How to Use Weekday Function in VBA (With Example)

Authored by

stats writer

November 18, 2025

The **Weekday function** is a critical tool within VBA (Visual Basic for Applications) designed to extract the day of the week from any given date value. Unlike standard Excel formulas that might return a day name, this function returns the result as a simple, easily processable integer. This capability is invaluable when writing complex automation scripts or business logic that requires conditional execution based on whether a date falls on a weekend, a Monday, or any other specific day.

Understanding how to properly implement the **Weekday** function allows developers to create robust and efficient data processing workflows directly within Microsoft Excel. The following sections will detail the function's syntax, its argument requirements, and provide a comprehensive example of its use in a practical Excel environment.

Introduction to the VBA Weekday Function

The primary purpose of the **Weekday** function in VBA is to accept a date expression and return a number between 1 and 7, which corresponds directly to the day of the week. This numerical output makes it highly suitable for looping structures and logical tests within a macro. By default, the function assumes that Sunday is the first day of the week, corresponding to the return value 1.

While the example below utilizes the function within a loop to process a range of cells, the **Weekday** function can be used anywhere a standard function is expected. For instance, it can be utilized in conditional statements (`If...Then...Else`) to perform actions only on weekdays or to calculate due dates based on the proximity of a weekend. Its flexibility makes it a cornerstone of date manipulation routines in VBA programming.

Here is a common implementation pattern showing how this function is used within a subroutine to iterate through data:

```
Sub FindWeekday()

Dim i As Integer

For i = 2 To 9
Range("B" & i) = WorksheetFunction.Weekday(Range("A" & i))
Next i

End Sub
```

This specific macro iterates through cells A2 to A9, applying the **Weekday** function to each date found in column A and subsequently placing the resulting integer value into the corresponding cell in column B (B2 to B9). This demonstrates its capability to handle batch processing efficiently.

Interpreting the Return Value System

The crucial aspect of using the **Weekday** function is understanding the standardized numerical assignment for each day. While the return system can be adjusted using optional arguments (the `FirstDayOfWeek` parameter), the default VBA setting mandates that Sunday is assigned the value 1. This is an important distinction to remember when designing logical tests based on the output.

The sequence proceeds linearly through the standard week structure, culminating in Saturday being represented by the value 7. The consistent assignment ensures that the output is predictable and reliable for programming purposes.

The following list details the default integer correlation for each day of the week:

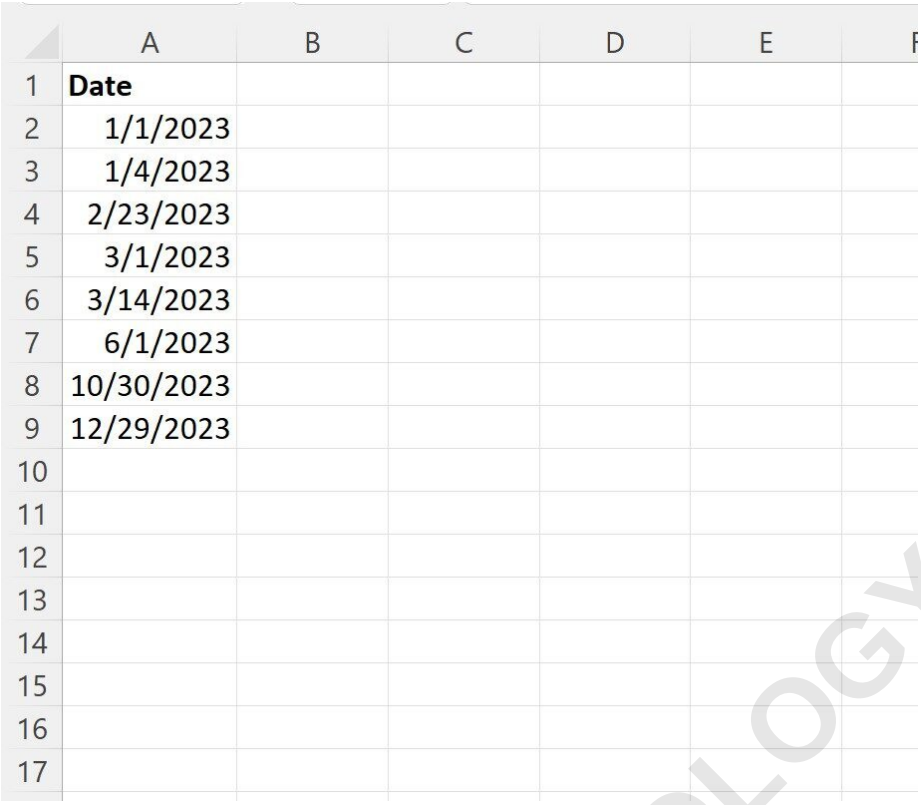
1: Sunday
2: Monday
3: Tuesday
4: Wednesday
5: Thursday
6: Friday
7: Saturday

If your regional settings or programming requirements dictate that the week should begin on Monday (where Monday = 1), you would need to utilize the optional `FirstDayOfWeek` argument. However, for most standard applications, relying on the default Sunday start (vbSunday) is sufficient and requires less explicit code.

Step-by-Step Practical Application

To illustrate the power and simplicity of the **Weekday function**, consider a scenario where we have a column containing various dates in an Excel spreadsheet. Our objective is to determine the day of the week for each date and log that information into an adjacent column, thereby allowing for subsequent filtering or analysis based on the day.

Suppose we begin with the following dataset occupying cells A2 through A9 in our worksheet. This data setup is typical for reporting or scheduling tasks where the day context is missing:



	A	B	C	D	E	F
1	**Date**					
2	1/1/2023					
3	1/4/2023					
4	2/23/2023					
5	3/1/2023					
6	3/14/2023					
7	6/1/2023					
8	10/30/2023					
9	12/29/2023					
10						
11						
12						
13						
14						
15						
16						
17						

We aim to populate Column B with the corresponding day index. This transformation is necessary because Excel handles dates as serial numbers, and converting them to a weekday index requires a specific function call, which we will encapsulate within a VBA subroutine.

Analyzing the Example VBA Macro

To achieve the desired output, we construct a concise and efficient subroutine named `FindWeekday`. This code leverages a standard `For...Next` loop structure, which is the most effective method for iterating through a continuous range of data cells.

The crucial element within the loop is the use of `WorksheetFunction.Weekday`. Since the **Weekday** function exists both as a standard VBA language function and as an Excel worksheet function, accessing it via the `WorksheetFunction` object ensures explicit calling, although in simple cases, the direct VBA call would also suffice.

The complete macro structure is defined as follows:

Sub FindWeekday()

Dim i As Integer

For i = 2 To 9

```
Range("B" & i) = WorksheetFunction.Weekday(Range("A" & i))
Next i

End Sub
```

The loop variable `i` controls the row number, starting at row 2 and ending at row 9, corresponding exactly to our dataset range A2:A9. In each iteration, the date value from column A is passed to the **Weekday** function, and the returned integer is written directly to the cell in column B. This automation significantly reduces the potential for error compared to manually entering formulas across dozens or hundreds of rows.

Output and Verification of Results

Upon successfully running the `FindWeekday` subroutine, the values in Column B are instantly calculated and populated. The resulting output demonstrates the effective conversion of date values into their corresponding numerical weekday indexes based on the standard 1=Sunday mapping:

	A	B	C	D	E
1	**Date**	**Weekday**			
2	1/1/2023	1			
3	1/4/2023	4			
4	2/23/2023	5			
5	3/1/2023	4			
6	3/14/2023	3			
7	6/1/2023	5			
8	10/30/2023	2			
9	12/29/2023	6			
10					
11					
12					
13					
14					
15					
16					
17					

Column B now clearly displays the day of the week (as an integer) for every date listed in column

A. To ensure complete understanding, it is helpful to verify a few sample results against a calendar:

The date **1/1/2023** falls on a Sunday. According to the default index system (Sunday=1), the **Weekday function** correctly returns **1**.
The date **1/4/2023** is a Wednesday. This corresponds to the integer value **4** in the output.
The date **2/23/2023** falls on a Thursday. This yields the expected integer result of **5**.

This verification process confirms that the macro performed its intended task accurately, providing a robust method for date analysis within Excel. The resulting integers can now be easily used in subsequent VBA code for classification or filtering purposes.

Alternative: Using WeekdayName for String Output

While the **Weekday** function excels at returning numerical representations, sometimes the requirement is to display the actual name of the day (e.g., "Monday" or "Tue"). If your goal is to return the day of the week as a formatted name string rather than an integer, you should opt to use the related VBA function, **WeekdayName**, instead.

The **WeekdayName** function takes the integer returned by the **Weekday** function and converts it into a localized, human-readable string. This is particularly useful when the output column is intended for end-user readability rather than further machine processing. This option provides clarity and improves the user interface of automated reports.

Conclusion and Further Resources

Mastering the **Weekday** function is essential for anyone utilizing VBA for data manipulation, particularly when dealing with chronological data. Its simple structure and predictable integer output make it highly versatile for constructing robust date logic, allowing programmers to easily identify weekends, specific workdays, or plan automated tasks based on the rhythm of the week.

For those seeking deeper technical understanding, including details on the optional `FirstDayOfWeek` argument and handling locale-specific date formatting, the complete official documentation for the VBA **Weekday** function provides comprehensive guidance.