Excel Formula: If Cell Color is Green Then Do Something

Authored by

stats writer

November 17, 2025

The Challenge of Referencing Cell Color in Standard Excel

A common hurdle faced by users of Microsoft Excel is the inability of standard formulas to directly interact with cell formatting attributes, such as background color. While we often rely on the powerful IF statement to perform logical checks on cell values, attempting to check if a cell color is green or red using functions like **=IF(A1="green", ...)** simply won't work. Excel formulas are designed to process data inputs, not display metadata.

To overcome this significant limitation and implement a condition that triggers an action based on **cell color**--for instance, returning "All-Star" if a cell is green--we must employ a clever workaround. This technique involves utilizing a legacy macro function combined with Excel's **Defined Name** feature, allowing us to extract the numerical color index associated with a cell's background.

This approach transforms a visual attribute (color) into quantifiable data (a numerical code), making it accessible to standard logical functions like the **IF statement**. The following comprehensive example illustrates the step-by-step process required to successfully implement this functionality.

Introducing the Solution: Leveraging the Legacy GET.CELL Macro Function

Since modern Excel functions cannot query formatting properties, we must turn to the older, hidden GET.CELL function. This function originated in the XLM (Excel Macro Language) era and is highly effective for retrieving specific information about a cell's formatting, location, or content.

Crucially, GET.CELL cannot be used directly within a spreadsheet cell in modern Excel versions; it must be encapsulated within a Defined Name. The function accepts two primary arguments: the type of information requested (the index number) and the cell reference to analyze. To retrieve the background color index of a cell, we use the argument number **38**.

By defining a custom name that executes **=GET.CELL(38,)**, we create a pseudo-function capable of returning a numerical index corresponding to the cell's fill color, which we can then use as the logical test within our subsequent **IF statement**.

Step 1: Setting Up Your Data and Defining the Goal

To demonstrate this technique, imagine a dataset detailing basketball players where the background color signifies a specific status. In this scenario, a green cell indicates that the player has been selected as an **All-Star**. Our objective is to create a dynamic formula that automatically labels the players based on this visual cue.

We begin with the following data structure. Column A contains the player names, and we need to

populate Column B with the conditional status based on the background color of the corresponding cell in Column A:

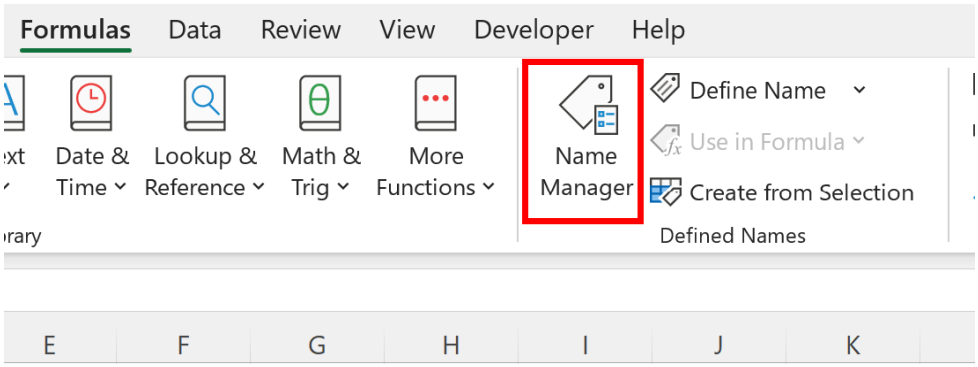
	A	B	C	D	E
1	**Athlete**				
2	Andy				
3	Bob				
4	Chad				
5	Doug				
6	Eric				
7	Frank				
8	Greg				
9	Henry				
10	Isaac				
11	John				
12	Kendall				
13	Luke				
14					
15					
16					

Our goal is straightforward: if the cell color in column A is green, column B should return "All-Star"; otherwise, it should return "Not All-Star." This simple conditional setup highlights the power of transforming formatting into usable data points for logical processing.

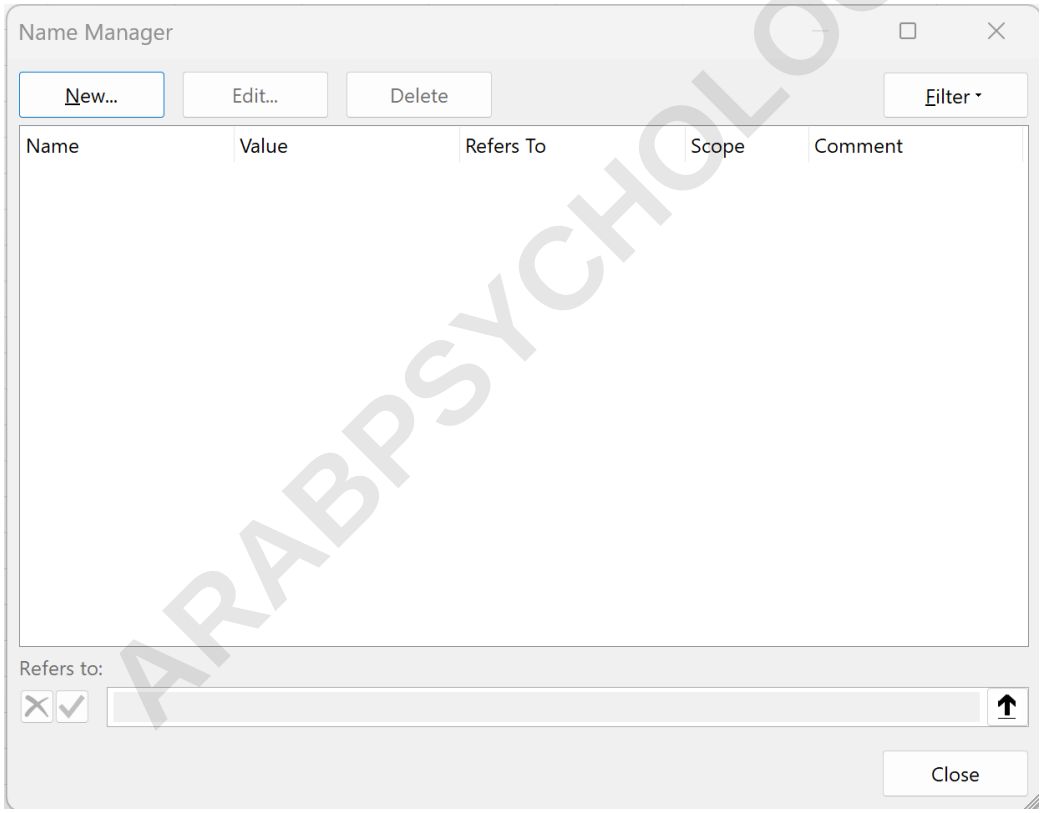
Step 2: Accessing the Name Manager Interface

The first procedural step requires us to access the **Name Manager**, which is the gateway to creating custom Defined Names necessary for implementing the GET.CELL functionality. The Name Manager allows users to define constants, formulas, or cell references that can be used throughout the workbook, effectively streamlining complex calculations and references.

To begin, navigate to the **Formulas** tab located along the top ribbon interface of Excel. Within the Defined Names group, locate and click the **Name Manager** icon. This action will launch the dialog box where we will create our custom function definition.



Once the **Name Manager** window is open, you will see a list of any existing defined names in the current workbook. Since we are creating a new custom function, click the **New** button situated in the top-left corner of the window. This action will open the "New Name" dialog box, prompting us to define the details of our custom color-checking function.



Step 3: Defining the Custom Function GreenCell

In the "New Name" dialog box, we must carefully define three key elements: the name of the function, the scope, and the formula it refers to. For clarity and ease of use, we will name our function **GreenCell**.

In the **Name** box, type **GreenCell**. Ensure the scope is set correctly (typically to the current Workbook or Sheet1, depending on your needs, though Workbook scope is generally safer for global use).

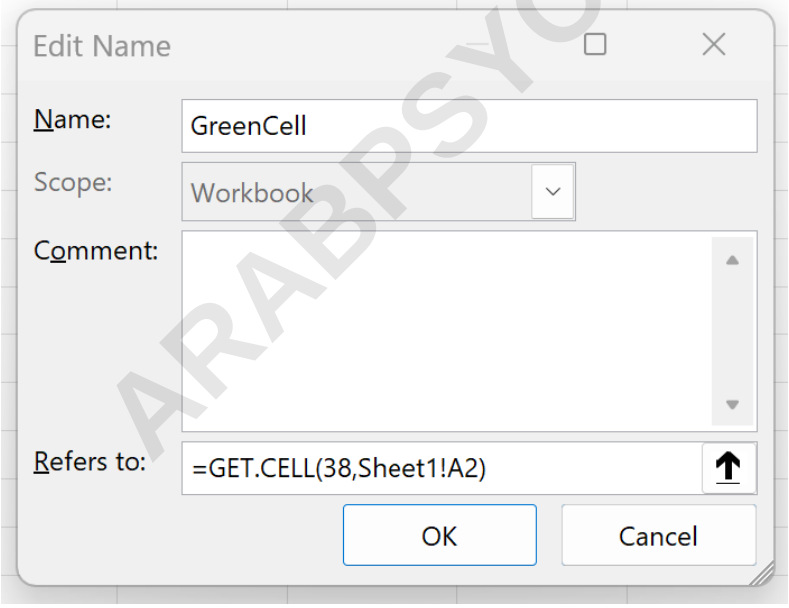
The crucial step involves typing the following formula into the **Refers to** box. This formula utilizes the hidden GET.CELL function:

=GET.CELL(38,Sheet1!A2)

Let's break down this syntax:

38: This is the numerical argument code for GET.CELL that specifically retrieves the background fill color index number.
Sheet1!A2: This is the cell reference. It is vital that we use a relative reference here, even though we specify a fixed cell (A2). Because we did not include dollar signs (e.g., A2), when we apply the **GreenCell** name in cell B2 and drag it down, Excel treats the reference as relative to the position where the defined name is used. When used in B3, it will automatically look at A3; when used in B4, it will look at A4, and so on.

After entering the Name and the formula, click **OK** to save the new Defined Name. You can then close the Name Manager window.



Step 4: Extracting and Verifying the Specific Color Code

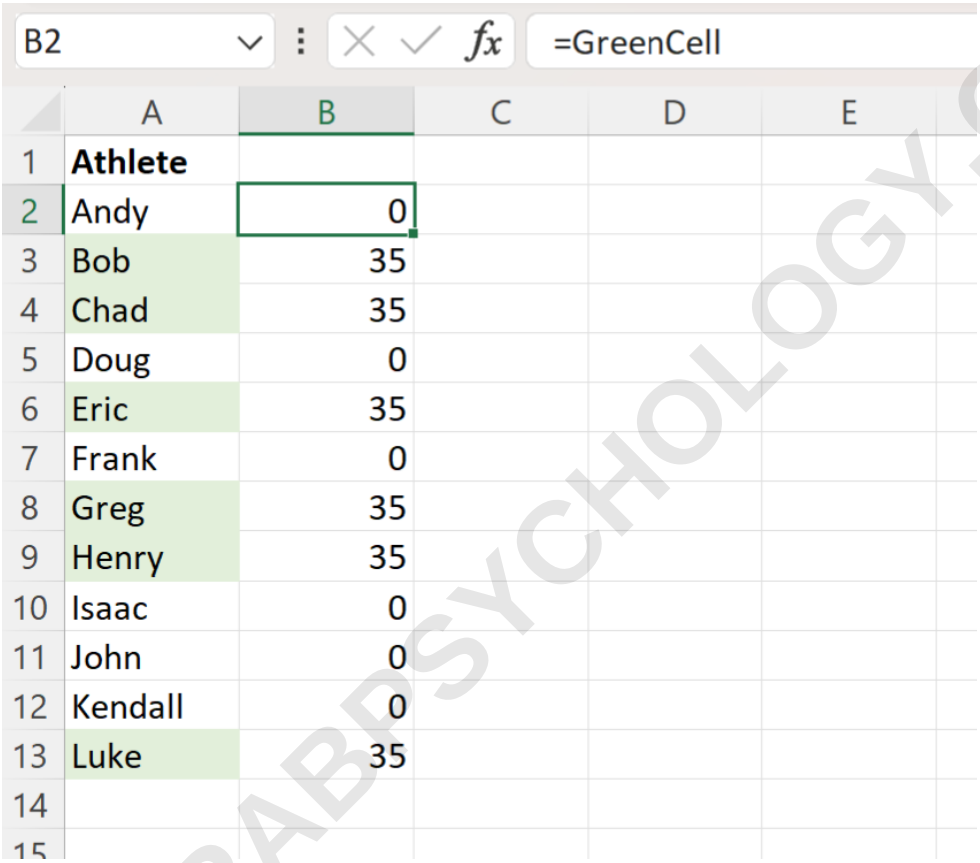
Before implementing the final IF statement, we must determine the exact numerical code that GET.CELL assigns to the specific shade of green used in our data. Color codes in Excel are not

universal for a single color name; they depend heavily on the palette, theme, or custom color settings used.

To verify the code, simply type the newly created defined name into cell **B2**:

=GreenCell

Next, click and drag this formula down column B. This action will apply the defined name to each corresponding cell in column A, revealing the numerical color indices.



Upon reviewing the results, we can clearly observe that the formula returns a color code of **35** for every cell in column A that possesses the specific shade of green used for the **All-Star** designation. Cells without this fill color will return a code of **0** (or another numerical code if they possess a different fill color). This verification step is critical, as the value **35** is the precise trigger we will use in our final logical formula.

Step 5: Implementing the Conditional Logic with the IF Formula

With the color code successfully identified (in this case, 35), we can now integrate this finding into a robust IF statement. This formula will check if the result of our custom **GreenCell** function equals

the code 35, and based on the result, assign the appropriate label.

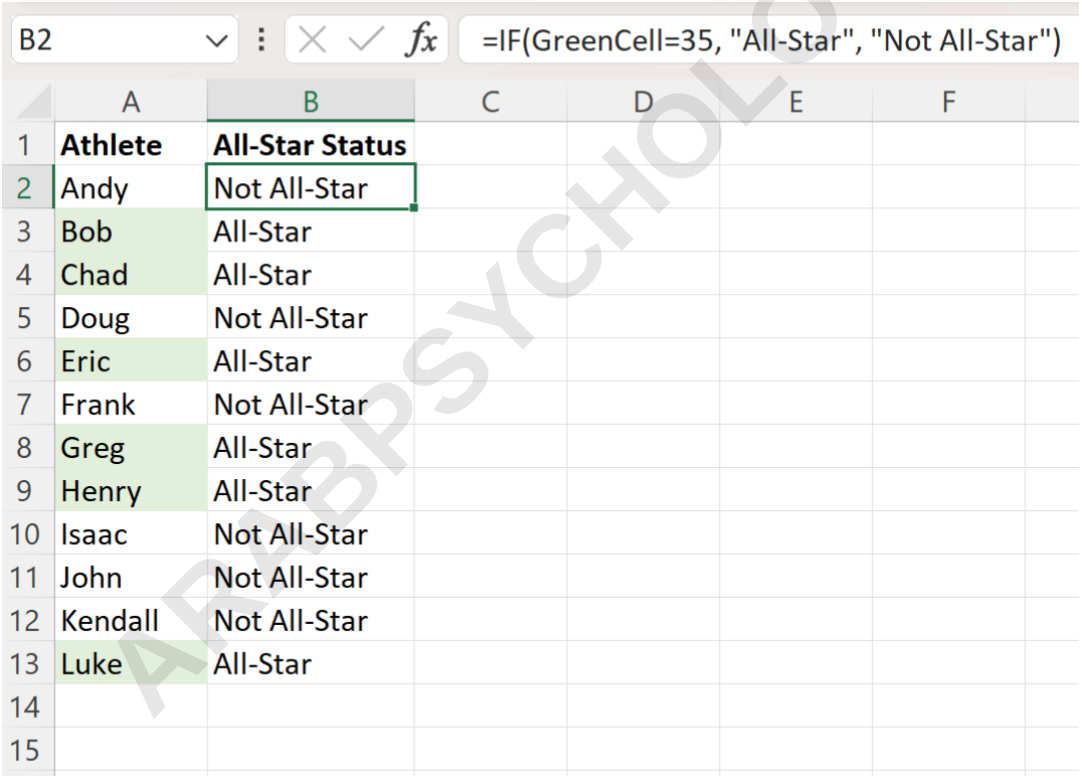
Type the following formula into cell **B2**, replacing the previous test formula:

=IF(GreenCell=35, "All-Star", "Not All-Star")

This formula is structured as follows:

Logical Test: Is the result of the **GreenCell** defined name equal to 35?
Value if True: If the cell is green (code 35), return "All-Star".
Value if False: If the cell is not green (any other code), return "Not All-Star".

Once entered in B2, click and drag this formula down to apply the conditional logic to all remaining cells in column B. The result instantly classifies the players based on the background color of their names.



	A	B	C	D	E	F
1	**Athlete**	**All-Star Status**				
2	Andy	Not All-Star				
3	Bob	All-Star				
4	Chad	All-Star				
5	Doug	Not All-Star				
6	Eric	All-Star				
7	Frank	Not All-Star				
8	Greg	All-Star				
9	Henry	All-Star				
10	Isaac	Not All-Star				
11	John	Not All-Star				
12	Kendall	Not All-Star				
13	Luke	All-Star				
14						
15						

As demonstrated in the resulting table, column B now accurately returns the specified status, "All-Star" or "Not All-Star," dynamically linked to the corresponding cell color in column A, achieving the initial objective.

Understanding Color Codes and Adaptability

It is paramount to recognize that the numerical color code extracted by the GET.CELL function is highly dependent on the current workbook's color palette and the exact shade applied. In our specific example, the shade of green utilized yielded the code **35**.

If you were to use a different shade of green--perhaps a darker or lighter tone, or a custom RGB color--the numerical code returned by **GreenCell** would change. This variability is why Step 4 (Extracting and Verifying the Specific Color Code) is mandatory. You must always test the color first before integrating the resulting number into your final **IF statement**.

Furthermore, this technique generally works best when the formatting is applied manually. If the green color is applied via Conditional Formatting, the behavior of GET.CELL can sometimes be inconsistent depending on the version of Excel and how the formatting rule is defined. For reliable results, manual fill color application is recommended when using this specific workaround.

Key Takeaways and Limitations of the GET.CELL Method

The use of the Defined Name and the legacy GET.CELL function provides a powerful, if unconventional, solution to a fundamental limitation in Excel. It effectively bridges the gap between formatting and calculation by translating visual data into numerical inputs.

However, it is essential to be aware of certain operational limitations associated with this technique:

No Automatic Recalculation: Unlike standard formulas, which automatically update when dependent cell values change, formulas relying on GET.CELL do not automatically recalculate if the cell color is manually changed. If you change a cell from green to white, the formula in column B will likely still show the previous result until a recalculation is forced.
Forcing Recalculation: To ensure the formula updates after a color change, you must manually force Excel to recalculate the sheet. This is typically achieved by pressing **F9** or by making a minor edit to any cell used in the calculation chain.
Macro Security: Since GET.CELL is technically an XLM macro function, users may encounter security prompts in certain environments, although it generally operates without issue when used solely through the Defined Name feature.

By understanding both the power and the limitations of this method, you can confidently integrate cell formatting checks into your logical workflows, making your spreadsheets more visually descriptive and functionally dynamic.