

Function index

Generated on October 14, 2026

Every function and syntax entry of Amtiri Script, from A to Z. Each one links to its entry, with its signature, its parameters and tested examples.

Token	Name	Description
<code>!=</code>	Not equal (C style)	True when the two values differ.
<code>&&</code>	And (short-circuit)	True only when both sides are true.
<code>*</code>	Multiply	Multiplies numbers.
<code>+</code>	Add	Adds numbers.
<code>-</code>	Negate	Flips the sign of a number and keeps its kind: a whole number stays whole and a decimal stays decimal.
<code>-</code>	Subtract	Subtracts the right value from the left one.
<code>.</code>	Field	Reads one named field out of a record - a control function's state, a database row, an inline object.
<code>/</code>	Divide	Divides the left value by the right one.
<code><</code>	Less than	True when the left value sorts before the right one.
<code><=</code>	Less than or equal	True when the left value sorts before the right one, or matches it.
<code><></code>	Not equal	True when the two values differ.
<code>==</code>	Equals	True when both values match.
<code>></code>	Greater than	True when the left value sorts after the right one.
<code>>=</code>	Greater than or equal	True when the left value sorts after the right one, or matches it.
<code>[]</code>	Item at	Reads one item out of a list by its position, counting from zero.
<code>^</code>	Power	Raises the left value to the power of the right one.
<code> </code>	Or (short-circuit)	True when either side is true.
<code>addDays</code>	Add days	Shifts a date by a number of days.
<code>addHours</code>	Add hours	Shifts a date and time by a number of hours.
<code>addMinutes</code>	Add minutes	Shifts a date and time by a number of minutes.
<code>addMonths</code>	Add months	Shifts a date by a whole number of months.

<code>addWeeks</code>	Add weeks	Shifts a date by a number of weeks.
<code>addYears</code>	Add years	Shifts a date by a whole number of years.
<code>and</code>	And	True only when every condition given is true.
<code>array</code>	List	An ordered list written inline, [a, b, c].
<code>arrayAdd</code>	Append	Gives a new list with values appended to the end, leaving the original list untouched.
<code>arrayAverage</code>	Average	The arithmetic mean of a list's items.
<code>arrayConcat</code>	Concatenate	Joins two lists into a new one, leaving both originals untouched.
<code>arrayFilter</code>	Filter	Keeps only the items a condition accepts.
<code>arrayMax</code>	List maximum	The largest item of a list, or null when the list is empty.
<code>arrayMin</code>	List minimum	The smallest item of a list, or null when the list is empty.
<code>arraySum</code>	Sum	Adds up every item of a list.
<code>bit</code>	Bit test	Tests one bit of a whole number - how a packed status word from a PLC or Modbus device is split into individual alarms.
<code>case</code>	Case	Compares one value against a series of candidates and yields the matching result - a lookup table.
<code>ceil</code>	Ceiling	Rounds up, or up to the nearest multiple of module when it is given.
<code>clamp</code>	Clamp	Holds a value inside a range: anything below min reads as min, anything above max reads as max.
<code>closePanel</code>	Close panel	Takes away the panel occupying this id on the current screen, wherever it was opened from.
<code>cos</code>	Cosine	The cosine of an angle given in radians.
<code>count</code>	Count	How many items a list holds.
<code>ctrlKalman</code>	Kalman filter	Deprecated: use the Kalman module.
<code>ctrlPID</code>	PID controller	Deprecated: use the PID module.
<code>ctrlPWM</code>	PWM controller	Deprecated: use the PWM module.
<code>darker</code>	Darker	Darkens a colour by a factor, towards black.
<code>date</code>	Date	Builds a date and time from its parts.
<code>date_of</code>	Date part	Drops the time of day, giving the same date at midnight.
<code>dayOfMonth</code>	Day of month	The day part of a date, starting at 1.
<code>dayOfWeek</code>	Day of week	Which day of the week a date falls on, Monday being 1.
<code>delayedSet</code>	Delayed set	Writes a value into a variable once a delay has passed.
<code>double</code>	Decimal	A decimal number.
<code>dowhile</code>	Do while	Repeats an expression for as long as a condition holds, testing after each pass - so it always runs at least once.
<code>evl</code>	Error default	Substitutes a default when evaluating a value raises an error, instead of letting the whole script fail.

<code>exec</code>	Sequence	Evaluates several expressions in order and yields the last one's result.
<code>exp</code>	Exponential	e raised to the power of value - the inverse of ln.
<code>fact</code>	Factorial	The factorial of a whole number, value!.
<code>false</code>	False	The boolean a comparison yields when it does not hold.
<code>floor</code>	Floor	Rounds down, or down to the nearest multiple of module when it is given.
<code>for</code>	For	Repeats an expression over a counter: the counter is set once, tested before each pass and advanced after it.
<code>forEach</code>	For each	Converts every item of a list.
<code>format</code>	Format	Builds a text from a pattern and its arguments, using Java format specifiers - %s for text, %d for a whole number, %.2f for two decimals.
<code>functions</code>	List functions	Every function name registered on the environment, sorted - an interpreter introspection aid.
<code>gadiant</code>	Gradient (old spelling)	The old, misspelt name of gradient, still accepted so stored formulas keep working.
<code>gradient</code>	Gradient	Picks the colour a named gradient assigns to a value - the usual way to drive an indicator from a reading.
<code>history</code>	History	Every sample a logger recorded for one variable over a time window, as a time series ready for the tsXxx functions.
<code>hsl</code>	HSL colour	Builds a colour from hue, saturation and lightness - easier than RGB when a colour has to vary smoothly with a reading.
<code>if</code>	If	Chooses between two values on a condition.
<code>inn</code>	Not-a-number default	Substitutes a default when a value is not a number: NaN, which is what a square root of a negative reading produces, or anything that is not numeric at all, such as text or null.
<code>integer</code>	Number	A whole number written straight into the script.
<code>ivl</code>	Infinite default	Substitutes a default when a value came out infinite - the guard to pair with a division whose divisor can reach zero.
<code>length</code>	Length	How many items a list or record holds, or how many characters a text has.
<code>lighter</code>	Lighter	Lightens a colour by a factor, towards white.
<code>ln</code>	Natural logarithm	The logarithm of a value in base e.
<code>log</code>	Logarithm (base 10)	The logarithm of a value in base 10.
<code>loggerValueAt</code>	Value at	What one variable's value was at a given instant, according to a logger: the last value it logged at or before that instant, of the type it was logged with.
<code>loggerValuesAt</code>	Values at	What several variables' values were at a given instant, according to a logger - loggerValueAt for each of them, one read of the logger per variable, gathered into one record.
<code>max</code>	Maximum	The largest of the values given.
<code>mbDouble</code>	Double	Decodes a 64-bit IEEE 754 floating-point value from four consecutive registers, the first holding the highest word (byte order ABCDEFGH).

<code>mbFloat</code>	Float	Decodes a 32-bit IEEE 754 floating-point value from two consecutive registers, the first holding the high word (byte order ABCD, as the Modbus master's own float register type): <code>mbFloat([16256, 0], 0)</code> is 1.0.
<code>mbFloatHL</code>	Float (high /low)	Decodes a 32-bit IEEE 754 floating-point value from two consecutive registers, the first holding the high word, with the two bytes of each register swapped (byte order BADC): <code>mbFloatHL([32831, 0], 0)</code> is 1.0.
<code>mbInt</code>	Int	Decodes a signed 32-bit whole number from two consecutive registers, the first holding the low word (word order CDAB): <code>mbInt([0, 1], 0)</code> is 65536.
<code>mbIntHL</code>	Int (high /low)	Decodes a signed 32-bit whole number from two consecutive registers, the first holding the high word, with the two bytes of each register swapped (byte order BADC): <code>mbIntHL([256, 0], 0)</code> is 65536.
<code>min</code>	Minimum	The smallest of the values given.
<code>mod</code>	Modulus	The remainder of dividing value by base - a wrapped counter, or an every-nth test.
<code>month</code>	Month	The month part of a date, 1 to 12.
<code>not</code>	Not	Inverts a condition.
<code>now</code>	Now	The current date and time in UTC.
<code>null</code>	Null	The absence of a value.
<code>nvl</code>	Null default	Substitutes a default when a value is null.
<code>object</code>	Object	A record of named fields written inline, {key: value}.
<code>openDashboard</code>	Open dashboard	Replaces what the current view is showing with another dashboard - how an Action button navigates the operator somewhere.
<code>openPanel</code>	Open panel	Shows a panel over the current view, centred at an offset from the component that ran this script.
<code>or</code>	Or	True when at least one condition given is true.
<code>psplit</code>	Period split	Cuts a time window into equal consecutive segments - the segment list <code>tsSplit</code> expects, and the basis of any bucketed aggregate.
<code>random</code>	Random	A fresh pseudo-random number between 0 inclusive and 1 exclusive on every evaluation.
<code>rgb</code>	RGB colour	Builds a colour from red, green and blue, each a fraction from 0 to 1: <code>rgb(1, 0, 0)</code> is pure red, and values of 1 and above are capped.
<code>round</code>	Round	Rounds to the nearest whole number, or to the nearest multiple of module when it is given - <code>round(x, 0.5)</code> snaps to halves.
<code>scale</code>	Scale	Maps a value from one range onto another - the usual way to turn a raw sensor reading into engineering units.
<code>scalein</code>	Scale (confined)	Maps a value from one range onto another exactly as <code>scale</code> does, then confines the result to the destination range, whichever way round it runs - an out-of-range reading cannot drive an output past its limits.
<code>serie</code>	Number range	Builds a list of consecutive whole numbers.
<code>sin</code>	Sine	The sine of an angle given in radians.
<code>sqrt</code>	Square root	The square root of a value.
<code>string</code>	Text	A fixed piece of text, in double quotes.
<code>substr</code>	Substring	The part of a text between two positions, counting from 0 and excluding the character at to.

<code>tan</code>	Tangent	The tangent of an angle given in radians.
<code>to_double</code>	To decimal	Converts a value to a decimal number.
<code>to_int</code>	To whole number	Converts a value to a whole number, truncating any decimals.
<code>to_lower</code>	To lower case	Converts a text to lower case.
<code>to_string</code>	To text	Converts a value to its text form.
<code>to_upper</code>	To upper case	Converts a text to upper case.
<code>toDate</code>	From day number	Turns a day number counted from 1899-12-31 back into a date and time - the inverse of <code>toDays</code> .
<code>toDays</code>	To day number	A date as a decimal count of days since 1899-12-31, the time of day as its fraction, so a duration becomes a subtraction.
<code>toLocale</code>	To local time	Reads a UTC date and time as the configured time zone.
<code>toUTC</code>	To UTC	Reads a date and time expressed in the configured time zone back as UTC - the inverse of <code>toLocale</code> , and what to apply to any date a person entered.
<code>toUTM</code>	To UTM	Converts a latitude and longitude to Universal Transverse Mercator coordinates - metres, so distances between two points can be worked out by subtraction.
<code>trim</code>	Trim	Removes leading and trailing whitespace from a text.
<code>true</code>	True	The boolean a comparison yields when it holds.
<code>tsAverage</code>	Time average	The time-weighted mean of a series - a sample held for an hour counts for an hour, not as one reading, which is what makes this the correct average for logged data.
<code>tsCenter</code>	Middle value	The value in force halfway through a series' time span.
<code>tsDistribution</code>	Distribution	How long a series spent at each distinct value - the run-time-per-state breakdown behind a machine-state or alarm report.
<code>tsFilter</code>	Filter series	Keeps only the samples a condition accepts, breaking the series where a run of rejected samples begins.
<code>tsFirst</code>	First value	The first sample of a series, as its [time, value] pair: <code>tsFirst(s)[1]</code> is the value, <code>tsFirst(s)[0]</code> its time.
<code>tsLast</code>	Last value	The last sample of a series, as its [time, value] pair: <code>tsLast(s)[1]</code> is the value, <code>tsLast(s)[0]</code> its time.
<code>tsMax</code>	Series maximum	The largest value in a series.
<code>tsMerge</code>	Merge	Interleaves several named series onto one timeline, so every instant carries the last known value of each - what a multi-variable chart or export needs.
<code>tsMin</code>	Series minimum	The smallest value in a series.
<code>tsSplit</code>	Split	Cuts a series into one sub-series per time segment - pair it with <code>psplit</code> to bucket a window, then aggregate each bucket.
<code>type_of</code>	Type of	The name of a value's underlying type, for diagnosing a script that is receiving something other than what it expects.

<code>variables</code>	List variables	Every variable name defined on the environment, sorted - an interpreter introspection aid.
<code>while</code>	While	Repeats an expression for as long as a condition holds, testing before each pass.
<code>with</code>	With	Evaluates an expression with a value bound to a name - a local alias, so an expensive sub-expression is written and computed once.
<code>xor</code>	Exclusive or	True when an odd number of the conditions given are true.
<code>year</code>	Year	The year part of a date.

This page describes Amtiri Script 5.5.0 as shipped with Data Orchester engine 6.22.0 and Site 1.27.0.