

# How to Easily Change Font Color in VBA: 3 Simple Methods

Authored by  
**stats writer**

November 20, 2025

## RECOMMENDED CITATION

stats writer (2025). *How to Easily Change Font Color in VBA: 3 Simple Methods*.  
PSYCHOLOGICAL SCALES. Retrieved from <https://scales.arabpsychology.com/?p=97992>

Mastering cell formatting is crucial for effective data visualization in Excel. Within VBA (Visual Basic for Applications), developers have several powerful techniques to programmatically adjust font colors. This comprehensive guide outlines the three primary methods for modifying the font color property of a Range object: using intrinsic color constants (VBA Color Names), leveraging the RGB function, and utilizing Hex color codes.

The core mechanism for applying color rests upon setting the `.Font.Color` property of the target Range object. This property expects a Long data type value, which represents the numerical equivalent of the desired color. While using basic color constants is quick, the RGB function offers unparalleled precision, and Hex color codes provide compatibility with standard web design palettes.

The following three methods provide distinct ways to control font color when writing routines in VBA for Excel:

## Method 1: Changing Font Color Using Intrinsic VBA Color Constants (Color Names)

The simplest approach involves using the intrinsic color constants provided directly by VBA. These are predefined variables that map directly to common colors, such as `vbRed`, `vbBlue`, `vbBlack`, `vbWhite`, `vbGreen`, and `vbYellow`. Using these constants significantly improves code readability and speed of development, although the selection is limited to only eight basic colors.

When assigning one of these constants to the `.Font.Color` property, VBA automatically converts the constant name into its corresponding long integer value. This method is ideal for quick formatting tasks where precise color matching is not required.

```
Sub ChangeColor()  
Range("A1").Font.Color = vbRed  
End Sub
```

This particular macro targets cell **A1** and changes the font color to bright red using the built-in `vbRed` constant. This is the most straightforward syntax for single-color application.

## Method 2: Changing Font Color Using the VBA RGB Function

For situations demanding custom colors beyond the basic eight constants, the `RGB()` function is the preferred tool. The RGB function allows you to specify a color by combining varying intensities of the three primary light colors: Red, Green, and Blue. Each component is defined by an integer value ranging from 0 (minimum intensity) to 255 (maximum intensity).

The function takes three arguments: `RGB(Red_Value, Green_Value, Blue_Value)`. By precisely adjusting these three parameters, you can generate over 16 million unique colors, providing maximum customization for your reports and dashboards. When executed, the function returns a single Long integer representing the combined color value that the `.Font.Color` property expects.

#### **Sub ChangeColor()**

```
Range("A1").Font.Color = RGB(255,0,0)
```

```
End Sub
```

This macro sets the font color in cell **A1**. Since the parameters are defined as (255, 0, 0), indicating maximum Red intensity and zero Green and Blue intensity, the result is pure red. This approach ensures precise color matching based on standardized RGB values.

### **Method 3: Changing Font Color Using Hexadecimal Color Codes**

The third powerful method for defining font color involves using Hex color codes. This method is particularly useful when integrating colors defined by graphic designers or copied from web standards, which typically use a six-character hexadecimal sequence (e.g., #RRGGBB).

To utilize a Hex color code in VBA, you must preface the code with the literal value `&H`. This prefix signals to the compiler that the following digits are a hexadecimal number.

#### **Sub ChangeColor()**

```
Range("A1").Font.Color = &H0000FF
```

```
End Sub
```

This macro sets the font color in cell **A1** using the hexadecimal code `&H0000FF`. Although in standard web design, red is represented as `#FF0000`, Excel employs a specialized interpretation of hexadecimal values, requiring specific attention to the color order.

### **Understanding Hexadecimal Color Order in Excel VBA**

When dealing with standard Hex color codes, they are typically ordered Red-Green-Blue (RGB). For example, pure red is `FF0000`. However, Excel VBA processes these long integer values in the reverse order: Blue-Green-Red (BGR). This is a crucial distinction that often trips up new developers.

Therefore, to achieve the color represented by `RRGGBB`, you must input the value as `BBGGRR`.

For example:

To get pure Red (standard RGB: FF0000), you must input `&H0000FF` (Blue=00, Green=00, Red=FF).

To get pure Blue (standard RGB: 0000FF), you must input `&HFF0000` (Blue=FF, Green=00, Red=00).

Always remember to type **&H** in front of the hex color code you want to use. This prefix is essential for VBA to correctly interpret the sequence as a hexadecimal number rather than a decimal integer.

The following examples demonstrate how to apply these formatting techniques in real-world scenarios across different ranges.

## Example 1: Applying Color Using VBA Color Names to a Single Cell

### Scenario: Targeting a Single Cell for Quick Formatting

Consider a simple scenario where text in cell **A1** currently displays in the default black font color. Our goal is to quickly highlight this single cell using a standard VBA constant, such as `vbRed`.

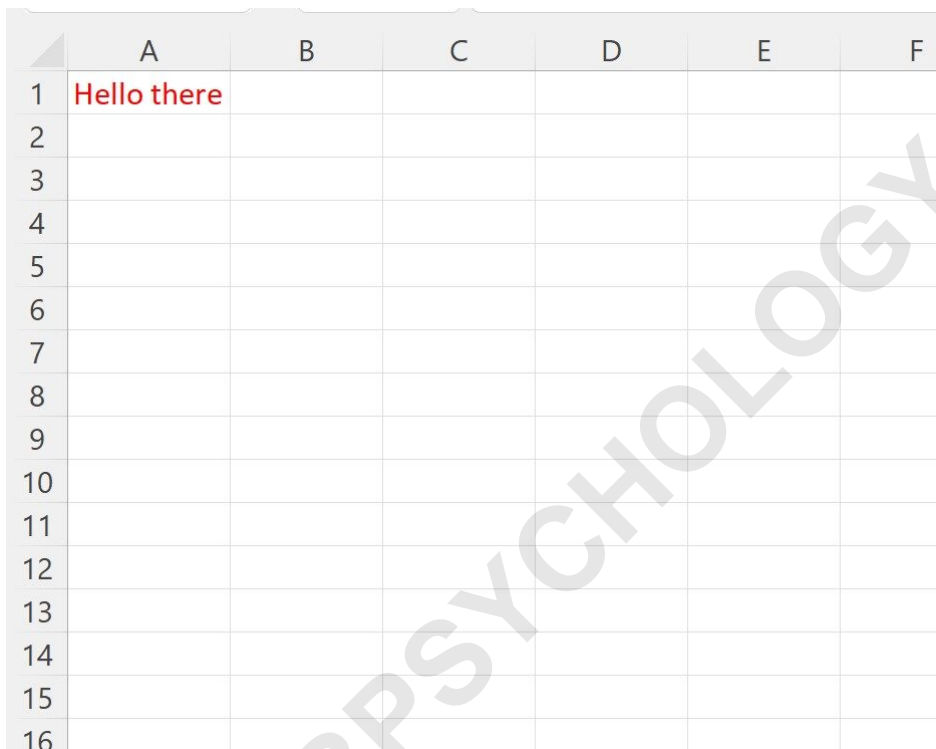
Initial state of the cell **A1**:

	A	B	C	D	E
1	Hello there				
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					

The corresponding macro uses the intrinsic color constant for simplicity:

```
Sub ChangeColor()  
Range("A1").Font.Color = vbRed  
End Sub
```

Upon execution of this routine, the output clearly shows the immediate effect of applying the constant value to the `.Font.Color` property:



	A	B	C	D	E	F
1	Hello there					
2						
3						
4						
5						
6						
7						
8						
9						
10						
11						
12						
13						
14						
15						
16						

As expected, the font color in cell **A1** has been successfully changed to red. This illustrates the efficiency of using VBA color constants for fundamental color requirements.

## Example 2: Applying Custom Color Using RGB Values to a Range

### Scenario: Applying Precision Coloring to Multiple Cells

Now, let's explore how to apply a color to an entire group of cells, specifically the range **A1:A5**, using the highly precise `RGB()` function. Assume the text strings in this range currently use the default black font color.

Initial range **A1:A5**:

	A	B	C	D	E
1	Hello there				
2	Hey ya'll				
3	Greetings				
4	Hi people				
5	Hello				
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					

We define the color red using its maximum RGB values (255, 0, 0) and apply it to the designated range using the following macro:

**Sub ChangeColor()**

**Range("A1:A5").Font.Color = RGB(255,0,0)**

**End Sub**

Running this macro results in the entire selected range being formatted simultaneously:

	A	B	C	D	E	F
1	Hello there					
2	Hey ya'll					
3	Greetings					
4	Hi people					
5	Hello					
6						
7						
8						
9						
10						
11						
12						
13						
14						
15						
16						
17						

Note how the font color in every cell within the **A1:A5** range is uniformly set to the precise red specified by the RGB function. This demonstrates the efficiency of applying complex formatting across contiguous ranges.

### Example 3: Applying Color Using Hexadecimal Codes to a Range

#### Scenario: Using Web-Standard Hex Codes for Range Formatting

Finally, we replicate the formatting on the same range **A1:A5**, but this time leveraging Hex color codes. This method is crucial for maintaining brand consistency when specific hexadecimal values are provided by design specifications.

Initial range **A1:A5** (same as previous example):

	A	B	C	D	E
1	Hello there				
2	Hey ya'll				
3	Greetings				
4	Hi people				
5	Hello				
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					

We utilize the BGR-formatted hexadecimal code `&H0000FF` (which corresponds to pure red, or RGB FF0000) to change the font color:

**Sub ChangeColor()**

**Range("A1:A5").Font.Color = &H0000FF**

**End Sub**

The execution produces the same visual outcome as the previous examples, confirming that all three methods can achieve identical results depending on the input format chosen:

	A	B	C	D	E	F
1	Hello there					
2	Hey ya'll					
3	Greetings					
4	Hi people					
5	Hello					
6						
7						
8						
9						
10						
11						
12						
13						
14						
15						
16						
17						

This confirms that using the BGR-reversed Hex color code, prefixed by `&H`, is a valid and robust method for defining specific colors in VBA.

## Summary of VBA Font Coloring Techniques

Choosing the appropriate method for setting font color depends primarily on your requirement for color precision and the source of your color definitions.

**VBA Color Constants (e.g., `vbRed`):** Fastest and easiest for basic, high-contrast colors (only 8 available).

**RGB Function (e.g., `RGB(255, 0, 0)`):** Best for custom colors requiring precise definition using the standard Red, Green, Blue component model.

**Hexadecimal Codes (e.g., `&H0000FF`):** Useful for developers familiar with web color standards, provided the necessary BGR reversal is applied for Excel compatibility.

For comprehensive details regarding the properties and methods available for cell formatting, you can find the complete documentation for the VBA **Font.Color** property on the official Microsoft website.