Office Scripts Introduction

Sample manual - first two chapters

Wise Owl Training

TABLE OF CONTENTS (1 of 5)

TABLE OF CONTENTS (2 of 5)

TABLE OF CONTENTS (3 of 5)

TABLE OF CONTENTS (4 of 5)

TABLE OF CONTENTS (5 of 5)

CHAPTER 1 - GETTING STARTED

1.1 Introduction to Office Scripts

This chapter introduces you to *Office Scripts*, beginning with the answers to a few simple questions.

What are Office Scripts?

Office Scripts are small programs used to automate tasks in Microsoft Excel. You write code in the *TypeScript* language and you can either record the steps or write your program from scratch.

```
function main(workbook: ExcelScript.Workbook) {

    // Get a reference to the Input worksheet
    let ws: ExcelScript.Worksheet = workbook
        .getWorksheet("Input");

    // Check that the worksheet exists
    if (!ws) {
        console.log("Input sheet missing");
        return;
    }

    // Get reference to next blank cell
    let rng: ExcelScript.Range = ws
        .getRange("A1")
        .getRangeEdge(ExcelScript.KeyboardDirection.down)
        .getOffsetRange(1, 0);

    // Enter value and format cell
    rng.setValue("Wise Owl");
    rng.getFormat().getFill().setColor("#f28400");
    rng.getFormat().getFont().setName("Arial");
}
```

A basic Office Script to add a value to the end of a list and apply some formatting.

What do I need to use Office Scripts?

According to Microsoft, these are the three requirements for using Office Scripts:

1) Excel for Windows, for Mac, or on the web.

2) OneDrive for Business.

3) Any commercial or educational Microsoft 365 license with access to the Microsoft 365 Office desktop apps.

Can't I already Automate Excel?

If you're thinking that Office Scripts sound a lot like *VBA macros*, you're right! In fact, there are three main ways to automate Excel, summarised in the table below:

Method	What you can do
Office Scripts	Automate the desktop or online version of Excel.
VBA Macros	Automate any desktop Office application.
Office Add-Ins	Create an add-in to extend the features of any desktop or online Office application.

Why use Office Scripts and not VBA?

Office Scripts allow you to do some things you can't do with VBA macros, as described in the table below:

Feature	Description
Excel online	You can use Office Scripts to automate workbooks in the web version of Excel.
Power Automate	You can run Office Scripts from a Power Automate flow.
External APIs	Office Scripts support calls to external APIs which can provide data to your files.
Better security	An Office Script only has access to the workbook in which it is running, unlike VBA which has access to your entire computer.

What are the Limitations?

Office Scripts have several limitations compared to VBA and Office Add-Ins, as described in the table below:

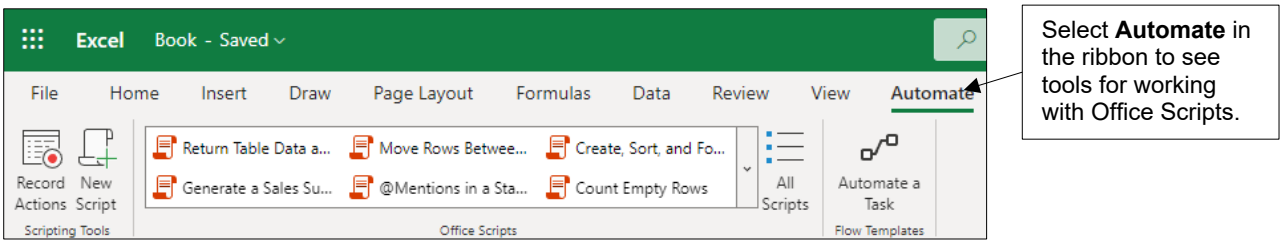
Feature	Description
Excel only	Currently, Office Scripts can only be used in Excel. Office Add-Ins can also work with Word, PowerPoint, Outlook, OneNote and Project. VBA can use OLE and COM libraries to control a variety of other applications, including the Microsoft Office apps.
No events	VBA and Office Add-Ins can respond to events to make code run automatically. Office Scripts must be run explicitly by the user.
Single workbook	An Office Script only has access to the workbook in which it is running. Even something as simple as copying a value from one workbook to another requires the use of a Power Automate flow.
User interface	Office Scripts don't have access to any of the user interface elements of Excel. This means that you can't display dialog boxes or similar UI elements. If you need to create a user-interface it would be better to use VBA or to create a full Office Add-In.

1.2 Preparing to Write Office Scripts

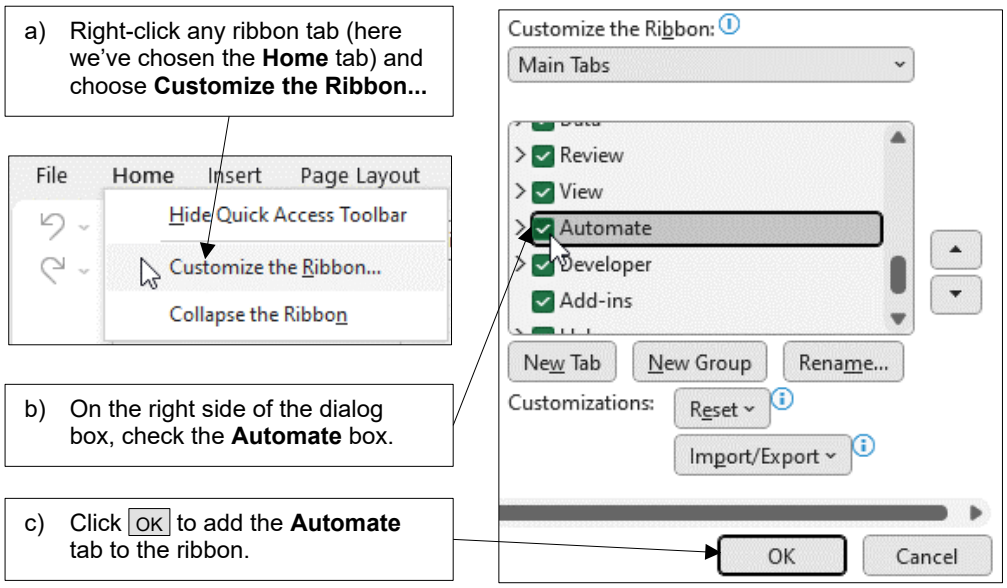
This section shows you the basic things you need to do before you can write your first script.

The Automate Ribbon Tab

The *Automate* tab of the ribbon contains the tools you need to write Office Scripts.

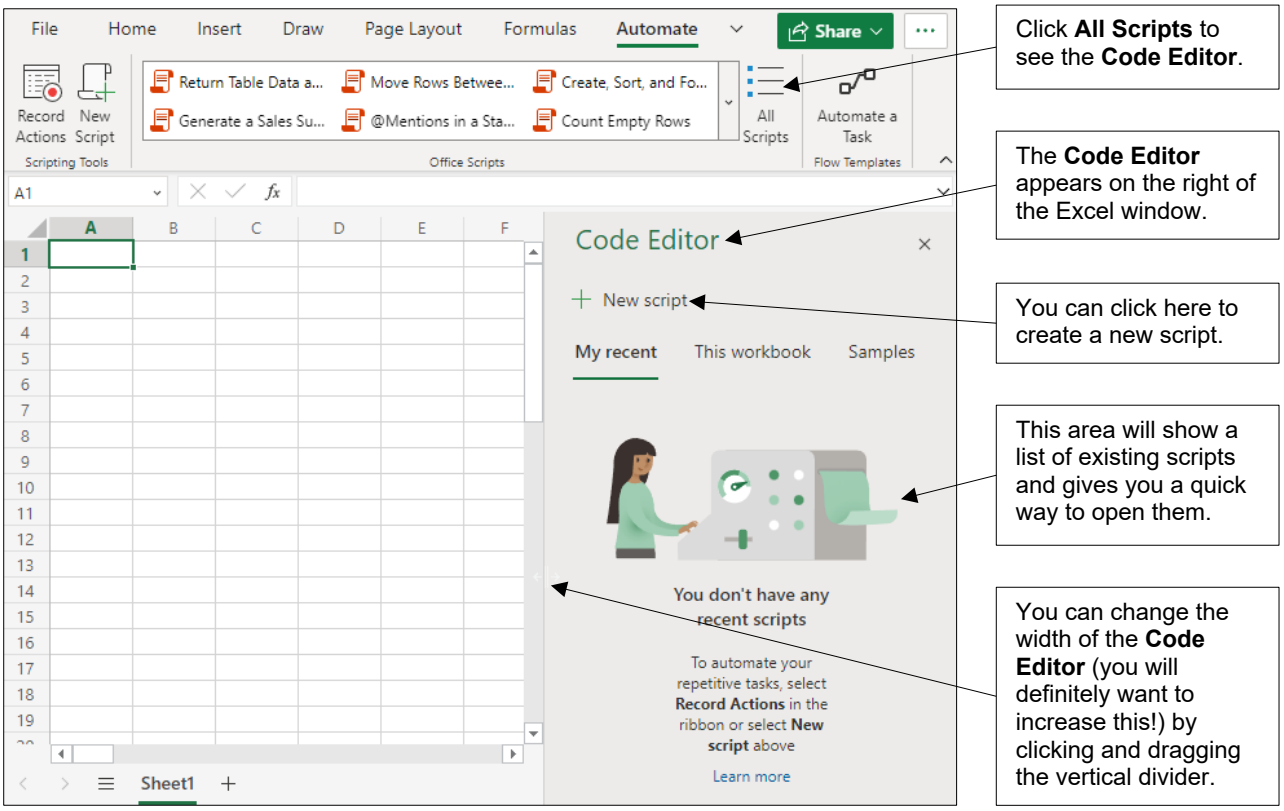


If you're using the desktop version of Excel, the **Automate** tab may not appear automatically. You can see how to display it in the diagram below:

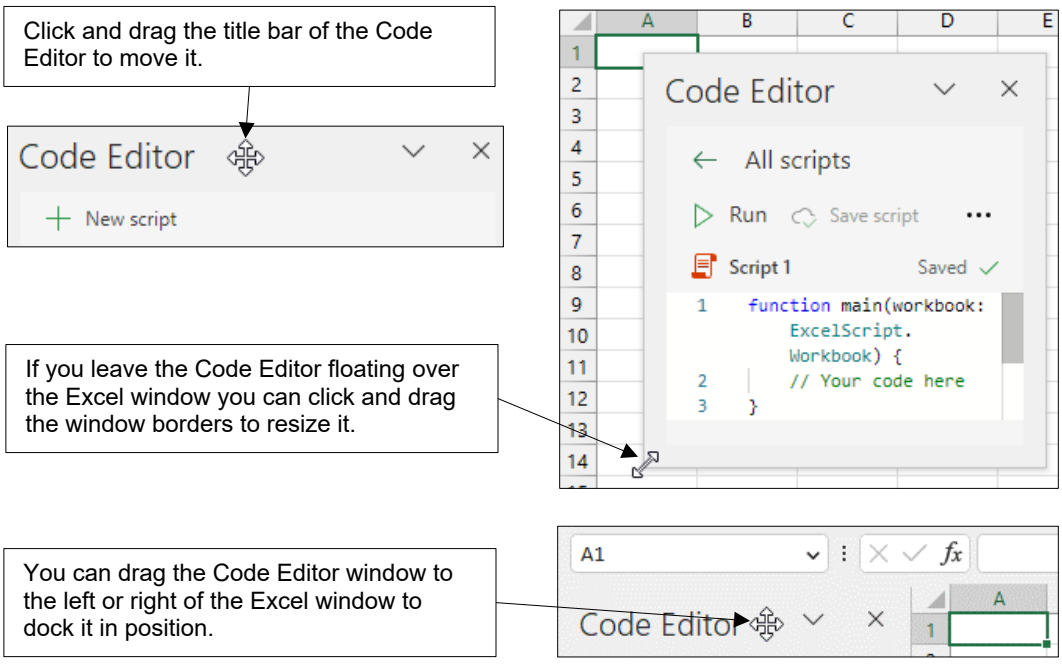


The Code Editor

You edit Office Scripts in the *Code Editor* window in Excel. You can open the **Code Editor** in several ways; a simple option is to choose **Automate | All Scripts** from the Excel ribbon.

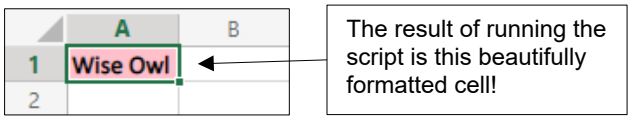


In the desktop version of Excel you can move and resize the Code Editor window by clicking and dragging its title bar.



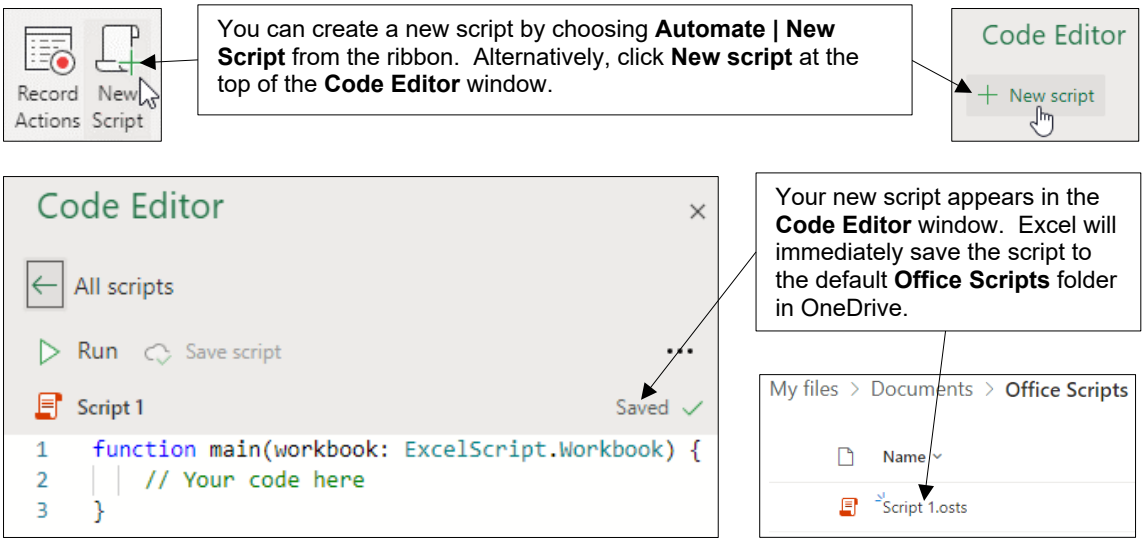
1.3 Writing Your First Script

This section shows you how to write a simple program to add text to a cell and apply some formatting to get used to the basics of writing code.



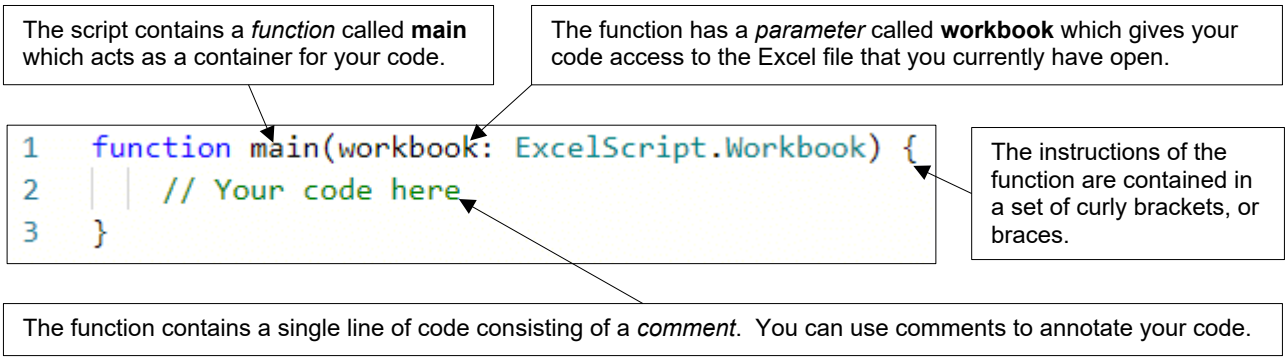
Creating a New Script

Start by creating a new workbook in either the desktop or online version of Excel and then choose to create a new Office Script.



The Basic Structure of a Script

You can see the basic structure of your new script in the diagram below:



Adding a Comment

You can add your own comments to your code to provide helpful reminders of what it is meant to do.

```
1   function main(workbook: ExcelScript.Workbook) {
2       // Your code here
3
4
5   }
```

Click at the end of the line **2** then press Enter twice to provide a new line on which to write your comment.

```
1   function main(workbook: ExcelScript.Workbook) {
2       // Your code here
3
4       // Enter my name in the selected cell
5   }
```

Type two forward slashes to begin the comment and follow these with anything you feel is appropriate.

Writing Basic Instructions

Our first instruction will write the name **Wise Owl** into the currently selected cell in the workbook. To do this:

1) On a new line in the script, begin typing the word **workbook** (take care when doing this – TypeScript is a case-sensitive language!).

As you begin typing you'll see a list of options matching what you've typed so far. To select an item from this *Intellisense* list you can press Tab or Enter or click on the word with the mouse.

```
1   function main(workbook: ExcelScript.Workbook) {
2       // Your code here
3
4       // Enter my name in the selected cell
5       work
6   }
            workbook                (parameter)
            Worker
```

2) After the word **workbook**, enter a full stop and begin typing **getActiveCell**.

```
4       // Enter my name in the selected cell
5       workbook.getA
6   }
            getActiveCell           (method)
```

Use the Intellisense list to help you finish entering the word.

3) After **getActiveCell**, open and close some round brackets.

```
4       // Enter my name in the selected cell
5       workbook.getActiveCell()
6   }
```

You can type in the open round bracket and the closing bracket will be added for you automatically.

4) Enter another full stop followed by **setValue** then an open round bracket and double-quotes.

```
4       // Enter my name in the selected cell
5       workbook.getActiveCell().setValue("")
6   }
```

The other double-quote and closing round bracket will be added for you automatically.

5) Between the double quotes, enter the name you want to write into the cell.

```
4       // Enter my name in the selected cell
5       workbook.getActiveCell().setValue("Wise Owl")
6   }
```

You can change the text in the double quotes to anything you like.

6) Complete the instruction by adding a semicolon to the end of the line.

```
4       // Enter my name in the selected cell
5       workbook.getActiveCell().setValue("Wise Owl");
6   }
```

Each TypeScript instruction ends with a semicolon.

Adding More Instructions

You can continue adding as many instructions as you need to complete your program. The diagram below shows the complete script with two extra instructions and relevant comments:

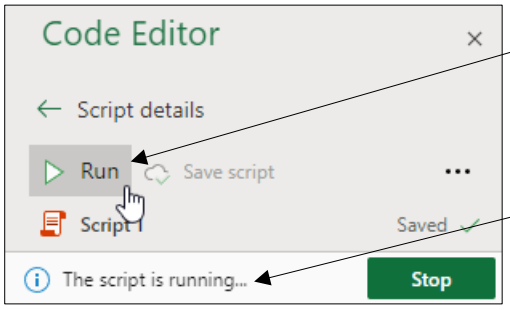
```
1   function main(workbook: ExcelScript.Workbook) {
2
3       // Enter my name in the selected cell
4       workbook.getActiveCell().setValue("Wise Owl");
5
6       // Change the cell fill colour
7       workbook.getActiveCell().getFormat().getFill().setColor("Pink");
8
9       // Make the font bold
10      workbook.getActiveCell().getFormat().getFont().setBold(true);
11  }
```

We've deleted the first comment to tidy up the code.

The pattern of the new instructions is similar to the first one we added.

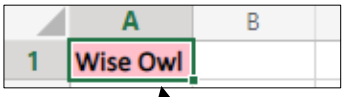
Running the Script

You can see how to run your finished script in the diagram below:



Click **Run** at the top of the Code Editor to run the script that you currently have open.

A message appears to tell you the code is running or, if you're unlucky, that the code has failed! You can click **Stop** if you need to abandon a script that appears to be stuck.



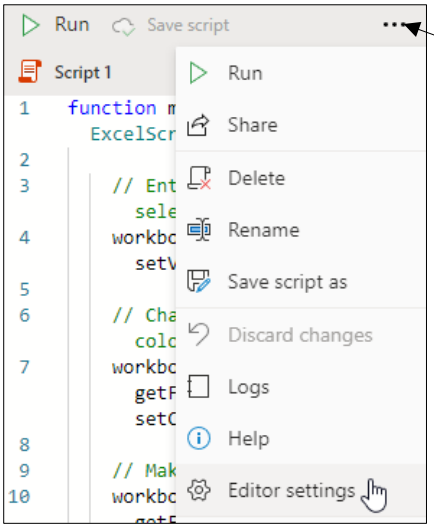
The beautiful end result!

Experienced VBA programmers should be careful not to press F5 to run your code. In the online version of Excel this will refresh the page and close the Code Editor!

1.4 Code Editor Settings

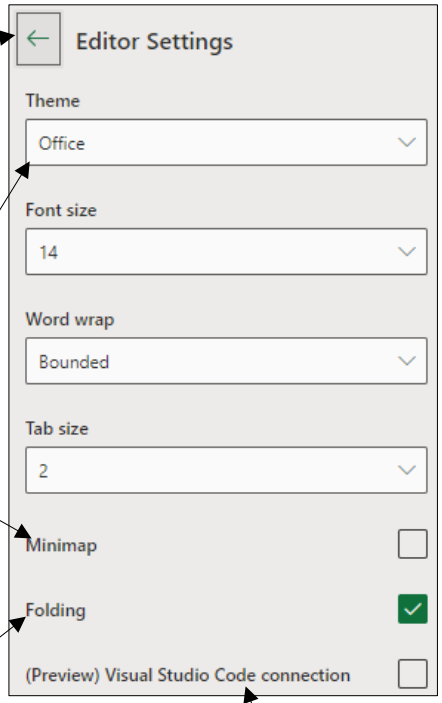
Changing Code Editor Settings

To change Code Editor settings you first need to open a script file.



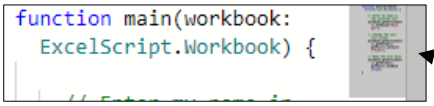
Click ... at the top right of the Code Editor and choose **Editor Settings** from the menu.

To return to your script, click this arrow.



You can change the **Theme** to alter the appearance of the editor. The example below uses the **Dark** theme.

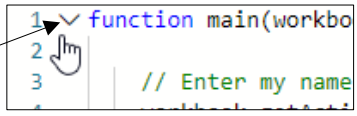
```
function main(workbook:
  ExcelScript.Workbook) {

  // Enter my name in
```



You can enable a **Minimap** of your script which appears at the top right of the code window.

The **Folding** option allows you to collapse and expand sections of your code contained within different types of brackets.

With the **Folding** option enabled you can click arrows like this one to collapse and expand sections of your script.

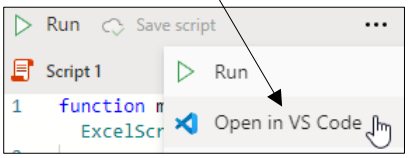


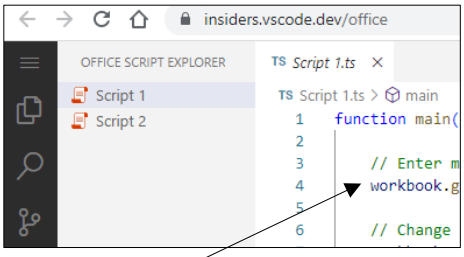
This option allows you to edit a script in the online version of Visual Studio Code.

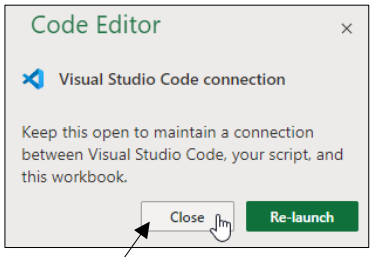
A Note on Visual Studio Code

If you're working in the online version of Excel, you can enable the **Visual Studio Code connection** option. You can then open a script in the Visual Studio Code web application, as shown below:

Use the menu at the top right of the code window and choose **Open in VS Code**.







You can edit the script in the new tab.

You can close the Visual Studio tab or click **Close** to return to the Code Editor in Excel.

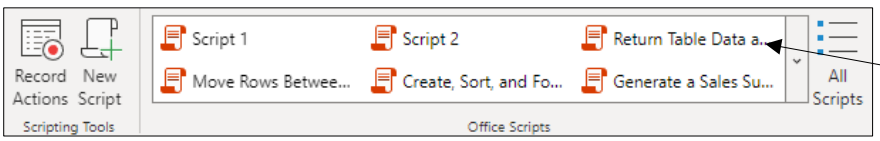
CHAPTER 2 - WORKING WITH SCRIPT FILES

2.1 Office Script Files

This section explains the basics of working with Office Script files.

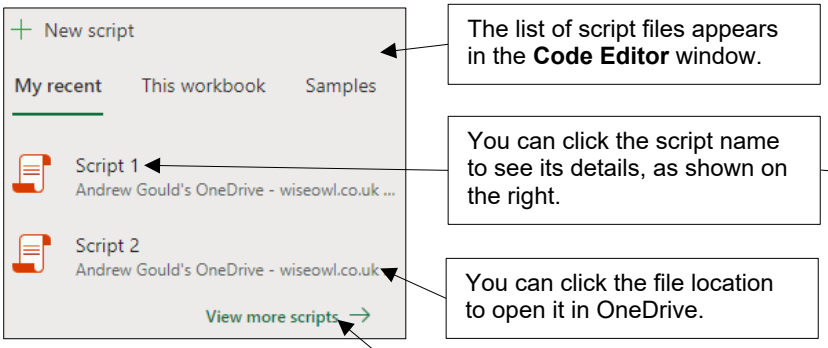
Viewing Existing Script Files

You can see your recently used and sample scripts in the **Automate** tab in the Excel ribbon.



Your Office Scripts appear in a gallery. You can click a script to see its details in the Code Editor window.

To see all your scripts, choose **Automate | All Scripts** from the Excel ribbon.

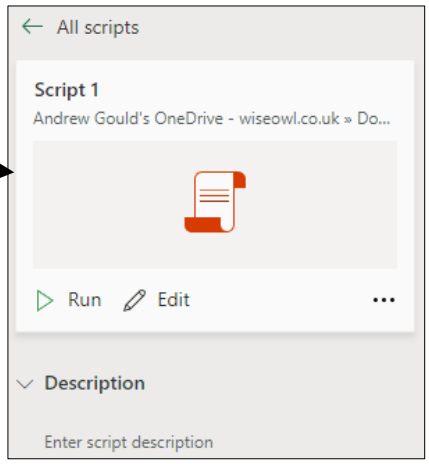


The list of script files appears in the **Code Editor** window.

You can click the script name to see its details, as shown on the right.

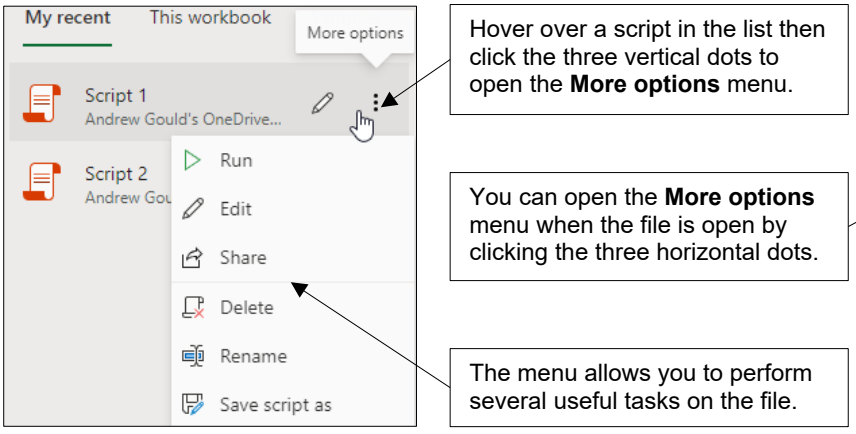
You can click the file location to open it in OneDrive.

Click **View more scripts** to open a dialog box which lets you browse to other folders in OneDrive and SharePoint.



The More Options Menu

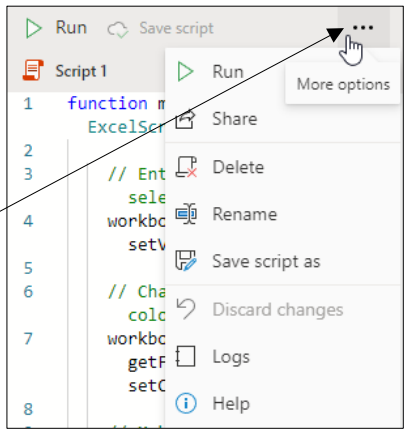
You can use a script's **More options** menu to perform several useful tasks on it.



Hover over a script in the list then click the three vertical dots to open the **More options** menu.

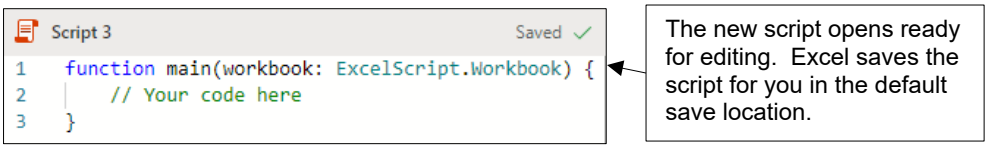
You can open the **More options** menu when the file is open by clicking the three horizontal dots.

The menu allows you to perform several useful tasks on the file.



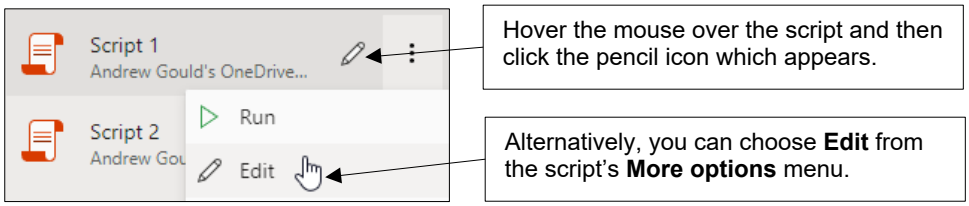
Creating a New Script File

The simplest way to create a new script is to choose **Automate | New Script** from the Excel ribbon.



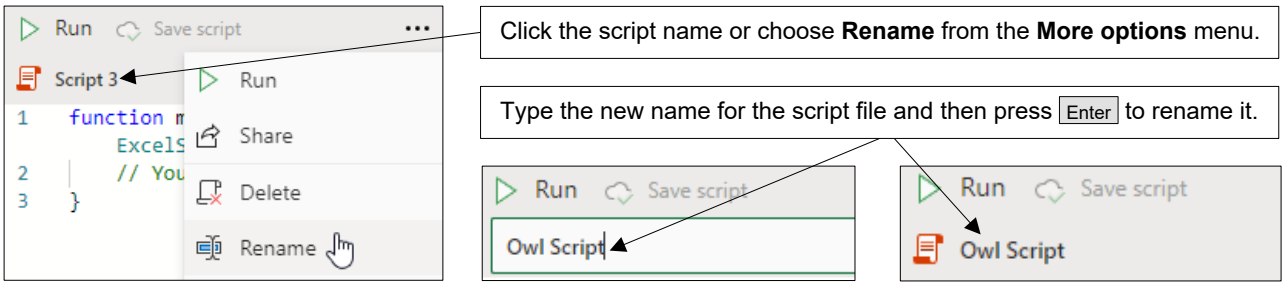
Opening an Existing Script

You can open an existing script to edit its code as shown in the diagram below:



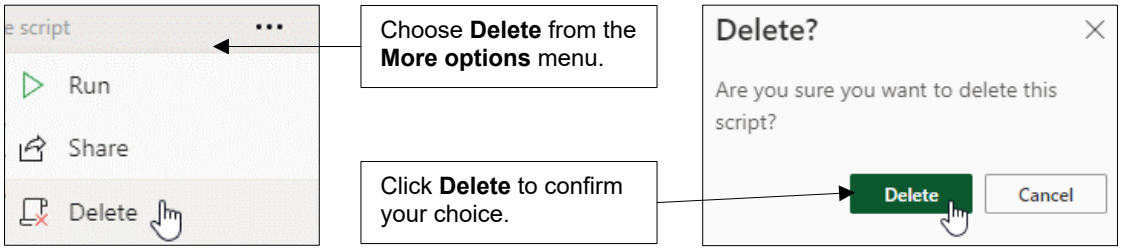
Renaming an Open Script

You can rename a script file, even if you're currently editing it.



Deleting a Script

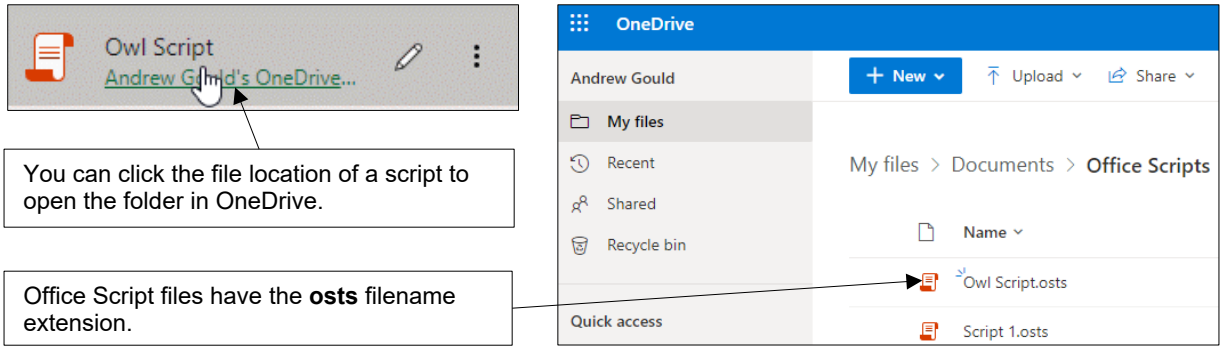
You can delete a script from the **More options** menu, even if you are currently editing it.



2.2 Saving Script Files

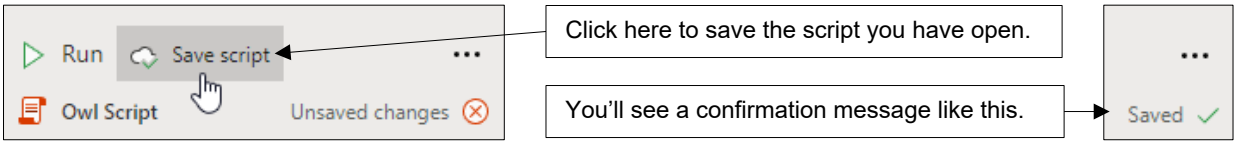
Where Office Scripts are Saved

By default, Office Script files are saved in the **Documents/Office Scripts** folder in OneDrive.



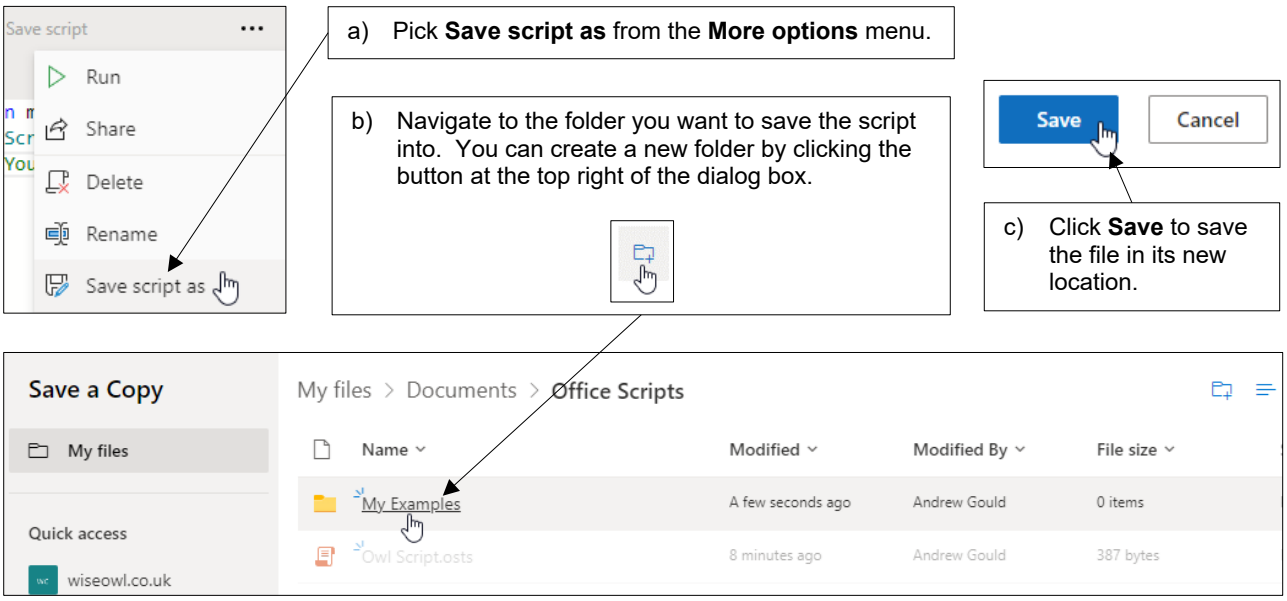
Saving an Open Script

You can save the script that you're currently working on in the default folder as shown below:



The Save As Option

You can choose to save your script in a different location as shown below:



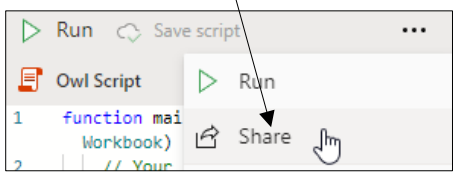
2.3 Sharing Scripts

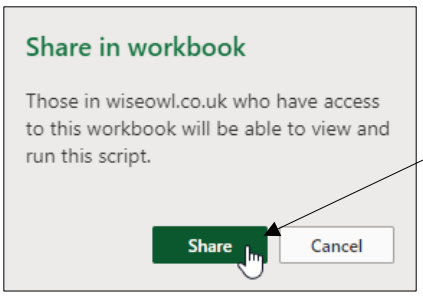
Scripts saved to your OneDrive are accessible only to you. If you'd like others to use your scripts, you'll need to share them in some way.

Sharing a Script

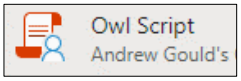
The first step in sharing a script is to share it in a workbook. To do this, first open the workbook you want to share the script in.

a) Choose **Share** from the script's **More options** menu.



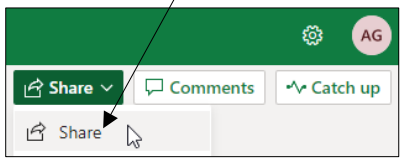


b) Click **Share** to confirm that you want to share the script. The script's icon changes to show that it is being shared:

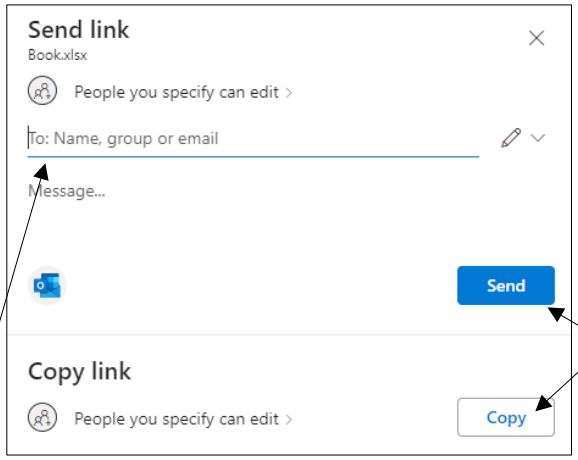


You'll also need to give other people access to the workbook, as shown in the diagram below:

a) Click the **Share** menu in the top right of the Excel window and choose **Share**.



b) In the dialog box, configure with whom the workbook is to be shared.

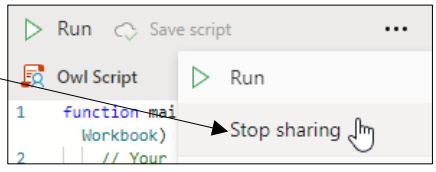


c) Share the file by sending a link in an email, or by copying the link and distributing it another way.

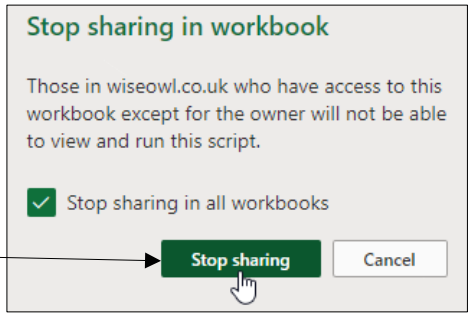
Stopping Sharing a Script

The diagram below shows how you can reverse the process of sharing an Office Script:

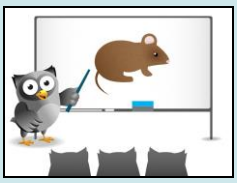
a) From the script's More options menu, choose **Stop sharing**.



b) On the dialog box which appears, choose whether to stop sharing in all workbooks or just the currently open one, and then click **Stop sharing.**



WHAT WE DO

		 ONLINE TRAINING	 MANCHESTER OR LONDON	 AT YOUR OFFICE	 BESPOKE CONSULTANCY
OFFICE 365	Microsoft Excel	✓	✓	✓	✓
	VBA macros	✓	✓	✓	✓
	Office Scripts	✓		✓	
	Microsoft Access				✓
POWER PLATFORM	Power BI and DAX	✓	✓	✓	✓
	Power Apps	✓		✓	
	Power Automate	✓	✓	✓	✓
SQL SERVER	Reporting Services	✓	✓	✓	✓
	Report Builder	✓		✓	✓
	Integration Services	✓	✓	✓	✓
	Analysis Services	✓		✓	
CODING LANGUAGES	SQL	✓	✓	✓	✓
	Visual C#	✓	✓	✓	✓
	Python	✓	✓	✓	✓

Wise Owl Training


Wise Owl
Training