How do I Set Print Area Using VBA (With Examples)

Authored by

stats writer

November 18, 2025

Introduction to Controlling Print Settings via VBA

Managing print settings within large or complex Excel workbooks often requires automation to ensure consistency and efficiency. Manually adjusting the print range for dozens of reports is time-consuming and prone to error. Fortunately, VBA (Visual Basic for Applications) provides powerful tools to programmatically define exactly which data range should be printed, bypassing the standard manual configurations. This capability is essential for generating standardized reports or for scripts that handle data extraction and presentation in a seamless workflow.

The primary mechanism for controlling these settings is through the **PageSetup object**, which is available for every worksheet. By manipulating properties within this object, we can dictate margins, headers, footers, page orientation, and--most importantly for this discussion--the specific range of cells designated as the **Print Area**. Utilizing VBA allows developers to tie the definition of the print area directly to user actions, such as selecting a range, or to conditional logic based on the data present in the spreadsheet.

Before committing any report to a physical printer, it is generally recommended practice to display a preview to confirm the layout and content are correct. Our goal here is to demonstrate the exact syntax required in VBA not only to set the desired **Print Area** dynamically but also to integrate the crucial step of showing a **Print Preview**, thereby ensuring accuracy before final output. The following syntax provides the fundamental structure for this automation task.

Sub SetPrintArea()

With Sheets("Sheet1")
.PageSetup.PrintArea = Selection.Address
.PrintPreview
End With

End Sub

The Core VBA Syntax for Defining Print Area

The macro structure shown above is concise yet highly effective. It begins with the standard definition of a subroutine, Sub SetPrintArea(). The core logic is encapsulated within a With block directed at a specific worksheet, in this case, **Sheets("Sheet1")**. Using the With structure is a best practice in VBA programming, as it minimizes redundant object references and improves code readability and execution speed. All subsequent commands within this block pertain directly to "Sheet1."

The critical line that establishes the printing boundaries is: .PageSetup.PrintArea =

Selection.Address. This instruction accesses the PageSetup object of the specified sheet and assigns a new value to the **PrintArea** property. The value assigned is derived from **Selection.Address**. The **Selection.Address** property dynamically returns the absolute range reference (e.g., "$A$1:$C$10") of the cells currently selected by the user on the active sheet. This ensures the **Print Area** is always synchronized with the user's focus.

This specific macro is designed to be user-driven: whatever range the user has highlighted immediately before running the routine will be designated as the sole printable content for **Sheet1**. The subsequent line, .PrintPreview, executes the command to open the print preview interface, allowing the user to review the configured print range visually. This ensures the user verifies the correct data is included before proceeding to physical printing or exporting.

Understanding the PageSetup.PrintArea Property

The **PrintArea** property is a fundamental component of the PageSetup object in Excel VBA. This property accepts a string argument that must be a valid A1-style cell reference defining the range to be printed. For instance, setting .PageSetup.PrintArea = "$A$1:$F$50" would fix the print range to those specific cells, regardless of the user's current selection. The power of the dynamic solution presented here lies in using **Selection.Address**, which constantly updates the string value of the **PrintArea** property based on the user's interactive input.

It is important to note that the **PrintArea** property expects the range to be specified using absolute references, hence why **Selection.Address** is perfectly suited, as it returns the address in this format. If you need to clear the print area entirely (to revert to printing the entire sheet content with data), you would set the property to an empty string: .PageSetup.PrintArea = "". Understanding how to dynamically set and reset this property is key to creating robust reporting tools within Excel.

When defining the **Print Area**, the system overrides any default printing settings that might otherwise attempt to print the full extent of the worksheet data. By carefully controlling this property via a macro, users gain granular control over output, preventing accidental printing of hidden data, intermediate calculations, or extraneous formatting elements located outside the designated report bounds. This level of precise control dramatically improves the quality and professionalism of generated documents.

Previewing vs. Printing: PrintPreview vs. PrintOut

The inclusion of the .PrintPreview method in the original macro is a safety feature that provides immediate feedback to the user regarding the outcome of the print area configuration. This method opens the standard **Print Preview** interface, showing exactly how the sheet, constrained by the newly set **Print Area**, will appear on the printed page. This step is invaluable for troubleshooting layout issues, checking page breaks, and confirming that the correct data range was selected prior

to consuming paper and toner.

However, there are scenarios, particularly in automated batch processing or when generating reports that must run silently, where displaying the preview is unnecessary or even disruptive. If the objective is to execute the printing command immediately after setting the print area, the developer must replace .PrintPreview with the .PrintOut method. The .PrintOut method sends the current sheet (constrained by the **Print Area**) directly to the default printer or to a specified printer if arguments are provided.

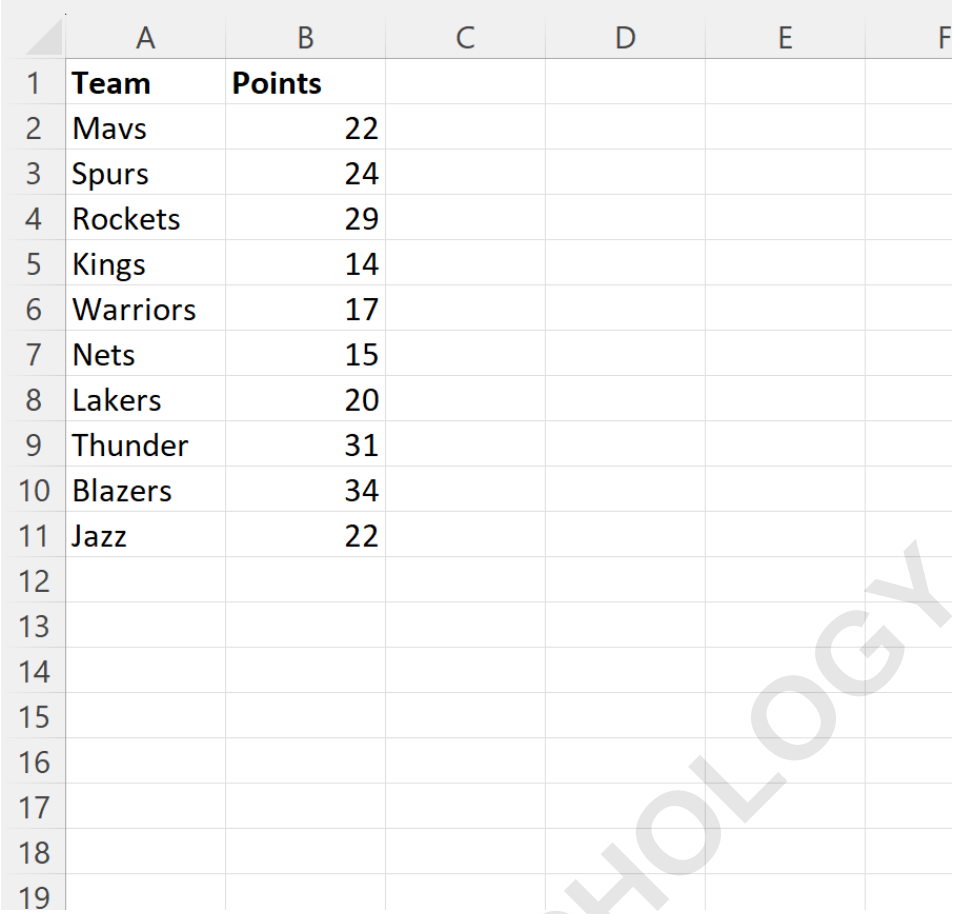
Note: If you intend to automatically print the selected range without user confirmation or previewing, replace .PrintPreview with .PrintOut in the VBA code block. The use of PrintPreview is highly recommended during development and testing phases, but .PrintOut is the necessary command for final, unattended execution. Always ensure your error handling is robust when using .PrintOut, especially in shared environments.

Practical Application: Setting the Print Area Dynamically

To illustrate the practical application of this macro, we will utilize a sample dataset containing information about basketball players. This example highlights how the **Print Area** can be instantly adapted to different reporting needs simply by changing the selection prior to execution. The goal is to isolate only the necessary data subset for printing, leaving ancillary information (like calculation columns or temporary notes) excluded from the final output.

Imagine the following scenario where our data is structured on **Sheet1**. The sheet contains player names, positions, and statistical data. We want the flexibility to print either only the header rows and a few specific players, or perhaps the entire dataset including column headers, depending on the requirement of the report being generated.

The following image displays the initial state of our sample data within the Excel worksheet. This data will serve as the basis for our dynamic printing demonstrations, confirming the importance of accurately defining the range before activating the print command.



	A	B	C	D	E	F
1	**Team**	**Points**				
2	Mavs	22				
3	Spurs	24				
4	Rockets	29				
5	Kings	14				
6	Warriors	17				
7	Nets	15				
8	Lakers	20				
9	Thunder	31				
10	Blazers	34				
11	Jazz	22				
12						
13						
14						
15						
16						
17						
18						
19						

We will use the foundational VBA code structure introduced earlier, which ties the PrintArea property directly to the active selection:

Sub SetPrintArea()

With Sheets("Sheet1")
.PageSetup.PrintArea = Selection.Address
.PrintPreview
End With

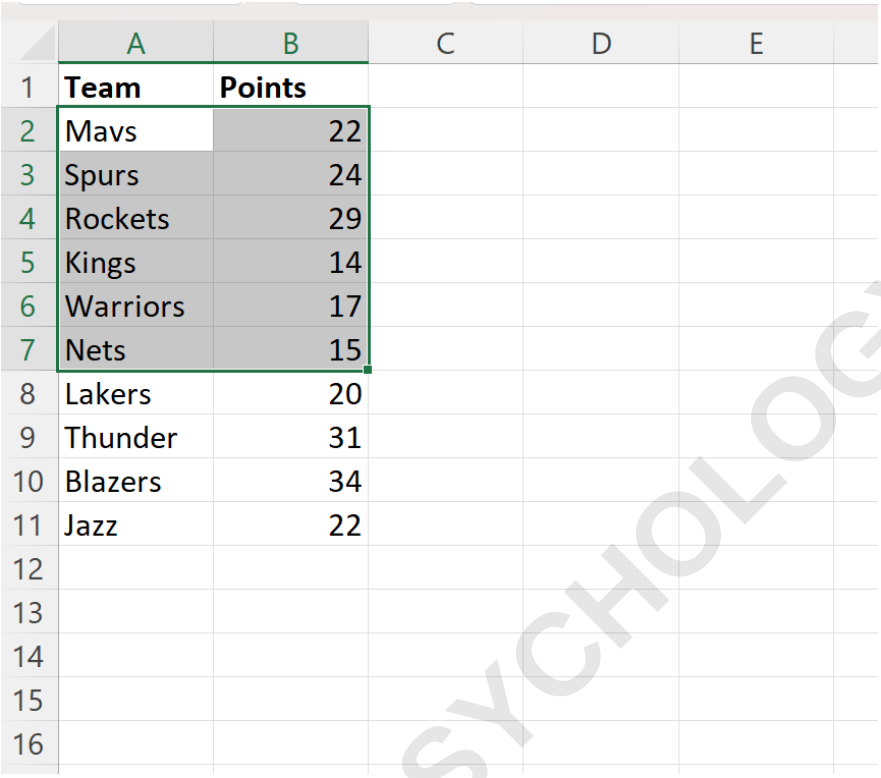
End Sub

Step-by-Step Demonstration: Selecting a Partial Range (A2:B7 Example)

In our first demonstration, we are interested only in printing the names and positions of the first few players, excluding the column headers located in row 1 and any subsequent data. To achieve this selective output, the user first navigates to **Sheet1** and precisely selects the cell range **A2:B7**. This action defines the scope of content we wish to include in our printed document, ignoring all other

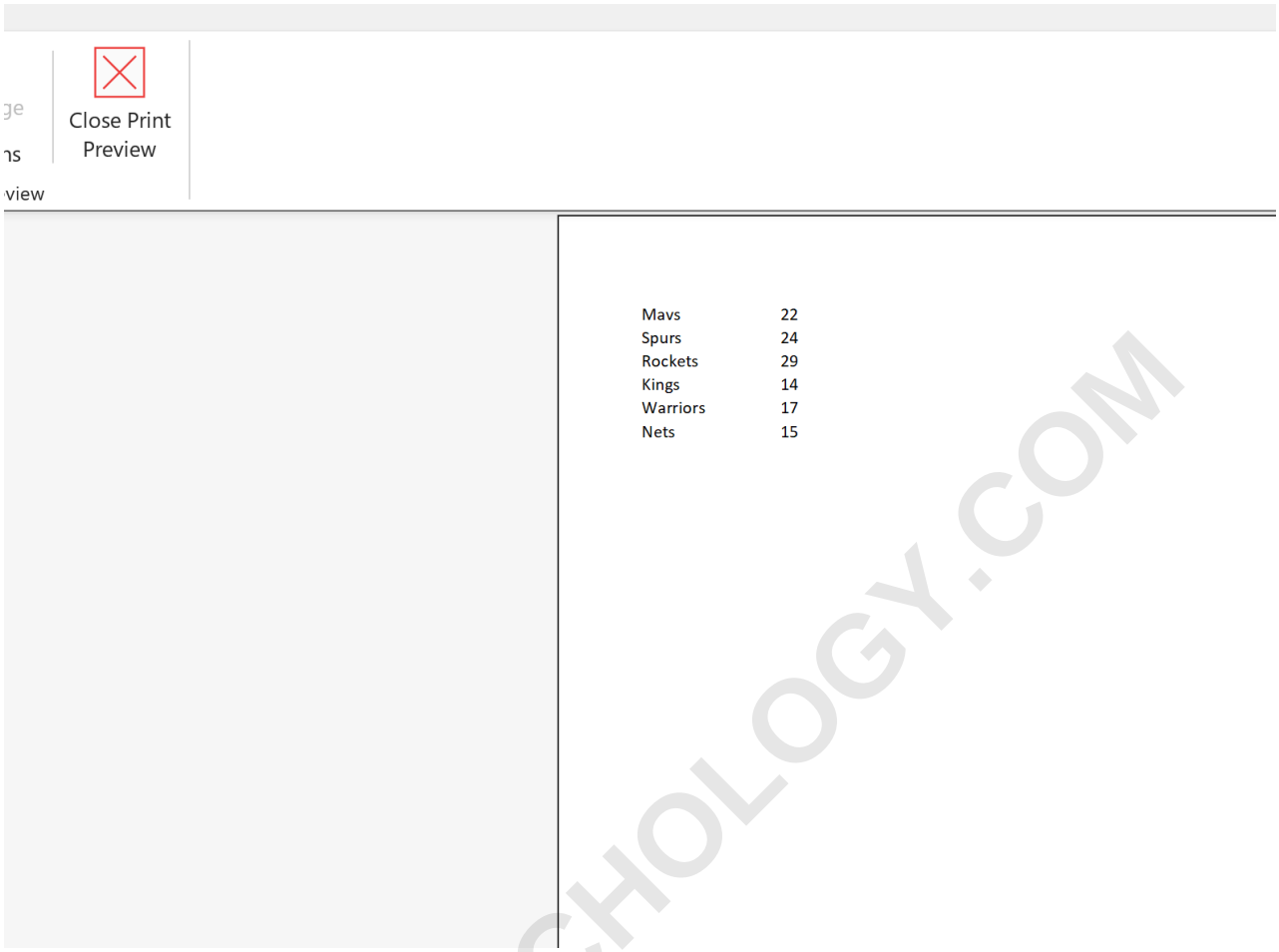
data on the sheet.

The following illustration shows the worksheet after this specific range has been highlighted by the user. This selection is critical because the Selection.Address method in the macro will now evaluate to the string "$A$2:$B$7" when the routine is executed, effectively setting the PrintArea to exactly these boundaries.



	A	B	C	D	E
1	**Team**	**Points**			
2	Mavs	22			
3	Spurs	24			
4	Rockets	29			
5	Kings	14			
6	Warriors	17			
7	Nets	15			
8	Lakers	20			
9	Thunder	31			
10	Blazers	34			
11	Jazz	22			
12					
13					
14					
15					
16					

Upon running the SetPrintArea macro, two key events occur simultaneously: first, the **Print Area** property for **Sheet1** is updated to **A2:B7**; second, the **Print Preview** window immediately appears. This preview confirms that only the selected names and positions are included, demonstrating that the code successfully localized the print output. The user is presented with the exact visual output shown in the subsequent image, confirming the successful definition of the partial range.

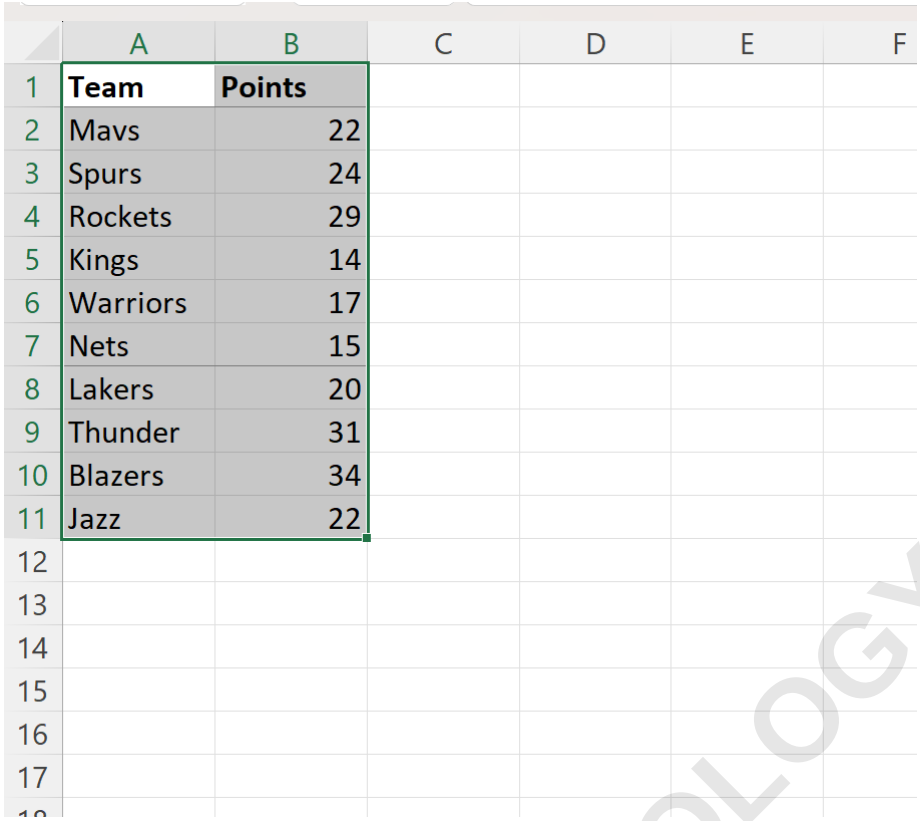


Flexibility and Dynamic Range Selection (A1:B11 Example)

One of the greatest advantages of using **Selection.Address** to set the PrintArea is the inherent flexibility it offers. If the reporting requirements change, or if a different segment of the data needs to be printed, the user simply needs to redefine their selection before executing the same macro. This eliminates the need for modifying the VBA code itself, making the solution highly scalable and user-friendly.

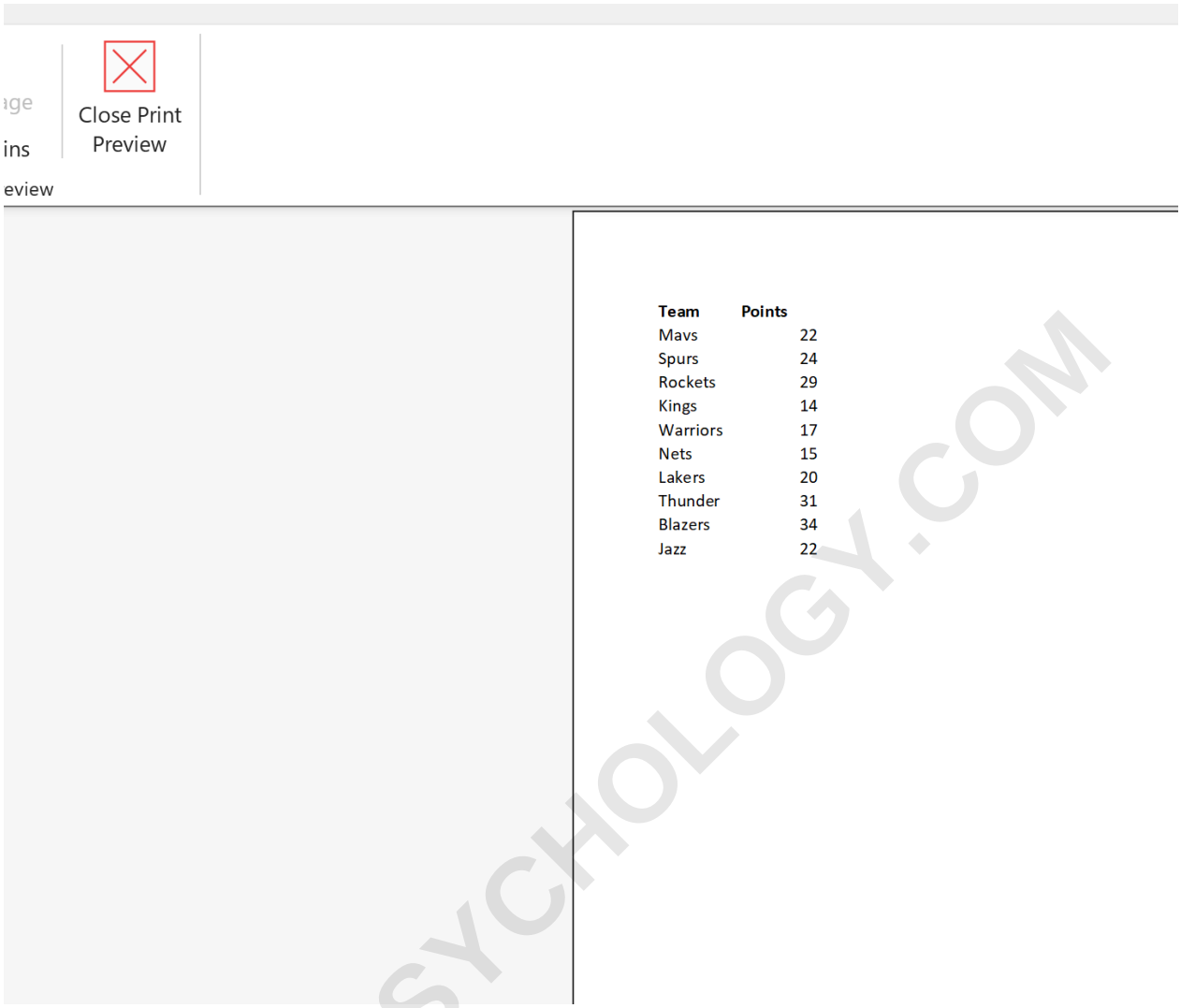
For instance, suppose we now need a report that includes the column headers (row 1) and all listed players, extending down to row 11. The user will select the range **A1:B11**, which encompasses the complete list of data we wish to output. This new selection overrides the previous boundaries established by the macro.

The visual representation below confirms the new, expanded selection, which now starts at the headers in row 1. This change instantly updates the necessary address property that the macro relies upon.



	A	B	C	D	E	F
1	**Team**	**Points**				
2	Mavs	22				
3	Spurs	24				
4	Rockets	29				
5	Kings	14				
6	Warriors	17				
7	Nets	15				
8	Lakers	20				
9	Thunder	31				
10	Blazers	34				
11	Jazz	22				
12						
13						
14						
15						
16						
17						

When the SetPrintArea macro is executed again, the **Print Area** is reset to **A1:B11**. The resulting **Print Preview** window now reflects this larger, more inclusive range. This successful demonstration confirms the macro's ability to instantly adapt to any contiguous range selected by the user, providing dynamic control over the printing process without any manual changes to the page setup menu.



Best Practices and Further Resources

When implementing print automation in VBA, always consider the user experience. While setting the **Print Area** based on **Selection.Address** is excellent for interactive use, for corporate reports or unattended processes, it is often safer to hardcode the range or calculate it based on data boundaries (e.g., finding the last populated row and column) to prevent unintended outputs if the user accidentally selects the wrong area.

Furthermore, always handle potential errors, such as when the user attempts to run the macro without any range selected, which can cause the code to halt. Implementing basic error handling using On Error Resume Next or checking if the selection is a valid range object can dramatically improve the stability of your code.

For developers looking for comprehensive details on all available printing parameters, the official documentation provides exhaustive information on the PrintArea property, covering advanced

scenarios such as non-contiguous range definitions and dealing with multiple pages. Understanding these properties ensures you can tailor your printing solutions to meet highly specific requirements.

Note: You can find the complete documentation for the **PrintArea** property in VBA via the Microsoft official website.

Related topics you might find helpful include:

VBA: How to Print to PDF
How to dynamically find the last row and column in an Excel sheet using VBA.
Using the PageSetup object to configure headers and footers programmatically.