HKUSPACE
香港大學專業進修學院
HKU School of Professional and Continuing Education

Business Process Automation with VBA and Python

Mr. Eddie Chow / 15 February 2025

HKUSPACE
香港大學專業進修學院
HKU School of Professional and Continuing Education

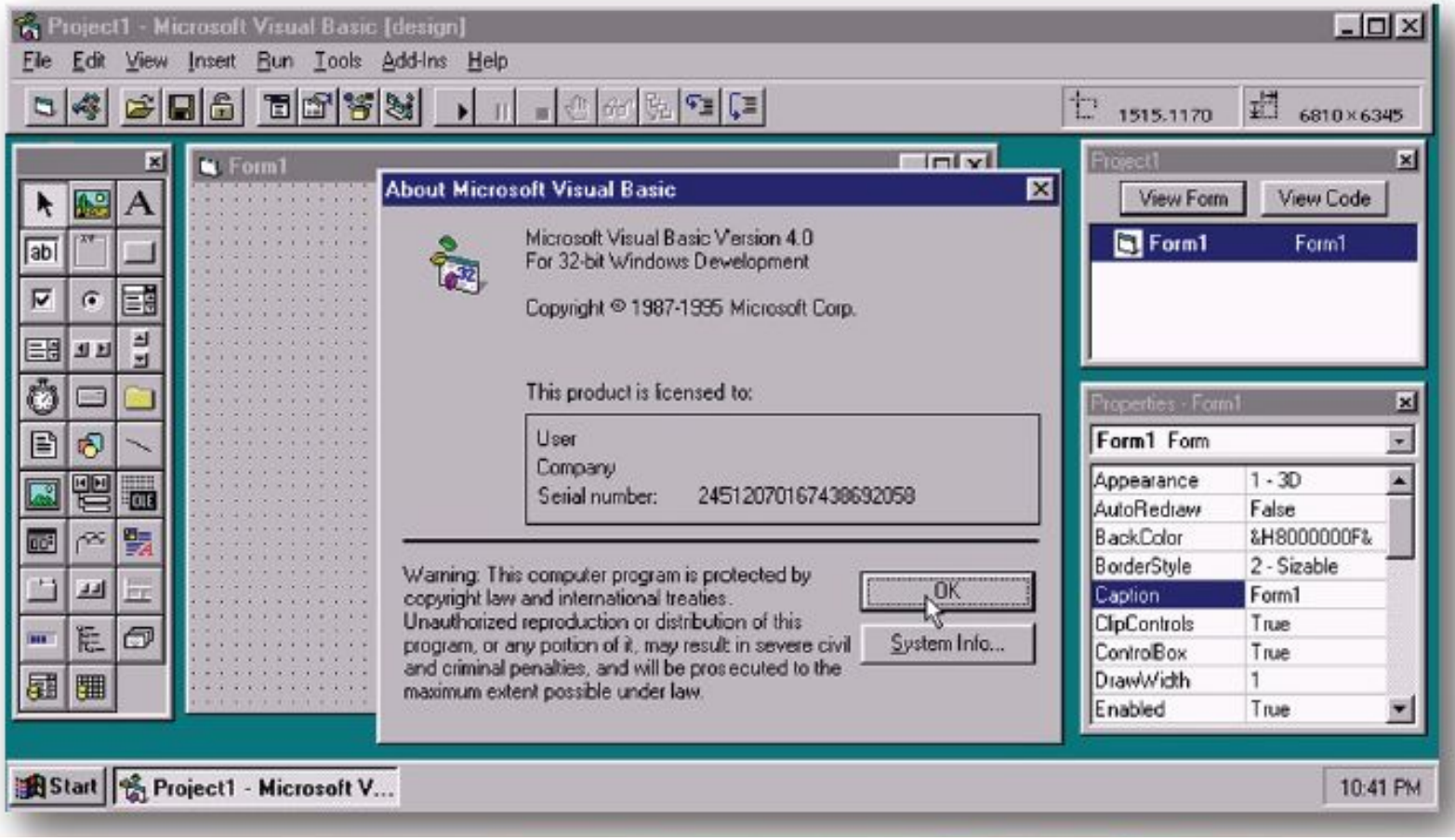
Table Of Contents

HKUSPACE
香港大學專業進修學院
HKU School of Professional and Continuing Education

- What is Process Automation, RPA, IPA?
- Overview of the technological building blocks related to process automation
- Contemporary tools for process automation
- Challenges and opportunities of business process automation
- Business implications of process automation

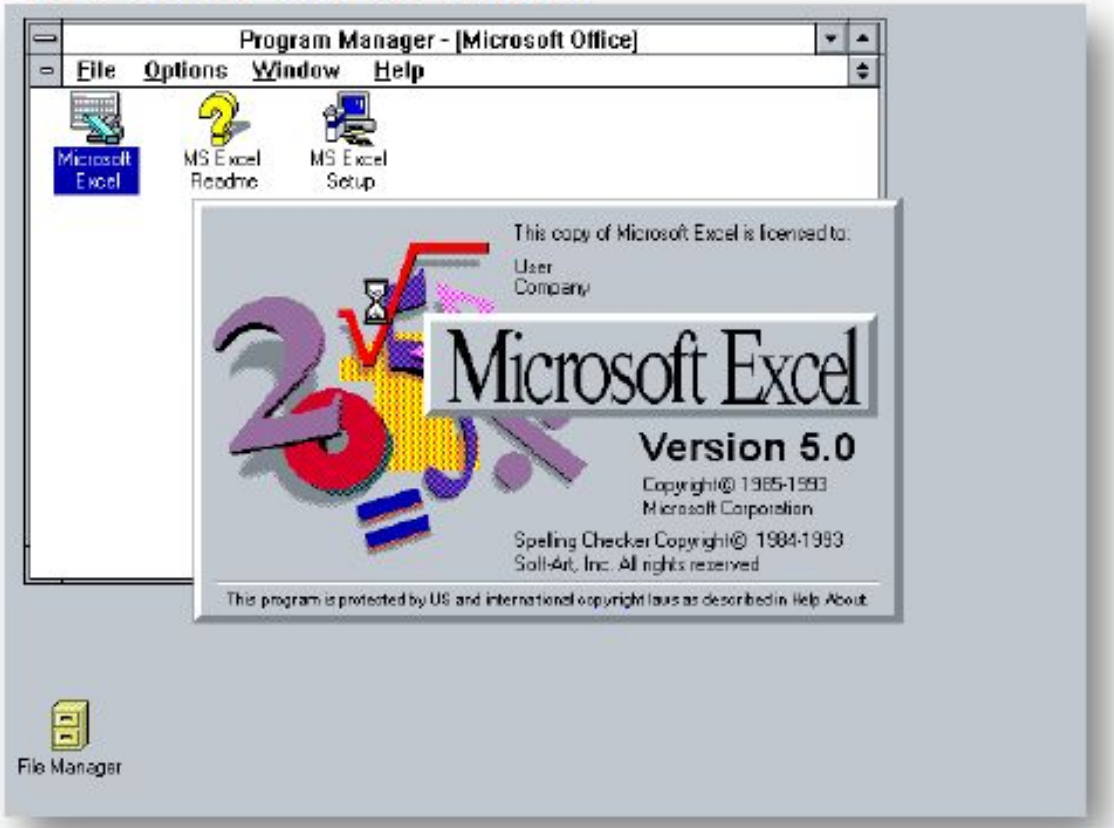
VBA - Visual Basic

- VB was “The” Microsoft way to write programs



VBA - Visual Basic for Applications

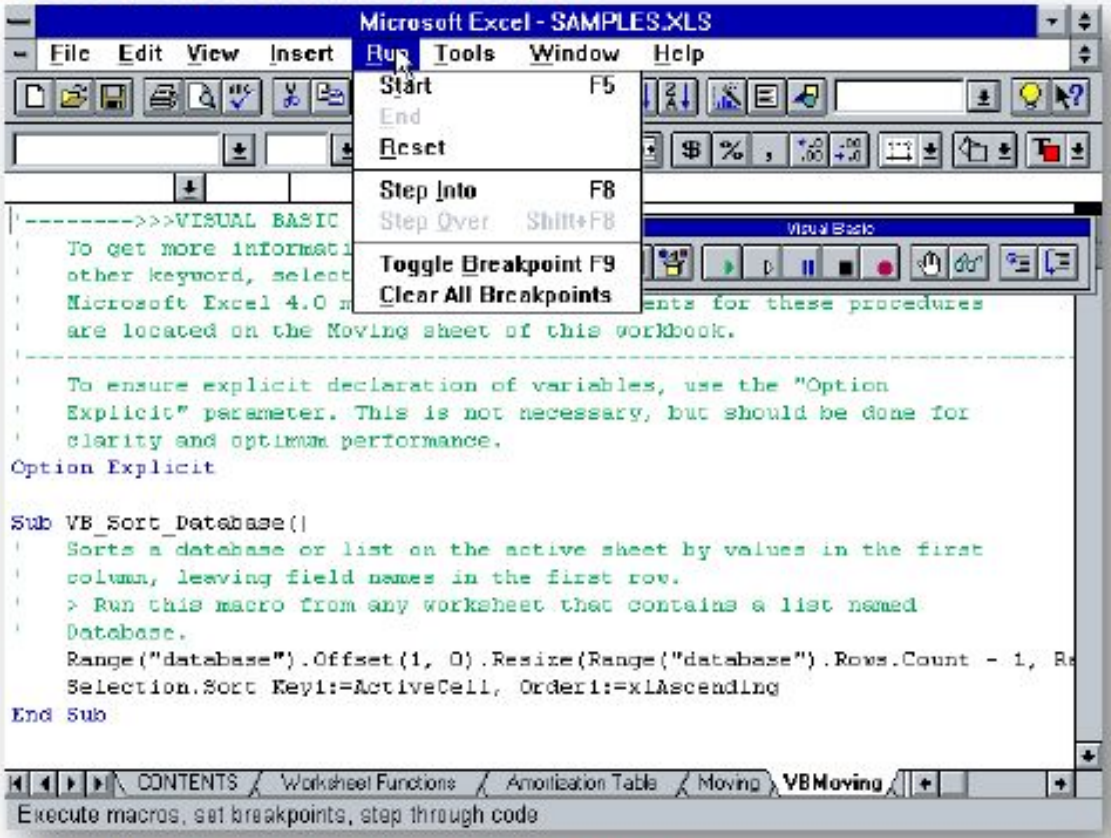
- Introduced to Excel 5.0 in 1993



1. *Microsoft Excel 5.x*. WinWorld. https://winworldpc.com/product/microsoft-excel/5x

VBA - Visual Basic for Applications

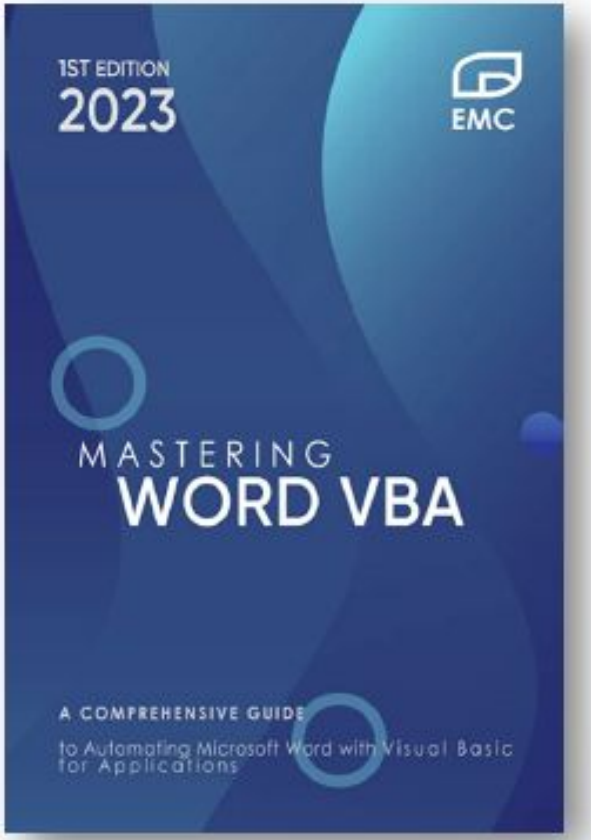
- Introduced to Excel 5.0 in 1993



1. *Microsoft Excel 5.x*. WinWorld. https://winworldpc.com/product/microsoft-excel/5x

VBA - Visual Basic for Applications

- Gradually extended to Microsoft Office
- VB classic ➔ ended in 2008
- VB.NET ➔ evolution ended in 2020[1]
- VBA ➔ still supported (e.g., M365)



1. Team, N. (2020, March 11). *Visual Basic support planned for .NET 5.0*. Visual Basic Blog. https://devblogs.microsoft.com/vbteam/visual-basic-support-planned-for-net-5-0/

Business Process Automation with VBA

VBA – Why?

- Automation
- User interaction
- Widely used in the industry

1. O. (2022, June 8). *Getting started with VBA in Office*. Getting Started With VBA in Office | Microsoft Learn. https://learn.microsoft.com/en-us/office/vba/library-reference/concepts/getting-started-with-vba-in-office

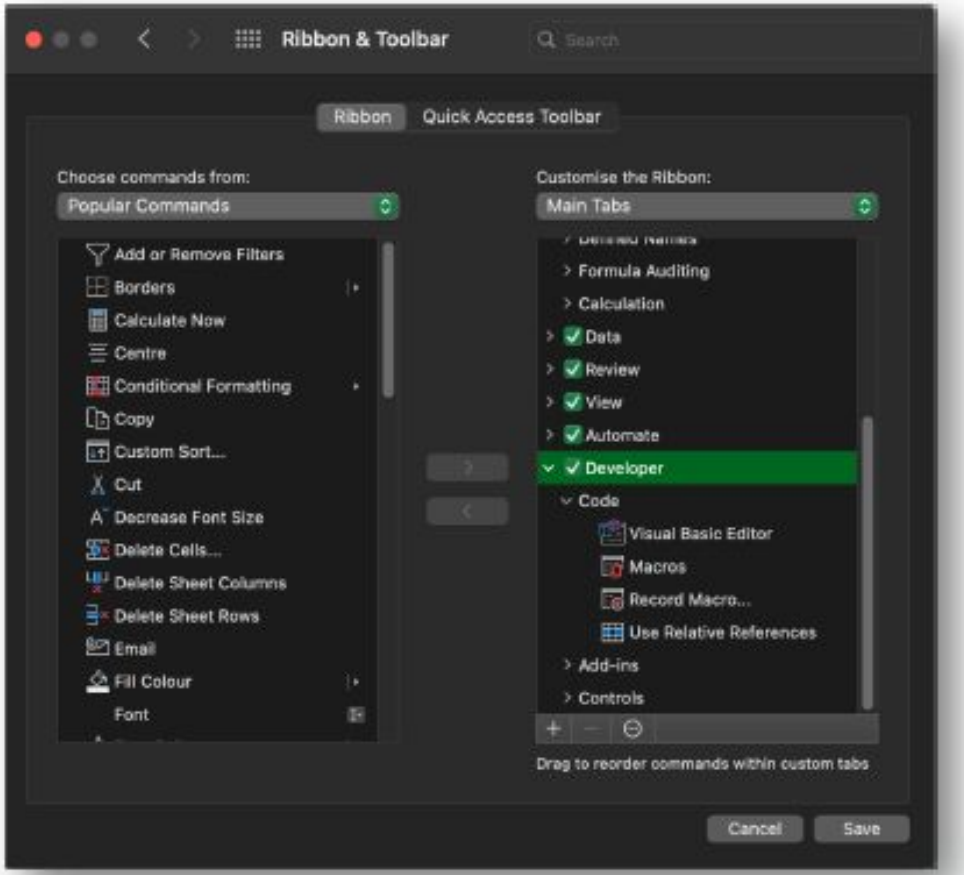
VBA - Automation

- Macros ➔ Microsoft's response to automation
- Naming:
  - **Micro**-instruction vs **Macro**-instruction
  - 1 key to perform series of commands / actions

1. *Why is Excel VBA called "Macros"?* Why Is Excel VBA Called "Macros"? https://www.excelforum.com/the-water-cooler/791161-why-is-excel-vba-called-macros.html

Macros – Where?

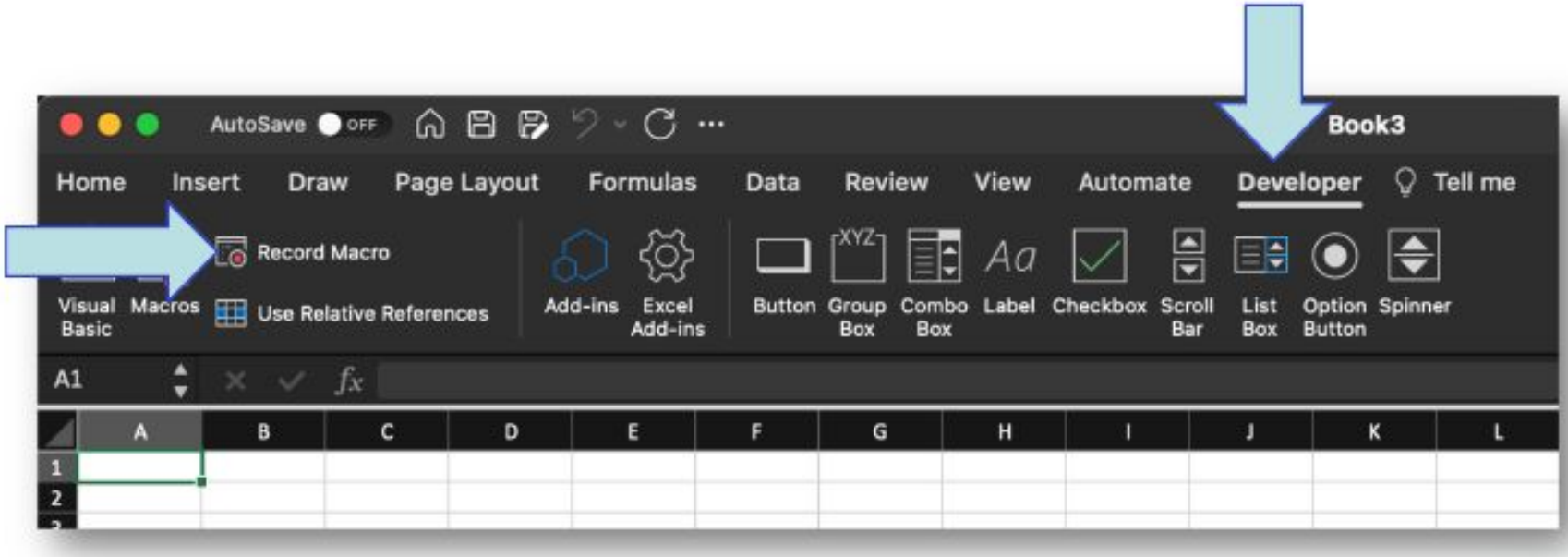
- "Developer" tab
  - Hidden by default
  - File
    ➔ Options
    ➔ Customize Ribbon
  - Turn on "Developer"



1. *Show the Developer tab - Microsoft Support*. Show the Developer Tab - Microsoft Support. https://support.microsoft.com/en-us/office/show-the-developer-tab-e1192344-5e56-4d45-931b-e5fd9bea2d45

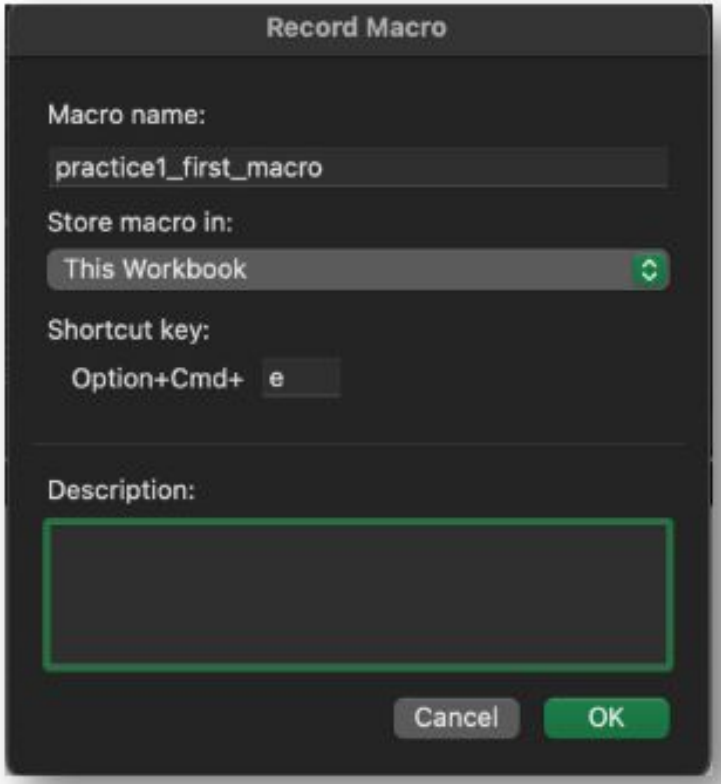
Practice 1 - Your first macro

- Click “Record Macro”



Practice 1 - Your first macro

- Type in a name (use "_" instead of spaces)
- Set a shortcut key
- Click OK

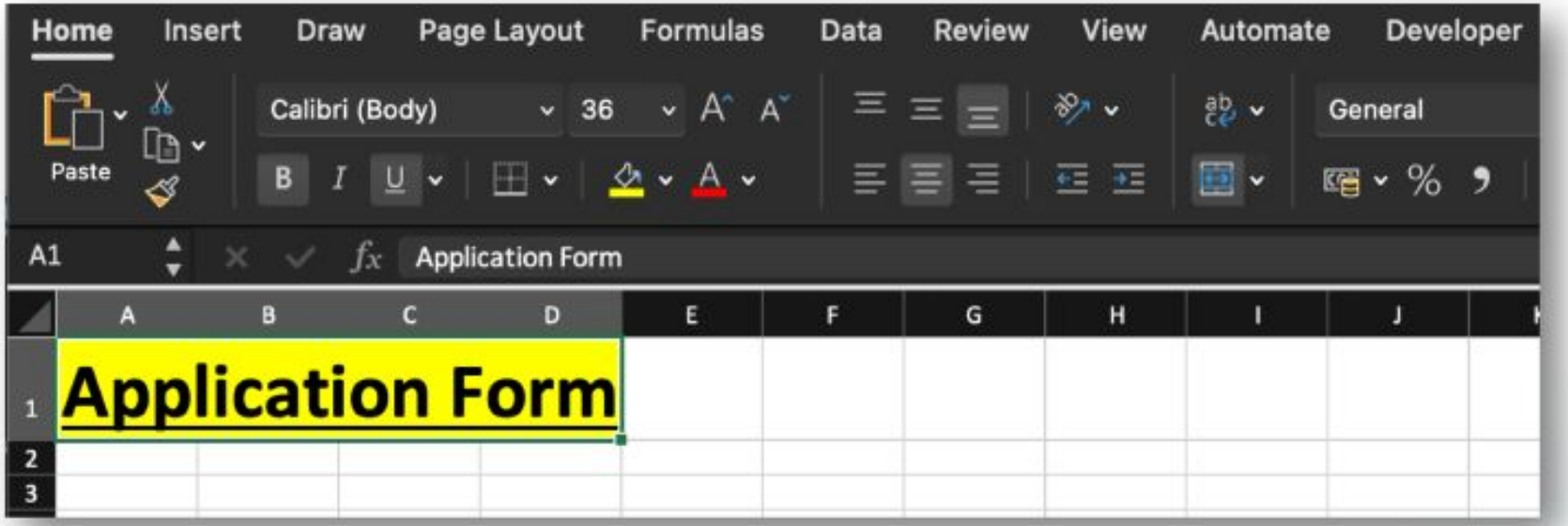


Practice 1 - Your first macro

- The recording has started
- Add a new sheet with the “+” at the bottom-left

Practice 1 - Your first macro

- Go back to “Home” tab
- Select Cells A1 to D1 ➔ Merge
- Enter “Application Form” at the top-left
- Style it: e.g., Font size + Bold + Underline

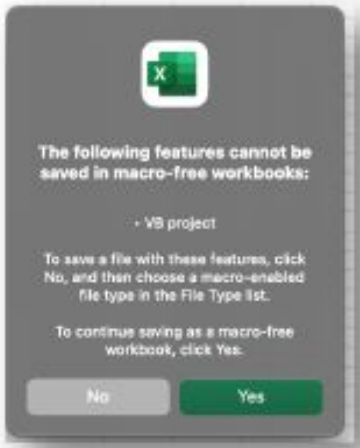


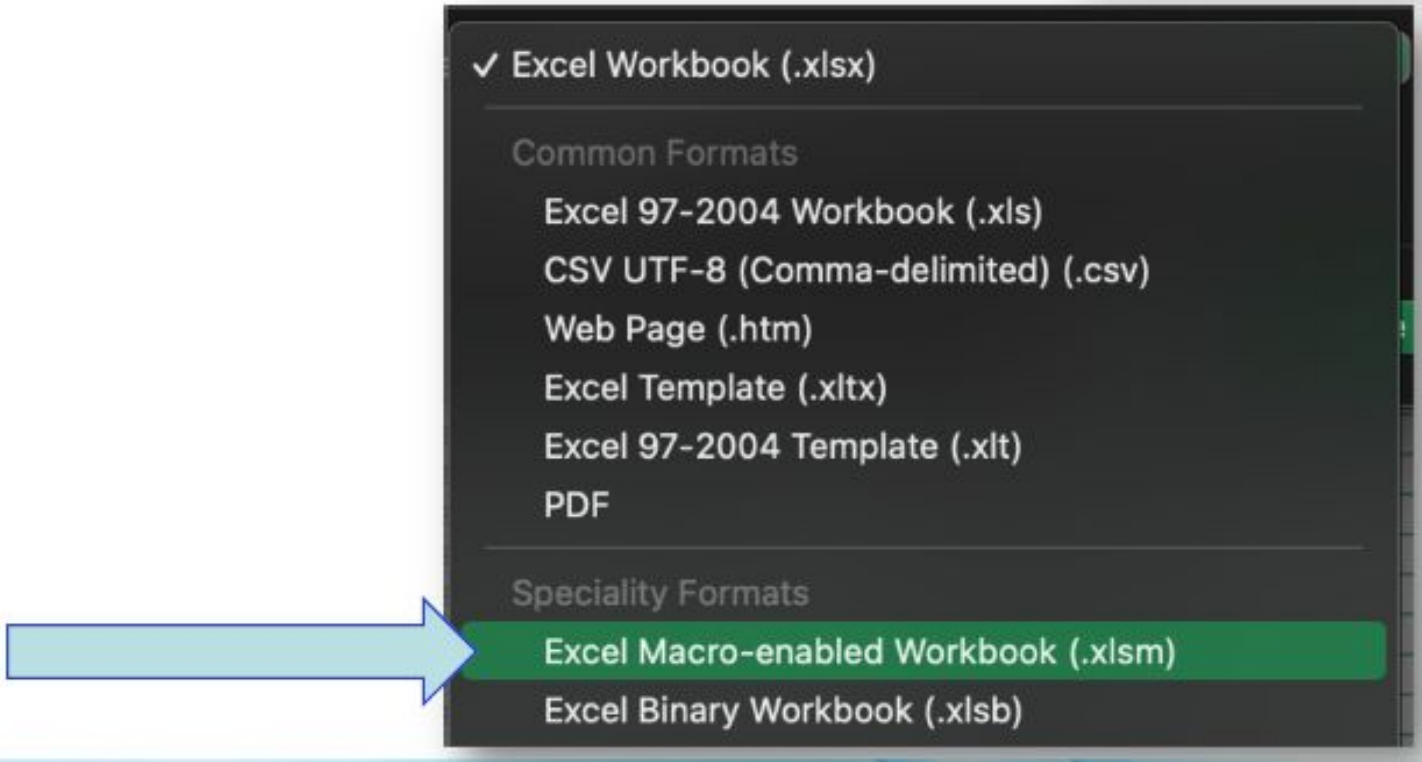
Practice 1 - Your first macro

- Go back to “Developer” tab
- Hit “Stop Recording”

Practice 1 - Your first macro

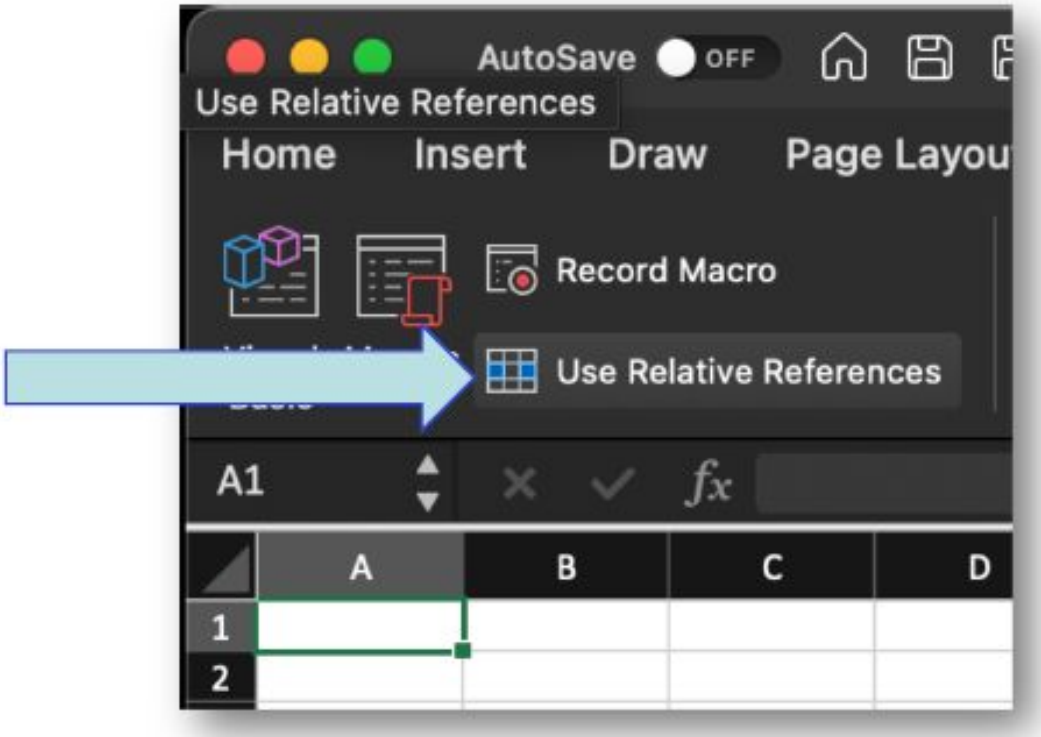
- Save as .xlsm to preserve your macros!





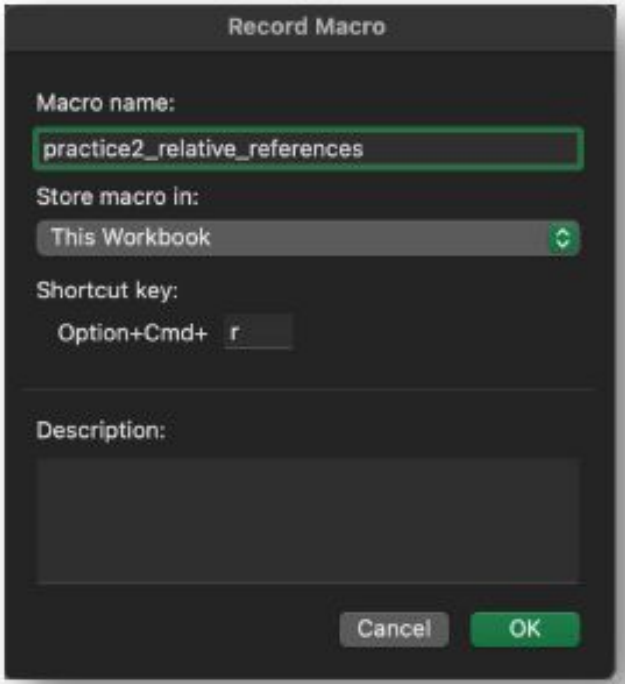
Relative Reference

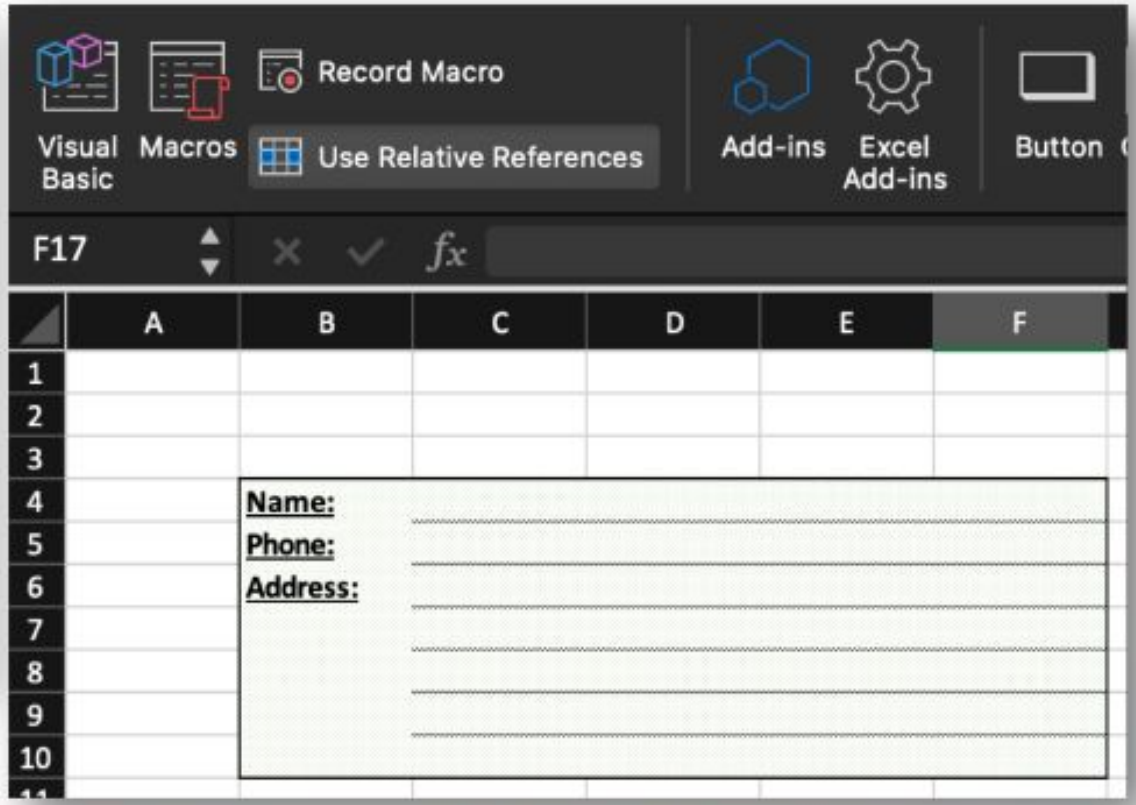
- Record and execute based on “current selection”
- Click BEFORE “Record Macro”



Practice 2 – Relative References

- Create a new macro to generate below



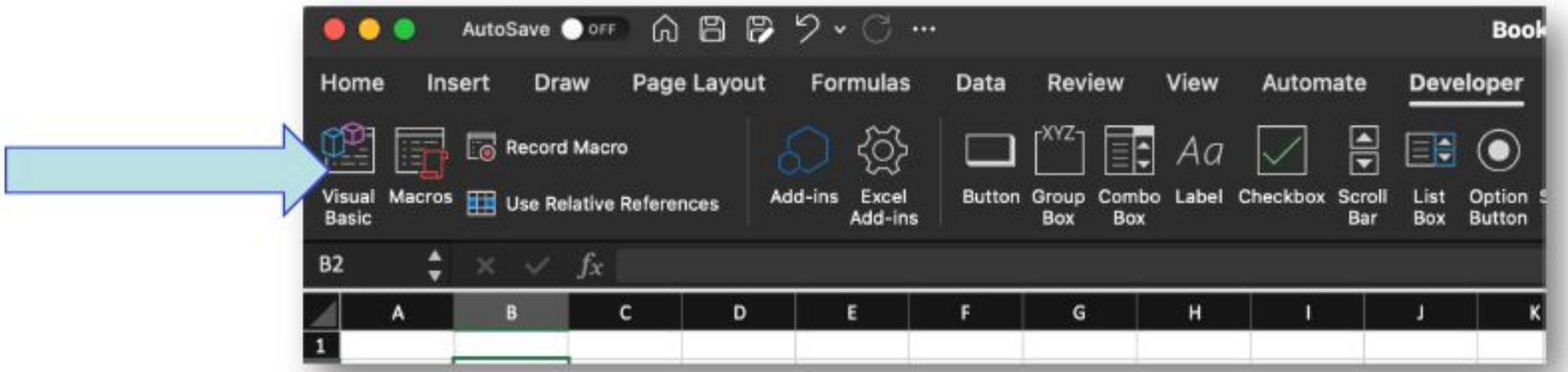


VBA - Why not just use macros?

- Macros ➔ great for repetition
- Need ➔ More flexibility and interactivity
- Solution ➔ Dig into the codes

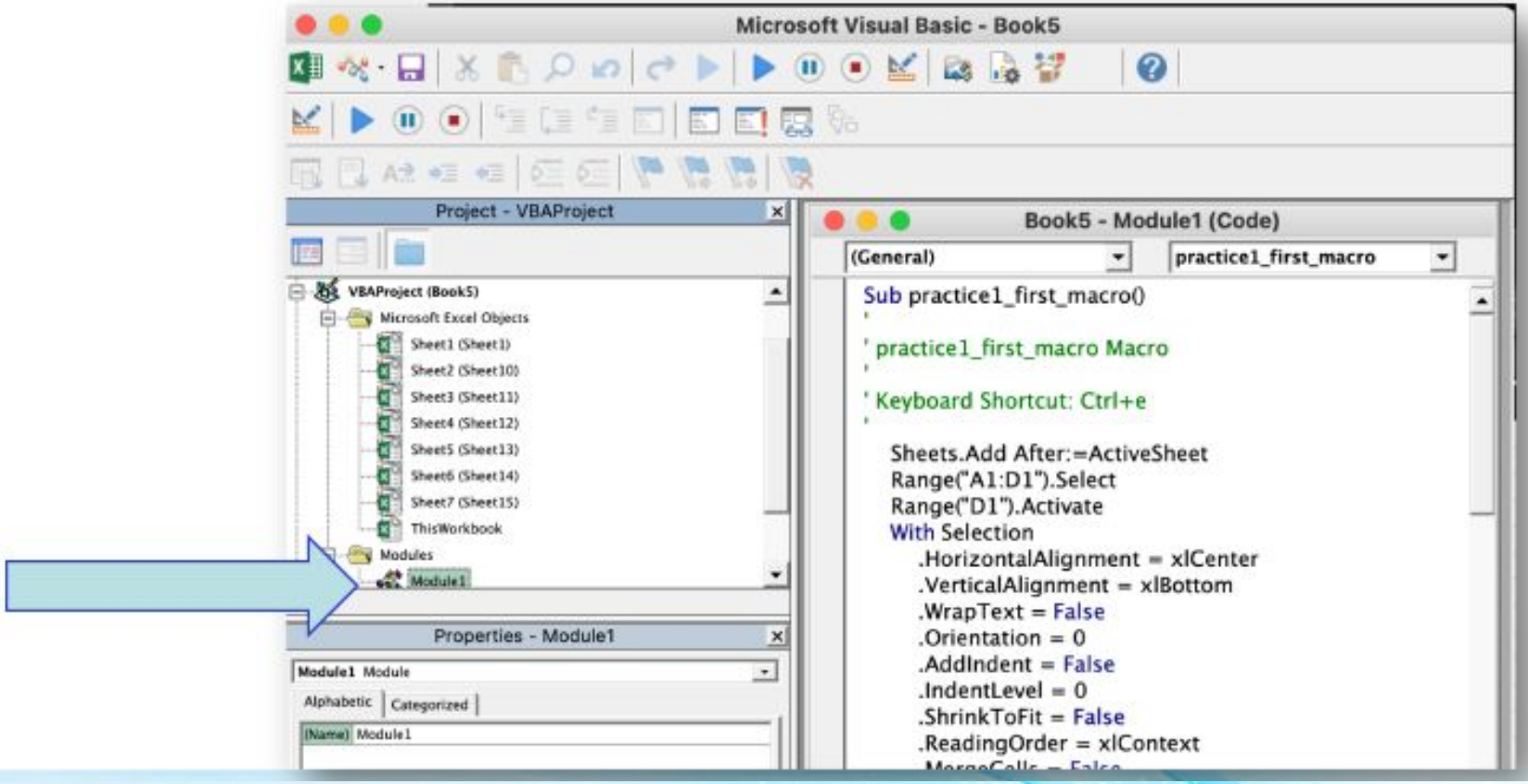
Entering Visual Basic Mode

- Developer tab ➔ Visual Basic



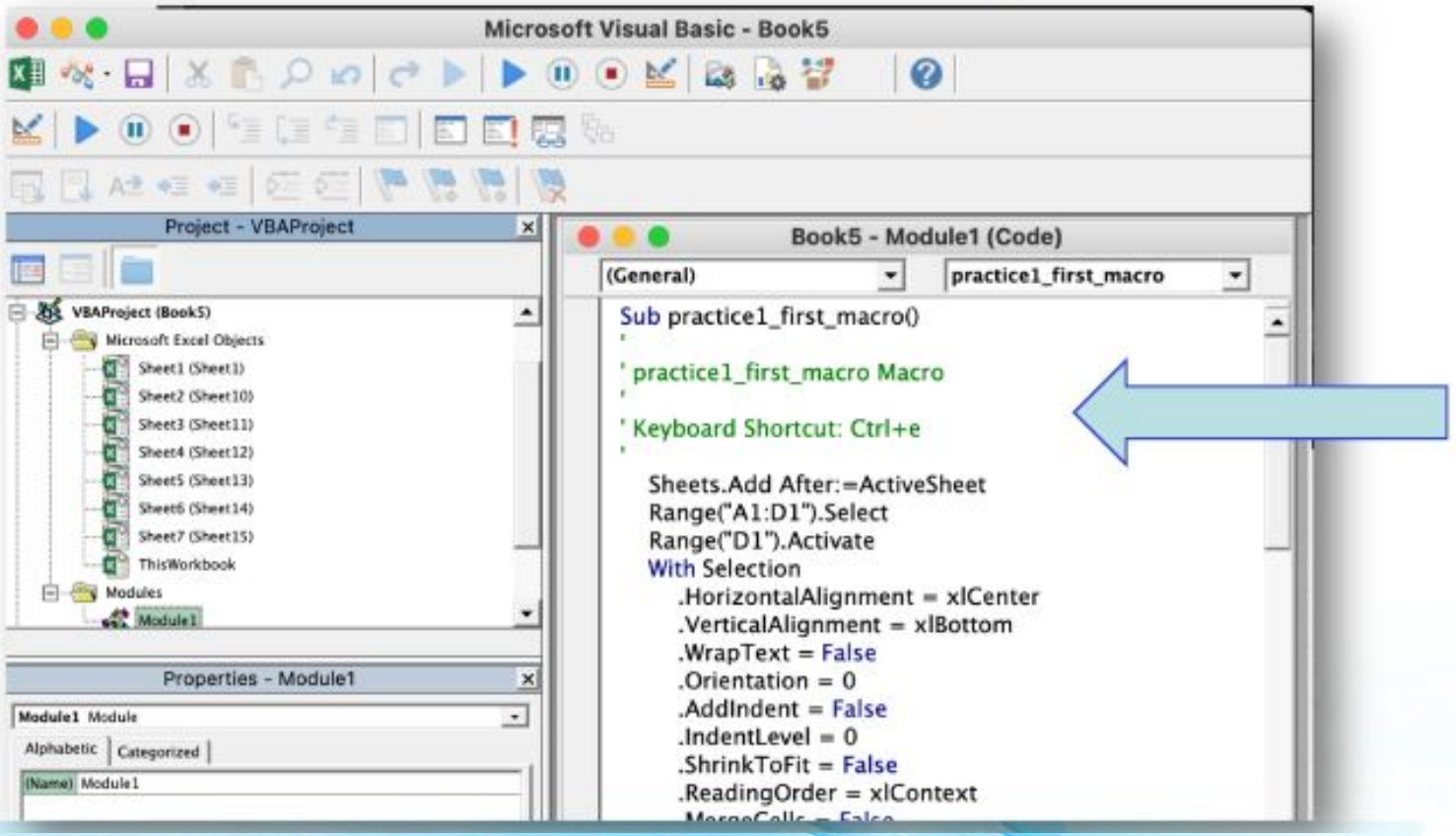
Entering Visual Basic Mode

- Integrated Development Environment (IDE)
- Modules = where codes are usually stored



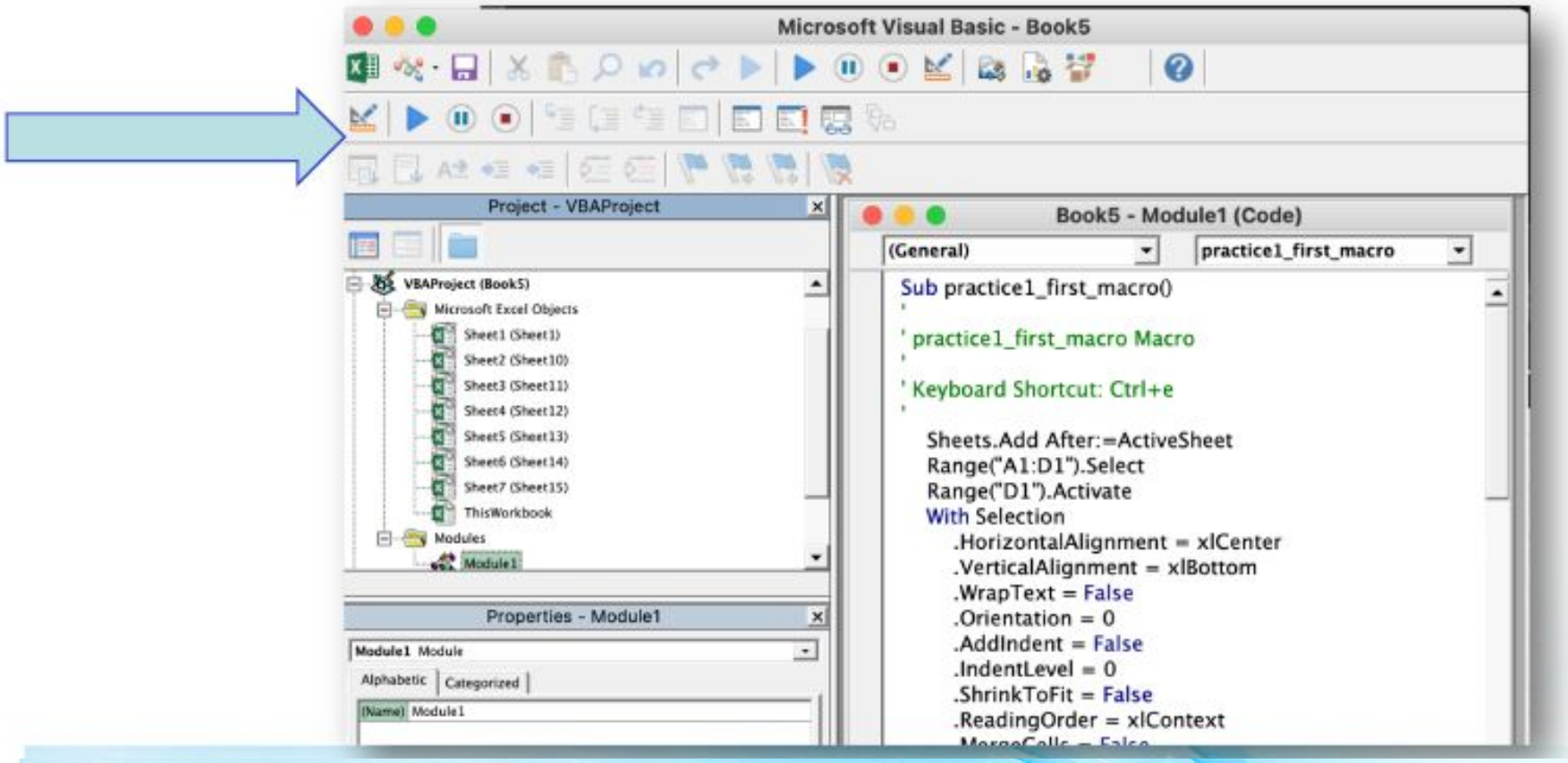
Entering Visual Basic Mode

- Main editor on the right
- Modules ➔ 1 or more “Sub” (sub-routines) or “Function”



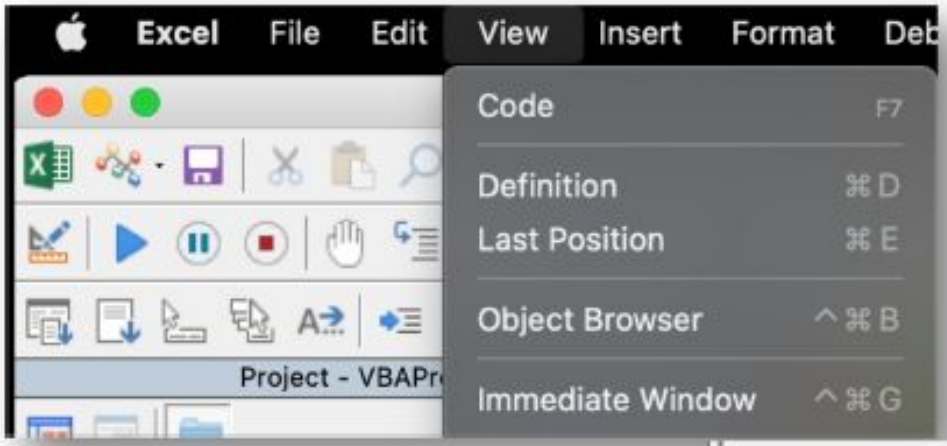
Entering Visual Basic Mode

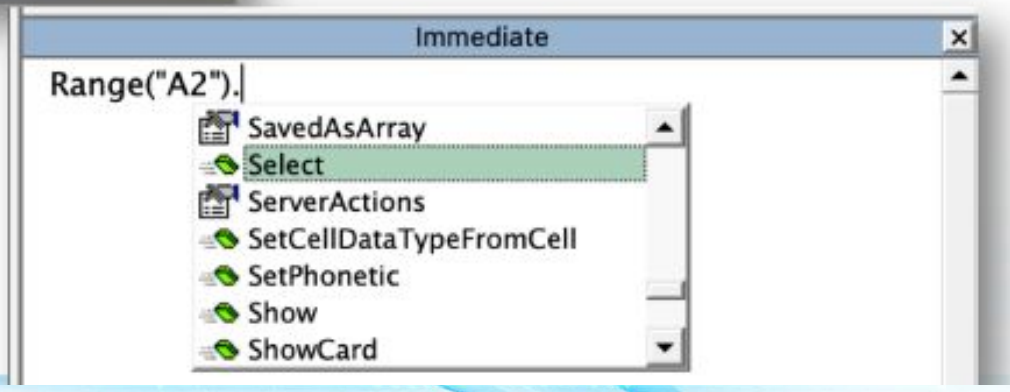
- Debugging toolbar at the top



Immediate Window

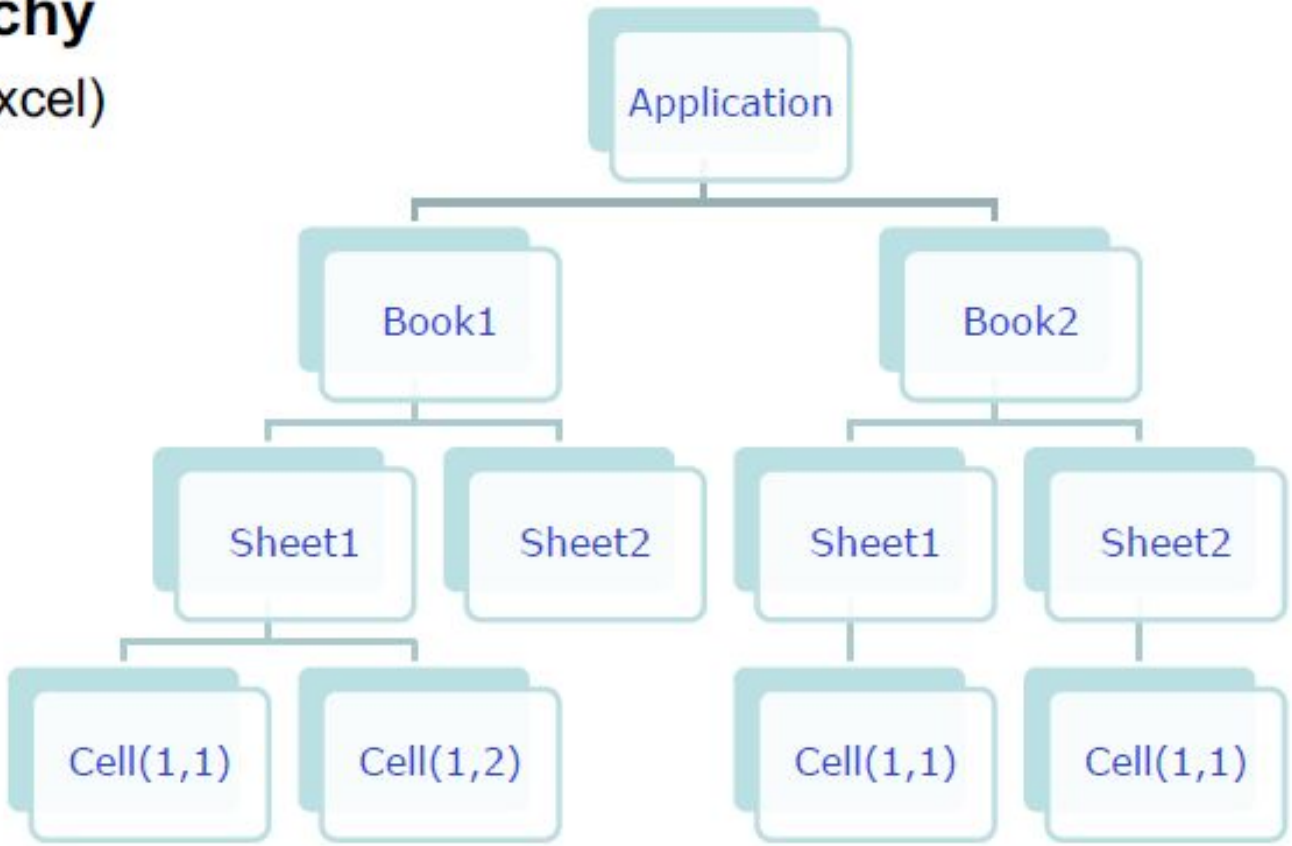
- View ➔ Immediate Window
- Useful for trying out codes (Thanks to Auto-complete)





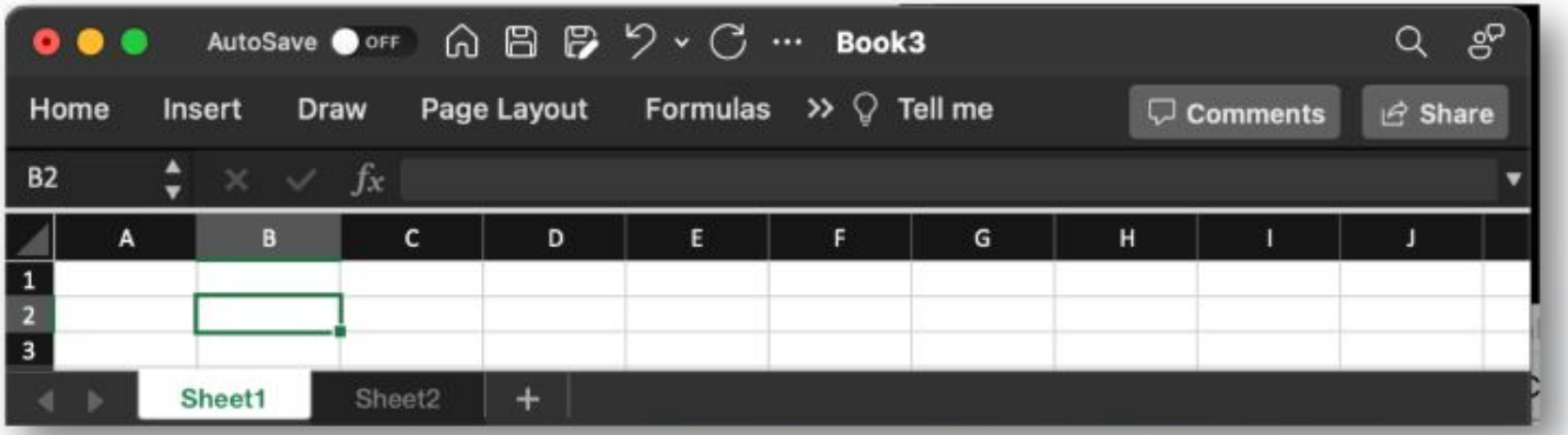
Object hierarchy

- Application (Excel)
- Workbooks
- Worksheets
- Cells



Object reference – Call something by name

- English: This man / that girl / “Mr. Chan”
- VBA
  - Application ➔ Excel.Application / Outlook.Application / etc
  - Workbooks("Book1")
  - Worksheets("Sheet1") / Worksheets(1)
  - Cells(2,3)
  - Range("B2")



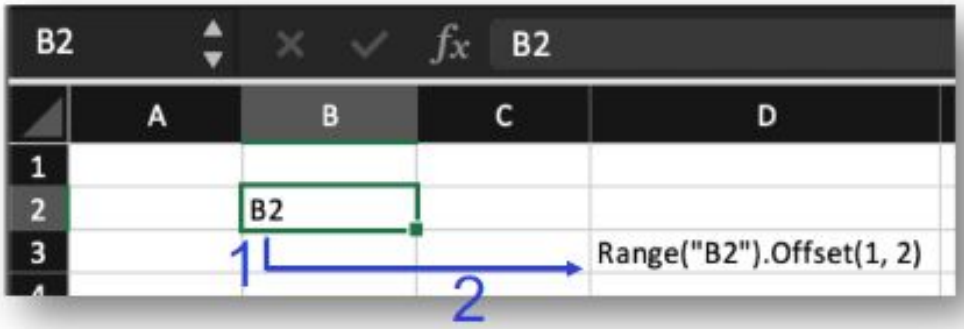
Object reference - Implied naming

- English: Here / Today
- VBA
  - ActiveWorkbook
  - ActiveSheet
  - Selection
    - If you write:
      - Range("A1:D1").Select
    - From that moment onwards:
      - Selection ➔ Same as Range("A1:D1")

* ThisWorkbook
  - This = Where the macro is stored
  - Active = Changeable by user

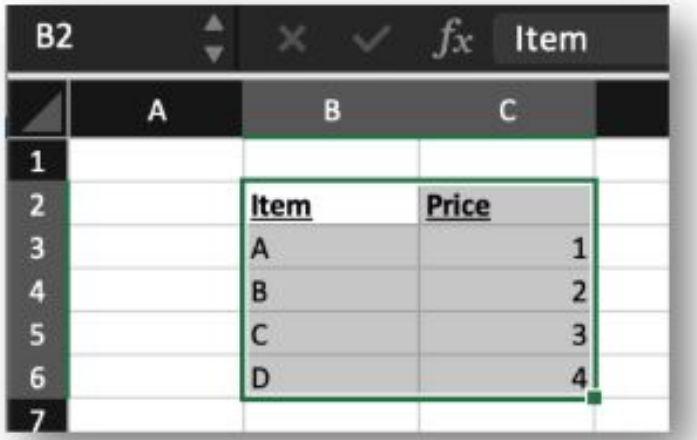
Object reference - Offset

- English: Is it that building? No, the one on the right
- VBA
  - Range("B2").Offset(1, 2)
  - Worksheets(ActiveSheet.Index + 1).Select

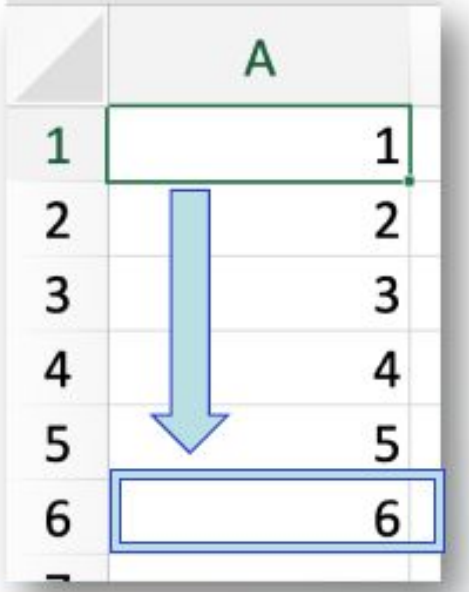


Object reference – Offset

- Region
  - Selection.CurrentRegion.Select



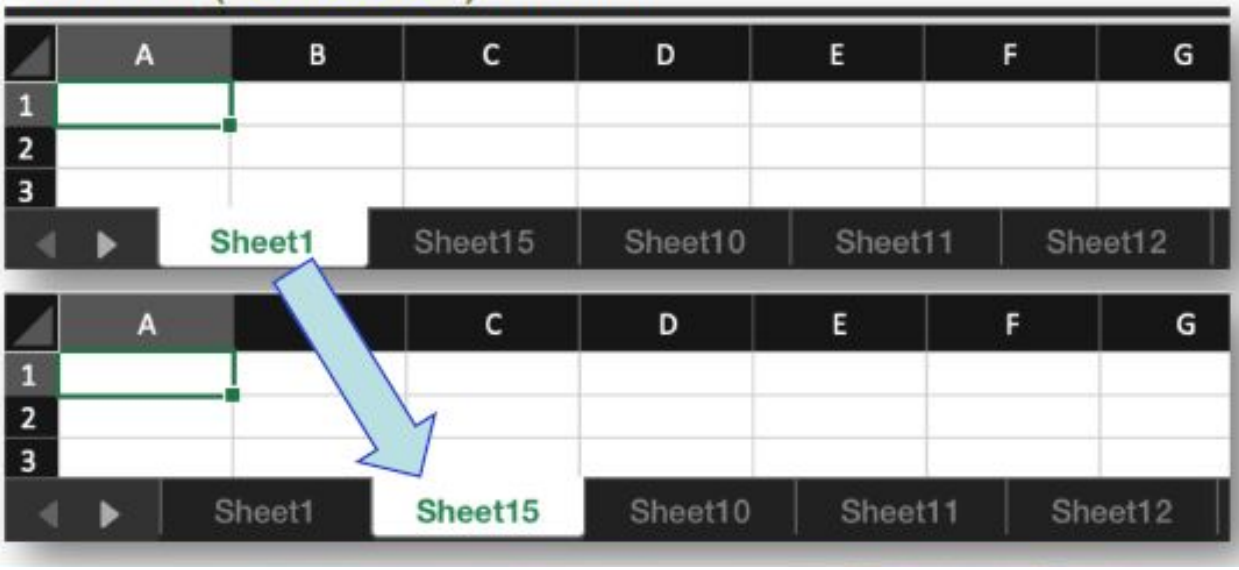
- Boundary
  - Range("A1").End(xlDown).Select




```
?Range("A1").End(xlDown).Row
 6
```

Object properties – Actions

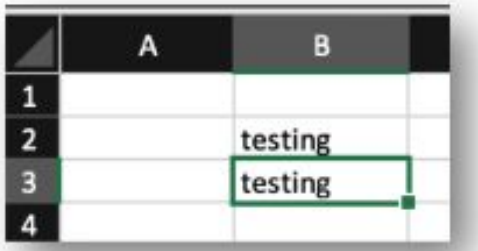
- English
  - Jackie runs / That person sings
- VBA (Actions are also known as Methods)
  - ActiveSheet.PrintOut
  - Workbooks("Book5").PrintPreview
  - Worksheets("Sheet15").Select



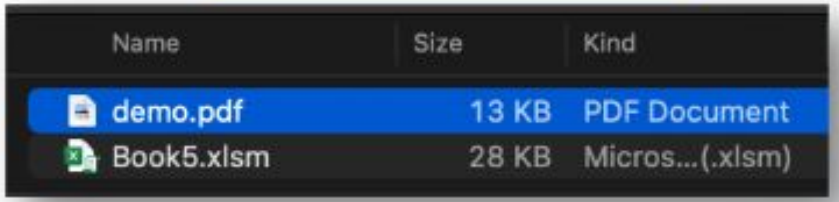
Object properties – More actions

- VBA – Some methods require parameters

- Copy & Paste
  - Range("B2").Copy Range("B3")

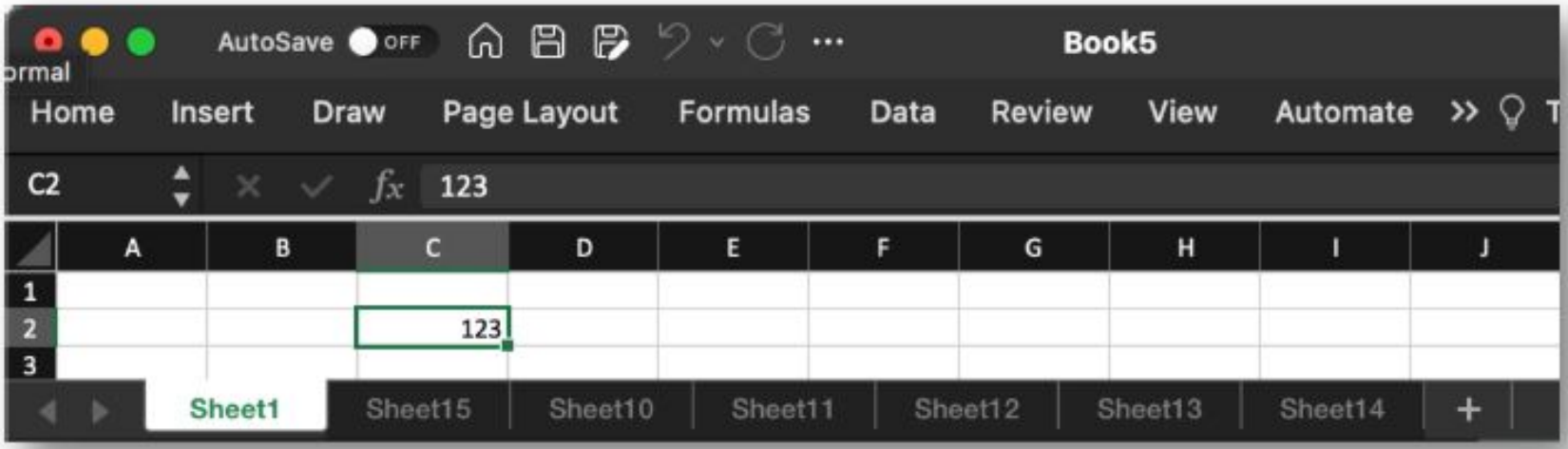


- Export to PDF
  - ActiveSheet.ExportAsFixedFormat Type:=xlTypePDF, Filename:="demo.pdf"



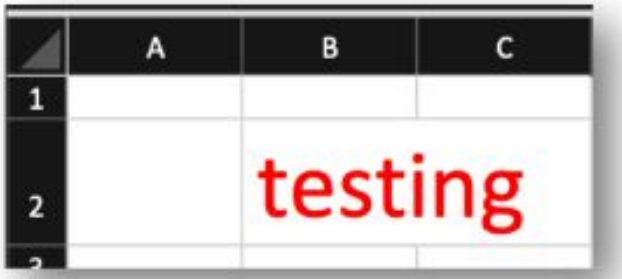
Object properties – Value

- English
  - HSBC ➔ Account 123-654321-012 ➔ Value ➔ $1000
- VBA
  - Workbooks("Book5").Worksheets("Sheet1").Cells(2,3).Value = 123



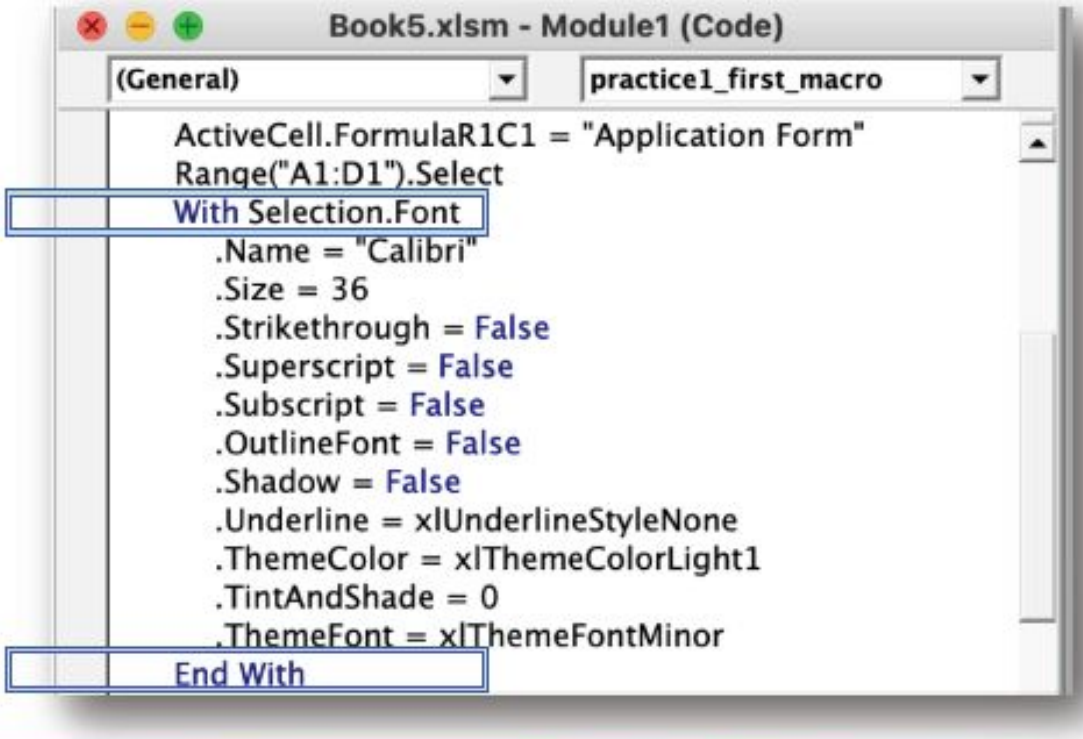
Object properties – Other properties

- VBA
  - Worksheets("Sheet1").Range("B2").Font.Size = 36
  - Worksheets("Sheet1").Range("B2").Font.Color = vbRed



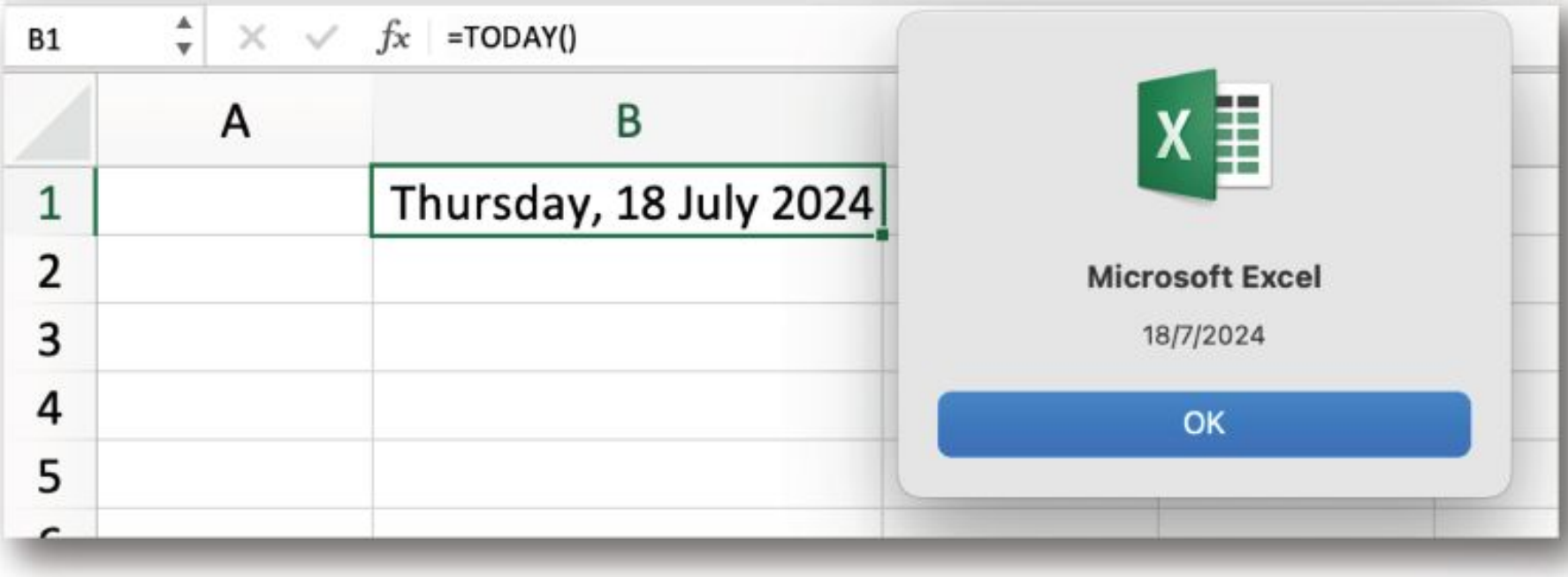
Object properties – Editing many properties

- "With" statement
  - Reduce redundancy
  - Example from Practice 1



Object properties – Value check

- Use “MsgBox”
  - MsgBox(Worksheets("Sheet1").Range("B1").Value)



Object properties – Value check

- Use “Debug.Print()”

```
Sub HighlightCell()
    If Range("B1").Value > 50 Then
        Range("B1").Interior.Color = vbYellow
    Else
        Range("B1").Interior.Color = vbGreen
    End If
    Debug.Print (Range("B1"))
End Sub
```

```
40
```

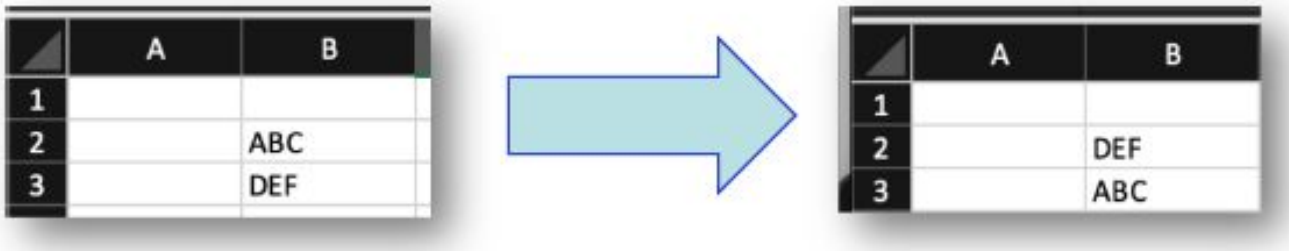
- Use “?” in the immediate window
  - ?ActiveWorkbook.Sheets.Count
  - ?Range("B2").Formula

```
Immediate
?ActiveWorkbook.Sheets.Count
 7
```

```
?Range("B2").Formula
=TODAY()
```

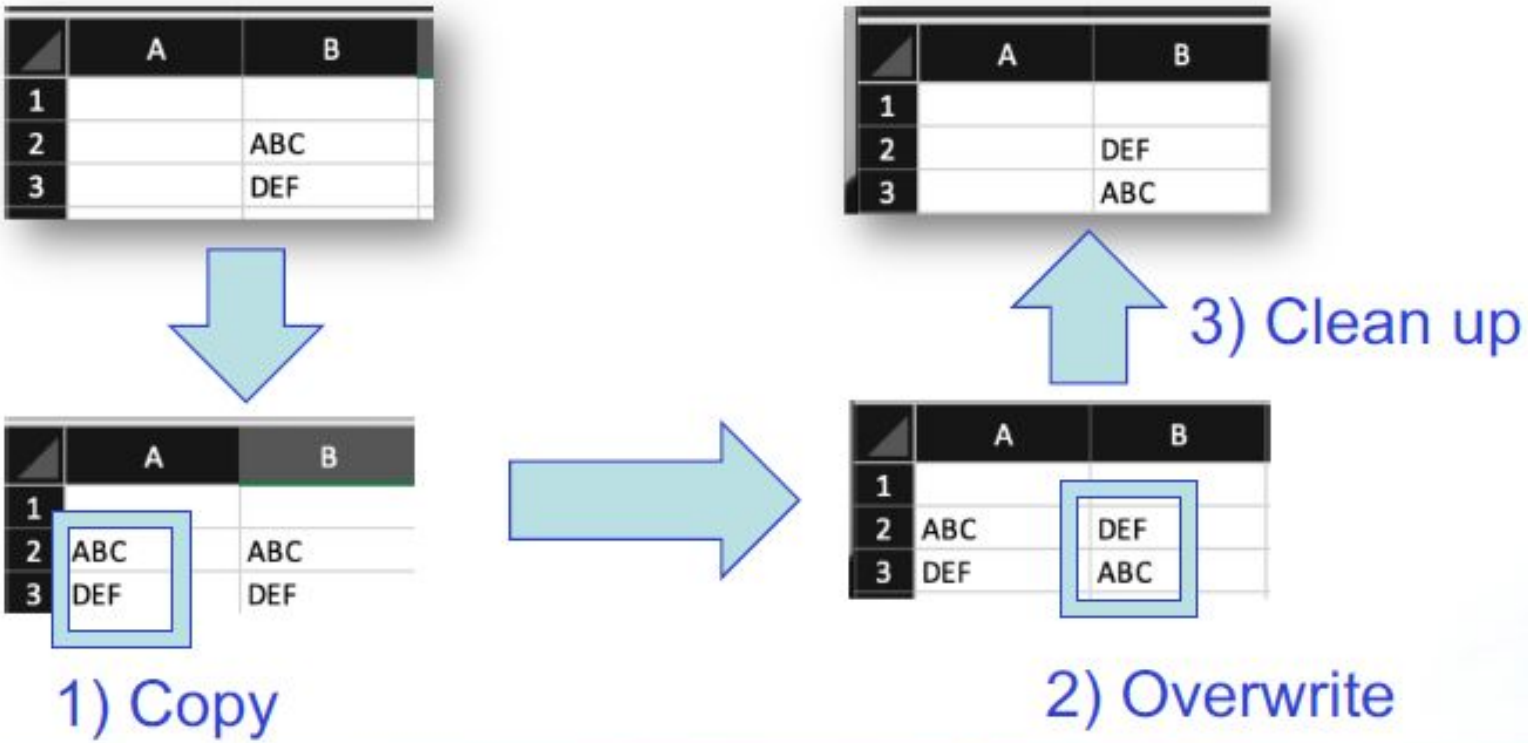
Checkpoint – Value swapping

- How to swap (switch) values between 2 cells?



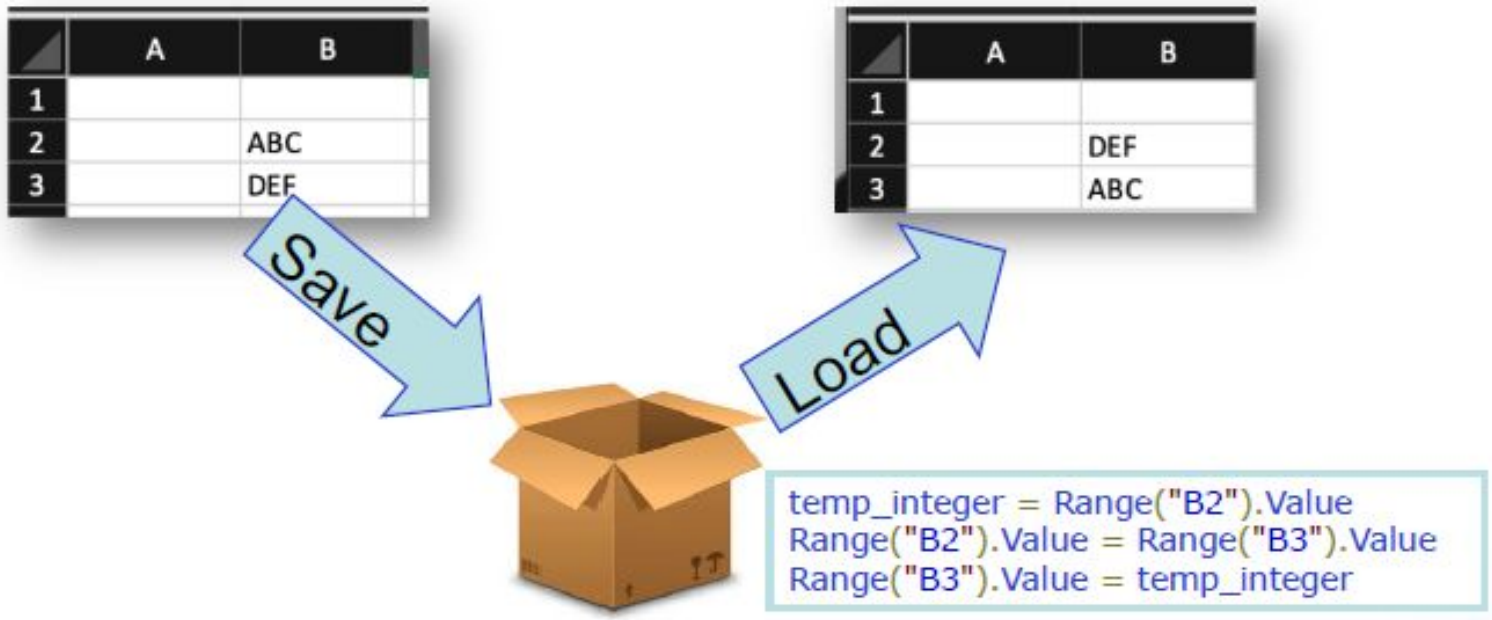
Checkpoint – Value swapping

- How to swap (switch) values between 2 cells?
  - 1 solution…



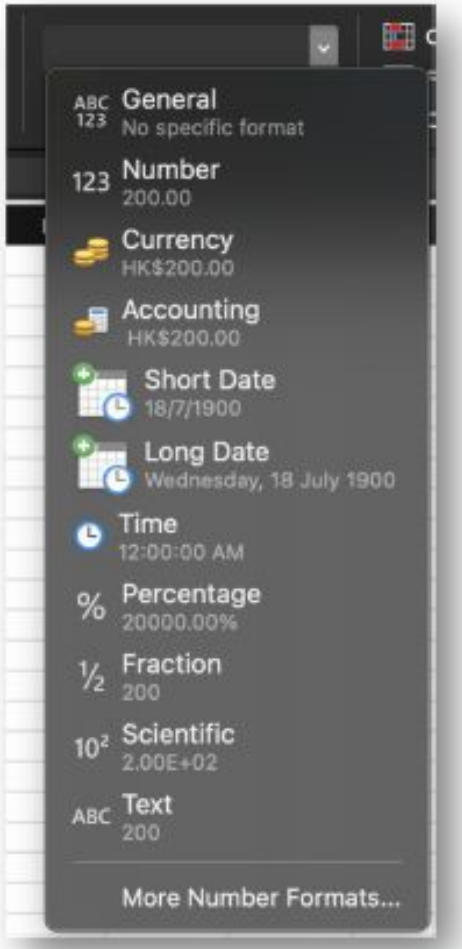
Variables – Motivation

- Storage of values without changing the sheets
- Meaningful references during calculation

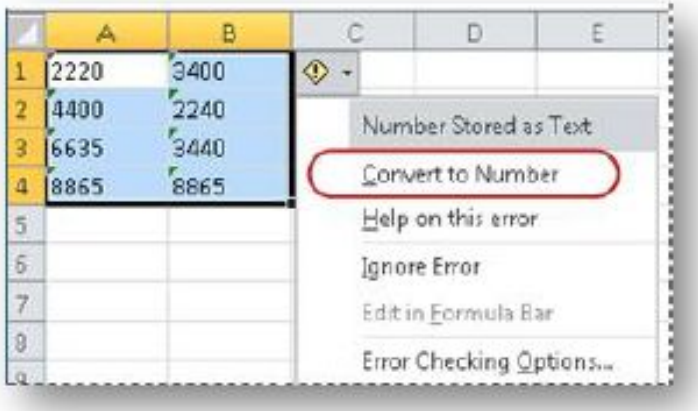


Variables – Data Types

- Excel is good at detecting types
  - Numbers / Date / Text



- Though sometimes can still be wrong



1. *Fix text-formatted numbers by applying a number format - Microsoft Support*. Fix Text-formatted Numbers by Applying a Number Format - Microsoft Support. https://support.microsoft.com/en-us/office/fix-text-formatted-numbers-by-applying-a-number-format-6599c03a-954d-4d83-b78a-23af2c8845d0

Variables – Data Types

- Best to define variables with types

Commonly Used VBA Data Types

Data Type	Bytes Used	Range of Values
Integer	2	-32,768 to 32,767
Long	4	-2.147 Billion to 2.147 Billion
Single	4	45 digits (left or right of decimal point)
Double	8	308 digits (left or right of decimal point)
Currency	8	-922 T to 922 T (4 digits after decimal point)
Date	8	1/1/100 to 12/31/9999
Boolean	2	True or False
String	1 per char	Varies according to number of characters
Variant	Varies	Any data type

Variables – Data Types

- Integer
  - Dim year As Integer
  - year = 2024
- Double
  - Dim pi As Double
  - pi = 3.14125

- Smart conversion during calculation

```
?TypeName( "A" & 3 )
String

?"A"&3
A 3
```

```
?TypeName(2)
Integer

?TypeName(3.3)
Double

?TypeName(2 * 3.4)
Double

?TypeName(5-3)
Integer
```

Variables – Data Types

- Date
  - Dim today As Date
  - today = DateSerial(2024, 7, 1)
  - today = Date()
- String
  - Dim message As String
  - message = "Welcome!"
- Boolean
  - Range("A1").Font.Bold = True
  - Range("A1").Font.Underline = False

Common functions - Numbers

- IsNumeric() ➔ Check if a text is number

```
?IsNumeric("123")
True

?IsNumeric("ABCD")
False
```

- CInt() / CDbl() ➔ Text to number

```
?CInt("123")
 123

?CInt("123.45")
 123

?CDbl("123.45")
 123.45

?CDbl("123")
 123
```

Common functions - Numbers

- Round() ➔ Rounding to a certain decimal place

```
?Round(5.1234)
 5

?Round(5.1234, 2)
 5.12
```

- Log() / Exp() ➔ For financial / engineering calculation

```
?exp(2)
 7.38905609893065

?log(10)
 2.30258509299405

?log(exp(1))
 1
```

Common functions - Date

- DateSerial ➔ Create a date literal

```
?DateSerial(2024, 12, 25)
25/12/2024
```

- Date() ➔ Today's date

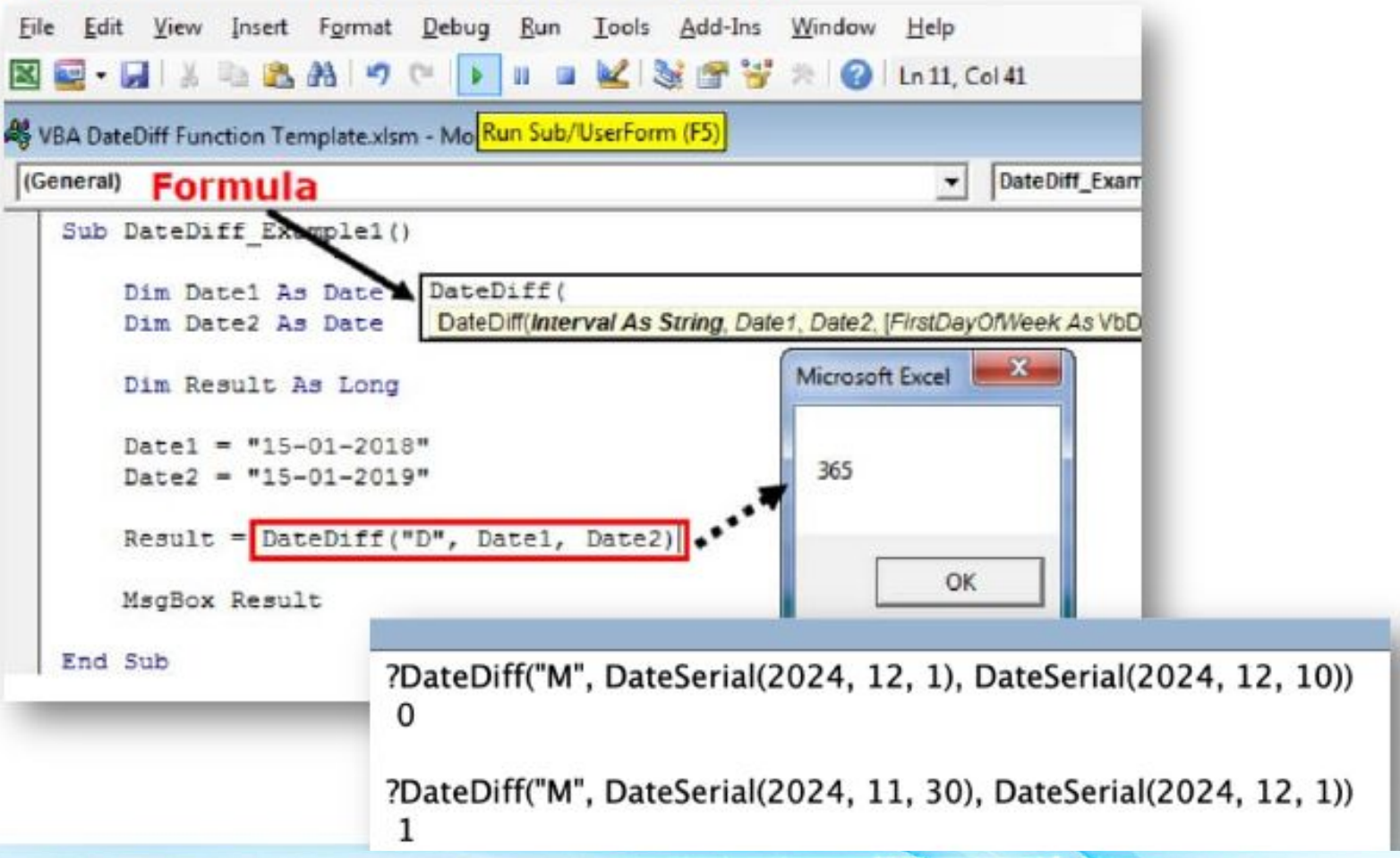
```
?Date()
18/7/2024
```

- Year() / Month() / Day() ➔ Extract

```
?Year(DateSerial(2024, 12, 25))
 2024

?Month(DateSerial(2024, 12, 25))
 12

?Day(DateSerial(2024, 12, 25))
 25
```

Common functions - Date

- DateDiff – Difference of 2 dates



Common functions - String

- CDate() ➔ Text to Date

```
?CDate("2023-03-25")
25/3/2023

?CDate("April 1, 2023")
1/4/2023

?CDate("25/2")
25/2/2023
```

- Format() ➔ Date to Text

```
?Format(DateSerial(2023, 3, 25), "M")
3

?Format(DateSerial(2023, 3, 25), "MM")
03

?Format(DateSerial(2023, 3, 25), "MMM")
Mar

?Format(DateSerial(2023, 3, 25), "MMMM")
March

?Format(DateSerial(2023, 3, 25), "DD-MMM-YYYY")
25-Mar-2023
```

Common functions - String

- Left() / Right() ➔ Substring

```
?Left("0005.HK", 4)
0005

?Right("0005.HK", 2)
HK
```

- Split() ➔ Tokenizing into a string array

```
?Split("A,B,C", ",")(0)
A

?Split("A,B,C", ",")(2)
C
```

```
?UBound(Split("A,B,C", ","))
 2
```

Common functions - String

- Trim() ➔ Remove empty spaces
  - Note: "" gives 1 quote

```
?""""" & Trim("    Hello!    ") & """""
"Hello!"
```

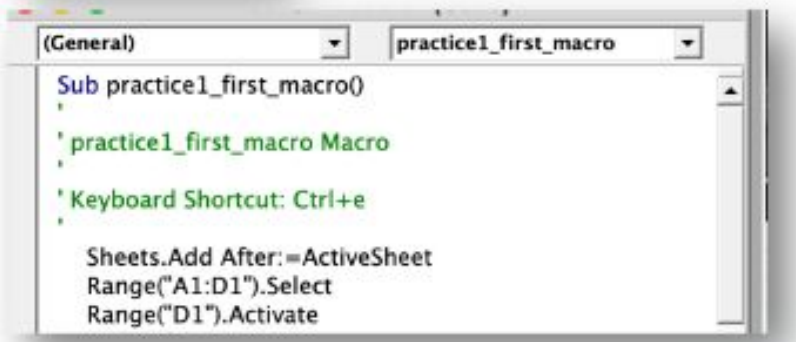
- InStr(x, y) ➔ Find y in x
  - Case sensitive

```
?InStr("Search here", "Search")
 1

?InStr("Search here", "here")
 8

?InStr("Search here", "nothing")
 0

?InStr("Search here", "search")
 0
```

```
?Lcase("HeLLo")
hello

?InStr(Lcase("Search here"), "search")
 1
```

Sub-routines

- Macros recorded are wrapped with "Sub … End Sub"
- Sub-routines are just reusable codes
- Within the scope of a sub-routine
  - Click the "Play" button to execute



Control flow – If, Then, Else

- Example: speed check


```
Sub HighlightCell()

    If Range("B1").Value > 50 Then
        Range("B1").Interior.Color = vbYellow
    Else
        Range("B1").Interior.Color = vbGreen
    End If

End Sub
```

Control flow – If, Then, Else

- Mark ➔ Grade

Mark	*Letter Grade*
90+	A+
85-89	A
80-84	A-
77-79	B+
73-76	B
70-72	B-
67-69	C+
63-66	C
60-62	C-
57-59	D+
50-56	D
0-49	F

Control flow – Select, Case

- Range check

```
?GetGrade(90)
A

?GetGrade(80)
B
```

```
Function GetGrade(score As Integer) As String
        Select Case score
        Case Is >= 90
                GetGrade = "A"
        Case Is >= 80
                GetGrade = "B"
        Case Is >= 70
                GetGrade = "C"
        Case Is >= 60
                GetGrade = "D"
        Case Else
                GetGrade = "F"
        End Select
End Function
```

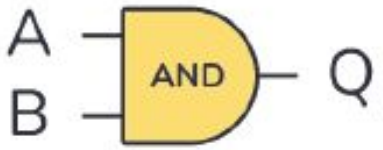
Control flow – Select, Case

- Grade ➔ Marks

```
Dim letter_grade As String
Dim grade_point As Double

Select Case letter_grade
Case "A", "A+"
        grade_point = 4.0
Case "A-"
        grade_point = 3.7
Case "B+"
        grade_point = 3.3
...
Case Else
        grade_point = 0.0
End Select
```

Control flow – Conditions

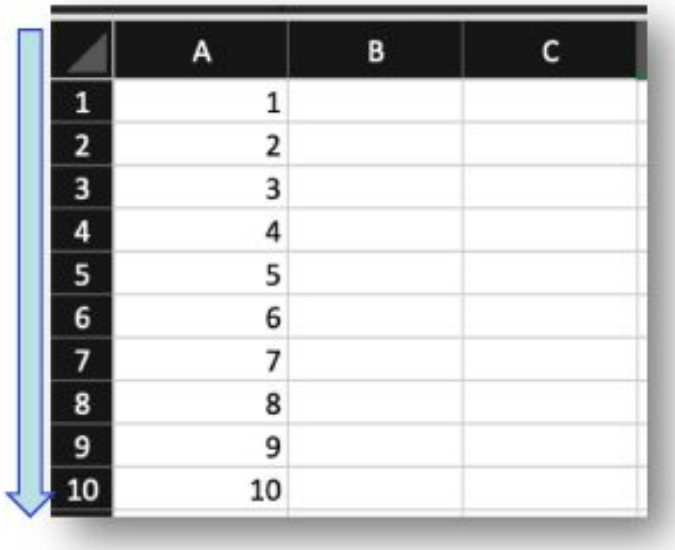
- AND
  - (True AND True) ➔ True
  - e.g., Month validity: 1 <= month AND month <= 12
- OR
  - (True OR False) ➔ True
  - e.g., Game Area: gold > 1000 OR level > 10
- NOT
  - NOT x=0 is same as x <> 0



A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1

Loops – Using "For"

- Iterating through the rows
- What is changing?
  - A1 ➔ A2 ➔ A3 ➔ A4…

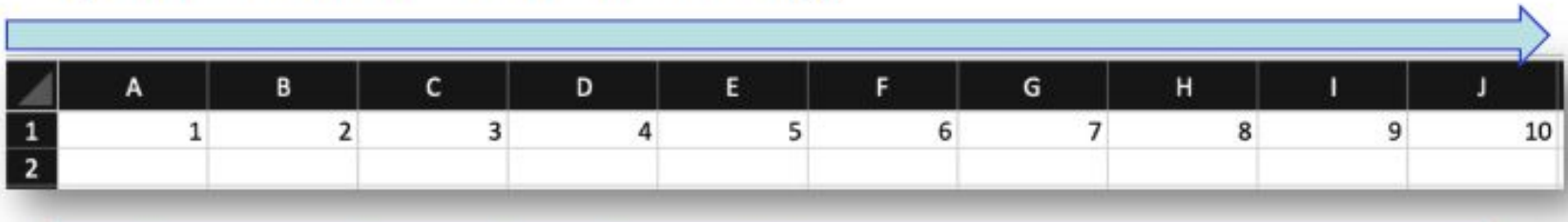


	A	B	C
1	1		
2	2		
3	3		
4	4		
5	5		
6	6		
7	7		
8	8		
9	9		
10	10		

```
Sub LoopThroughRows()
        For i = 1 To 10
                Range("A" & i).Value = i
        Next i
End Sub
```

Loops – Using "For"

- Iterating through the columns
- Again, track the change: A1 ➔ B1 ➔ C1 ➔ …
- Option 1: Make use of offset()



	A	B	C	D	E	F	G	H	I	J
1	1	2	3	4	5	6	7	8	9	10
2										

```
Sub LoopThroughColumns()
        For i = 1 To 10
                Range("A1").Offset(0, i - 1).Value = i
        Next i
End Sub
```

Loops – What is happening?

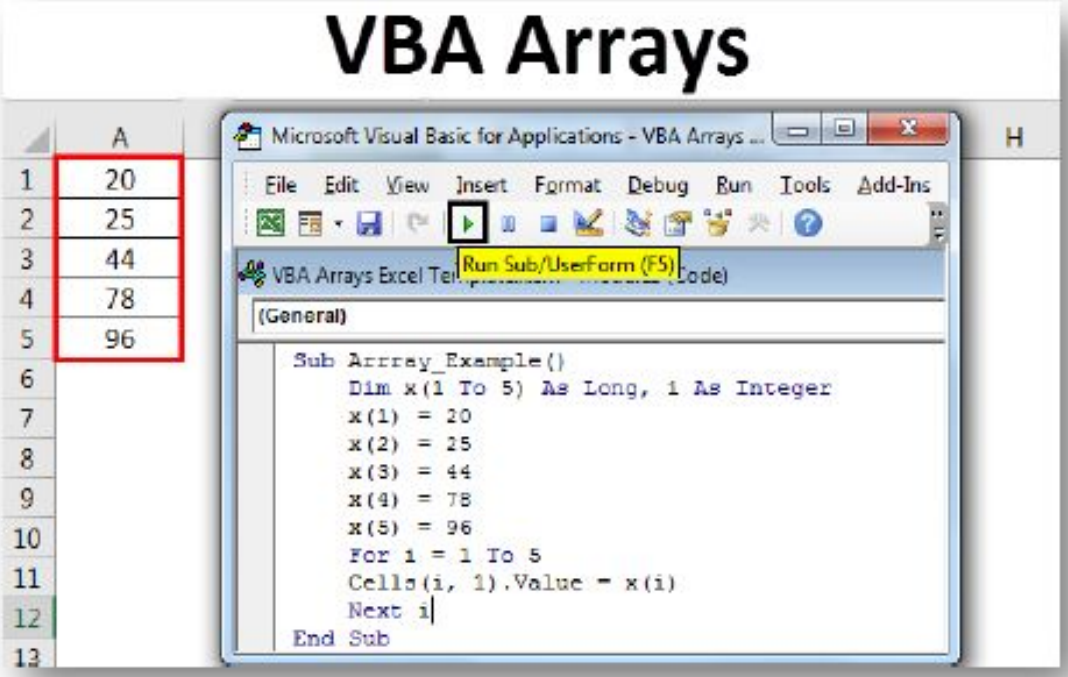
- Temporary variable + "Array"

```
Sub LoopThroughRows()
        For i = 1 To 10
                Range("A" & i).Value = i
        Next i
End Sub
```

```
Sub LoopThroughColumnsWithCells()
        Dim i As Integer: i = 1
        For Each cell In Range("A1:J1").Cells
                cell.Value = i
                i = i + 1
        Next cell
End Sub
```

Loops – What is happening?

- Array = many variables



Extra Lab on Generative AI

https://colab.research.google.com/drive/1E6Hdzvp0lgUl1cG_Ddd0FCYQ9-FG7OJ2

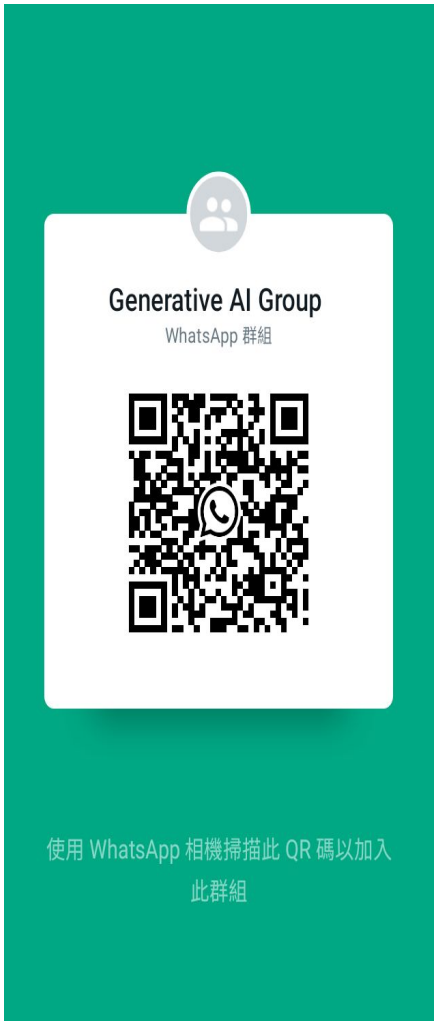
Main Program

```
prompt = """write a 10 words with 9 letters."""
#prompt = """write a proposal for starting an fashion ai business """

with torch.no_grad():
  token_ids = tokenizer.encode(prompt, add_special_tokens=False ,return_tensors="pt")
  output_ids = model.generate(
      token_ids.to(model.device),
      max_new_tokens=512,
      do_sample=True,
      temperature = 0.8
  )

output = tokenizer.decode(output_ids[0][token_ids.size(1) :])
print(output)
```

My Generative AI Group

Please Join My Generative AI Group!

I will update more latest information to all of you.



HKUSPACE
香港大學專業進修學院
HKU School of Professional and Continuing Education

THANK YOU