Numeric Formats Cheat Sheet

by Matthew Perryman (PezMat) via cheatography.com/26431/cs/7969/

Predefined Numeric Formats

G or **g**	General Number Displays number with no thousand separator. For example, `String s.F orm at( &H3FA, " G")` or `String.Fo rma t("{ 0:G }", &H3FA)` returns **1018**	
C or **c**	Currency Displays number with thousand separator, if appropriate; displays two digits to the right of the decimal separator. Output is based on system locale settings. For example, `String s.F orm at( 123 4567, " C")` or `String.Fo rma t("{ 0:C }", 1234567)` returns **£1,234,567.00**	★
F or **f**	Fixed Displays at least one digit to the left and two digits to the right of the decimal separator. For example, `String s.F orm at( 123 4567, " F")` or `String.Fo rma t("{ 0:F }", 1234567)` returns **1234567.00**	★
N or **n**	Standard Displays number with thousand separator, at least one digit to the left and two digits to the right of the decimal separator. For example, `String s.F orm at( 123 4567, " N")` or `String.Fo rma t("{ 0:N }", 1234567)` returns **1,234,-567.00**	★
Percent	Displays number multiplied by 100 with a percent sign (%) appended immediately to the right; always displays two digits to the right of the decimal separator. For example, `String s.F orm at( 0.8 0345, " Per cen t")` returns **80.35%**	
P or **p**	Displays number with thousandths separator multiplied by 100 with a percent sign (%) appended to the right and separated by a single space; always displays two digits to the right of the decimal separator. For example, `String s.F orm at( 0.8 0345, " P")` or `String.Fo rma t("{ 0:P }", 0.80345)` returns **80.35 %**	★
Scientific	Uses standard scientific notation, providing two significant digits. For example, `String s.F orm at( 123 4567, " Sci ent ifi c")` returns **1.23E+06**	
E or **e**	Uses standard scientific notation, providing six significant digits. For example, `String s.F orm at( 123 4567, " E")` or `String.Fo rma t("{ 0:E }", 1234567)` returns **1.234567E+006** **Note:** The case of the **e** in the Format denotes the case of the **e** in the output (i.e. using **e** returns **1.234567e+006**)	★

By **Matthew Perryman** (PezMat)
cheatography.com/pezmat/

Published 21st April, 2016.
Last updated 12th May, 2016.

Predefined Numeric Formats (cont)

D or **d**	Displays number as a string that contains the value of the number in Decimal (base 10) format. This option is supported for integral types (**Byte**, **Short**, **Integer**, **Long**) only. For example, `String s.F orm at( &H7F, " D")` or `String.Fo rma t("{ 0:D }", &H7F)` returns **127**
X or **x**	Displays number as a string that contains the value of the number in Hexadecimal (base 16) format. This option is supported for integral types (**Byte**, **Short**, **Integer**, **Long**) only. For example, `String s.F orm at(127, " X")` or `String.Fo rma t("{ 0:X }", 127)` returns **7F** **Note:** The case of the **x** in the Format denotes the case of the letters in the output (i.e. using **x** returns **7f**)
Yes/No	Displays No if number is 0; otherwise, displays Yes. For example, `String s.F orm at(0, " Yes /No ")` returns **No**
True/ False	Displays False if number is 0; otherwise, displays True. For example, `String s.F orm at(1, " Tru e/F als e")` returns **True**
On/Off	Displays Off if number is 0; otherwise, displays On. For example, `String s.F orm at(1, " On/ Off ")` returns **On**

★ Digits after the formatting character determine the number of decimal places to output

For example, `String s.F orm at( 123 4567, " C0")` or `String.Fo rma t("{ 0:C 0}", 1234567)` returns `£1,234,567`

The examples were created on a machine with 'English (United Kingdom)' Region settings

Smart Device Developer Notes

The **Yes/No**, **True/False**, and **On/Off** formats are not supported.

User-Defined Numeric Formats

0	Digit placeholder Displays a digit or a zero. If the expression has a digit in the position where the zero appears in the format string, display it; otherwise, displays a zero in that position
#	Digit placeholder Displays a digit or nothing. If the expression has a digit in the position where the **#** character appears in the format string, displays it; otherwise, displays nothing in that position
.	Decimal placeholder The decimal placeholder determines how many digits are displayed to the left and right of the decimal separator. If the format expression contains only **#** characters to the left of this symbol; numbers smaller than 1 begin with a decimal separator. To display a leading zero displayed with fractional numbers, use zero as the first digit placeholder to the left of the decimal separator
%	Percent placeholder Multiplies the expression by 100. The percent character (%) is inserted in the position where it appears in the format string

User-Defined Numeric Formats (cont)

,	Thousand separator The thousand separator separates thousands from hundreds within a number that has four or more places to the left of the decimal separator. Standard use of the thousand separator is specified if the format contains a thousand separator surrounded by digit placeholders (**0** or **#**) For example, consider the three following format strings: ▸ "#,0.", which uses the thousands separator to format the number 100 million as the string "100,000,000" ▸ "#0,.", which uses scaling by a factor of one thousand to format the number 100 million as the string "100000" ▸ "#,0,.", which uses the thousands separator and scaling by one thousand to format the number 100 million as the string "100,000"
:	Time separator In some locales, other characters may be used to represent the time separator. The time separator separates hours, minutes, and seconds when time values are formatted. The actual character used as the time separator in formatted output is determined by your system settings
/	Date separator In some locales, other characters may be used to represent the date separator. The date separator separates the day, month, and year when date values are formatted. The actual character used as the date separator in formatted output is determined by your system settings
E-, **E+**, **e-** or **e+**	Scientific format If the format expression contains at least one digit placeholder (**0** or **#**) to the left of **E-**, **E+**, **e-** or **e+**, the number is displayed in scientific format and `E` or `e` is inserted between the number and its exponent. The number of digit placeholders to the left determines the number of digits in the exponent. Use **E-** or **e-** to place a minus sign next to negative exponents. Use **E+** or **e+** to place a minus sign next to negative exponents and a plus sign next to positive exponents. You must also include digit placeholders to the right of this symbol to get correct formatting
-, **+**, **$**, , **(**, or **)**	Literal characters These characters are displayed exactly as typed in the format string. To display a character other than one of those listed, precede it with a backslash (****) or enclose it in double quotation marks (" ")

User-Defined Numeric Formats (cont)

\	Displays the next character in the format string. To display a character that has special meaning as a literal character, precede it with a backslash (\ ). The backslash itself isn't displayed. Using a backslash is the same as enclosing the next character in double quotation marks. To display a backslash, use two backslashes (\\ ) Examples of characters that can't be displayed as literal characters are the date-formatting and time-formatting characters (**a**, **c**, **d**, **h**, **m**, **n**, **p**, **q**, **s**, **t**, **w**, **y**, **/** and **:**), the numeric-formatting characters (**#**, **0**, **%**, **E**, **e**, comma and period), and the string-formatting characters (**@**, **&**, **<**, **>** and **!**)
("*ABC*")	Displays the string inside the double quotation marks (" "). To include a string in the style argument from within code, you must use Chr(34) to enclose the text (34 is the character code for a quotation mark ("))

Source

Strings.Format Method

By **Matthew Perryman** (PezMat)
cheatography.com/pezmat/

Published 21st April, 2016.
Last updated 12th May, 2016.