



VBA CODE EXAMPLES

[Return to VBA Code Examples](#)

VBA GUIDES

Formatting Numbers in Excel VBA [See Pricing](#)



IN THIS ARTICLE

Formatting Numbers in Excel VBA

How to Use the Format Function in VBA

- Creating a Format String

- Using a Format String for Alignment

- Using Literal Characters Within the Format String

- Use of Commas in a Format String

- Creating Conditional Formatting within the Format String

- Using Fractions in Formatting Strings

- Date and Time Formats

- Predefined Formats

- General Number

- Currency

- Fixed

- Standard

- Percent

- Scientific

- Yes/No

- True/False

- On/Off

- General Date

- Long Date

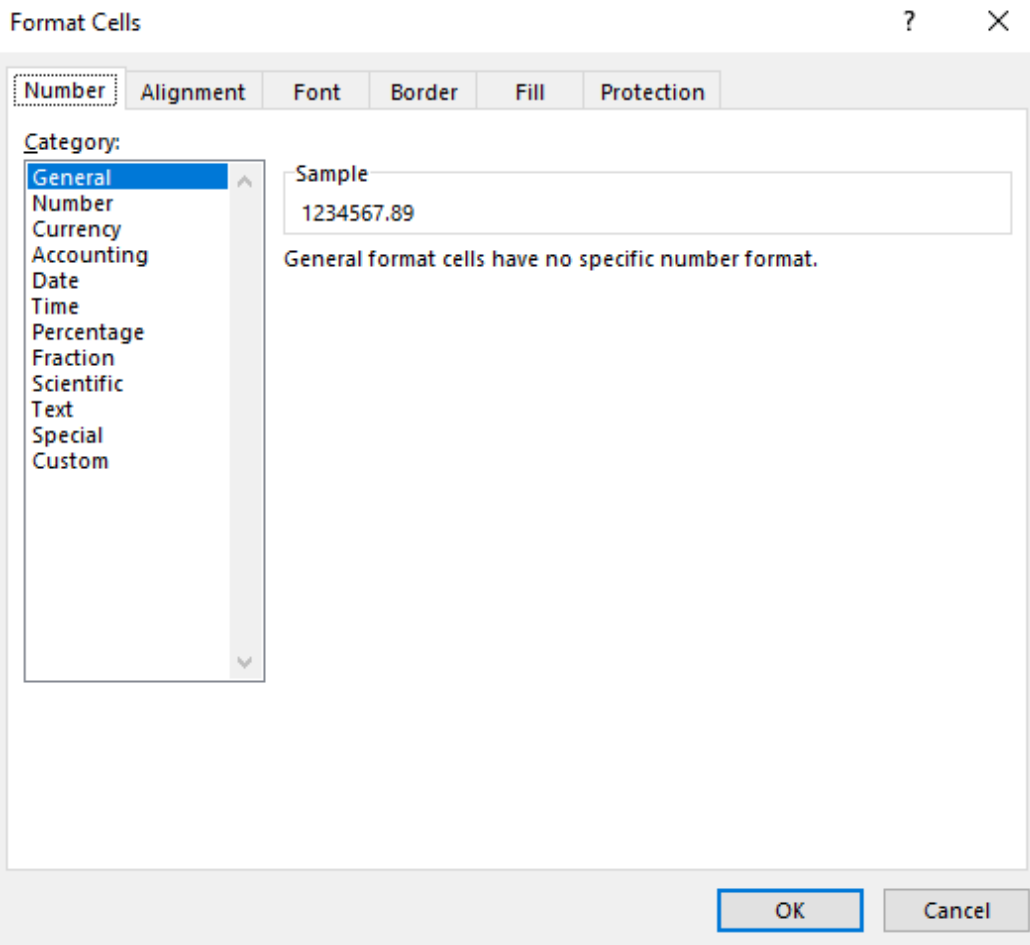
- Medium Date

- Short Date

Long Time
Medium Time
Short Time
Dangers of Using Excel's Pre-Defined Formats in Dates and Times
User-Defined Formats for Numbers

Ultimate VBA Add-in

Learn More

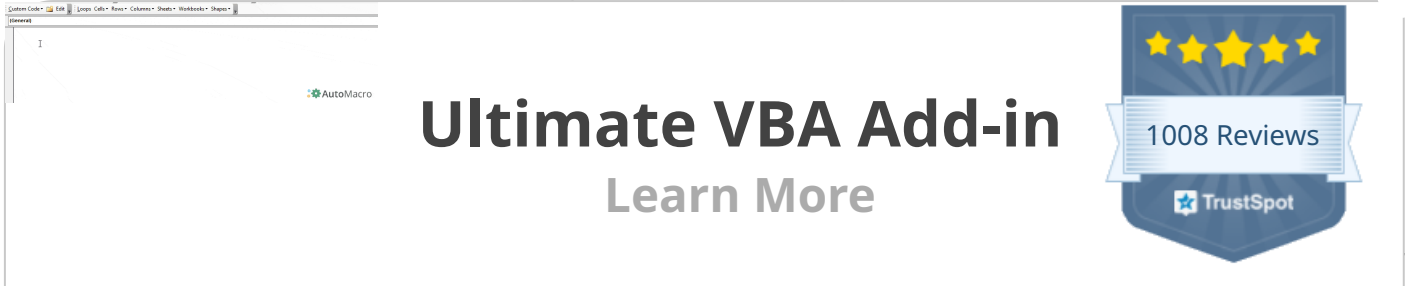
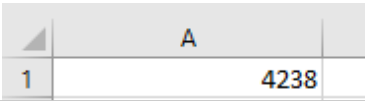


Formatting of numbers make the numbers easier to read and understand. The Excel default for numbers entered into cells is 'General' format, which means that the number is displayed exactly as you typed it in.

For example, if you enter a round number e.g. 4238, it will be displayed as 4238 with no decimal point or thousands separators. A decimal number such as 9325.89 will be displayed with the decimal point and the decimals. This means that it will not line up in the column with the round numbers, and will look extremely messy.

Also, without showing the thousands separators, it is difficult to see how large a number actually is without counting the individual digits. Is it in millions or tens of millions?

From the point of view of a user looking down a column of numbers, this makes it quite difficult to read and compare.



Ultimate VBA Add-in
Learn More

Excel. This applies to not only an entered value in a cell on a worksheet, but also things like [message boxes](#), [UserForm controls](#), [charts and graphs](#), and the [Excel status bar](#) at the bottom left hand corner of the worksheet.

The [Format function](#) is an extremely useful function in VBA in presentation terms, but it is also very complex in terms of the flexibility offered in how numbers are displayed.

How to Use the Format Function in VBA

If you are showing a message box, then the Format function can be used directly:

```
MsgBox Format(1234567.89, "#,##0.00")
```

This will display a large number using commas to separate the thousands and to show 2 decimal places. The result will be 1,234,567.89. The zeros in place of the hash ensure that decimals will be shown as 00 in whole numbers, and that there is a leading zero for a number which is less than 1

The hashtag symbol (#) represents a digit placeholder which displays a digit if it is available in that position, or else nothing.

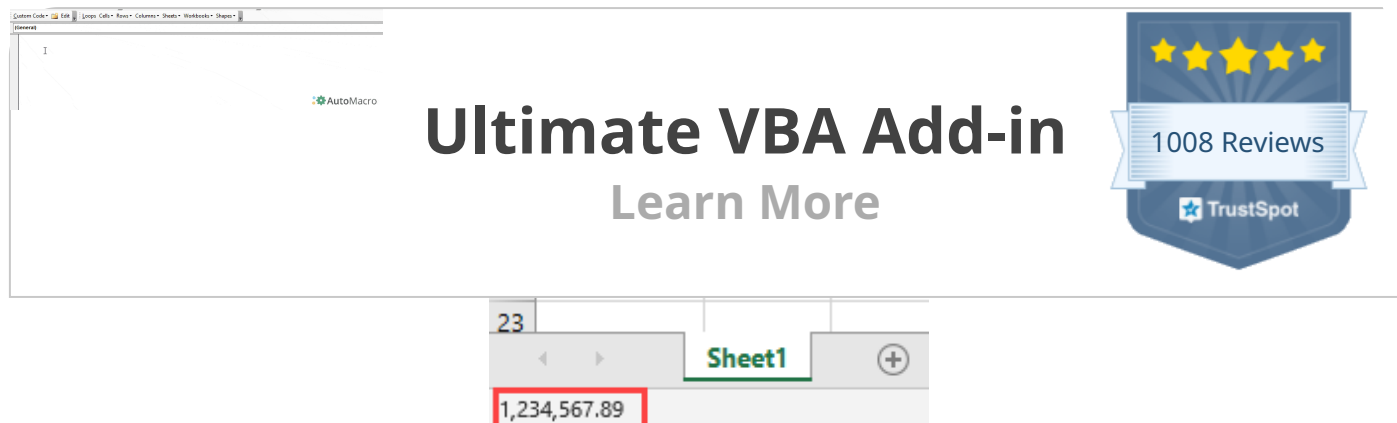
You can also use the format function to address an individual cell, or a [range of cells](#) to change the format:

```
Sheets("Sheet1").Range("A1:A10").NumberFormat = "#,##0.00"
```

This code will set the range of cells (A1 to A10) to a custom format which separates the thousands with commas and shows 2 decimal places.

If you check the format of the cells on the Excel front end, you will find that a new custom format has been created.

You can also format numbers on the Excel Status Bar at the bottom left hand corner of the Excel window:



You clear this from the status bar by using:

```
Application.StatusBar = ""
```

Creating a Format String

This example will add the text 'Total Sales' after each number, as well as including a thousands separator

```
Sheets("Sheet1").Range("A1:A6").NumberFormat = "#,##0.00" Total Sales""
```

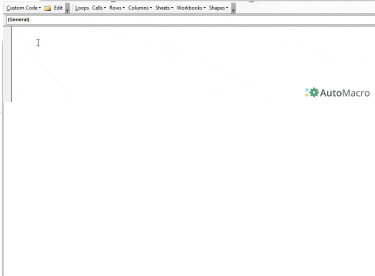
This is what your numbers will look like:

	A	B
1	4,238.00 Total Sales	
2	9,325.89 Total Sales	
3	15,876.00 Total Sales	
4	34.20 Total Sales	
5	239.52 Total Sales	
6	29,713.61 Total Sales	

Note that cell A6 has a 'SUM' formula, and this will include the 'Total Sales' text without requiring formatting. If the formatting is applied, as in the above code, it will not put an extra instance of 'Total Sales' into cell A6


Although the cells now display alpha numeric characters, the numbers are still present in numeric form. The 'SUM' formula still works because it is using the numeric value in the background, not how the number is formatted.

The comma in the format string provides the thousands separator. Note that you only need to



Ultimate VBA Add-in

Learn More



other numbers.

By using a single zero to the left of the decimal point and two zeros to the right of the decimal point in the format string, this will give the required result (0.80).

If there was only one zero to the right of the decimal point, then the result would be '0.8' and everything would be displayed to one decimal place.




Using a Format String for Alignment

We may want to see all the decimal numbers in a range aligned on their decimal points, so that all the decimal points are directly under each other, however many places of decimals there are on each number.

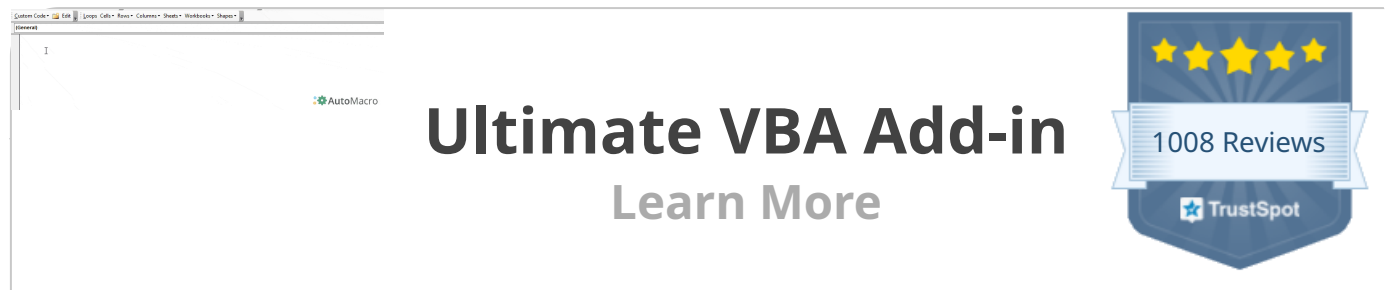
You can use a question mark (?) within your format string to do this. The '?' indicates that a number is shown if it is available, or a space

```
Sheets("Sheet1").Range("A1:A6").NumberFormat = "#,##0.00??"
```

This will display your numbers as follows:

A6				=SUM(A1:A5)
	A			
1				4,238.00
2				9,325.89
3				15,876.00
4				34.20
5				239.523
6				29,713.613
7				

All the decimal points now line up underneath each other. Cell A5 has three decimal places and this would throw the alignment out normally, but using the '?' character aligns everything perfectly.



refers to your local and changes it to the one appropriate for the locale that is set on the Windows Control Panel. This could have implications if your Excel application is being distributed in other countries and you want to ensure that whatever the locale is, the currency indicator is always the same.

You may also want to indicate that the numbers are in millions in the following example:

```
Sheets("Sheet1").Range("A1:A6").NumberFormat = "\$#,##0.00 \m"
```

This will produce the following results on your worksheet:

A6	<input type="button" value="X"/> <input type="button" value="✓"/> <input type="button" value="fx"/>	<input type="text" value="=SUM(A1:A5)"/>
	A	B
1	\$4,238.00 m	
2	\$9,325.89 m	
3	\$15,876.00 m	
4	\$34.20 m	
5	\$239.52 m	
6	\$29,713.61 m	
7		

In using a backslash to display literal characters, you do not need to use a backslash for each individual character within a string. You can use:

```
Sheets("Sheet1").Range("A1:A6").NumberFormat = "\$#,##0.00 \mill"
```

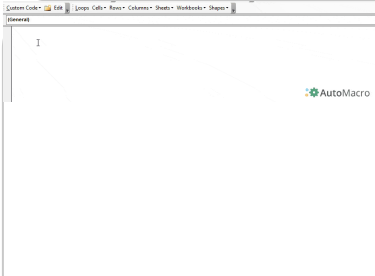
This will display 'mill' after every number within the formatted range.

You can use most characters as literals, but not reserved characters such as 0, #, ?

Use of Commas in a Format String


We have already seen that commas can be used to create thousands separators for large numbers, but they can also be used in another way.

By using them at the end of the numeric part of the format string, they act as scalers of thousands. In other words, they will divide each number by 1,000 every time there is a



Ultimate VBA Add-in

Learn More



This will show the numbers divided by 1,000 although the original number will still be in background in the cell.

If you put two commas in the format string, then the numbers will be divided by a million

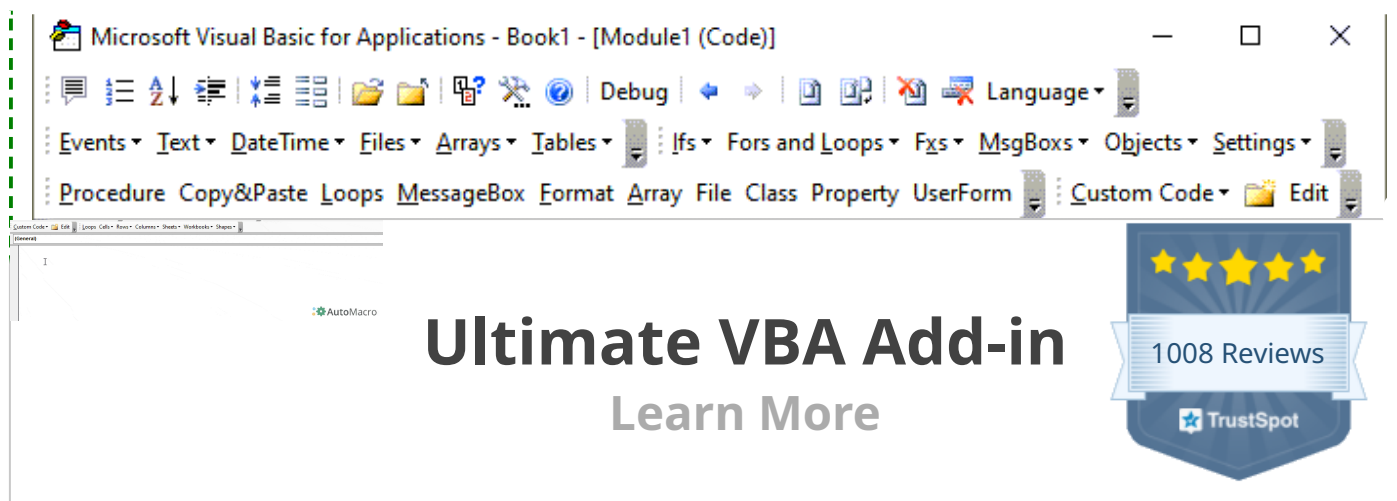
```
Sheets("Sheet1").Range("A1:A6").NumberFormat = "\$#,##0.00,,\m"
```

This will be the result using only one comma (divide by 1,000):

	A	B	C	D
1	\$4.24m			
2	\$9.33m			
3	\$15.88m			
4	\$0.03m			
5	\$0.24m			
6	\$29.71m			
7				

VBA Coding Made Easy

Stop searching for VBA code online. Learn more about AutoMacro - A VBA Code Builder that allows beginners to code procedures from scratch with minimal coding knowledge and with many time-saving features for all users!



Ultimate VBA Add-in

Learn More

1008 Reviews

TrustSpot

Creating Conditional Formatting within the Format String

You could set up conditional formatting on the front end of Excel, but you can also do it within your VBA code, which means that you can manipulate the format string programmatically to make changes.

You can use up to four sections within your format string. Each section is delimited by a semicolon (;). The four sections correspond to positive, negative, zero, and text

```
Range("A1:A7").NumberFormat = "#,##0.00;[Red]-#,##0.00;[Green] #,##0.00;[Blue]"
```



In this example, we use the same hash, comma, and zero characters to provide thousand separators and two decimal points, but we now have different sections for each type of value.

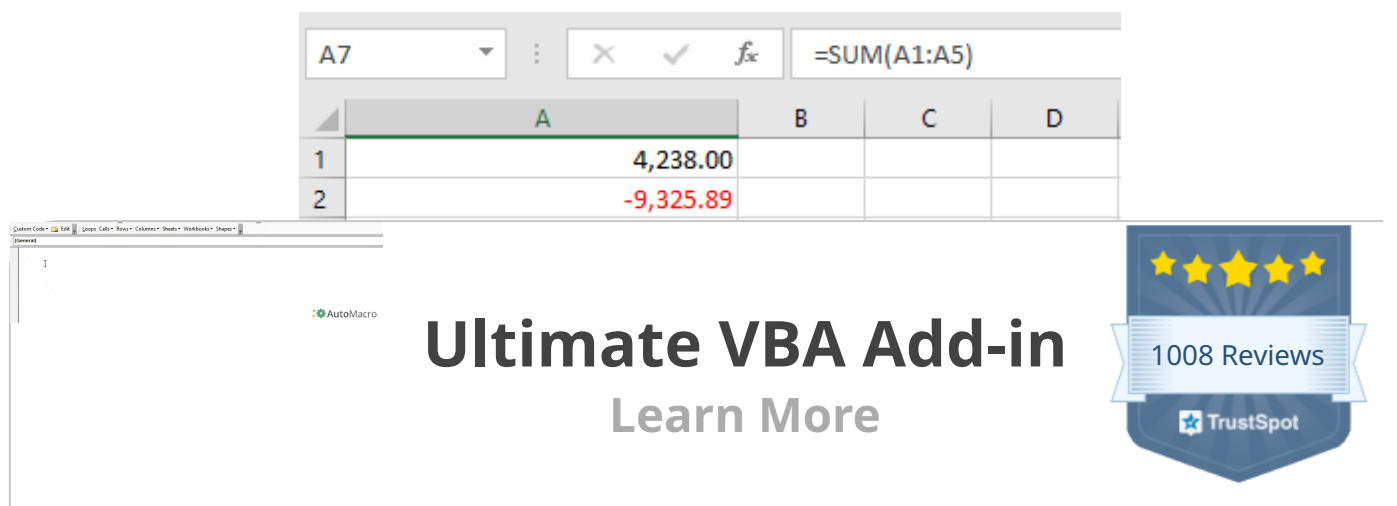
The first section is for positive numbers and is no different to what we have already seen previously in terms of format.

The second section for negative numbers introduces a color (Red) which is held within a pair of square brackets. The format is the same as for positive numbers except that a minus (-) sign has been added in front.

The third section for zero numbers uses a color (Green) within square brackets with the numeric string the same as for positive numbers.

The final section is for text values, and all that this needs is a color (Blue) again within square brackets

This is the result of applying this format string:



The screenshot shows an Excel spreadsheet with the following data:

	A	B	C	D
1	4,238.00			
2	-9,325.89			

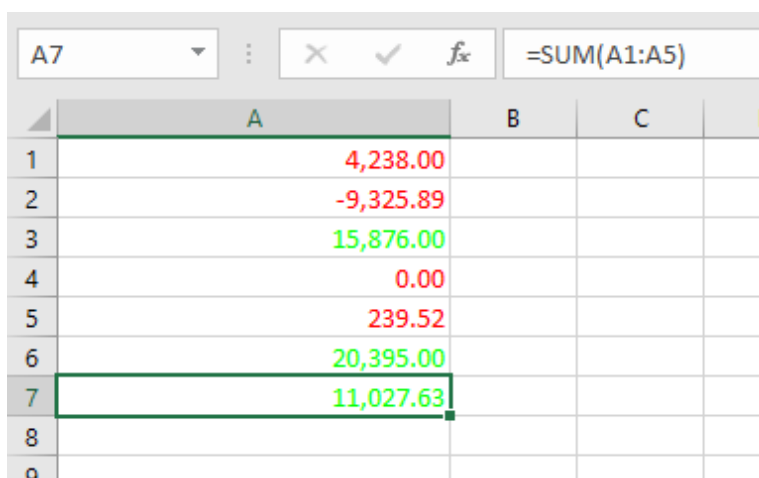
The formula bar shows `=SUM(A1:A5)`. Below the spreadsheet, the text "Ultimate VBA Add-in" is displayed with a "Learn More" link. To the right is a TrustSpot badge showing 1008 Reviews and 5 stars.

You can go further with conditions within the format string. Suppose that you wanted to show every positive number above 10,000 as green, and every other number as red you could use this format string:

```
Range("A1:A7").NumberFormat = "[>=10000][Green]#,##0.00;[<10000][Red]#,##0.00"
```

This format string includes conditions for `>=10000` set in square brackets so that green will only be used where the number is greater than or equal to 10000

This is the result:



The screenshot shows the same Excel spreadsheet as before, but with additional data and conditional formatting applied. The numbers greater than or equal to 10,000 are displayed in green, and all other numbers are displayed in red.

	A	B	C	D
1	4,238.00			
2	-9,325.89			
3	15,876.00			
4	0.00			
5	239.52			
6	20,395.00			
7	11,027.63			
8				
9				

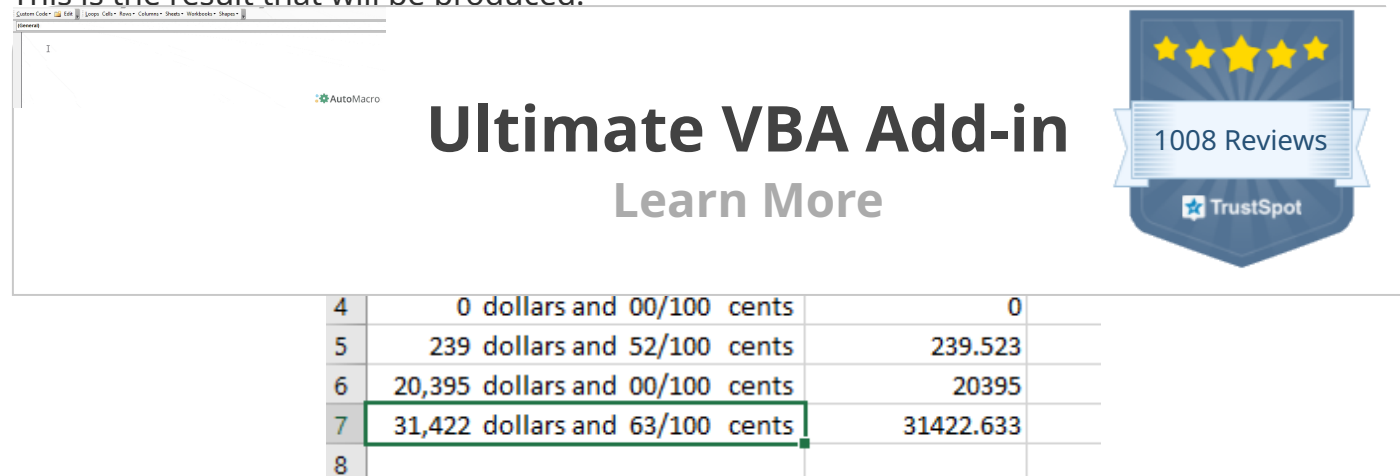
Using Fractions in Formatting Strings

Fractions are not often used in spreadsheets, since they normally equate to decimals which everyone is familiar with.

However, sometimes they do serve a purpose. This example will display dollars and cents:

```
Range("A1:A7").NumberFormat = "#,##0 "" dollars and "" 00/100 "" cents ""
```

This is the result that will be produced:



4	0 dollars and 00/100 cents	0
5	239 dollars and 52/100 cents	239.523
6	20,395 dollars and 00/100 cents	20395
7	31,422 dollars and 63/100 cents	31422.633
8		

Remember that in spite of the numbers being displayed as text, they are still there in the background as numbers and all the Excel formulas can still be used on them.

Date and Time Formats

Dates are actually numbers and you can use formats on them in the same way as for numbers. If you format a date as a numeric number, you will see a large number to the left of the decimal point and a number of decimal places. The number to the left of the decimal point shows the number of days starting at 01-Jan-1900, and the decimal places show the time based on 24hrs

```
MsgBox Format(Now(), "dd-mmm-yyyy")
```

This will format the current date to show '08-Jul-2020'. Using 'mmm' for the month displays the first three characters of the month name. If you want the full month name then you use 'mmmm'

You can include times in your format string:

```
MsgBox Format(Now(), "dd-mmm-yyyy hh:mm AM/PM")
```

This will display '08-Jul-2020 01:25 PM'

'hh:mm' represents hours and minutes and AM/PM uses a 12-hour clock as opposed to a 24-hour clock.

You can incorporate text characters into your format string:

```
MsgBox Format(Now(), "dd-mmm-yyyy hh:mm AM/PM" & today & "")
```

[illegible]

VBA Programming | Code Generator does work for you!

Predefined Formats

Excel has a number of built-in formats for both numbers and dates that you can use in your code. These mainly reflect what is available on the number formatting front end, although some of them go beyond what is normally available on the pop-up window. Also, you do not have the flexibility over number of decimal places, or whether thousands separators are used.

General Number

This format will display the number exactly as it is

```
MsgBox Format(1234567.89, "General Number")
```

The result will be 1234567.89

Currency

```
MsgBox Format(1234567.894, "Currency")
```

This format will add a currency symbol in front of the number e.g. \$, £ depending on your locale, but it will also format the number to 2 decimal places and will separate the thousands with commas.

The result will be \$1,234,567.89

Fixed

```
MsgBox Format(1234567.894, "Fixed")
```

This format displays at least one digit to the left but only two digits to the right of the decimal



Ultimate VBA Add-in

Learn More



```
MsgBox Format(1234567.894, "Standard")
```

This displays the number with the thousand separators, but only to two decimal places.

The result will be 1,234,567.89

[AutoMacro](#) | [Ultimate VBA Add-in](#) | [Click for Free Trial!](#)

Percent

```
MsgBox Format(1234567.894, "Percent")
```

The number is multiplied by 100 and a percentage symbol (%) is added at the end of the number. The format displays to 2 decimal places

The result will be 123456789.40%

Scientific

```
MsgBox Format(1234567.894, "Scientific")
```

This converts the number to Exponential format

The result will be 1.23E+06

Yes/No

```
MsgBox Format(1234567.894, "Yes/No")
```

This displays 'No' if the number is zero, otherwise displays 'Yes'



This displays 'False' if the number is zero, otherwise displays 'True'

The result will be 'True'

[AutoMacro](#) | [Ultimate VBA Add-in](#) | [Click for Free Trial!](#)

On/Off

```
MsgBox Format(1234567.894, "On/Off")
```

This displays 'Off' if the number is zero, otherwise displays 'On'

The result will be 'On'

General Date

```
MsgBox Format(Now(), "General Date")
```

This will display the date as date and time using AM/PM notation. How the date is displayed depends on your settings in the Windows Control Panel (Clock and Region | Region). It may be displayed as 'mm/dd/yyyy' or 'dd/mm/yyyy'

The result will be '7/7/2020 3:48:25 PM'

Long Date

```
MsgBox Format(Now(), "Long Date")
```

This will display a long date as defined in the Windows Control Panel (Clock and Region |



```
MsgBox Format(Now(), "Medium Date")
```

This displays a date as defined in the short date settings as defined by locale in the Windows Control Panel.

The result will be '07-Jul-20'

[AutoMacro](#) | [Ultimate VBA Add-in](#) | [Click for Free Trial!](#)

Short Date

```
MsgBox Format(Now(), "Short Date")
```

Displays a short date as defined in the Windows Control Panel (Clock and Region | Region). How the date is displayed depends on your locale. It may be displayed as 'mm/dd/yyyy' or 'dd/mm/yyyy'

The result will be '7/7/2020'

Long Time

```
MsgBox Format(Now(), "Long Time")
```

Displays a long time as defined in Windows Control Panel (Clock and Region | Region).

The result will be '4:11:39 PM'

Medium Time

```
MsgBox Format(Now(), "Medium Time")
```

Displays a medium time as defined by your locale in the Windows Control Panel. This is usual



Ultimate VBA Add-in

Learn More



```
MsgBox Format(Now(), "Short Time")
```

Displays a medium time as defined in Windows Control Panel (Clock and Region | Region). This is usually set as 24-hour format with hours and minutes

The result will be '16:18'

[AutoMacro](#) | [Ultimate VBA Add-in](#) | [Click for Free Trial!](#)

Dangers of Using Excel's Pre-Defined Formats in Dates and Times

The use of the pre-defined formats for dates and times in Excel VBA is very dependent on the settings in the Windows Control Panel and also what the locale is set to

Users can easily alter these settings, and this will have an effect on how your dates and times are displayed in Excel

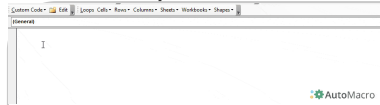
For example, if you develop an Excel application which uses pre-defined formats within your VBA code, these may change completely if a user is in a different country or using a different locale to you. You may find that column widths do not fit the date definition, or on a user form the ActiveX control such as a combo box (drop down) control is too narrow for the dates and times to be displayed properly.

You need to consider where the audience is geographically when you develop your Excel application

User-Defined Formats for Numbers

There are a number of different parameters that you can use when defining your format string:

Character	Description
Null String	No formatting



Ultimate VBA Add-in

[Learn More](#)



is rounded to the number of decimal places shown by the zeros. If there are more digits before the decimal point than zeros these will be displayed normally.

#

Digit placeholder. This displays a digit or nothing. It works the same as the zero placeholder above, except that leading and trailing zeros are not displayed. For example 0.75 would be displayed using zero placeholders, but this would be .75 using # placeholders.

. Decimal point.

Only one permitted per format string. This character depends on the settings in the Windows Control Panel.

%

Percentage placeholder. Multiplies number by 100 and places % character where it appears in the format string

, (comma)

Thousand separator. This is used if 0 or # placeholders are used and the format string contains a comma. One comma to the left of the decimal point indicates round to the nearest thousand. E.g. ##0, Two adjacent commas to the left of the thousand separator indicate rounding to the nearest million. E.g. ##0,,

E- E+

Scientific format. This displays the number exponentially.

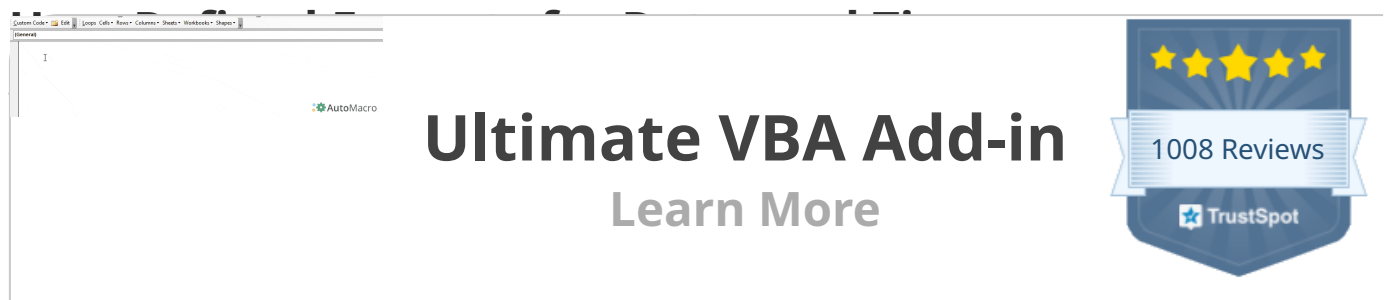
: (colon)

Time separator – used when formatting a time to split hours, minutes and seconds.

/

Date separator – this is used when specifying a format for a date

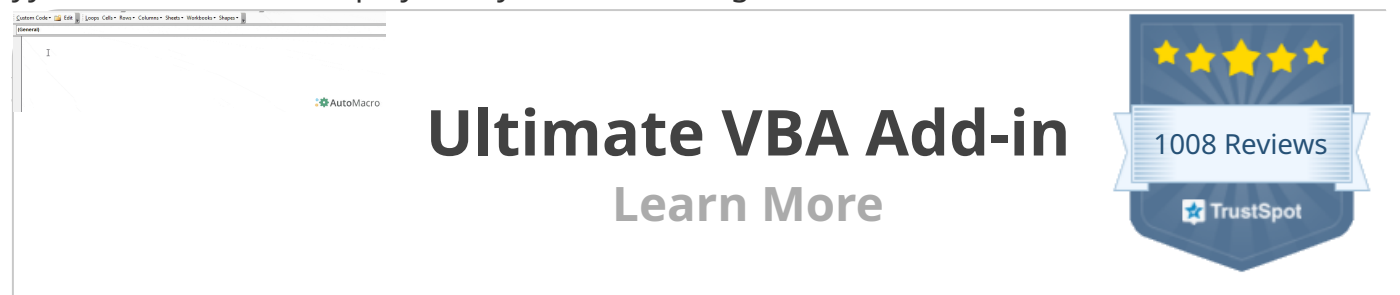
- + £ \$ () Displays a literal character. To display a character other than listed here, precede it with a backslash (\)

A screenshot of the 'Ultimate VBA Add-in' interface. The interface features a dark blue header with the text 'Ultimate VBA Add-in' in large white font, and 'Learn More' in a smaller white font below it. To the right of the text is a TrustSpot badge showing five yellow stars and the text '1008 Reviews'. The background of the interface is a light gray grid pattern. In the top left corner, there is a small menu bar with options like 'Custom Code', 'Edit', 'Insert', 'Cells', 'Rows', 'Columns', 'Sheets', 'Workbooks', and 'Steps'. Below the menu bar, there is a small 'AutoMacro' icon.

d	Display the day as a number without leading zero
dd	Display the day as a number with leading zero
ddd	Display the day as an abbreviation (Sun – Sat)
dddd	Display the full name of the day (Sunday – Saturday)
dddddd	Display a date serial number as a complete date according to Short Date in the International settings of the windows Control Panel
ddddddd	Displays a date serial number as a complete date according to Long Date in the International settings of the Windows Control Panel.
w	Displays the day of the week as a number (1 = Sunday)
ww	Displays the week of the year as a number (1-53)
m	Displays the month as a number without leading zero
mm	Displays the month as a number with leading zeros
mmm	Displays month as an abbreviation (Jan-Dec)
mmmm	Displays the full name of the month (January – December)
q	Displays the quarter of the year as a number (1-4)

y Displays the day of the year as a number (1-366)

yy Displays the year as a two-digit number

A screenshot of the 'Ultimate VBA Add-in' interface. The interface shows a menu bar with options like 'Custom Code', 'Edit', 'Insert Cells', 'Rows', 'Columns', 'Insert', 'Worksheet', and 'Steps'. Below the menu bar, there is a large text area with the text 'Ultimate VBA Add-in' and 'Learn More' below it. To the right of the text area is a TrustSpot review badge showing five stars and '1008 Reviews'. The badge is blue with a white ribbon across the middle containing the text '1008 Reviews'. The TrustSpot logo is at the bottom of the badge.

n Displays the minute as a number without leading zero

nn Displays the minute as a number with leading zero

s Displays the second as a number without leading zero

ss Displays the second as a number with leading zero

ttttt Display a time serial number as a complete time.

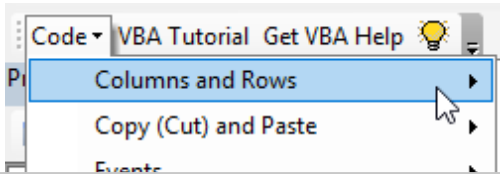
AM/PM Use a 12-hour clock and display AM or PM to indicate before or after noon.

am/pm Use a 12-hour clock and use am or pm to indicate before or after noon

A/P Use a 12-hour clock and use A or P to indicate before or after noon

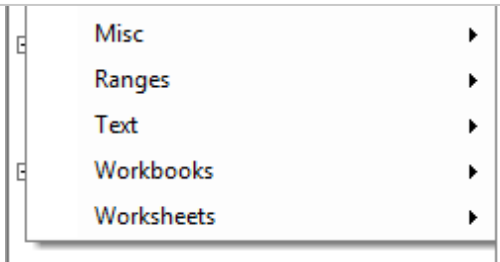
a/p Use a 12-hour clock and use a or p to indicate before or after noon

VBA Code Examples Add-in



Ultimate VBA Add-in

[Learn More](#)



(No installation required!)

[Free Download](#)

VBA Code Generator



[Learn More](#)

AD



Ultimate VBA Add-in

Learn More



AD



**Ukrainian
children
need your
support.**

DONATE NOW



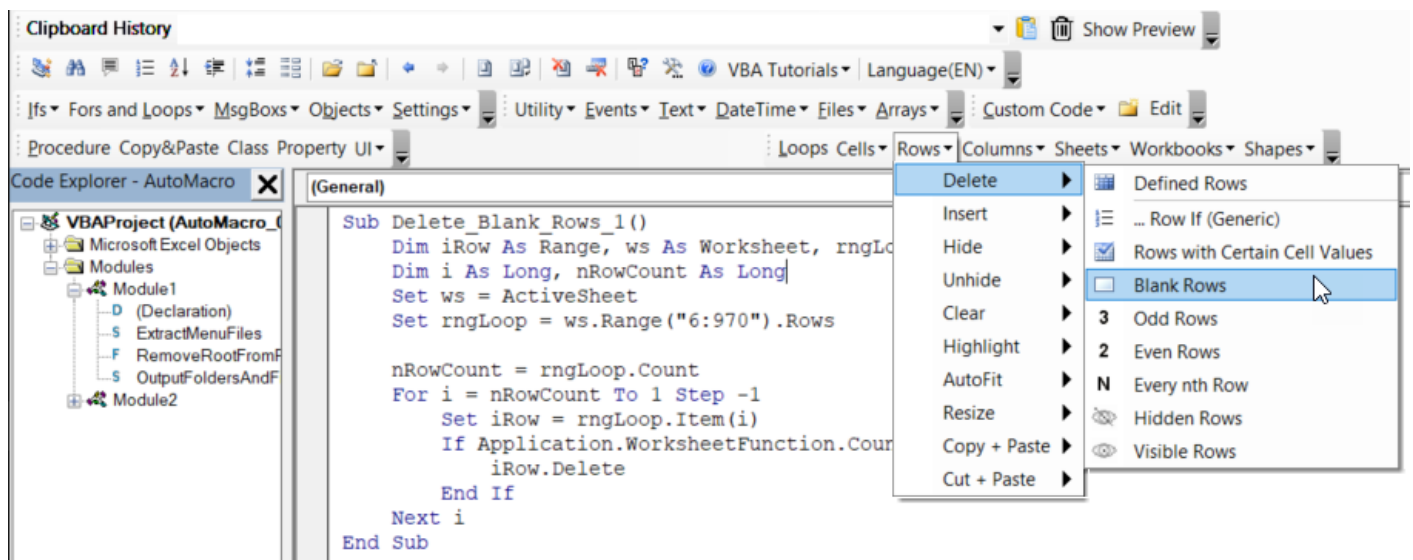
[Return to VBA Code Examples](#)

Ultimate VBA Add-in

Learn More



1008 Reviews



AutoMacro: VBA Add-in with Hundreds of Ready-To-

Use VBA Code Examples



Ultimate VBA Add-in

Learn More



VBA CODE GENERATOR

VBA TUTORIAL

VBA CODE EXAMPLES FOR EXCEL

EXCEL BOOT CAMP

FORMULAS TUTORIAL

EXCEL FORMULAS EXAMPLES LIST

FUNCTIONS LIST

SHORTCUT COACH

SHORTCUT TUTORIAL



The advertisement features a central white box with the text "Ultimate VBA Add-in" in large, bold, black font, and "Learn More" in a smaller, grey font below it. To the left of this box is a screenshot of an Excel VBA editor window showing a code snippet: "1. Declare variable 'myVar' as a string variable type". Above the central box is an orange banner with a "FREE" label in a red box. Below the central box is another orange banner with the text "VBA TUTORIAL" in large white letters, followed by the "Automate Excel.com" logo and a "GET STARTED" button. To the right of the central box is a blue TrustSpot badge showing five yellow stars and the text "1008 Reviews".

Ultimate VBA Add-in
Learn More

VBA TUTORIAL
Automate Excel.com
GET STARTED

1008 Reviews
TrustSpot

Register for **FREE** and get:

- VBA EXAMPLES ADD-IN
- VBA CHEATSHEETS PDFS
- VBA TUTORIAL PDFS

[Register for free](#)



Ultimate VBA Add-in

Learn More

