..

Display Format Reference

Date published: 2020-10-30
Date modified: 2022-09-21

https://docs.cloudera.com/

Legal Notice

Contents

Display format reference

Cloudera Data Visualization implements standard number formatting through Python's format specification mini-language. The general format of the specifier is:

```
[[fill]align][sign][symbol][0][width][,][.precision][type]
```

For further details, see the related information.

Related Information

Standard format specifier
Format style table
Quantity prefixes
Additional format examples
Date/time formats

Standard format specifier

Format of the specifier follows:

```
[[fill]align][sign][symbol][0][width][,][.precision][type]
```

where:

- fill is a character other than { or }.

  The fill works with one of the alignment options. If the alignment character is invalid, both fill and alignment are ignored.
- align is one of the following:
  - < forces the field to be left-aligned within the available space; this is the default setting.
  - > forces the field to be right-aligned within available space.
  - = forces padding after the sign, if any, but before the digits; used for printing numeric fields in the form +000 000120.
  - ^ forces the field to be centered within the available space.
- sign is one of the following:
  - + means that a sign must be used for both positive and negative numbers.
  - - means that a sign must be used for negative numbers only; this is the default setting.
  - " " (space) means that a leading space must be used on positive numbers, and a minus sign on negative numbers.
- symbol is one of the following:
  - $, to represent currency
  - #, which is valid only for integers of specific output: 0b for binary, 0o for octal, and 0x for hexadecimal options.
- 0 enables zero-padding.
- , (comma) signals to use a comma for a thousands separator.
- width is a decimal integer that defines the minimum field width; if not specified, the content determines the width.

  If preceded by 0, the field will be zero-padded. This produces the same result as alignment of = and fill of 0.

- precision is a decimal that specifies how many digits to display after the decimal point of a floating point value that is formatted with type f, F, or %, or before and after the decimal point of a floating point value that is formatted with type g, r, and p.

  For non-number types, it indicates the maximum fields size, or how many characters to use in field content.

  The precision field does not support integer values.

- type is one of the following:
  - % is for percentage. Multiplies the number by 100, and displays the number in fixed format, f, followed by percent sign.
  - b is for binary format, and outputs numbers in base 2.
  - c is for character; it converts integers to corresponding Unicode characters.
  - d is for decimal integer; it outputs numbers in base 10.

    Use Number.toString() method.
  - e is for exponent notation for floating point and decimal numbers; it prints the number in scientific notation, using e to indicate exponent.

    For example, it would print 345 as 3.45e2.

    Use Number.toExponential() method.
  - E is for exponent notation for floating point and decimal numbers; it prints the number in scientific notation, using E to indicate exponent.

    For example, it would print 345 as 3.45E2.

    Use Number.toExponential() method.
  - f is for fixed floating point. Displays the number as a fixed-point number.

    Use Number.toFixed() method.
  - F is for fixed floating point, same as f; also converts nan to NAN, and inf to INF. Displays the number as a fixed-point number.

    Use Number.toFixed() method.
  - g is general format for floating point and decimal values.

    For a precision p>=1, it rounds the number to p significant digits, and formats the result depending on magnitude, either in fixed-point format, or in scientific notation.

    Both insignificant trailing zeros and decimal point are dropped if unnecessary.

    A precision of 0 is treated the same as precision of 1.

    Regardless of precision, positive infinity is rendered as inf, negative infinity as -inf, positive zero as 0, negative zero as -0, and NaN as nan.

    Use Number.toPrecision() method.
  - G is general format for floating point and decimal values, same as g. Also switches to E notation if the number is too large. Represents infinity and Nan as uppercase: INF, -INF, NAN.

    For a precision p>=1, it rounds the number to p significant digits, and formats the result depending on magnitude, either in fixed-point format, or in scientific notation.

    Both insignificant trailing zeros and decimal point are dropped if unnecessary.

    A precision of 0 is treated the same as precision of 1.

    Regardless of precision, positive infinity is rendered as inf, negative infinity as -inf, positive zero as 0, negative zero as -0, and NaN as nan.

    Use Number.toPrecision() method.
  - n is general format for floating point and decimal number, same as g. However, it uses current locale settings to insert the appropriate number separator characters.

    For example, one thousand one hundred and one-tenth would be rendered as 1,100.1 in United States and as 1.100,1 in France.
  - o is for octal format; it outputs numbers in base 8.
  - p is rounded percentage; like r type, but multiplied by 100 and with suffix %.
  - r is rounded to precision significant digits, padded with zeros whenever necessary, same as for f type. If prec ision is not specified, behaves like g type.

- s is for string format; it is the default type for string, and does not have to be specified.

  If used for floating point or decimal numbers, it is the metric prefix, SI for the International System of Units. It would render micro (0.000001 or 10-6) as 1.00µ, and it would render tera (1,000,000,000,000 or 1012) as 1.00T.
- x is for hexadecimal format; it outputs numbers in base 16, using lower-case letters for digits over 9.

  a for 10, b for 11, c for 12, d for 13, e for 14, and f for 15.
- X is for hexadecimal format; it outputs numbers in base 16, using upper-case letters for digits over 9.

  A for 10, B for 11, C for 12, D for 13, E for 14, and F for 15.
- None is the same as s for strings, and d for integers. It is similar to g for floating point and decimal values, with at least one digit past the decimal point, and a default precision of 12.

Format style table

The following table shows the format style options, and demonstrates how these options would change the appearance of a sample input number, 1235.00.

Table 1: Format Style Usage and Examples

Format Style	Values	Description	Examples	
			Mask	Output
Fill	All characters	Accepts all characters except curly brackets: { and }		
Align	<	Forces left-alignment within the specified space.	*<6	1235**
	>	Forces right-alignment within the specified space (default).	*>6	**1235
	^	Forces central alignment within the specified space.	*^6	*1235*
Sign	Plus +	Uses a sign for both positive and negative numbers.	+	+1235
	Minus -	Uses a sign for only negative numbers (default).	-	-1235
	Space	Uses a leading space on positive numbers, and a minus sign on negative numbers.	' '	1235
Symbol	$	Prefixes the $ (Dollar) currency symbol.	$	$1235
	£	Prefixes the £ (Pounds Sterling) currency symbol.	£	£1235
	¥	Prefixes the ¥ (Japanese Yen) currency symbol.	¥	¥1235
	#	Prefixes the # (Indian Rupee) currency symbol.	#	#1235
	€	Prefixes the € (Euro) currency symbol.	€	€1235
0	0	Enables zero-padding.	06	001235
Width	width	Defines the minimum field width. If not specified, the width is determined by the content.	6	1235
,	,	Enables the use of a comma to separate every third digit.	,	1,235
Precision	.precision	Indicates how many digits must be displayed after the decimal point, for a value formatted with types e, f and %, or before and after the decimal point for values formatted with types g, r, and p.	See f, g, r, p, and % types for examples	
Types	b	Binary: Outputs the number in base 2.	b	10011010011
	c	Character: Converts the integer to the corresponding Unicode character.	c	ä
	d	Integer: Outputs the number in base 10. Ignores non-integer values.	d	1235

Format Style	Values	Description	Examples	
			Mask	Output
	e	Exponent notation: Displays the number in scientific notation, using the letter e to indicate the exponent.	e	1.235e+3
			.1e	1.2e+3
	f	Fixed point: Displays the number as a fixed-point number.	f	1235
			.1f	1235.0
	g	General: For a given precision p >= 1, this rounds the number to p significant digits and then formats the result in either fixed-point format or in scientific notation, depending on its magnitude.	g	1235
			.2g	1.2e+3
			.5g	1235.0
	n	Formatted Number: This is the same as d, except that it uses the current locale setting to insert the appropriate number separator characters.	n	1,235
			.2n	1.2e+3
			.5n	1,235.0
	o	Octal: Outputs the number in base 8.	o	2323
	r	Rounded: Similar to general format, but does not use scientific notation.	r	1235
			.2r	1200
			.5r	1235.0
	s	String: Rounded and using scientific notation, but with a unit suffixed. See, Name and Symbol in Quantity prefixes. • Renders micro (0.000001 or 10-6) as 1.00µ. • Renders tera (1,000,000,000,000 or 1012) as 1.00T.	s	1.235k
			.2s	1.2k
			.5s	1.2350k
	S	String: Rounded and using regular currency unit suffixes. See, Quantity prefixes. • Renders 1,000,000,000 as 1B, for billions. • Renders 1,000,000,000,000 as 1T, for trillions.	$S	$1235
			$S,	$1,235
	x	Hex: Outputs the number in base 16, using the lower-case letters for the digits greater than 9. a for 10, b for 11, c for 12, d for 13, e for 14, and f for 15.	x	4d3
	X	Hex: Outputs the number in base 16, using the lower- case letters for the digits greater than 9. A for 10, B for 11, C for 12, D for 13, E for 14, and F for 15.	X	4D3
	%	Percentage: Multiplies the number by 100 and displays it in fixed f format, followed by a percent sign.	%	123500%
			.1%	123500.0%

Related Information

Quantity prefixes

Quantity prefixes

Table 2: Quantity Prefixes, Relative Values, Symbols, and Scientific Notation

Word	Decimal	10n	Name (Scientific)	Symbol (Sientific)	Symbol (Currency)
septillion	1 000 000 000 000 000 000 000 000	1024	yotta	Y	Y

Word	Decimal	10n	Name (Scientific)	Symbol (Sientific)	Symbol (Currency)
sextillion	1 000 000 000 000 000 000 000	1021	zetta	Z	Z
quintillion	1 000 000 000 000 000 000	1018	exa	E	E
quadrillion	1 000 000 000 000 000	1015	peta	P	P
trillion	1 000 000 000 000	1012	tera	T	T
billion	1 000 000 000	109	giga	G	B
million	1 000 000	106	mega	M	M
thousand	1 000	103	kilo	k	k
hundred	100	102	hecto	h	-
ten	10	101	deca	da	-
one	1	100			-
tenth	0.1	10-1	deci	d	-
hundredth	0.01	10-2	centi	c	-
thousandth	0.001	10-3	milli	m	-
millionth	0.000 001	10-6	micro	µ	-
billionth	0.000 000 000 001	10-9	nano	n	-
trillionth	0.000 000 000 000 001	10-12	pico	p	-
quadrillionth	0.000 000 000 000 000 001	10-15	femto	f	-
quintillionth	0.000 000 000 000 000 000 001	10-18	atto	a	-
sextillionth	0.000 000 000 000 000 000 000 001	10-21	zepto	z	-
septillionth	0.000 000 000 000 000 000 000 000 001	10-24	yocto	y	-

Additional format examples

Table 3: Number Formatting Examples

Input	Action	Syntax	Output
1234	Separate thousands by commas.	,	1,234
1234	Show $ value with commas.	$,	$1,234
0.123	Show % value with 2 decimal points of precision.	.2%	12.30%
12345	Display SI syntax.	.2s	12k
123	Show precision to 2 decimal places.	.2f	123.00
123	Include zero-padding for fixed width of 7.	07.2f	0123.00
123	Set sign for positive or negative values with precision to 2 decimal points.	+.2f	+123.00

Date/time formats

Date and time may be represented by a wide variety of tokens. We provide here a quick reference, adapted from Moment.js interface documentation.

Table 4: Tokens for Date and Time Representation, with Examples

Token	Input	Example	Description
Y	YYYY	2014	4- or 2-digit year
	YY	14	2-digit year
	Y	-25	Year with any number of digits and a sign. Used for years that are not in 4-digit form, and for years Before the Common Era.
Q	Q	1..4	Quarter of year. Sets the month to the first month in quarter.
M	M MM	1..12	Month number
	MMM MMMM	Jan..December	Month name in locale set by moment.locale()
D and d	D DD	1..31	Day of month
	Do	1st..31st	Day of month with ordinal
	DDD DDDD	1..365	Day of year
	ddd dddd	Mon...Sunday	Day name in locale set by moment.locale()
X and x	X	1410715640.579	Unix timestamp
	x	1410715640579	Unix ms timestamp
G and g	GGGG	2014	ISO 4-digit week year
	GG	14	ISO 2-digit week year
	gggg	2014	Locale 4-digit week year
	gg	14	Locale 2-digit week year
W and w	W WW	1..53	ISO week of year
	w ww	1..53	Locale week of year
E and e	E	1..7	ISO day of week
	e	0..6	Locale day of week
H and h	H HH	0..23	24-hour time
	h hh	1..12	12-hour time used with a and A
A and a	a A	am pm	Post- or ante-meridian. Note that one character a or p are also valid.
m	m mm	0..59	Minutes
s and S	s ss	0..59	Seconds
	S SS SSS	0.999	Fractional seconds
Z	Z ZZ	+12:00	Offset from UTC, as +-HH:mm, +-HHmm, or Z.