How to Easily Preview Your Documents in VBA Before Printing

Authored by

stats writer

November 19, 2025

The functionality of a print preview is indispensable in applications driven by Visual Basic for Applications (VBA). Integrating print preview capabilities into your automated processes allows users to visualize the final output of a document, worksheet, or user form before committing resources to physical printing. This preliminary step is crucial not only for saving paper and time but also for rigorous quality control, ensuring that generated reports adhere precisely to established formatting standards.

When working with complex data models or dynamically generated reports in applications like Microsoft Excel or Access, relying solely on code execution without visual confirmation can lead to costly errors. Utilizing the Print Preview method within VBA empowers developers and end-users alike to debug layout issues, confirm pagination, and make final adjustments to elements such as margins, headers, and footers directly from the preview window. Understanding how to correctly call this method, whether for an entire document or a specific range, is foundational to creating robust and user-friendly automation scripts.

Introduction to Print Preview in VBA

The primary purpose of the Print Preview function in VBA is to provide a non-destructive simulation of the printing process. When a user runs a macro that includes this command, the application generates a virtual representation of the hard copy. This feature is far more efficient than relying on standard application menu navigation, especially when automating repetitive reporting tasks.

In the context of Excel, the print preview window dynamically renders how worksheet data, charts, and specified print areas will appear across pages. This allows the user to confirm vital formatting details, such as the orientation (portrait or landscape), the scaling percentage, and whether column and row headers are correctly set to repeat on subsequent pages. Without this programmatic checkpoint, developers would need to manually navigate the application interface, diminishing the efficiency gains provided by VBA.

Furthermore, the print preview interface is interactive. Even though the preview is initiated via code, the user retains control within the preview window. They can zoom in and out, navigate between pages, and often access the Print Setup dialog box to make minor, last-minute modifications to the document structure before finalizing the printing action. This blend of automation and user control ensures professional results every time a document is generated.

The Importance of Pre-Printing Validation

Implementing rigorous validation steps before printing is a hallmark of professional automation development. While VBA scripts handle the data processing and calculation perfectly, visual layout problems--such as columns truncating off the page edge or data spilling onto unwanted pages--are common. Using the print preview is the most reliable way to catch these aesthetic and structural

inconsistencies.

Consider the benefits that pre-printing validation offers in a corporate environment. Automated reports often deal with varying data volumes. A report that looks perfect with 50 rows of data might break when it expands to 500 rows. The print preview provides immediate feedback on how overflow data is handled by the current page setup. This ensures compliance with business requirements, where documents must often fit within specific page limits or formats.

Key areas where print preview enhances validation include:

Margin Compliance: Quickly verifying that standard corporate margins are maintained and that content does not interfere with potential binding areas.

Pagination Check: Ensuring that logical breaks occur where intended and that critical information is not split unnecessarily across two pages.

Header/Footer Integrity: Confirming that dynamic page numbering, dates, and confidential disclaimers appear correctly on every printed sheet.

Scaling Issues: Adjusting the worksheet to fit on a specific number of pages, a common requirement easily managed within the preview context.

Core Print Preview Methods in Excel VBA

In Excel VBA, the PrintPreview method is exceptionally versatile. It can be applied to various objects within the Excel object hierarchy, including `Workbook`, `Worksheet`, and specific `Range` objects. The simplest syntax involves calling the method directly on the object you intend to print. The primary choice you must make when scripting is determining the scope of the preview: whether it should encompass the entire active sheet or only a predefined subset of cells.

The following two methods represent the most frequently used approaches for displaying a print preview dynamically before executing the actual printing command. Both utilize the same underlying method but apply it to different objects to define the print scope. Mastering these two techniques is essential for any developer writing automated reporting scripts.

It is important to remember that using `.PrintPreview` does not automatically trigger printing; it merely displays the preview window. This ensures that the user has the final authority to proceed with the hard copy, or alternatively, to cancel the operation or adjust settings before printing.

You can use the following methods in VBA to display a print preview before actually printing a sheet:

Method 1: Displaying Print Preview for the Entire Sheet

Method 1: Print Preview for Entire Sheet

This method is utilized when the entire visible worksheet, as defined by its current print settings, needs to be reviewed. By applying the `PrintPreview` method directly to the `ActiveSheet` object, the VBA engine assumes the scope covers all data and elements currently present on that worksheet, respecting any defined print titles or repeated rows/columns.

The code required for this operation is concise and highly efficient, making it ideal for quick reviews of standard reports where the scope is always the full sheet data:

```
Sub UsePrintPreview()
ActiveSheet.PrintPreview
End Sub
```

This particular macro will provide a print preview for the entire currently active sheet. The resulting window will show exactly how the pages will be laid out based on the current page setup parameters.

Method 2: Displaying Print Preview for a Specific Selection

Method 2: Print Preview for Selected Area

In many scenarios, you may only want to print a subset of the data available on the worksheet. For instance, a user might select a specific table or a range of cells (e.g., A1:F50) and wish to preview only that selection. In this case, applying the `PrintPreview` method to the `Selection` object is the appropriate technique. This forces the print output to restrict itself only to the boundaries of the highlighted cells.

If no specific range is selected by the user when this code is executed, the `Selection` object often defaults to the entire used range of the sheet, though it is best practice to ensure a range is selected programmatically or manually before calling this function. The code structure remains minimal:

```
Sub UsePrintPreview()
Selection.PrintPreview
End Sub
```

This particular macro will provide a print preview for only the currently selected area of the sheet. This is incredibly useful for generating ad-hoc reports based on user interaction.

Setting up the Workbook for Examples

The following examples demonstrate how to utilize each method effectively, using a sample active sheet in Excel containing basic sales data. This setup provides a clear visual context for how the different VBA commands affect the scope and appearance of the final previewed output.

We will assume that we are working with a standard Excel worksheet populated with various columns and rows of information. This sheet serves as our environment for testing the `ActiveSheet.PrintPreview` versus the `Selection.PrintPreview` commands.

The illustrative data structure used for these examples is shown below. Notice the extent of the data and its layout, which will determine what is included in the full sheet preview versus the selected area preview:

	A	B	C	D	E	F
1	**Team**	**Points**	**Assists**			
2	Mavs	22	12			
3	Heat	19	9			
4	Nets	14	4			
5	Rockets	20	6			
6	Rockets	30	5			
7	Mavs	34	7			
8	Mavs	30	7			
9	Heat	29	8			
10	Nets	14	6			
11	Nets	29	3			
12						
13						
14						
15						
16						
17						
18						
19						

This visual reference is key to understanding the differences demonstrated in the subsequent walkthroughs. The goal is to see how the VBA code interacts with this source data to generate accurate previews.

Detailed Walkthrough: Printing the ActiveSheet

Suppose our requirement is to capture the entirety of the report on the active sheet. We need to ensure that all populated cells are included in the print job, regardless of whether a specific range is currently highlighted.

We can create the following macro to perform a print preview. This will allow us to see what the entire document will look like, including page breaks and margins, before we actually send the job to the printer:

```
Sub UsePrintPreview()
ActiveSheet.PrintPreview
End Sub
```

Upon execution of this simple procedure, the Excel application automatically invokes the print preview handler. The resulting window reflects the full scope of the ActiveSheet, typically fitting the content onto as many pages as necessary according to the existing page setup.

When we run this macro, the following print preview window appears, showing the first page of the fully structured report:

Team	Points	Assists
Mavs	22	12
Heat	19	9
Nets	14	4
Rockets	20	6
Rockets	30	5
Mavs	34	7
Mavs	30	7
Heat	29	8
Nets	14	6
Nets	29	3

This visual confirmation shows us exactly what the page will look like if we print the entire currently active sheet. We can verify that the margins are correct and that the necessary data columns are not clipped.

Detailed Walkthrough: Previewing a Defined Range

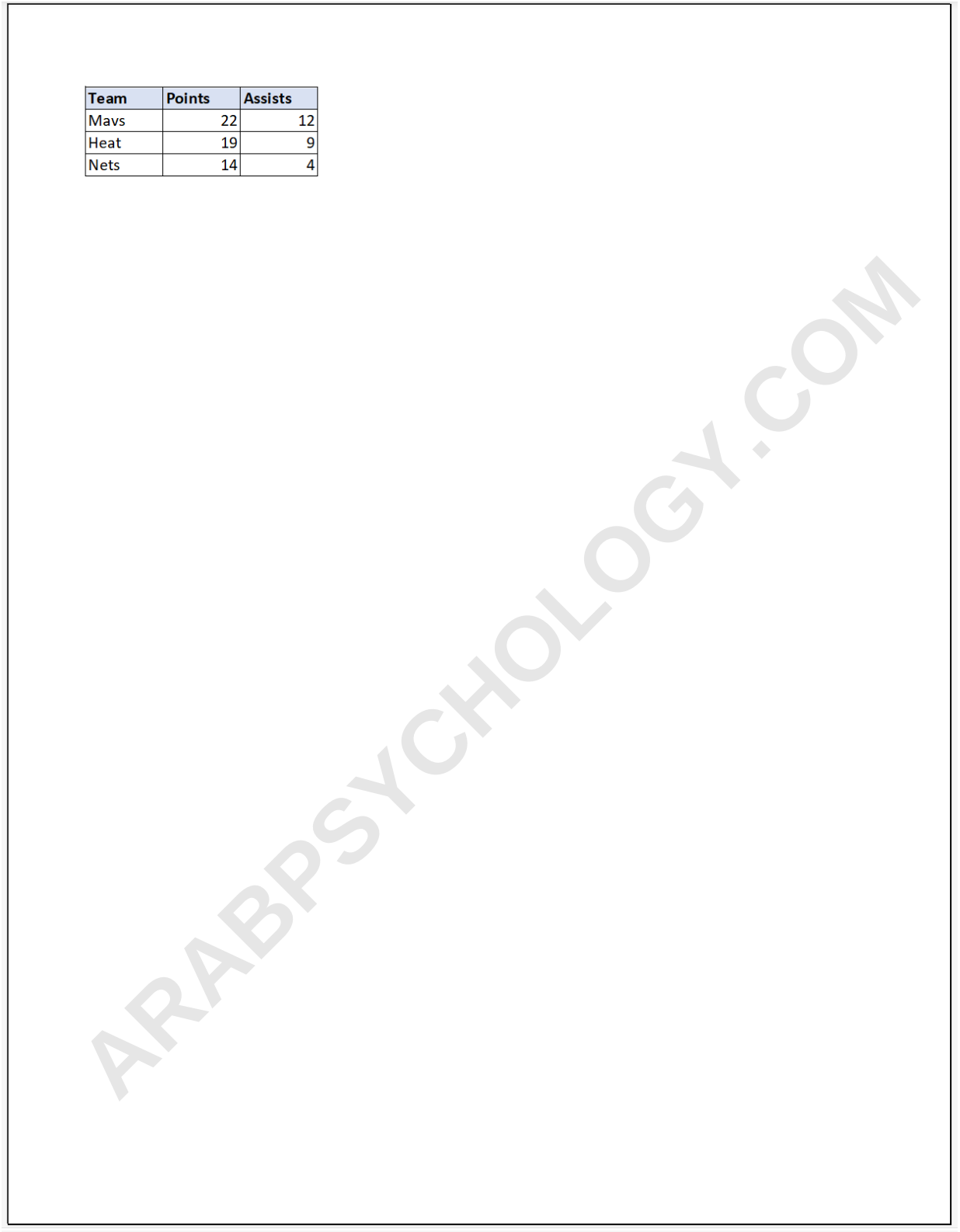
Now, let us consider a scenario where the user has highlighted a specific area of the data--perhaps only the first three columns, or a subset of rows--and needs to print only that segment. To achieve this, we must ensure the Selection object is targeted by the `PrintPreview` method.

Before running the code, the user would manually select the desired range (e.g., A1:C20). The VBA code then leverages this pre-existing selection to define the print boundaries. This is highly effective when dealing with large datasets where only specific tables or figures need to be isolated for printing.

We can create the following macro to perform a print preview. This procedure ensures that only the highlighted area is considered for printing, minimizing wasted paper and ensuring only relevant data is output:

```
Sub UsePrintPreview()
Selection.PrintPreview
End Sub
```

When we run this macro after selecting a range, the following print preview window appears, demonstrating the confined scope of the output:



Team	Points	Assists
Mavs	22	12
Heat	19	9
Nets	14	4

This output confirms exactly what the page will look like if we print only the selected area. Notice how the content is scaled and positioned relative to the single page, entirely ignoring the data outside the selected range boundaries.

Advanced Considerations and Best Practices

While the basic usage of `ActiveSheet.PrintPreview` and `Selection.PrintPreview` covers most requirements, expert VBA developers often need to implement more nuanced control over the printing process. For instance, you might want to adjust the page orientation or set a specific print quality before displaying the preview.

The `PrintPreview` method itself can take an optional argument, `EnableChanges`, which is a Boolean value. If set to `True` (the default), the user can make changes to the page setup within the preview window. If set to `False`, the preview is read-only, preventing users from altering the established formatting, which is critical in controlled reporting environments.

Furthermore, for highly customized reports, instead of relying on the user's selection, it is often better to define the range explicitly within the code. For example, using `Worksheets("Report").Range("A1:G100").PrintPreview` ensures that the exact required area is previewed every time, eliminating user error related to incorrect selection. This programmatic approach ensures consistency and reliability across all automated report generations.

It is highly recommended that developers consult the official documentation for the `PrintPreview` method to understand all available parameters and associated printing properties within VBA.

Note: You can find the complete documentation for the **PrintPreview** method in VBA online, which offers further details on its optional parameters and integration with other print-related objects.

Related Resources

VBA: How to Print to PDF