COMP1022Q
Introduction to Computing with Excel VBA

Handling Excel Events

Gibson Lam and David Rossiter

Outcomes

- After completing this presentation, you are expected to be able to:

  1. Understand Excel events

  2. Write VBA code to handle a workbook event

  3. Write VBA code to handle a worksheet event

Overview

- So far, we have only used macros to start running some VBA code

- In this presentation, we will talk about *events* and how you can make VBA code to do something when the event occurs

- 'Workbook' means 'the Excel file'

Excel Events

- Lots of events might happen in Excel

- Here are some examples:

  - When you select a cell in a worksheet, your action generates a 'selection change' event

  - When you open an Excel file (=workbook), your action generates a 'workbook open' event

- In this discussion, we will look at Excel events associated with a workbook

Handling the Events

- You can write some code for a specific event so that when that event occurs, the system runs the code that you have written for the event

- The code that you write for an event is called an *event handler*, which means code that 'handles' the event

- For example, you can write some code so that when you select a cell on a worksheet, the cell will change to have a red background
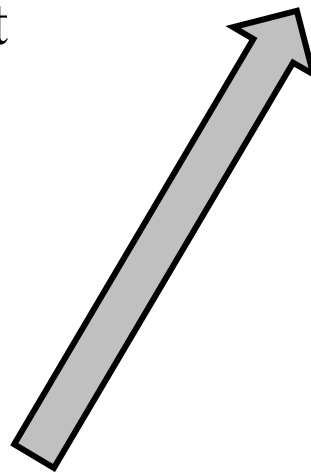
Handling an Event – an Example

```
Sub Worksheet_SelectionChange(   ...   )
    ActiveCell.Interior.Color = vbRed
End Sub
```

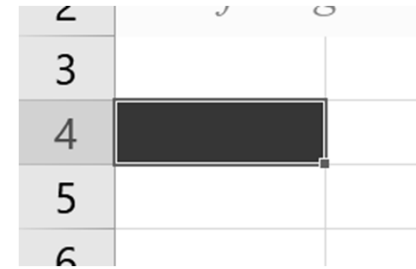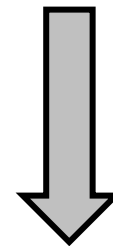*ActiveCell means the currently selected cell*

1. You write a VBA subroutine to handle the 'selection change' event
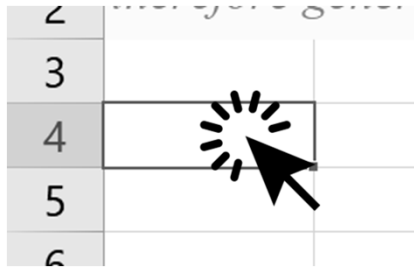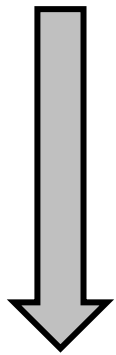
2. You select a cell on the worksheet, that means a 'selection change' event is generated

3. Because the 'selection change' event has occurred, the event handler code (shown above) is run

4. The event handler code (shown above) does something i.e. changes the newly selected cell to red

This part means:
- Something is passed to the subroutine
- The thing passed to the subroutine is a `Range` object (meaning that it is a cell)
- It contains details of the cell which was selected, although those details aren't used in the Sub
- When it comes to the Sub, it is put in a variable called `Target`, although that variable isn't used
- If we have time we might look at what `ByVal` means later, let's ignore it for now

If we have time we might look at what `Private` means later, let's ignore it for now

```
Private Sub Worksheet_SelectionChange(

    ByVal Target As Range )

    ActiveCell.Interior.Color = vbRed

End Sub
```

The Full Subroutine Code

Workbook and Worksheet in VBA

- When you open the VBA editor you can see the workbook (i.e. the Excel file) and worksheet(s) as part of the project, like this:

- What we can do in the workbook and worksheet(s) is to write VBA code to handle events that occur to them

- The events that occur for the workbook are different to the events that occur for a worksheet
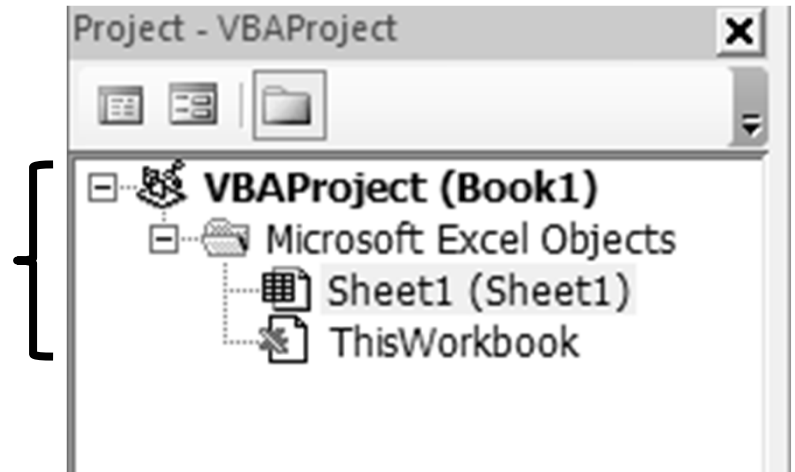
Examples of Events

- Here are two commonly used events for **workbooks**:
    - *Open* – occurs when the Excel file is opened
    - *BeforeClose* – occurs before the Excel file is closed

- Here are two commonly used events for **worksheets**:
    - *Change* – occurs when something is changed in a cell
    - *SelectionChange* – occurs when the selection is changed, i.e. when you select cell(s) different to the currently selected cell(s)

Writing VBA Code for the Workbook Open Event

- In the next few slides we will demonstrate one way to write simple VBA code for the 'workbook open' event

- The procedure is:

  1. Open the VBA code area for the workbook in the VBA editor

  2. Select the event

 3. Write the VBA code to handle the event - this is a subroutine

Opening the VBA Code
for the Workbook

- Double-click on 'ThisWorkbook' to open the VBA code area for the workbook

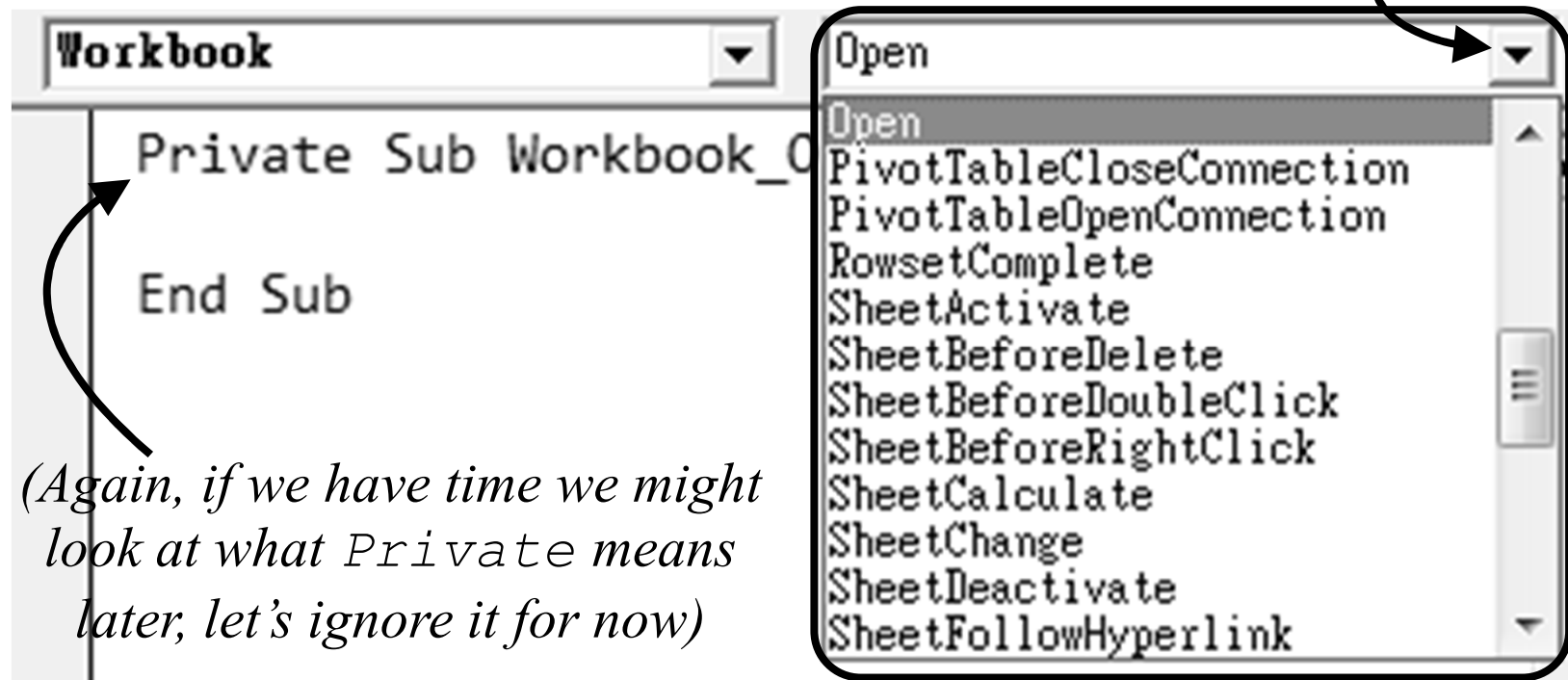- At the moment, the VBA code window for the workbook is empty
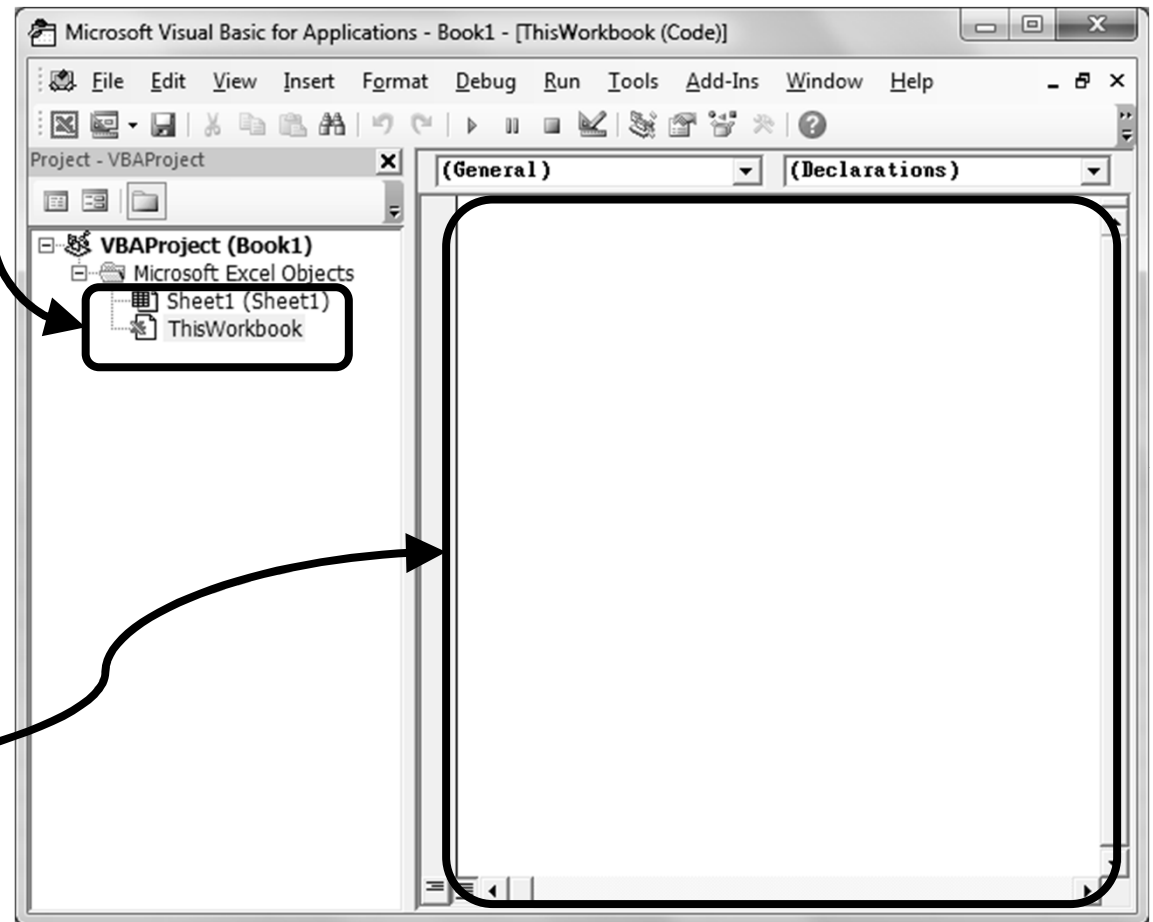
Selecting Events of the Workbook

- At the top of the VBA code window we can then select the event we want to write our VBA code for

- First, we select 'Workbook' in the box on the left

- Once 'Workbook' is selected, the VBA editor automatically assumes you want to write code for the 'Open' event

- Therefore you will see the *event handler*, `Workbook_Open()`, which is for the workbook open event, showing up in the editor
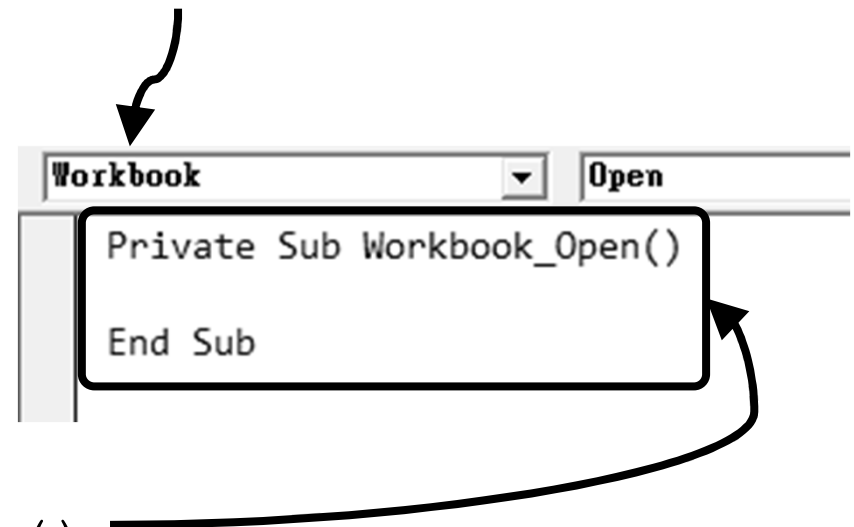
Exploring the Events

- In case you want to write code for another workbook event, you can select the event you want


```
Workbook                    ▼    Open                           ▼

    Private Sub Workbook_O          Open
                                    PivotTableCloseConnection
                                    PivotTableOpenConnection
    End Sub                         RowsetComplete
                                    SheetActivate
                                    SheetBeforeDelete
                                    SheetBeforeDoubleClick
                                    SheetBeforeRightClick
(Again, if we have time we might     SheetCalculate
  look at what Private means         SheetChange
  later, let's ignore it for now)    SheetDeactivate
                                    SheetFollowHyperlink
```

- In our example, we want to write code for the 'workbook open' event, so we don't need to select something else right now

Writing the Example Code

- Let's write one line of VBA code in the subroutine, like this:

```
Private Sub Workbook_Open()
    Msgbox "How are you doing today?"
End Sub
```

Remember that Sub means Subroutine, which just means a group of VBA code

Running the Code

- To run the code, we can use the ▶ button, similar to what we do when we write a macro

- Or, to properly check whether the event is working, we can save the file, close it and then open it again

- Right after the file has opened, we will see the message box popping up like this:

Writing VBA Code for Other Events

- You can now write VBA code for any workbook event using the same method

- You can also try to write VBA code for worksheet events

- Note that not all event handlers can be tested using the ▶ button

  - For example, you cannot use the ▶ button to run the selection change event