

How to Display the Last Saved Date in Excel

Authored by
stats writer

February 2, 2026

RECOMMENDED CITATION

stats writer (2026). *How to Display the Last Saved Date in Excel*. PSYCHOLOGICAL SCALES. Retrieved from <https://scales.arabpsychology.com/?p=129121>

Tracking the precise moment an **Excel** file was last modified and saved is crucial for effective data management and version control. While simple formulas might seem like a quick fix, they often fail to capture the true "last saved" timestamp. For instance, attempting to use the `=TODAY()` formula only provides the current date, which updates every time the workbook is opened, rather than displaying the persistent, historical save time metadata.

If your goal is only to stamp a cell with the current date (which updates daily) and then manually change the cell format, you might follow a procedure similar to this. However, it is essential to understand that this method **does not** provide the actual date and time the file was last saved to disk. For achieving the reliable, permanent last saved date, a robust approach utilizing **Visual Basic for Applications (VBA)** is required, which we detail in the main section below.

Open the **Excel spreadsheet** where you wish to insert a date stamp.

Click on the cell where you want the current date to appear.

In the formula bar, type `=TODAY()`. This function automatically inserts the current system date into the selected cell.

To prevent this date from updating every time the sheet recalculates (effectively making it static after the initial entry), you would typically copy the cell and paste it as values. Alternatively, to format the display, right-click on the cell and select "Format Cells."

In the **Number** tab, select "Date" as the category and choose your preferred date format.

Click "OK" to apply the formatting changes.

Crucially, this formula only displays the current date. It is important to note that this technique is insufficient for tracking the actual moment the file was last saved, which requires a **macro**.

By understanding the limitations of simple formulas, we can proceed to the definitive solution involving custom VBA code to reliably extract the workbook's internal metadata.

A Comprehensive Guide: How to Insert the Last Saved Date in Excel Using VBA (With Detailed Example)

Often, professionals require a reliable method to display the exact date and time an **Excel file was last saved within the worksheet itself. This feature is invaluable for auditing, ensuring data freshness, and maintaining strict version control when collaborating on**

complex spreadsheets.

The most effective and accurate way to achieve this dynamic display is by writing a small piece of code in **VBA**. Although working with VBA might initially sound complex, even for users unfamiliar with coding, the required process is exceptionally straightforward. The following comprehensive, step-by-step example is designed to guide you precisely through the implementation, resulting in a persistent, accurate save time display.

Prerequisite: Understanding the Need for VBA

Unlike simple calculations, extracting the 'Last Save Time' requires accessing the workbook's fundamental metadata, which standard Excel formulas cannot retrieve directly. **VBA macros** provide the necessary functionality to query the `BuiltinDocumentProperties` object, ensuring that the date displayed is an accurate reflection of the file's last write time recorded by the operating system.

This implementation involves defining a custom function that acts as a bridge between the worksheet

and the workbook's internal properties. Once defined, this function behaves exactly like any native Excel function, allowing you to call it from any cell in your workbook.

Step 1: Enabling the Developer Tab in Excel

Before writing any VBA code, we must ensure that the **Developer** tab is visible on the main Excel ribbon. By default, this tab is often hidden, as it contains advanced tools for creating macros, user forms, and add-ins. Making it visible is a necessary preparatory step for accessing the Visual Basic Editor (VBE).

To enable the Developer tab, navigate through the following sequence:

Click the File tab located in the top-left corner of the Excel window.

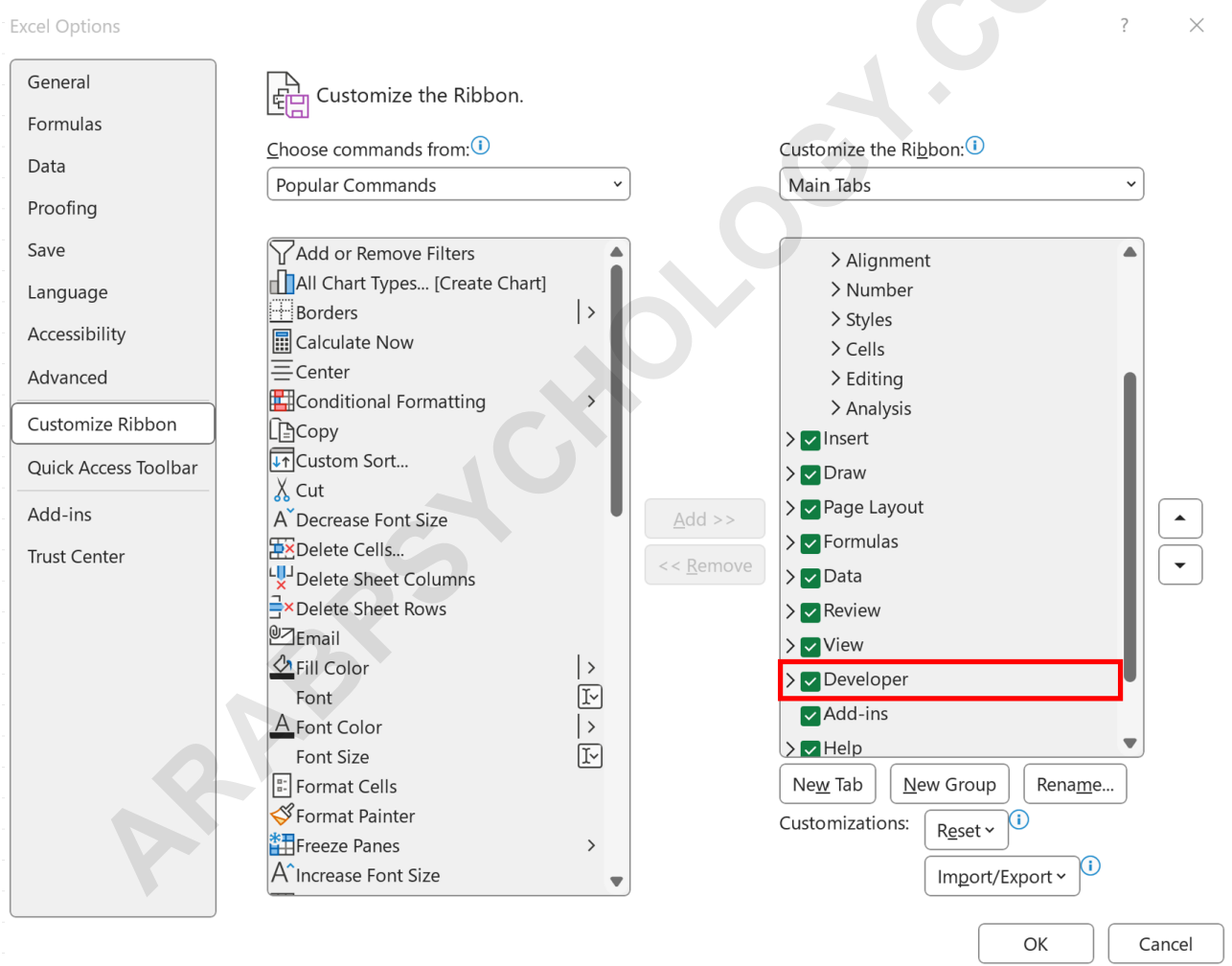
Select Options near the bottom of the File menu. This opens the Excel Options dialog box.

In the left pane of the dialog box, click Customize Ribbon.

Under the section titled Main Tabs on the right-hand side, look for and check the box next to Developer.

Click OK to save the changes and close the dialog box.

Upon completion of these steps, the Developer tab will now be prominently displayed on the top ribbon, granting you access to the crucial tools required for creating and managing VBA functions and macros.



Step 2: Accessing the Visual Basic Editor and Inserting a New Module

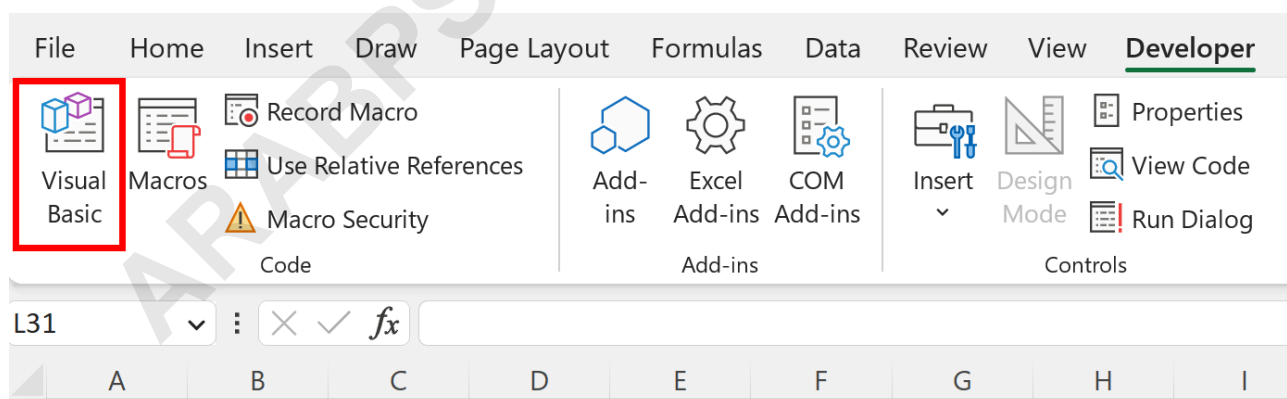
With the Developer tab enabled, the next phase is to

enter the Visual Basic Editor (VBE), which is the environment where we will write and store our custom date-tracking function. This code must reside within a standard module to be callable as a worksheet function.

Follow these precise instructions to open the VBE and prepare your coding environment:

Click the newly visible Developer tab on the Excel ribbon.

Within the Code group, click the Visual Basic icon. This action immediately opens the Visual Basic Editor in a new window. Alternatively, you can use the keyboard shortcut `Alt + F11` to launch the VBE.



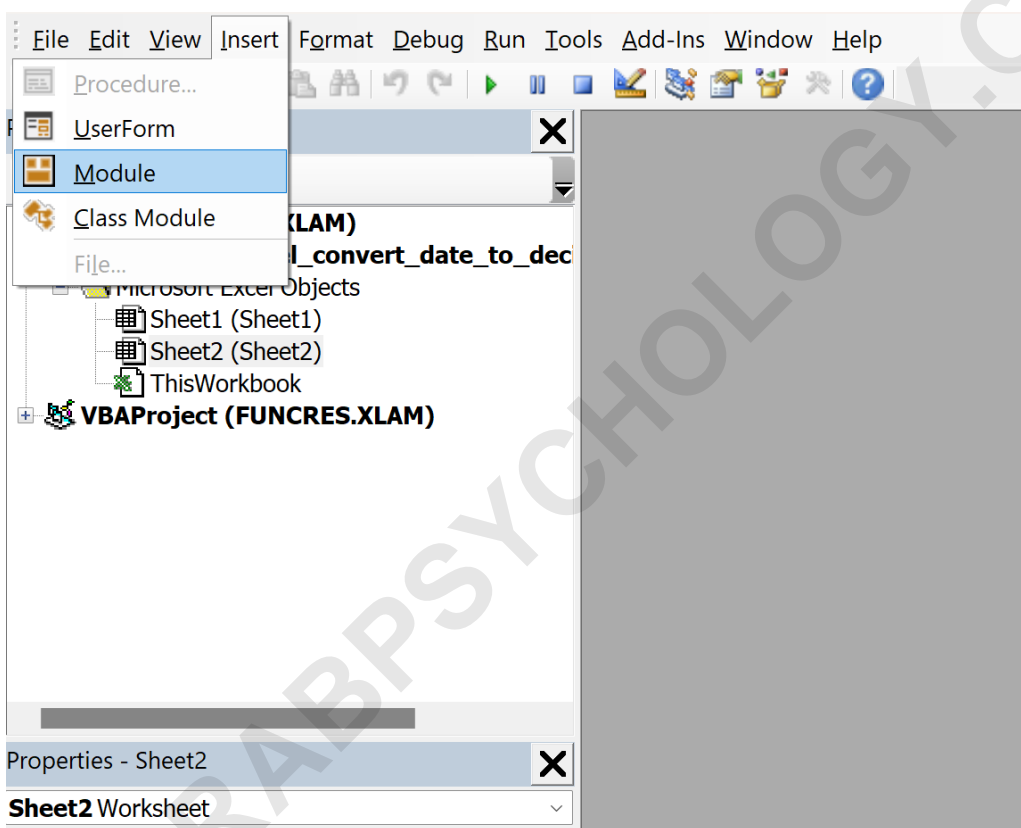
Once inside the VBE, you must insert a new module. Modules are containers for procedures (subroutines and functions) that can be executed anywhere in the

workbook.

In the VBE menu bar, click the Insert tab.

From the dropdown menu that appears, select Module.

A new, blank module window will open on the right side of the VBE, ready to accept your code.



Step 3: Creating the LastSavedDate Function using VBA

The core of this solution lies in a short but powerful **VBA** function that specifically targets the workbook property storing the last save time. This property is part of the `BuiltinDocumentProperties` collection, a predefined set of

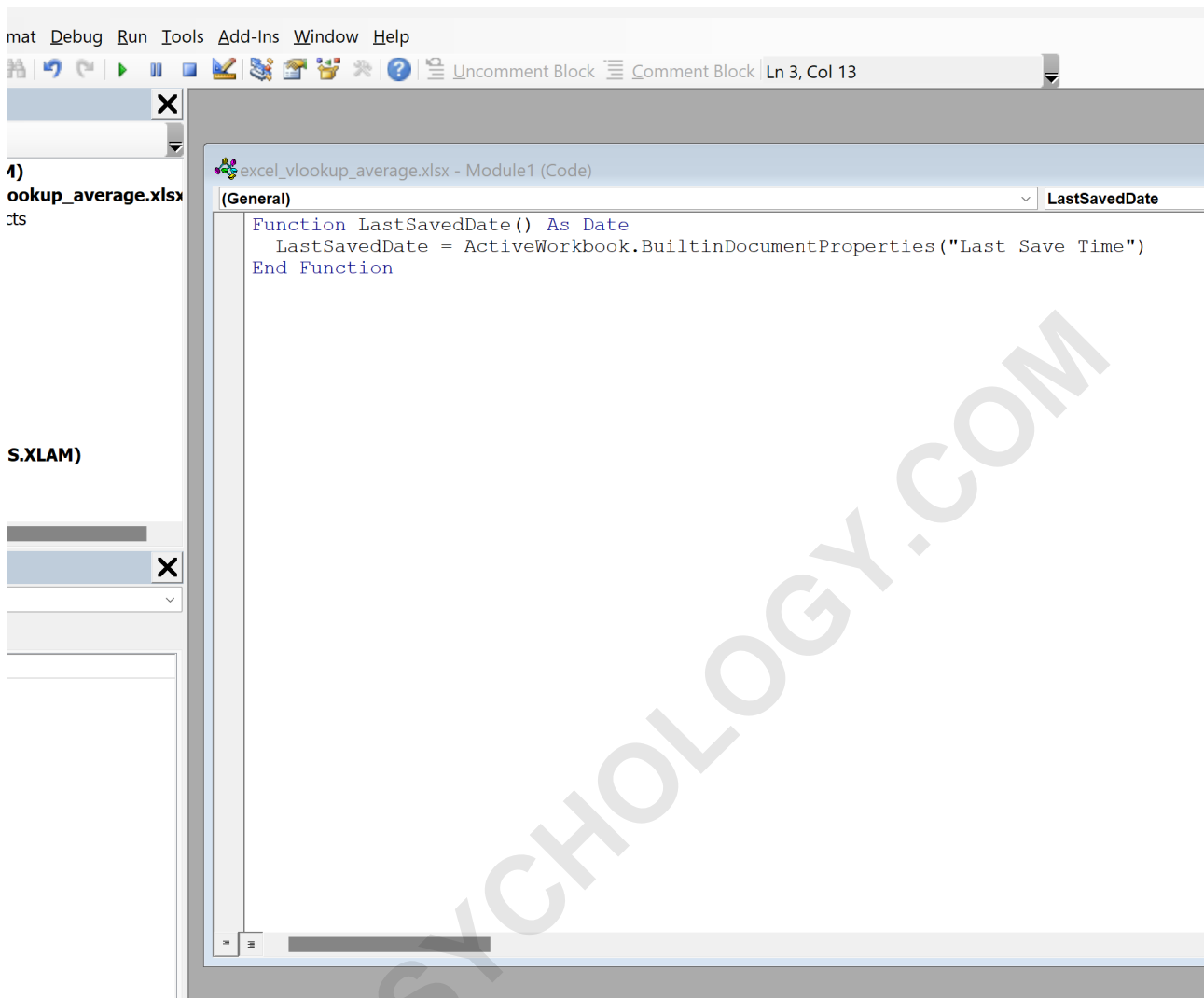
metadata tags associated with every Excel file.

Carefully paste the following code snippet directly into the module code editor window:

```
Function LastSavedDate() As Date
    LastSavedDate =
    ActiveWorkbook.BuiltinDocumentProperties("Last Save
    Time")
End Function
```

This custom function, named `LastSavedDate()`, is declared to return a value of the **Date** data type. It retrieves the required value by accessing the `ActiveWorkbook.BuiltinDocumentProperties` collection and specifying the exact property name: `"Last Save Time"`. This ensures that the function returns the authoritative timestamp recorded by Excel.

The following screenshot illustrates where this code should be placed within the VBE environment:



After pasting the code, ensure you save the workbook. Note that since this workbook now contains a macro, it must be saved as an Excel Macro-Enabled Workbook (.xlsm) file type to preserve the VBA module.

Step 4: Implementing the Custom Function in Your Worksheet

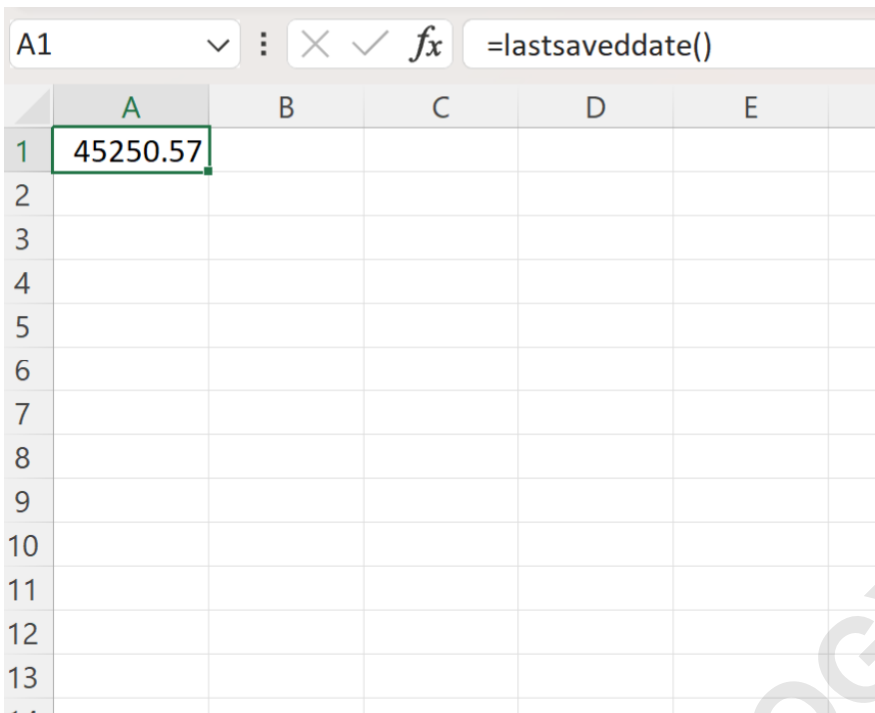
The final step in the process is utilizing the newly created custom function within the Excel worksheet.

Since we defined this procedure as a public function in a module, it is now available to be called just like standard functions such as `SUM` or `AVERAGE`.

To display the last saved date for the current workbook, simply type the function name followed by empty parentheses into any cell you desire:

`=LastSavedDate()`

For demonstration purposes, we will input this formula into cell A1 of our worksheet. As soon as you press Enter, the function will execute, querying the file metadata and returning the timestamp.



The screenshot shows an Excel spreadsheet with the following data:

	A	B	C	D	E
1	45250.57				
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					

Step 5: Formatting the Output as a Date

When the `LastSavedDate()` function initially returns its value, it often appears as a large numeric serial value (e.g., 45250.75), which is how Excel internally stores dates and times. To convert this serial number into a human-readable date format, we must apply appropriate cell formatting.

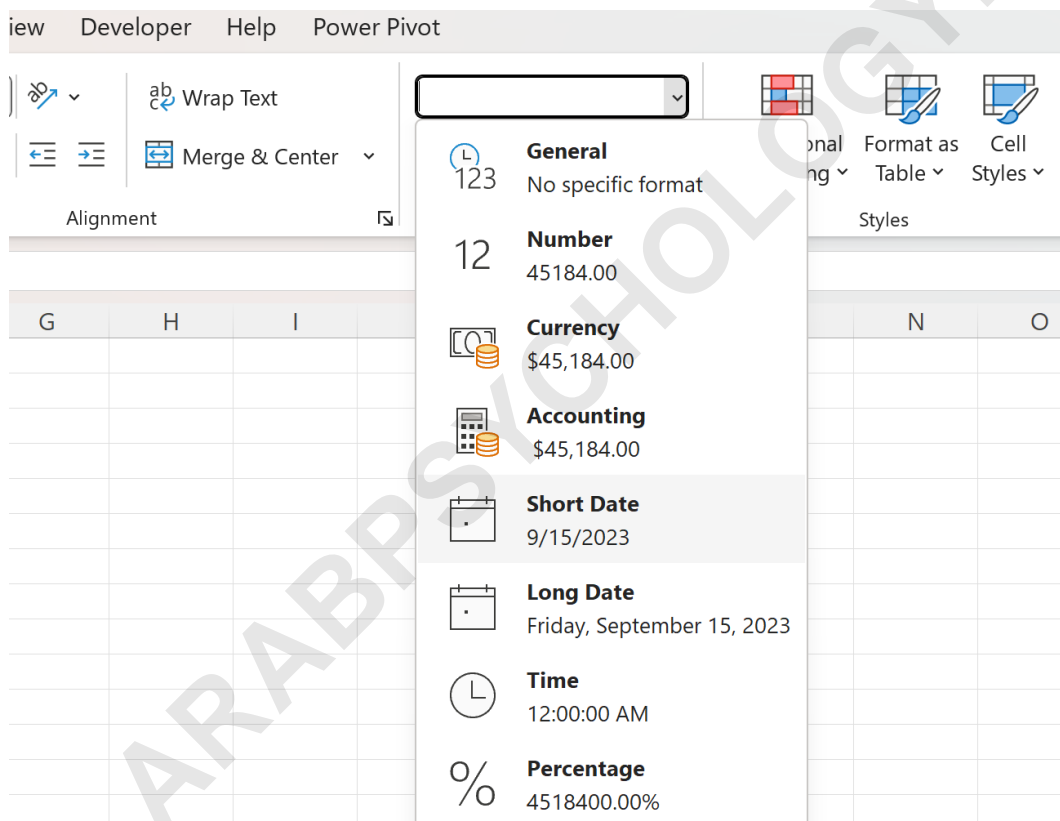
To correctly display the output as a date, perform the following formatting steps:

Click on the cell containing the `=LastSavedDate()` formula (in our example, cell A1).

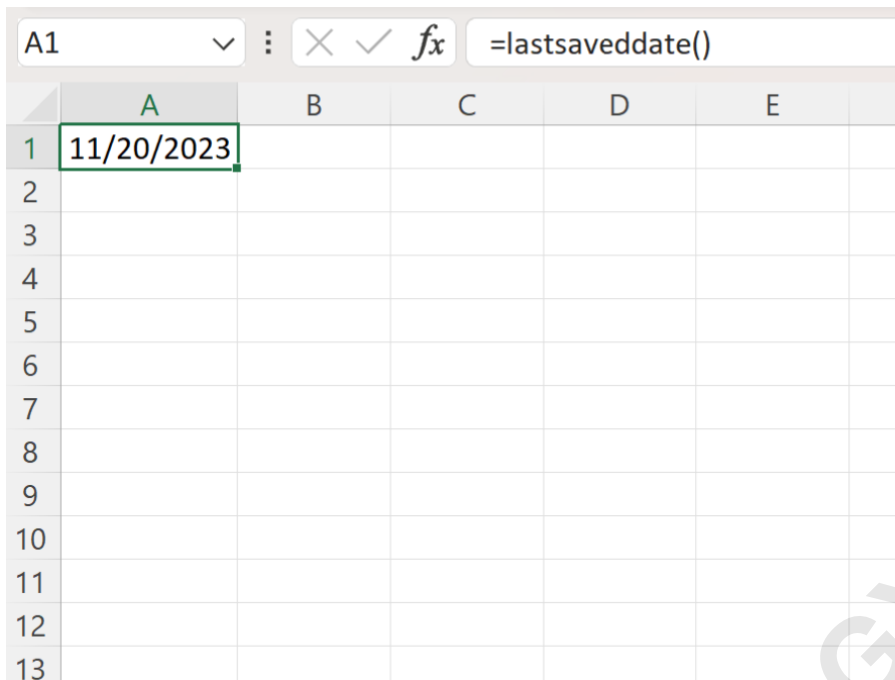
Navigate to the Home tab on the Excel ribbon.

Locate the Number Format dropdown menu (usually displaying "General").

Click the dropdown arrow and select Short Date (or Long Date, depending on your preference). This format applies the standard date representation based on your regional settings.



Once the formatting is applied, the numeric serial value is instantly translated into the expected date format:



The screenshot shows an Excel spreadsheet with the following data:

	A	B	C	D	E
1	11/20/2023				
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					

The formula bar at the top shows the formula `=lastsaveddate()` entered in cell A1. The spreadsheet grid shows columns A through E and rows 1 through 13. Cell A1 contains the date 11/20/2023.

In this particular example, the output clearly shows that this **Excel** workbook was last saved on 11/20/2023. Crucially, this value will automatically update every time you save the workbook, providing a perpetually accurate record of the last modification.

Conclusion: Ensuring Data Integrity with Automated Timestamps

Implementing a custom `LastSavedDate()` function through **VBA** offers a powerful and reliable solution for tracking file integrity. By leveraging `BuiltinDocumentProperties`, users can embed dynamic metadata directly into their worksheets, eliminating manual date entry errors and ensuring that collaborators always know the precise vintage of the

data they are viewing.

This technique is essential for regulatory compliance, audit trails, and efficient team collaboration on large, evolving datasets. Remember to save your file as an .xlsm to maintain the macro functionality.

Further Learning and Related Topics

If you are interested in extending your knowledge of Excel automation and metadata manipulation, the following tutorials explain how to perform other common operations: