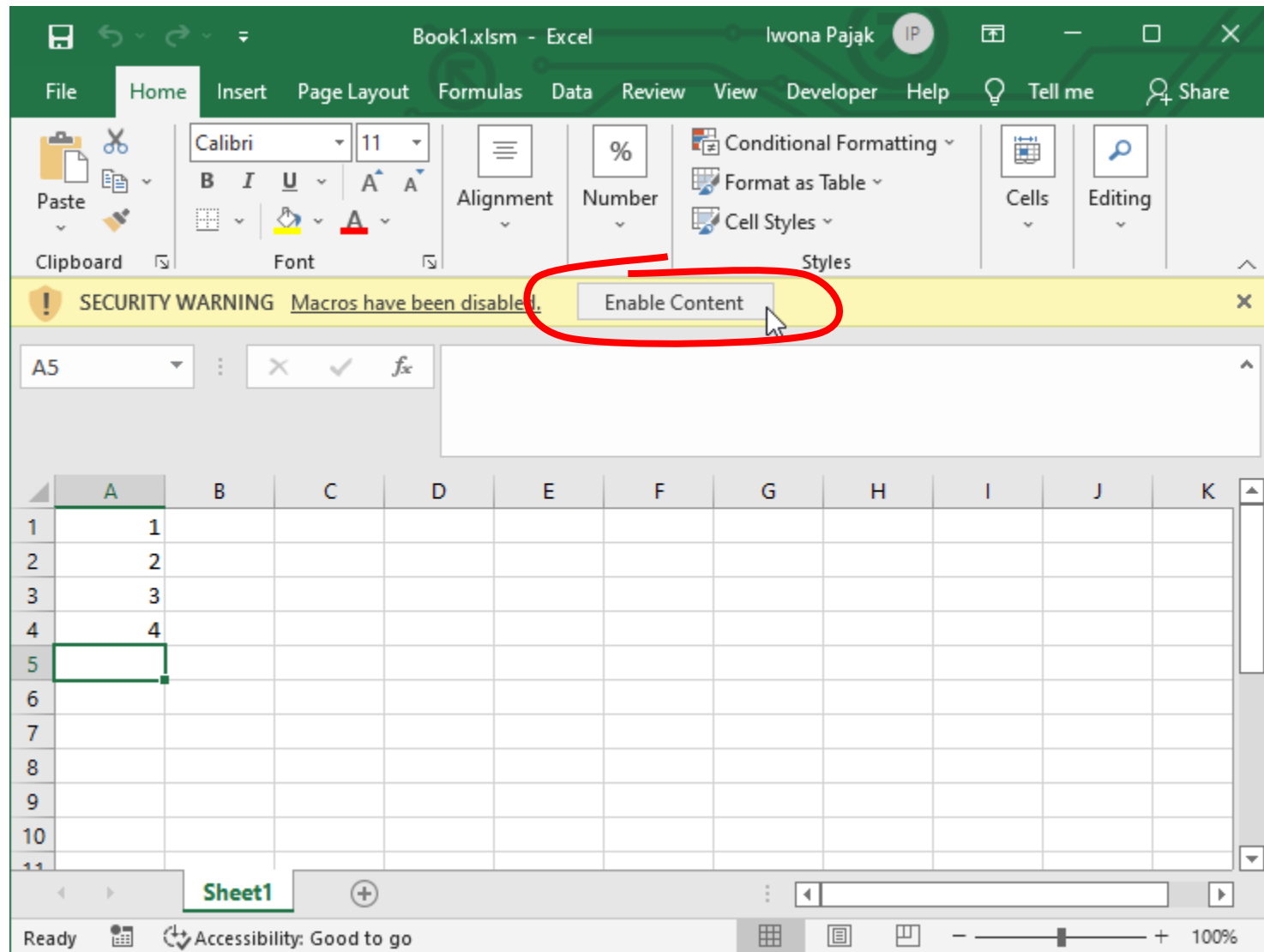
Saving workbook



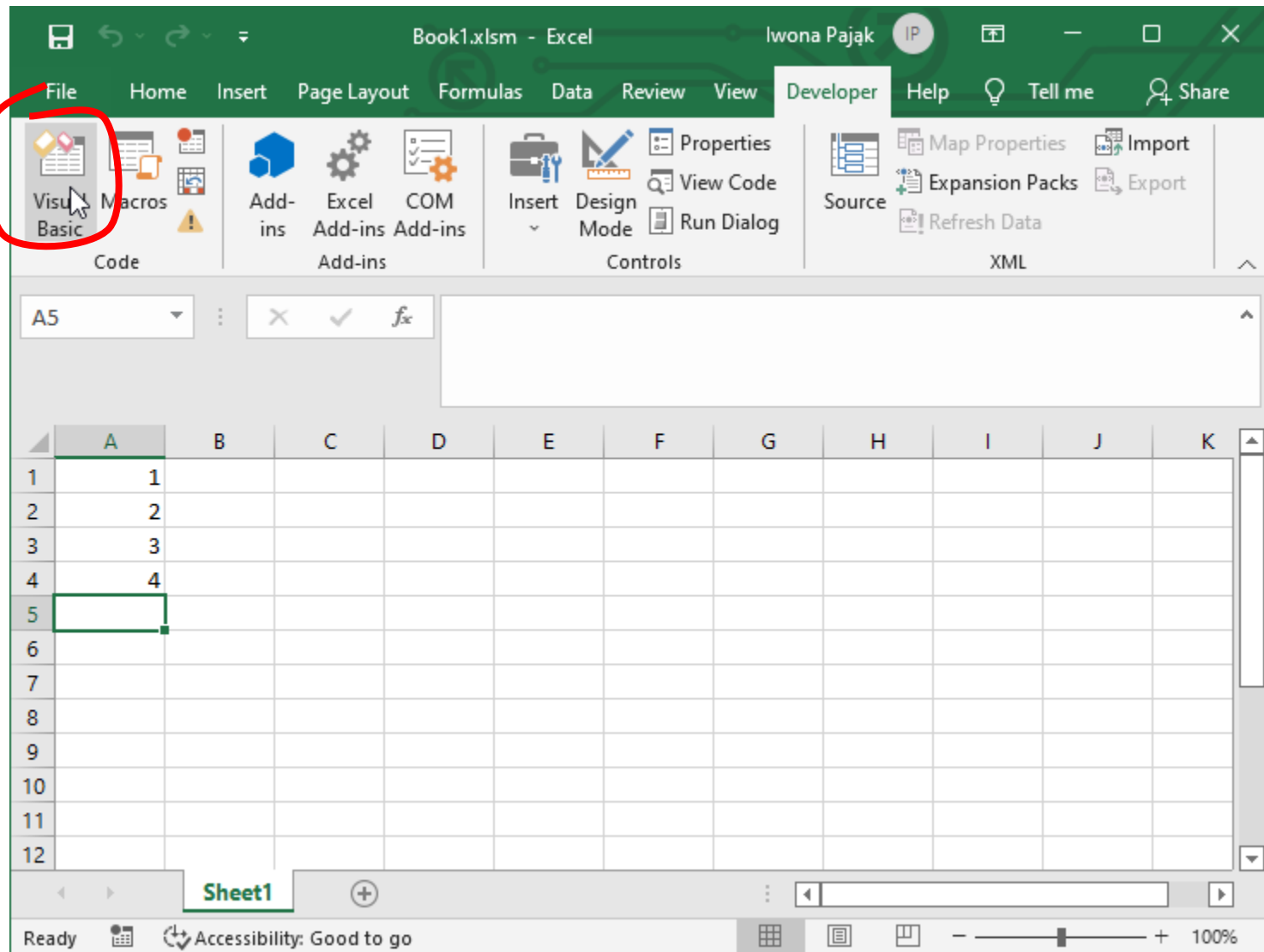
Saving workbook



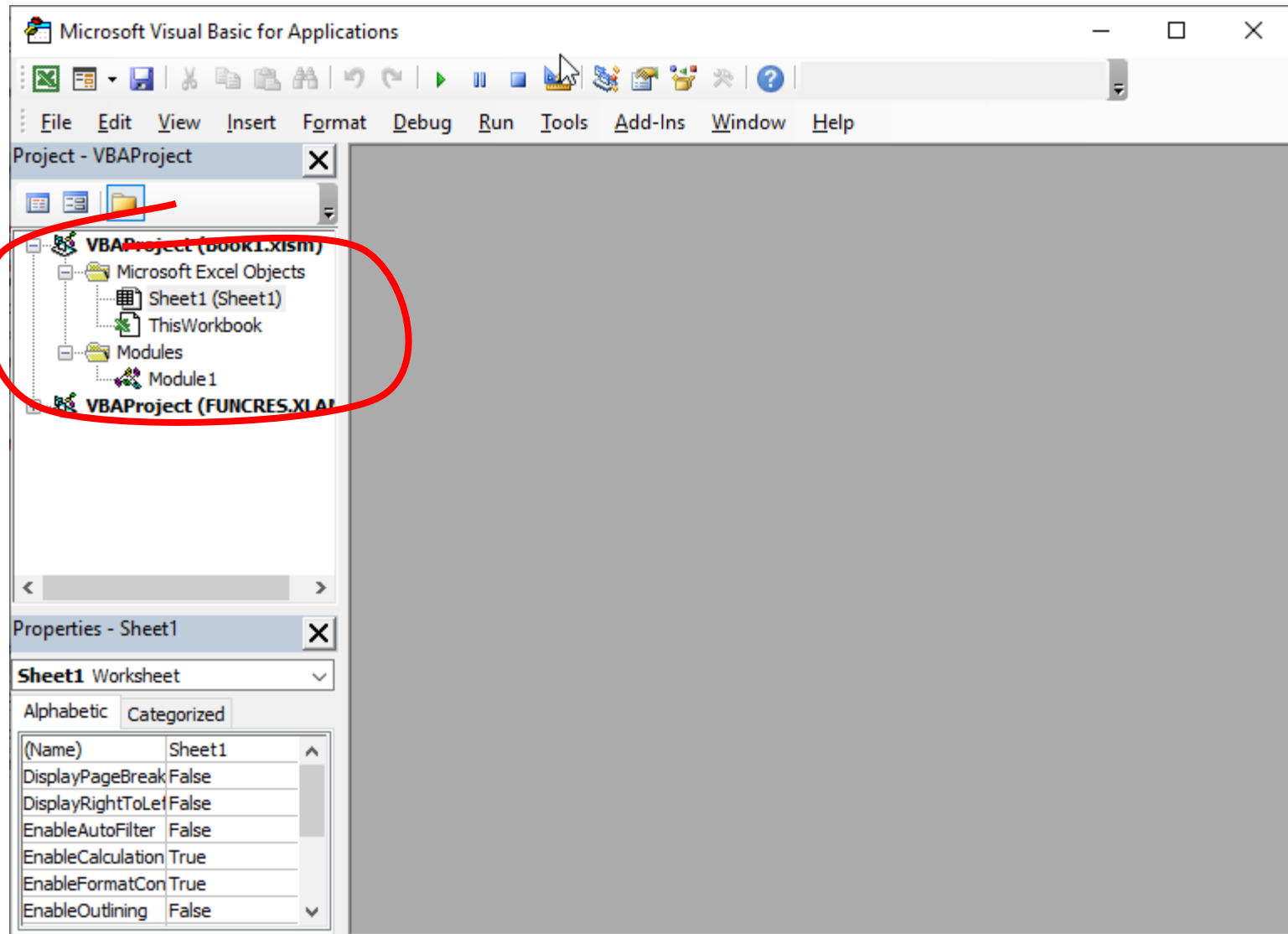
Opening workbook



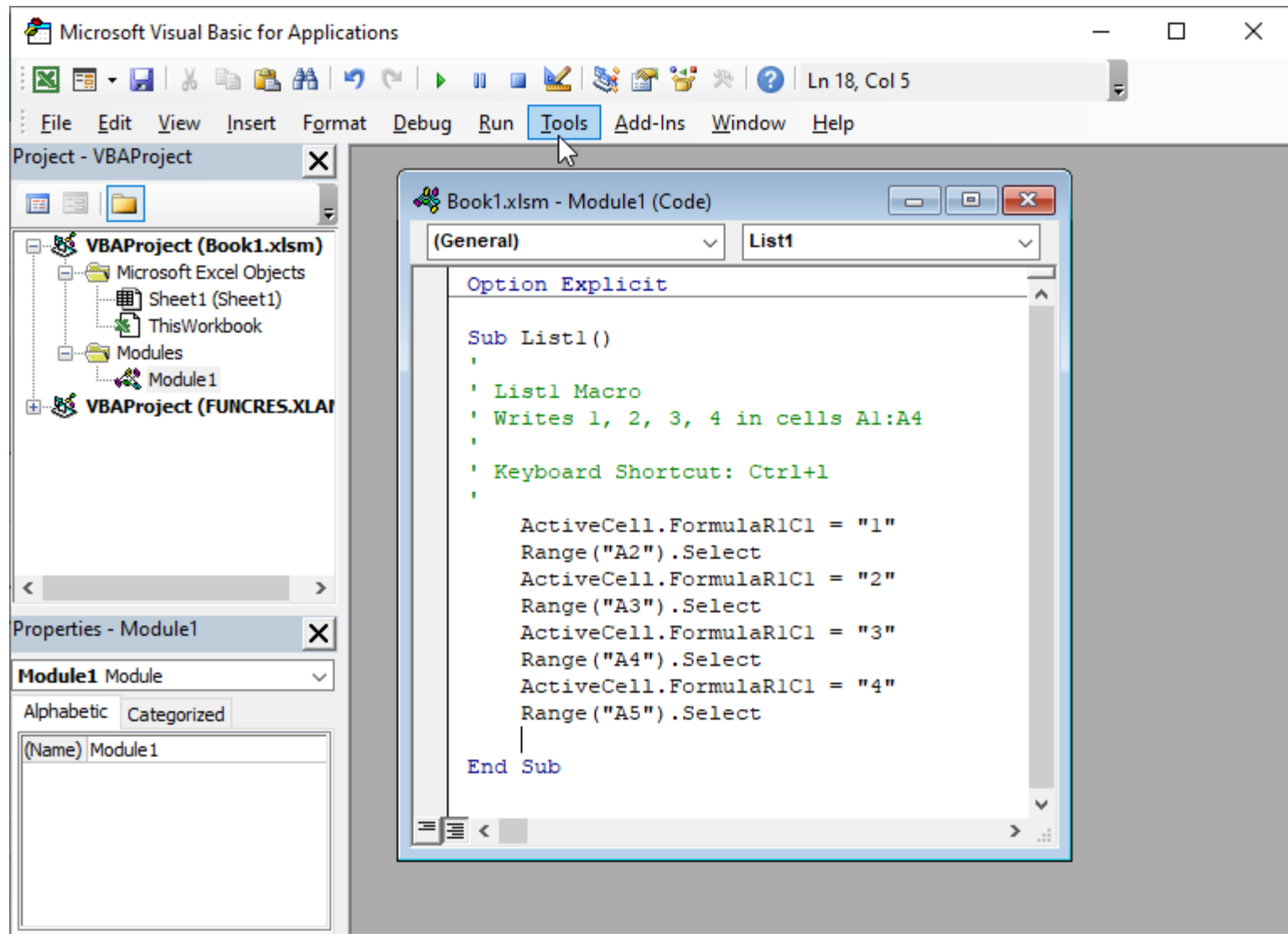
Editing macro – macro List1



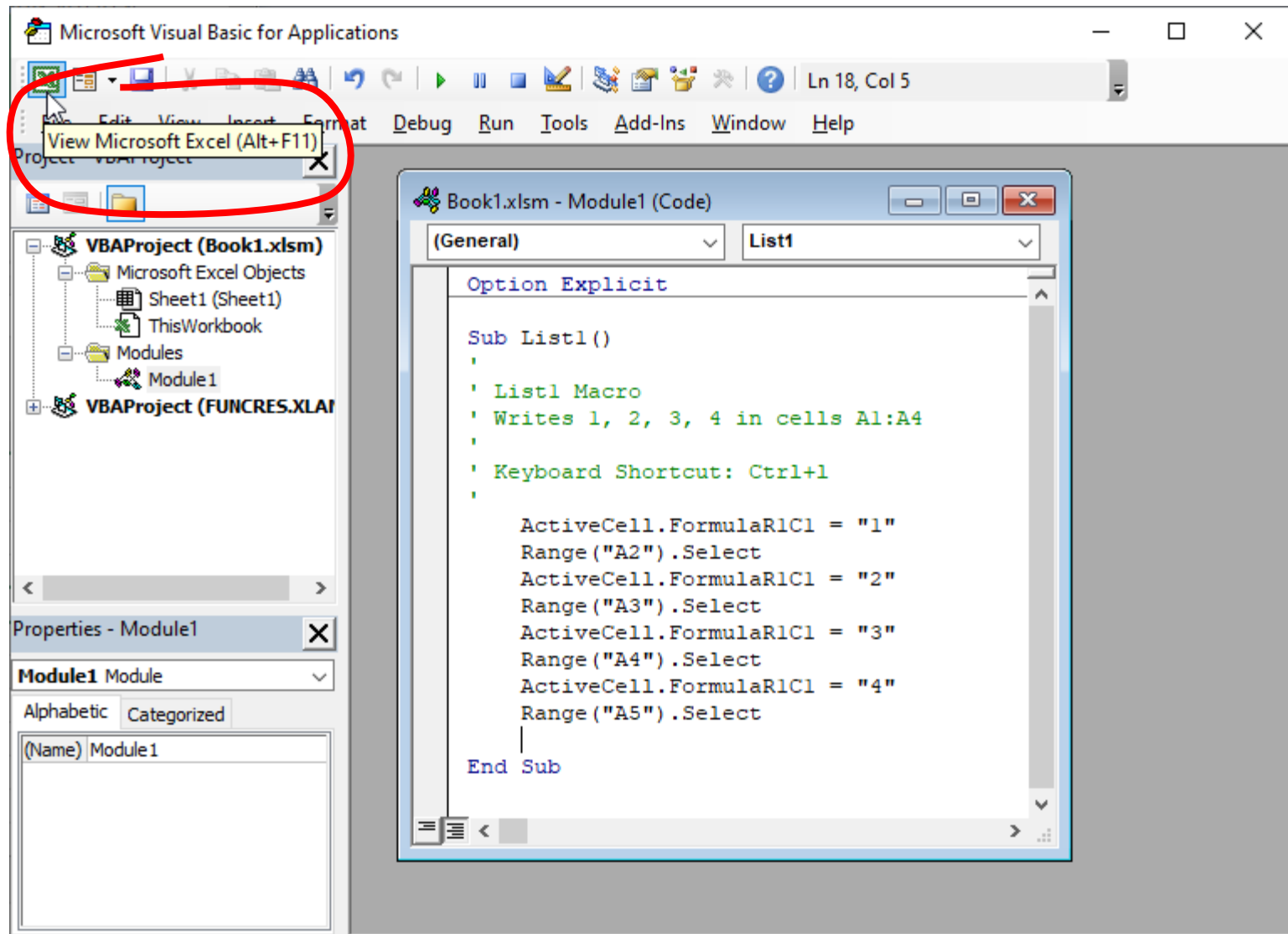
Editing macro – macro List1



Editing macro – macro List1



Editing macro – macro List1



Running macro – macro List1 (method 1)

Book1.xlsm - Excel Iwona Pajak

File Home Insert Page Layout Formulas Data Review View Developer Help Tell me Share

Visual Basic Macros Code

Macro

Macro name:

List1

List1

Run

Step Into

Edit

Create

Delete

Options...

Macros in: All Open Workbooks

Description

Writes 1, 2, 3, 4 in cells A1:A4

Cancel

1. select **Macros**

2. choose macro **List1**

3. select **Run**

Note! strange result

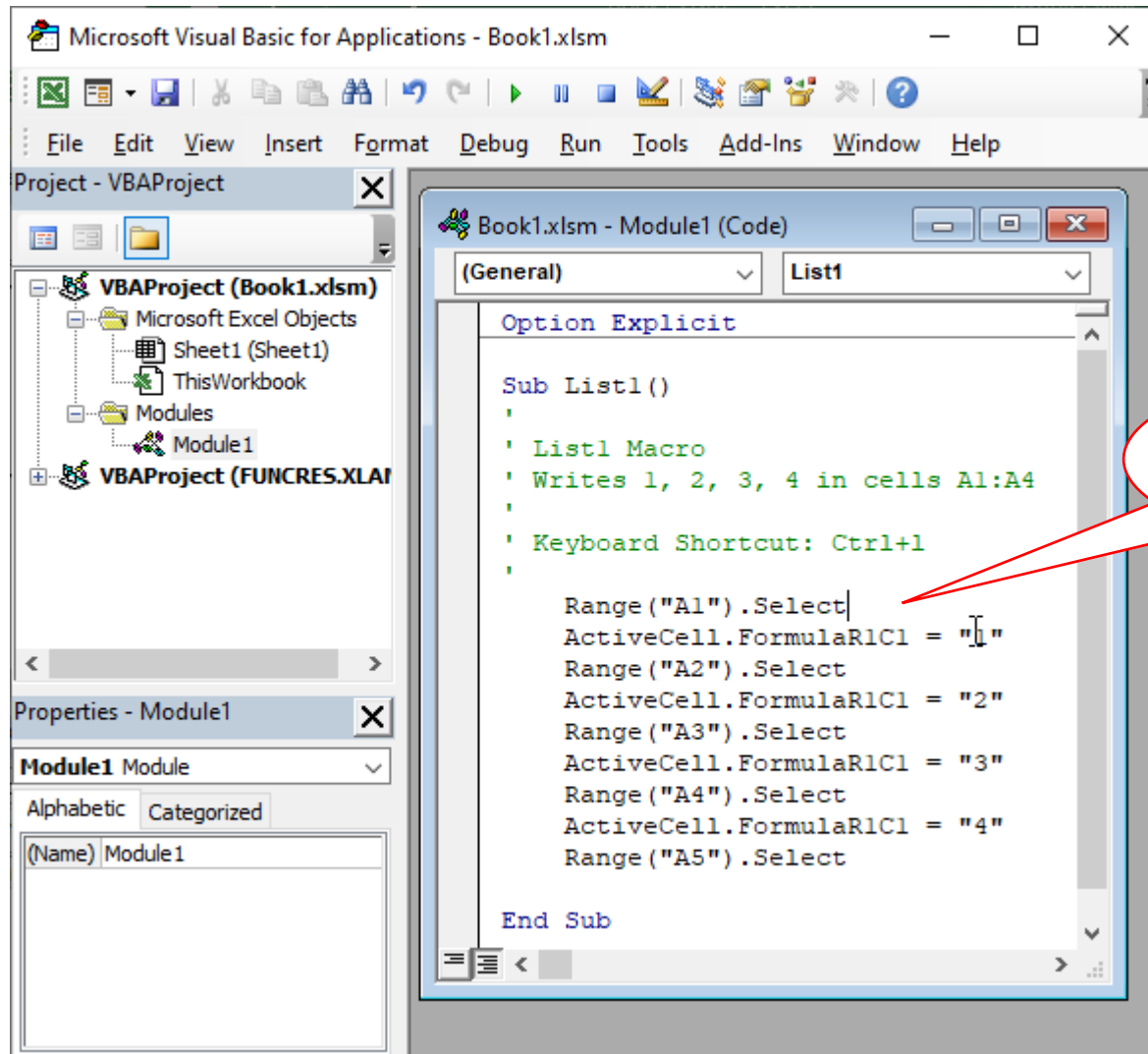
during recording cell A1 was not selected

	A	B
1	1	
2	2	
3	3	
4	4	
5		
6		
7		
8		
9		
10		
11		
12		

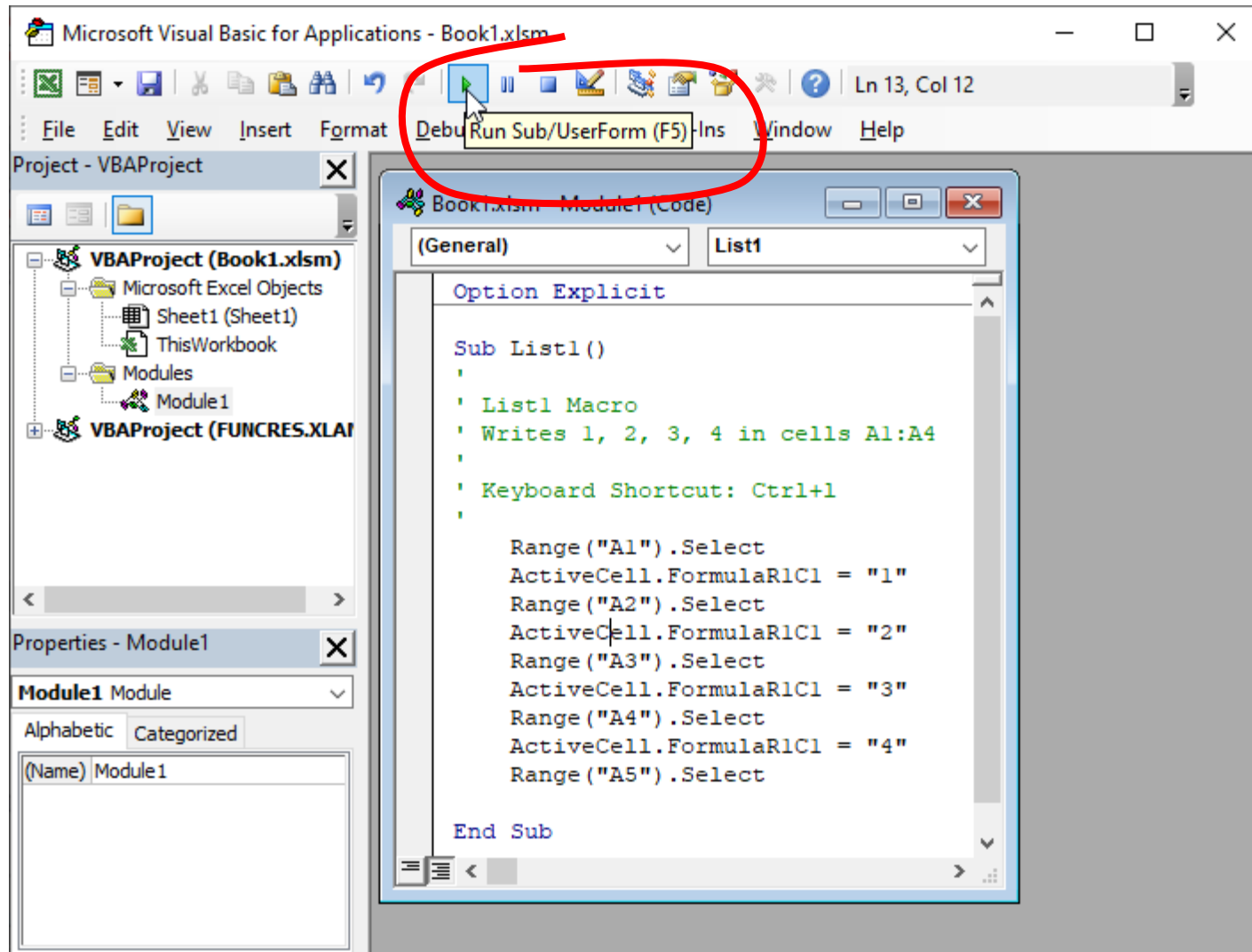
Sheet1

Ready Accessibility: Good to go

Editing macro – macro List1

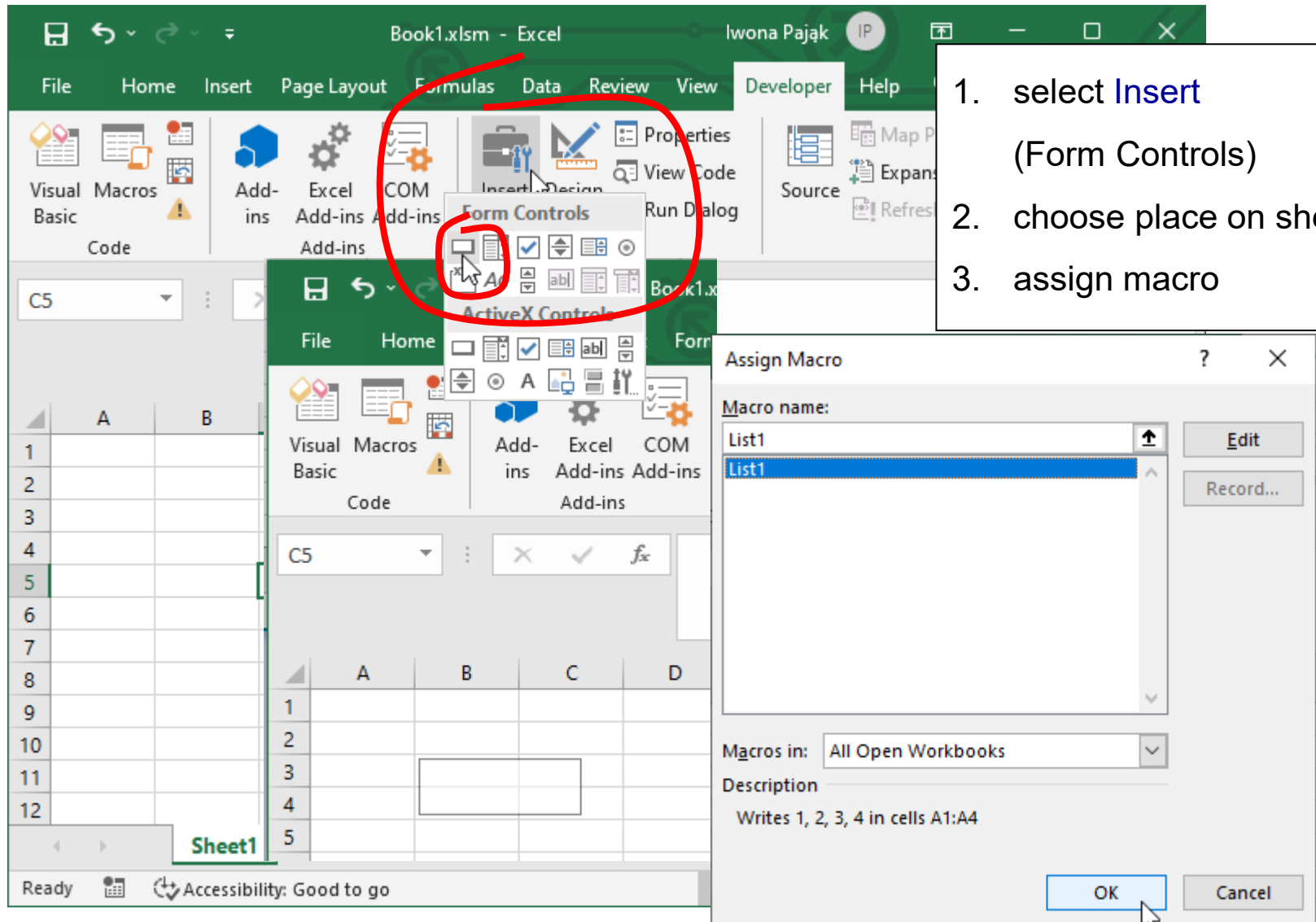


Running macro – macro List1 (method 2)



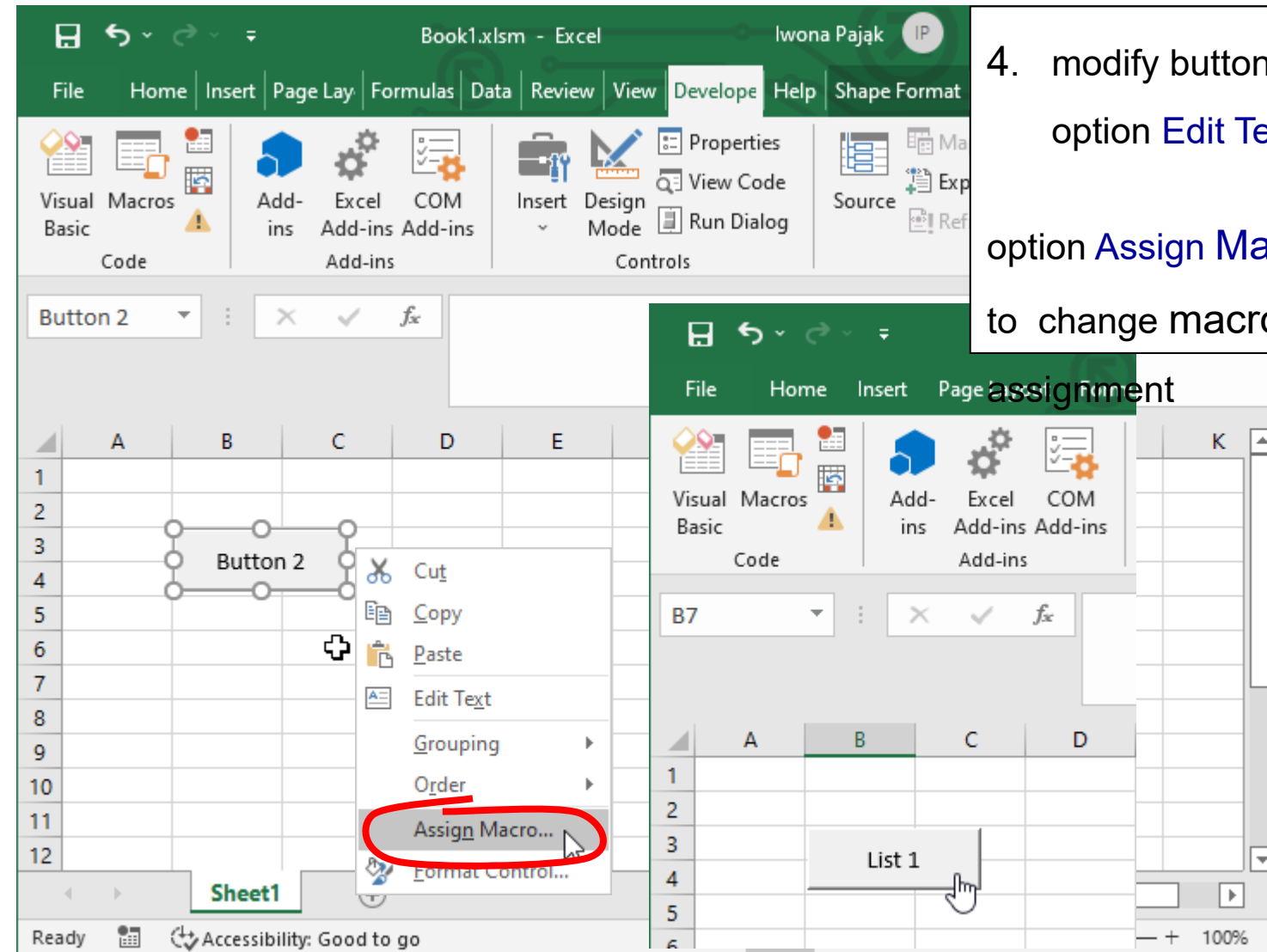
Running macro – macro List1 (method 3)

1. select **Insert** (Form Controls)
2. choose place on sheet
3. assign macro



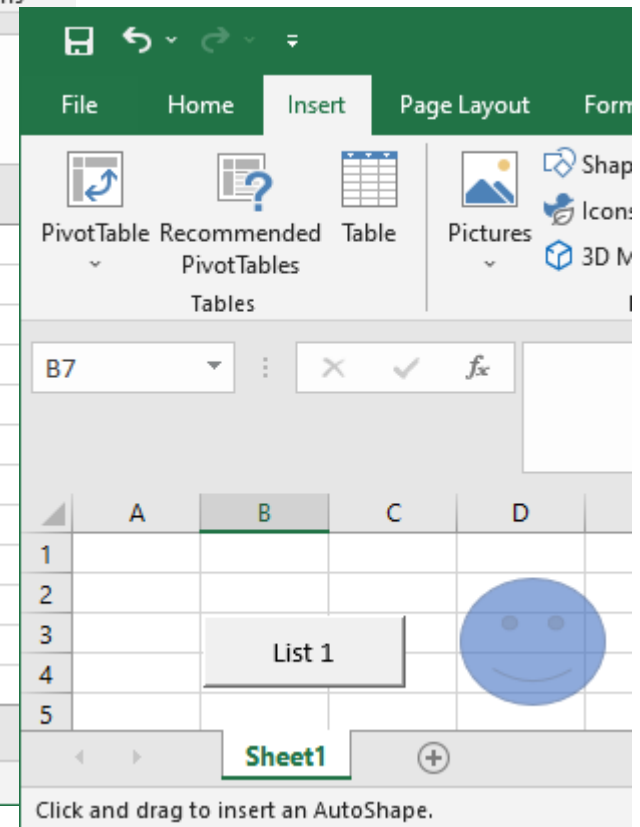
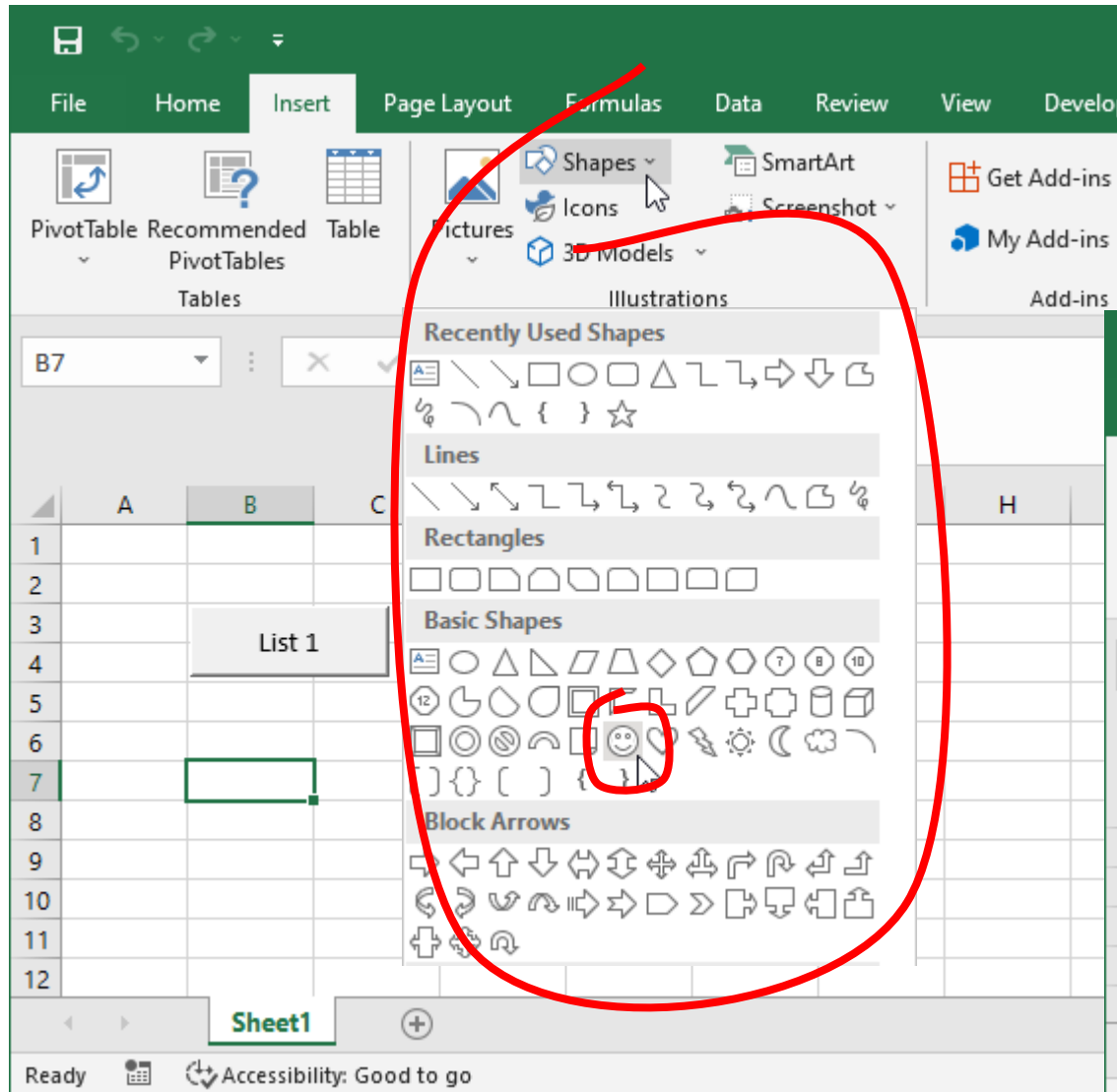
Running macro – macro List1 (method 3)

4. modify button caption
option **Edit Text**
option **Assign Macro** allows
to change macro



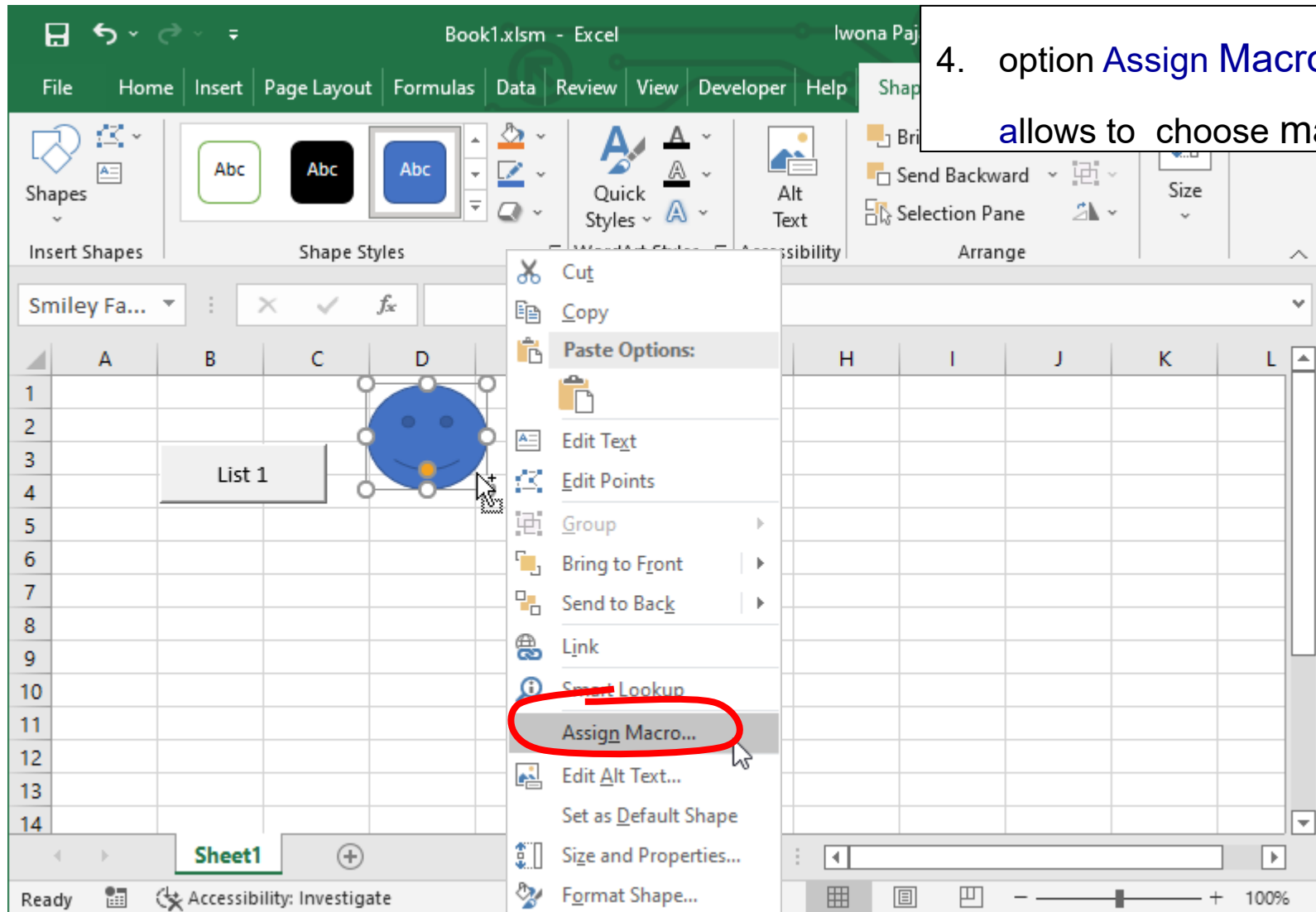
Running macro – macro List1 (method 4)

1. select Shapes
2. select shape
3. choose place on sheet



Running macro – macro List1 (method 4)

4. option **Assign Macro**
allows to choose macro





Class Range

Object Browser

Object Browser – the tool accessible in VBA editor, allows to browse classes, their properties and methods.

The screenshot shows the VBA Object Browser window. The 'Range' object is selected in the search results. The 'Classes' pane on the left lists various objects, with 'Range' selected. The 'Members of 'Range'' pane on the right lists properties and methods, with 'Cells' selected. A context menu is open over 'Cells', showing options like 'Copy', 'View Definition', 'Find Whole Word Only', 'Group Members', 'Show Hidden Members', 'References...', 'Properties...', 'Help', 'Dockable', and 'Hide'. The 'Help' option is highlighted. Red arrows point to various parts of the interface with labels: 'Searching element' points to the search box; 'Search results' points to the search results table; 'Element list (classes, modules, types)' points to the 'Classes' pane; 'Content of selected element (properties, methods, etc.)' points to the 'Members of 'Range'' pane; 'Syntax' points to the 'Help' option in the context menu. A legend on the right side of the image shows icons for properties (property sheet icon), methods (function icon), events (lightning bolt icon), and elements of types (class icon). A legend on the left side of the image shows icons for global objects (globe icon), classes (class icon), modules (module icon), and types (type icon).

Searching element

Search results

Element list (classes, modules, types)

global objects

classes

modules

types

Content of selected element (properties, methods, etc.)

Syntax

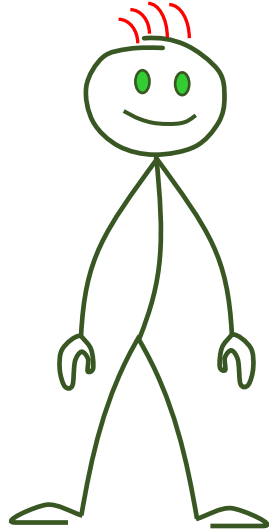
properties

methods

events

elements of types

Object model of everything

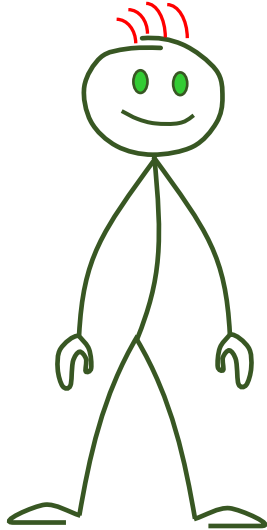


Student	
Property	Value
Name	John
Family name	Smith
Height	180
Weight	80
Age	21
Year of study	3
Address	Warsaw
Hair	
Color	red
Length	20
Eyes	
Color	green

Range	
Property	Value
Value	3
Formula	=A2+B2
Height	15
Width	48
Left	96
Top	15
Address	\$C\$2
Font	
Color	255
Size	11
Interior	
Color	65535

C2		X ✓ fx		=A2+B2	
	A	B	C	D	E
1					
2	1	2	3		
3					

Object model of everything



Student	
Method	Parameters
Stand up	
Be quiet	
Come to the blackboard	
Sit down	destination, e.g. chair
Answer	question, e.g. 2+2=?

Range	
Method	Parameters
Select	
Clear	
Calculate	
Copy	destination, e.g. Range("c3")
Merge	across, e.g. False (merge cells in each row of the range as separate merged cells)

C2					
	A	B	C	D	E
1					
2	1	2	3		
3					

Working with VBA – Object Browser

The screenshot displays the Microsoft Visual Basic for Applications environment. The main window is titled "Book1.xlsm - Module1 (Code)". The code editor shows a VBA subroutine named `List1()` that uses the `Select` method on the `Range` object. The code is as follows:

```
Option Explicit

Sub List1()
    ' List1 Macro
    ' Writes 1, 2, 3, 4 in cells A1:A5
    ' Keyboard Shortcut: Ctrl+l
    Range("A1").Select
    ActiveCell.FormulaR1C1 = "1"
    Range("A2").Select
    ActiveCell.FormulaR1C1 = "2"
    Range("A3").Select
    ActiveCell.FormulaR1C1 = "3"
    Range("A4").Select
    ActiveCell.FormulaR1C1 = "4"
    Range("A5").Select
End Sub
```

The Object Browser window is open on the right, showing the "Members of 'Range'" list. The `Select` method is highlighted. The "Classes" list on the left shows the `Range` class selected. The "Function Select()" section at the bottom indicates it is a member of `Excel.Range`.

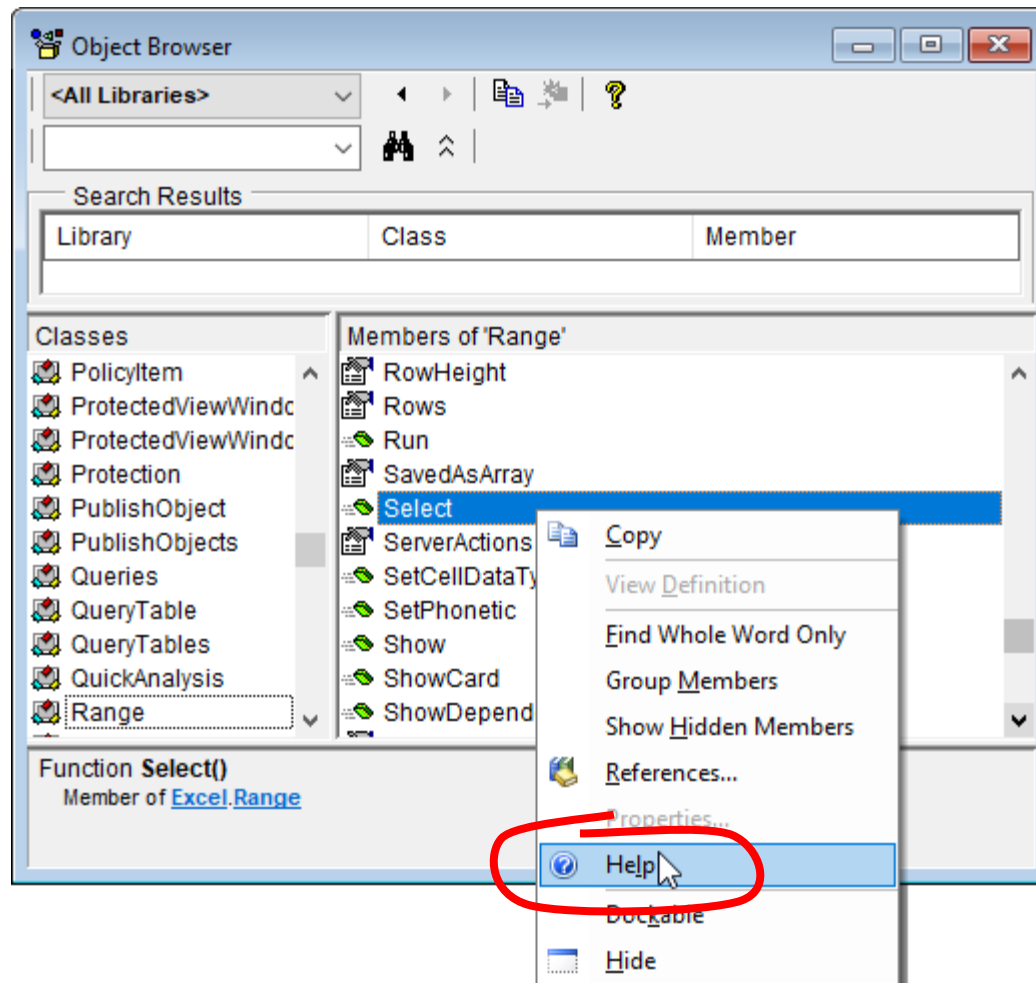
Library	Class	Member
	Protection	
	PublishObject	
	PublishObjects	
	Queries	
	QueryTable	
	QueryTables	
	QuickAnalysis	
	Range	
	Ranges	
	RecentFile	
	RecentFiles	

Members of 'Range'
Row
RowDifferences
RowHeight
Rows
Run
SavedAsArray
Select
ServerActions
SetCellDataTypeFromCell
SetPhonetic
Show

Function `Select()`
Member of `Excel.Range`

Select is parameterless method from class **Range**

Working with VBA – Object Browser



Range.Select method (Excel) | Microsoft Docs

docs.microsoft.com/en-us/office/vba/api/excel.range.select?f1url=%3F...

Docs / Office VBA Reference / Excel / Object model / Range object / Methods / Select

Filter by title

- Parse
- PasteSpecial
- PrintOut
- PrintPreview
- RemoveDuplicates
- RemoveSubtotal
- Replace
- RowDifferences
- Run
- Select**
- SetCellDataTypeFromCell
- ...

Range.Select method (Excel)

Article • 09/13/2021 • 2 minutes to read • 6 contributors

In this article

- [Syntax](#)
- [Return value](#)
- [Remarks](#)

Selects the object.

Syntax

expression.Select

expression A variable that represents a **Range** object.

Working with VBA – Object Browser

The screenshot displays the Microsoft Visual Basic for Applications environment. The main window is titled "Book1.xlsm - Module1 (Code)" and shows a VBA subroutine named `List1()`. The code includes comments and a loop that sets the `FormulaR1C1` property of the active cell for cells A1 through A5. The line `ActiveCell.FormulaR1C1 = "1"` is highlighted with an orange box. The Object Browser window is open on the right, showing the "Range" class selected in the "Classes" pane. The "Members of 'Range'" pane lists various properties and methods, with `FormulaR1C1` highlighted. The "Property `FormulaR1C1` As Variant" section at the bottom indicates it is a member of the `Excel.Range` class. A red arrow points from the text below to the `FormulaR1C1` property in the Object Browser.

```
Option Explicit

Sub List1()
    ' List1 Macro
    ' Writes 1, 2, 3, 4 in cells A1:A4
    ' Keyboard Shortcut: Ctrl+l

    Range("A1").Select
    ActiveCell.FormulaR1C1 = "1"
    Range("A2").Select
    ActiveCell.FormulaR1C1 = "2"
    Range("A3").Select
    ActiveCell.FormulaR1C1 = "3"
    Range("A4").Select
    ActiveCell.FormulaR1C1 = "4"
    Range("A5").Select
End Sub
```

Object Browser Search Results:

Library	Class	Member
Classes	PublishObjects	
	Queries	
	QueryTable	
	QueryTables	
	QuickAnalysis	
	Range	FormulaArray
		FormulaHidden
		FormulaLocal
		FormulaR1C1
		FormulaR1C1Local
		FunctionWizard
		Group
		HasArray
		HasFormula
		HasRichDataType
		HasSpill
	Ranges	
	RecentFile	
	RecentFiles	
	RectangularGradien	
	ReflectionFormat	

Property `FormulaR1C1` As Variant
Member of `Excel.Range`

FormulaR1C1 is property from class **Range**

Range.FormulaR1C1 property (Excel) | docs.microsoft.com

Docs / Office VBA Reference / Excel / Object model / Range object / Properties / FormulaR1C1

Filtruj według tytułu

- FormulaR1C1
- FormulaR1C1Local
- HasArray
- HasFormula
- HasRichDataType
- Height
- Hidden
- HorizontalAlignment
- Hyperlinks
- ID
- IndentLevel

Range.FormulaR1C1 property (Excel)

Artykuł • 13.09.2021 • Czas czytania: 2 min • Współautorzy: 6

W tym artykule

- Syntax
- Remarks
- Example

Returns or sets the formula for the object, using R1C1-style notation in the language of the macro. Read/write **Variant**.

Syntax

Filtruj według tytułu

FormulaR1C1

FormulaR1C1Local

HasArray

HasFormula

HasRichDataType

Height

Hidden

HorizontalAlignment

Hyperlinks

ID

IndentLevel

Interior

Setting the formula of a multiple-cell range fills all cells in the range with the formula.

Example

This example sets the formula for cell B1 on Sheet1.

VB

Kopiuj

```
Worksheets("Sheet1").Range("B1").FormulaR1C1 = "=SQRT(R1C1)"
```

Support and feedback

Have questions or feedback about Office VBA or this documentation?
Please see [Office VBA support and feedback](#) for guidance about the ways you can receive support and provide feedback.

Range – selected properties

Content

Value – value entered into the cell

Text – content of range as formatted text

Formula, FormulaR1C1 – formula contained in range in A1 or R1C1 notation

FormulaLocal, FormulaLocaleR1C1 – localized formula (A1 or R1C1 notation)

Relative ranges

Cells – collection of cells

Columns, Rows – collections of columns or rows

Offset – range shifted by given number rows and columns

Resize – range resized to given number rows and columns

Range – selected properties

Appearance

Font – font used in range (contains name, color, size, etc.)

Interior – style of interior (contains color, pattern, etc.)

Other

Address – address of range (e.g. C3:F5)

Count – number of cells in range

Column, Row – number of first column or row

Macros

Macro

formally: subroutine (sub) or procedure, piece of code performing a specific task, used to

- break down large pieces of code into small parts
- call the same code multiple times

Sub name_of_sub()

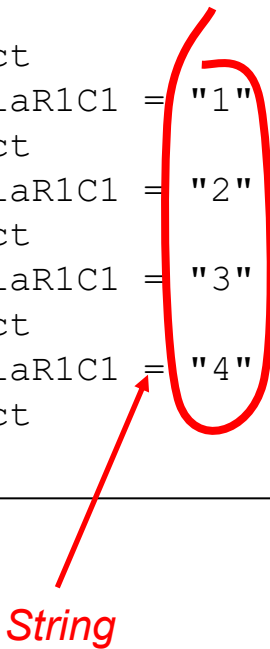
← body of sub

End Sub

Macro List2

Modifying macros – macro List2

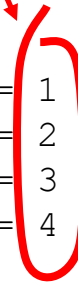
```
Sub List1()  
'  
' List1 Macro  
' Writes 1, 2, 3, 4 in cells A1:A4  
'  
' Keyboard Shortcut: Ctrl+l  
'  
    Range("A1").Select  
    ActiveCell.FormulaR1C1 = "1"  
    Range("A2").Select  
    ActiveCell.FormulaR1C1 = "2"  
    Range("A3").Select  
    ActiveCell.FormulaR1C1 = "3"  
    Range("A4").Select  
    ActiveCell.FormulaR1C1 = "4"  
    Range("A5").Select  
End Sub
```



String

*numbers
(Byte, Integer, Long,
Single, Double)*

```
Sub List2()  
'  
' List2 Macro  
' Writes 1, 2, 3, 4  
' in cells A1:A4  
'  
'  
    Range("A1").Value = 1  
    Range("A2").Value = 2  
    Range("A3").Value = 3  
    Range("A4").Value = 4  
End Sub
```



Macro List3

Macro List3

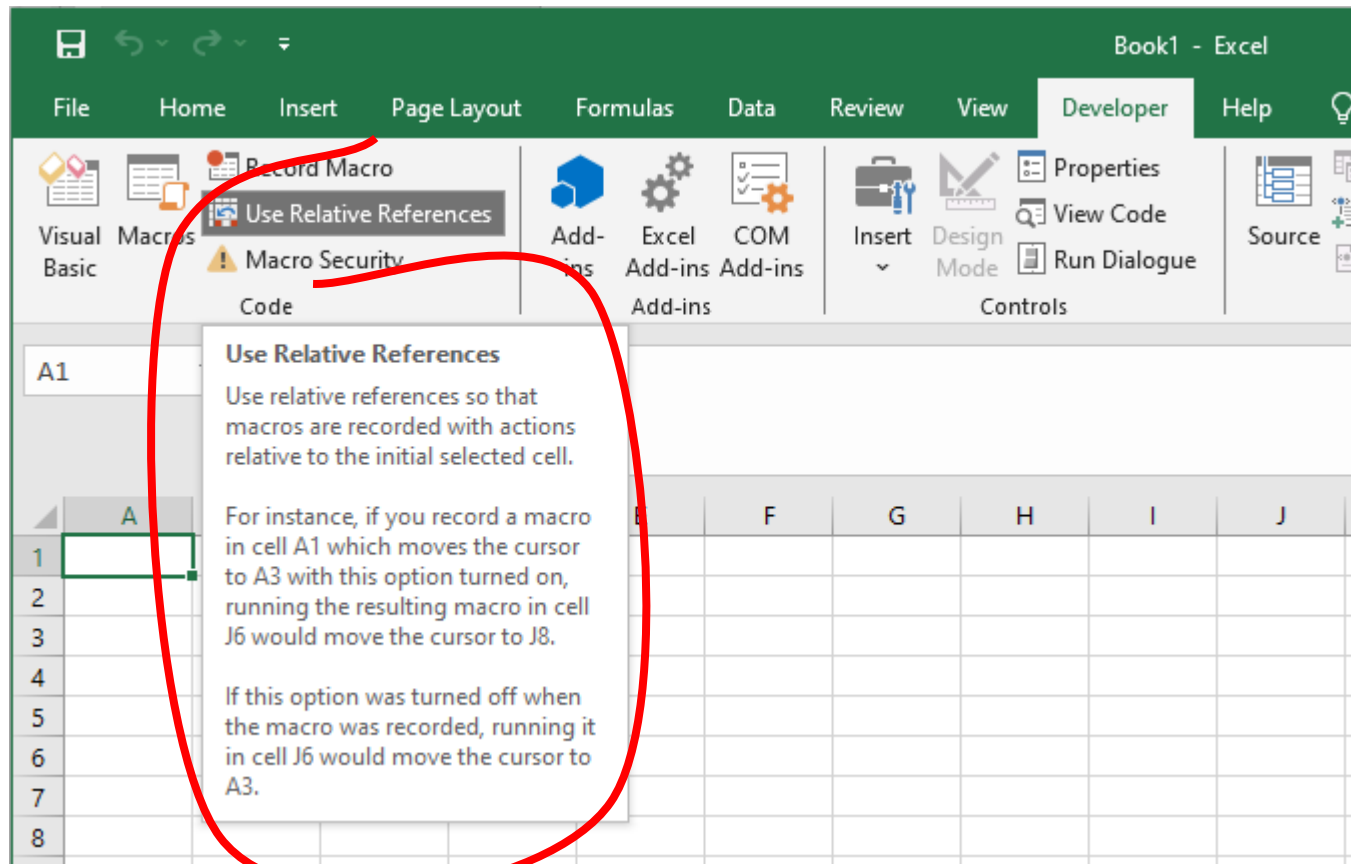
Assumptions

macro writes down:

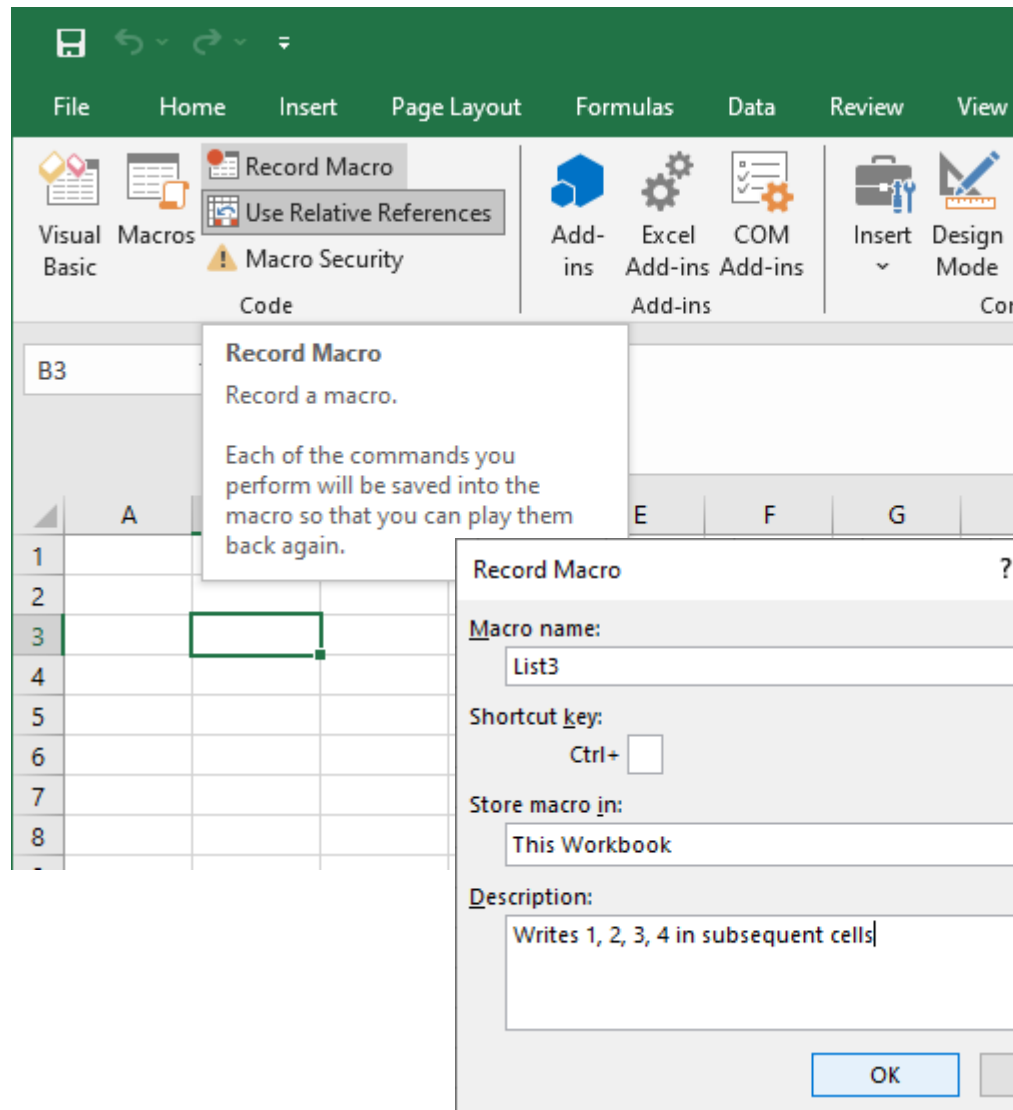
- digit 1 in the active cell
- digit 2 in the cell below
- digit 3 in the cell below
- digit 4 in the cell below

	A	B	C
1			
2		1	
3		2	
4		3	
5		4	
6			
7			

Recording macro – macro List3

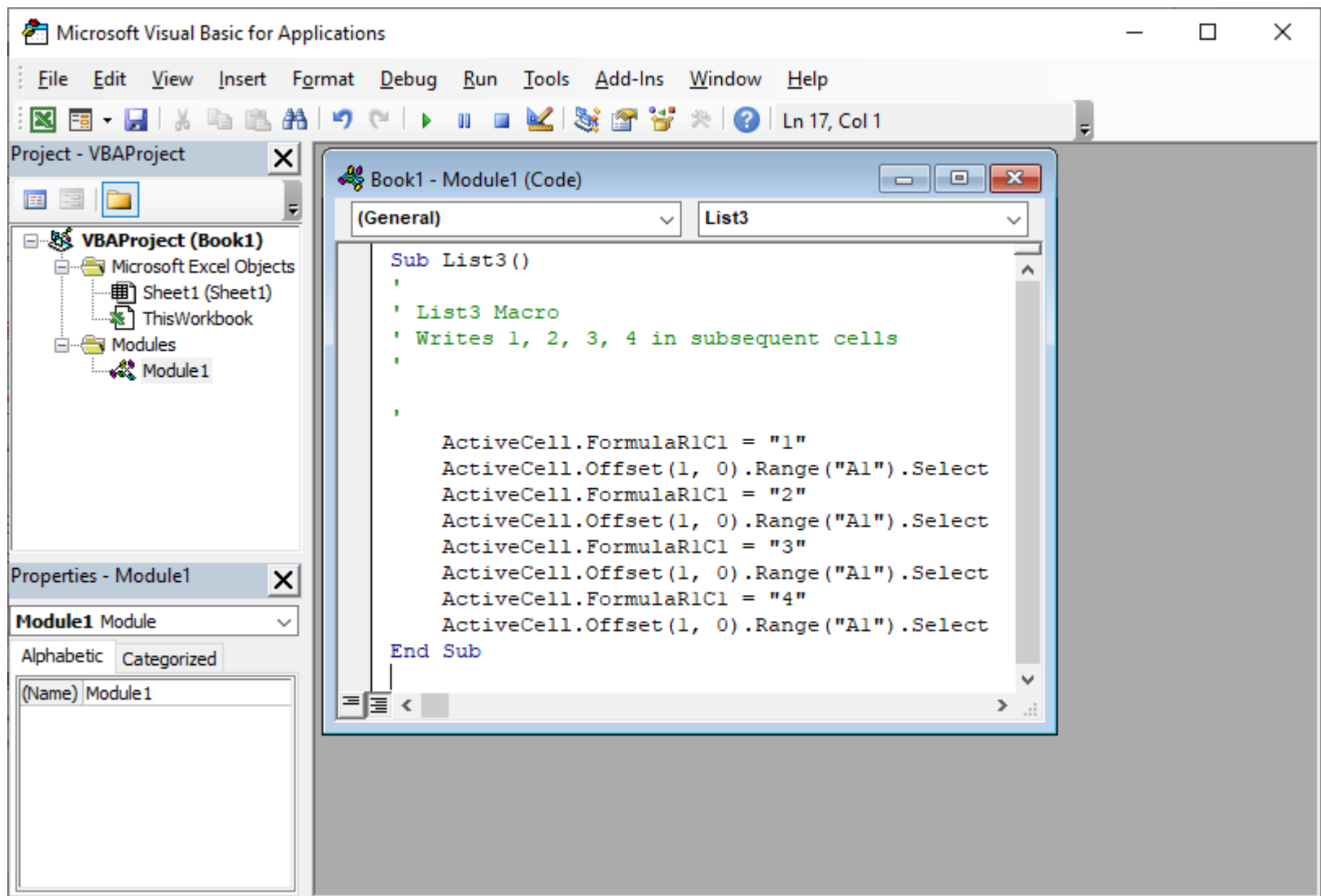


Recording macro – macro List3



1. select **Record Macro**
2. define the macro name, description, etc.
3. write **1**
4. set cursor to next cell
5. write **2**
6. set cursor to next cell
7. write **3**
8. set cursor to next cell
9. write **4**
10. set cursor to next cell
11. **stop recording**

Editing macro – macro List3



Macro List4

Modifying macros – macro List4

```
Sub List3()  
'  
' List1 Macro  
' Writes 1, 2, 3, 4 in subsequent cells  
'  
    ActiveCell.FormulaR1C1 = "1"  
    ActiveCell.Offset(1, 0).Range("A1").Select  
    ActiveCell.FormulaR1C1 = "2"  
    ActiveCell.Offset(1, 0).Range("A1").Select  
    ActiveCell.FormulaR1C1 = "3"  
    ActiveCell.Offset(1, 0).Range("A1").Select  
    ActiveCell.FormulaR1C1 = "4"  
    ActiveCell.Offset(1, 0).Range("A1").Select  
End Sub
```

```
Sub List4()  
'  
' List4 Macro  
' Writes 1, 2, 3, 4 in subsequent cells  
'  
    ActiveCell.Value = 1  
    ActiveCell.Offset(1, 0).Value = 2  
    ActiveCell.Offset(2, 0).Value = 3  
    ActiveCell.Offset(3, 0).Value = 4  
End Sub
```

Displaying messages

Displaying messages

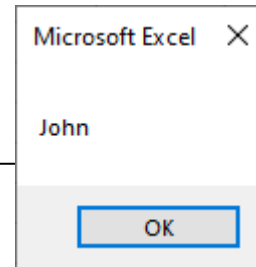
```
Sub Message1()  
'  
' Displaying message: Hello  
'  
    MsgBox "Hello"  
End Sub
```



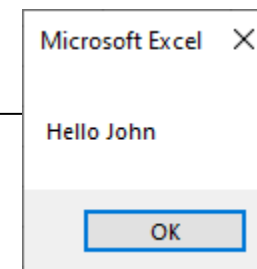
```
Sub List4()  
'  
' List4 Macro  
' Writes 1, 2, 3, 4 in subsequent cells  
'  
    MsgBox "Subsequent cells filled in 1..4"  
    ActiveCell.Value = 1  
    ActiveCell.Offset(1, 0).Value = 2  
    ActiveCell.Offset(2, 0).Value = 3  
    ActiveCell.Offset(3, 0).Value = 4  
End Sub
```

Displaying messages

	A	B	C
1			
2		John	
3			



```
Sub Message2()  
'  
' Displays the contents of cell B2  
'  
    MsgBox Range("B2").Value  
End Sub
```



```
Sub Message3()  
'  
' Displays message:  
' text Hello and the contents of cell B2  
'  
    MsgBox "Hello " & Range("B2").Value  
End Sub
```

VBA

Variables, assignment

Variable

A named storage location that can contain data that can be modified during program execution. Each variable has a name that **uniquely** identifies it within its scope. A data type can be specified or not.

Variable names must begin with an alphabetic character, must be unique within the same scope, can't be longer than 255 characters, and can't contain an embedded period or type-declaration character.

Dim statement

Declares variables and allocates storage space.

```
Dim a As Integer  
Dim b As Single, c As Single  
Dim d  
Dim length As Integer  
Dim surname As String  
Dim r As Range
```

Writing assignment statements

Assignment statements assign a value or expression to a variable or constant. Assignment statements always include an equal sign (=).

```
a = 2  
b = 2.5  
c = 5/3  
d = 3.2  
length = 5  
surname = "Doe"
```

```
Dim a As Integer  
Dim b As Single, c As Single  
Dim d  
Dim length As Integer  
Dim surname As String
```

Set statement

The **Set** statement is used to assign an object to a variable that has been declared as an object. The Set keyword is required.

```
Set r = Range("a1")
```

```
Dim r As Range
```

Macros modifications

Modifying macros – using variables

```
Sub List4()  
    ActiveCell.Value = 1  
    ActiveCell.Offset(1, 0).Value = 2  
    ActiveCell.Offset(2, 0).Value = 3  
    ActiveCell.Offset(3, 0).Value = 4  
End Sub
```

```
Sub List5b()  
    Dim n As Integer  
  
    n = 1  
    ActiveCell.Value = n  
    ActiveCell.Offset(1, 0).Value = n+1  
    ActiveCell.Offset(2, 0).Value = n+2  
    ActiveCell.Offset(3, 0).Value = n+3  
End Sub
```

```
Sub List5a()  
    Dim n1 As Integer, n2 As Integer  
    Dim n3 As Integer, n4 As Integer  
  
    n1 = 1  
    n2 = n1 + 1  
    n3 = n1 + 2  
    n4 = n1 + 3  
    ActiveCell.Value = n1  
    ActiveCell.Offset(1, 0).Value = n2  
    ActiveCell.Offset(2, 0).Value = n3  
    ActiveCell.Offset(3, 0).Value = n4  
End Sub
```

New macro – macro List6

Assumptions

macro

- gets value from the current cell
- puts value increased by 1 in the cell 1 row below
- puts value increased by 2 in the cell 2 rows below
- puts value increased by 3 in the cell 3 rows below

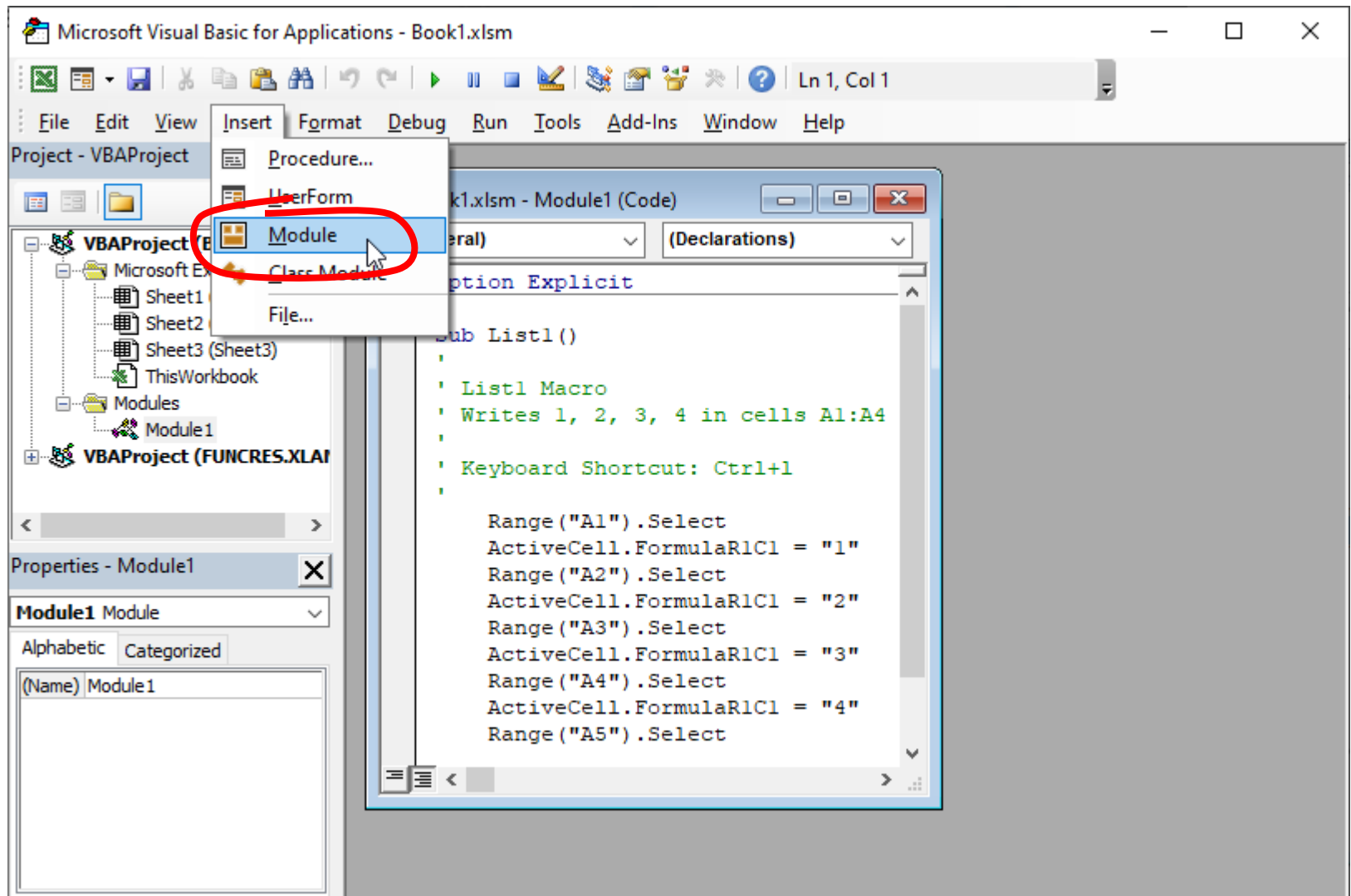
	A	B	C
1			
2		5	
3		6	
4		7	
5		8	
6			

```
Sub List5b()  
Dim n As Integer  
    n = 1  
    ActiveCell.Value = n  
    ActiveCell.Offset(1, 0).Value = n+1  
    ActiveCell.Offset(2, 0).Value = n+2  
    ActiveCell.Offset(3, 0).Value = n+3  
End Sub
```

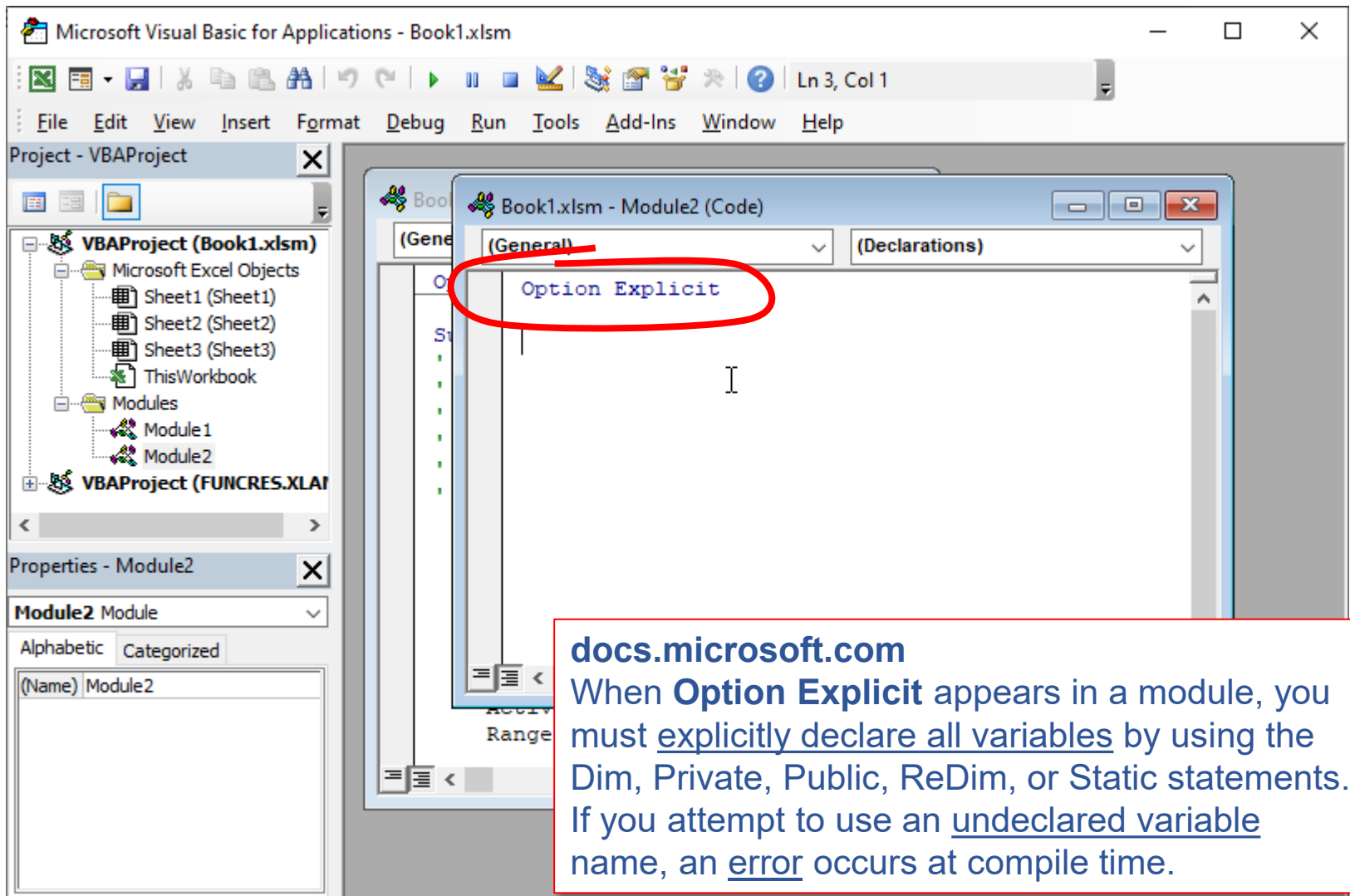
```
Sub List6()  
Dim n As Integer  
    n = ActiveCell.Value  
    ActiveCell.Offset(1, 0).Value = n+1  
    ActiveCell.Offset(2, 0).Value = n+2  
    ActiveCell.Offset(3, 0).Value = n+3  
End Sub
```


VBA environment

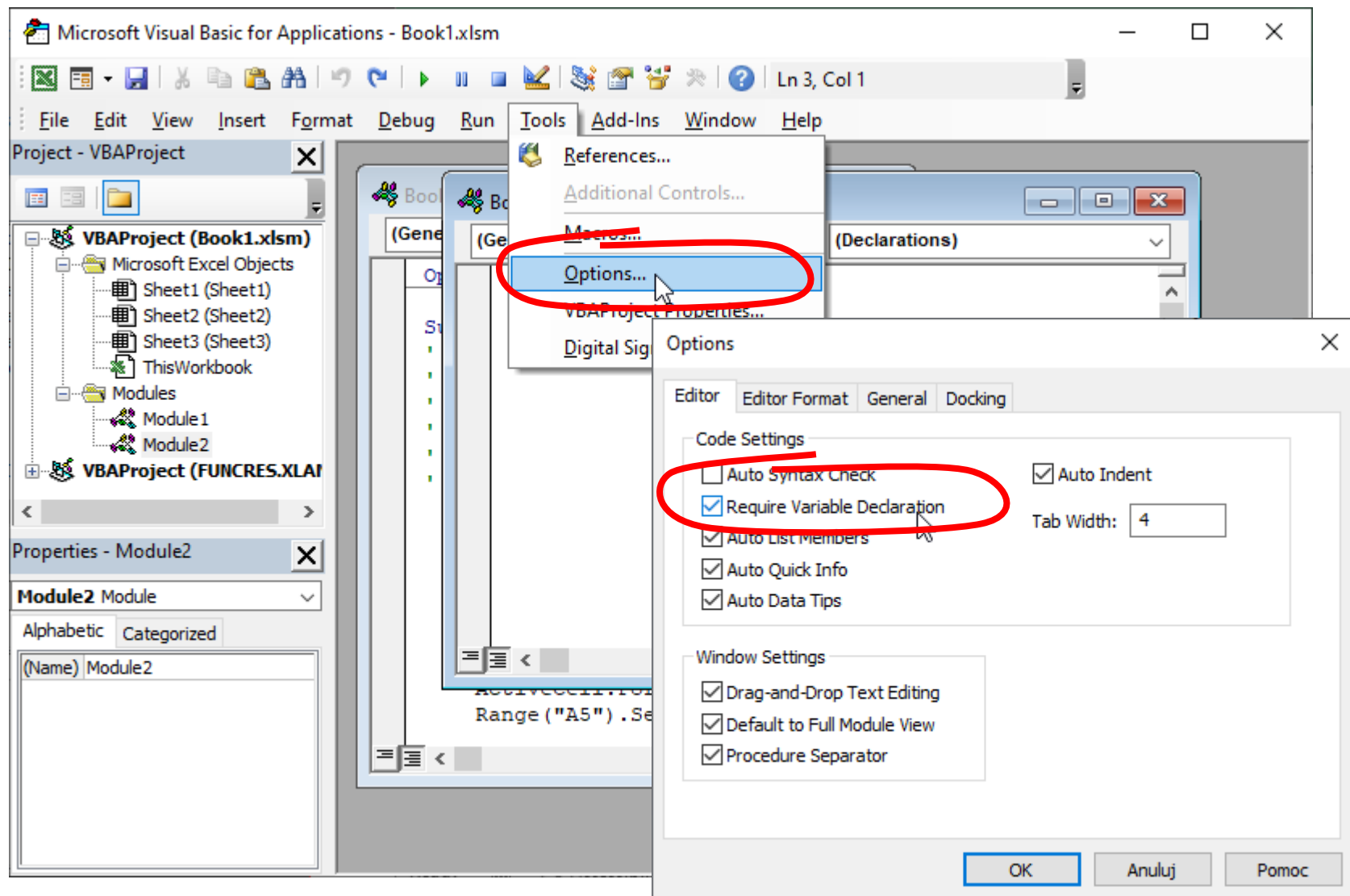
Creating new module



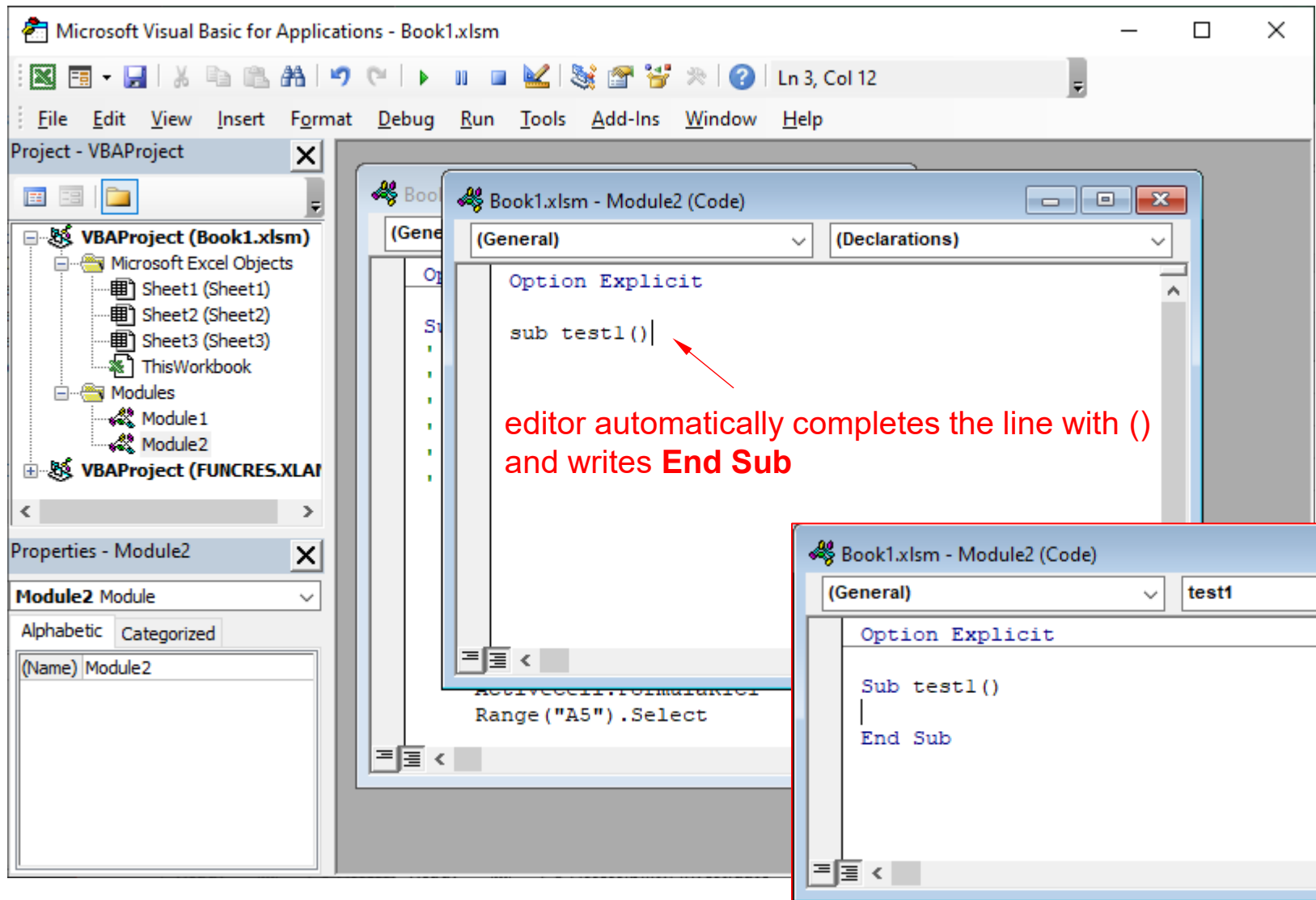
Option Explicit



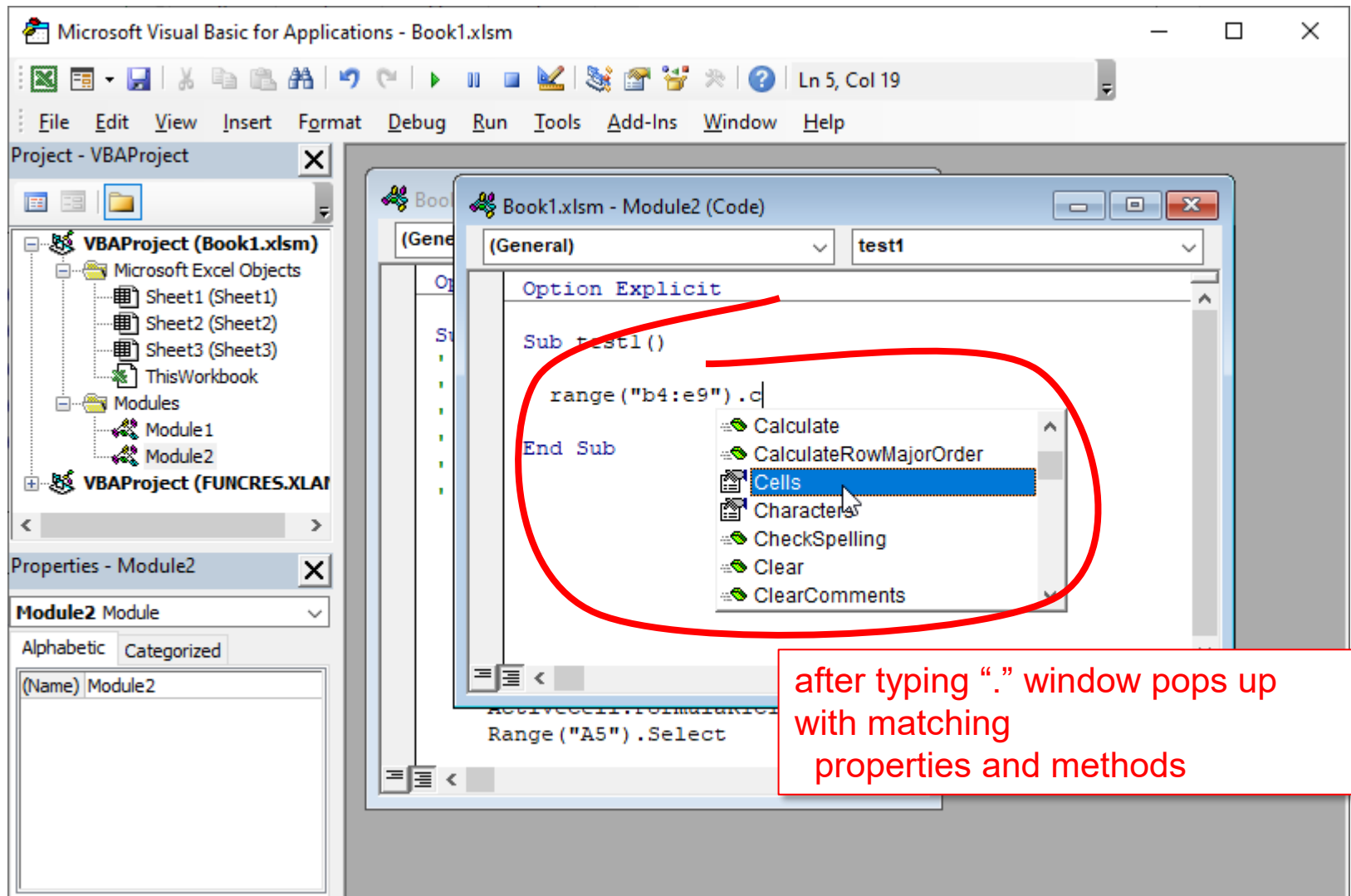
Option Explicit



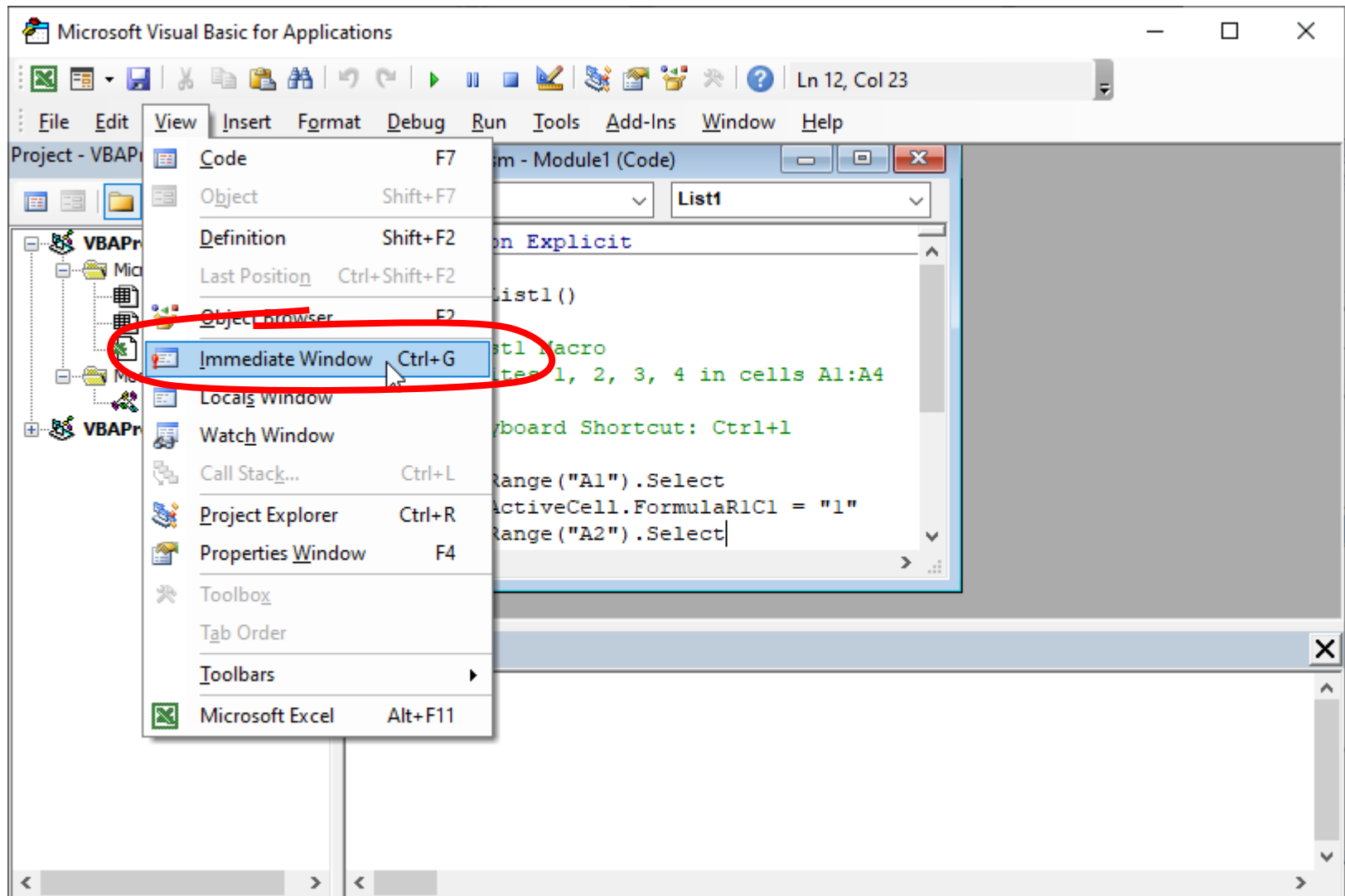
Editing macros



Editing macros



Working with VBA – Immediate Window



Working with VBA – Immediate Window

The screenshot shows the Microsoft Visual Basic for Applications environment for 'Book1.xlsm'. The VBA Project Explorer on the left shows the project structure. The Code window displays a macro named 'List1' with the following code:

```
Option Explicit

Sub List1()
'
' List1 Macro
' Writes 1, 2, 3, 4 in cells A1:A4
'
' Keyboard Shortcut: Ctrl+l

Range("A1").Select
ActiveCell.FormulaR1C1 = "1"
Range("A2").Select
```

The Immediate window at the bottom shows the command: `? Range("C2").Value`. A red circle highlights this command and a small Excel grid that shows the result of the command. The grid shows the formula bar with `=A2+B2` and the value 3 in cell C2.

	A	B	C	D	E
1					
2		1	2		
3					

after pressing Enter, result 3 is shown