WorldQuant Operators

Operators

add

- **Category:** Arithmetic
- **Scope:** REGULAR
- **Definition:** add(x, y, filter = false), x + y
- **Description:** Add all inputs (at least 2 inputs required). If filter = true, filter all input NaN to 0 before adding

log

- **Category:** Arithmetic
- **Scope:** REGULAR
- **Definition:** log(x)
- **Description:** Natural logarithm. For example: Log(high/low) uses natural logarithm of high/low ratio as stock weights.

subtract

- **Category:** Arithmetic

- **Scope:** REGULAR
- **Definition:** subtract(x, y, filter=false), x - y
- **Description:** x-y. If filter = true, filter all input NaN to 0 before subtracting

signed_power

- **Category:** Arithmetic
- **Scope:** REGULAR
- **Definition:** signed_power(x, y)
- **Description:** x raised to the power of y such that final result preserves sign of x
- **Documentation:** https://api.worldquantbrain.com/operators/signed_power
- **Documentation Content:**

```
sign(x) * (abs(x) ^ y)
x raised to the power of y such that final result preserves sign of x. For power of 2, x ^ y
will be a parabola but signed_power(x, y) will be odd and one-to-one function (unique value of
x for certain value of signed_power(x, y)) unlike parabola.



Example:
If x = 3, y = 2 â‡’ abs(x) = 3 â‡’ abs(x) ^ y = 9 and sign(x) = +1 â‡’ sign(x) * (abs(x) ^ y) =
signed_power(x, y) = 9
If x = -9, y = 0.5 â‡’ abs(x) = 9 â‡’ abs(x) ^ y = 3 and sign(x) = -1 â‡’ sign(x) * (abs(x) ^
y) = signed_power(x, y)
```

sign

- **Category:** Arithmetic
- **Scope:** REGULAR
- **Definition:** sign(x)
- **Description:** if input = NaN; return NaN

reverse

- **Category:** Arithmetic
- **Scope:** REGULAR
- **Definition:** reverse(x)
- **Description:** Â - x

power

- **Category:** Arithmetic
- **Scope:** REGULAR
- **Definition:** power(x, y)
- **Description:** x ^ y
- **Documentation:** https://api.worldquantbrain.com/operators/power

multiply

- **Category:** Arithmetic

- **Scope:** REGULAR
- **Definition:** multiply(x ,y, ... , filter=false), x * y
- **Description:** Multiply all inputs. At least 2 inputs are required. Filter sets the NaN values to 1
- **Documentation:** https://api.worldquantbrain.com/operators/multiply
- **Documentation Content:**

```
Examples:
```

min

- **Category:** Arithmetic
- **Scope:** REGULAR
- **Definition:** min(x, y ..)
- **Description:** Minimum value of all inputs. At least 2 inputs are required
- **Documentation:** https://api.worldquantbrain.com/operators/min
- **Documentation Content:**

```
Example:
```

max

- **Category:** Arithmetic
- **Scope:** REGULAR
- **Definition:** max(x, y, ..)
- **Description:** Maximum value of all inputs. At least 2 inputs are required
- **Documentation:** https://api.worldquantbrain.com/operators/max
- **Documentation Content:**

```
Example:
```

inverse

- **Category:** Arithmetic
- **Scope:** REGULAR
- **Definition:** inverse(x)
- **Description:** 1 / x

densify

- **Category:** Arithmetic
- **Scope:** REGULAR
- **Definition:** densify(x)
- **Description:** Converts a grouping field of many buckets into lesser number of only available buckets so as to make working with grouping fields computationally efficient
- **Documentation:** https://api.worldquantbrain.com/operators/densify

- **Documentation Content:**

  ```
  This operator converts a grouping field with many buckets into a lesser number of only the available buckets, making working with grouping fields computationally efficient. The example below will clarify the implementation.

  **Example**:

  Say a grouping field is provided as an integer (e.g., industry: tech -> 0, airspace -> 1, ...) and for a certain date, we have instruments with grouping field values among {0, 1, 2, 99}. Instead of creating 100 buckets and keeping 96 of them empty, it is better to just create 4 buckets with values {0, 1, 2, 3}. So, if the number of unique values in x is n, densify maps those values between 0 and (n-1). The order of magnitude need not be preserved.
  ```

abs

- **Category:** Arithmetic
- **Scope:** REGULAR
- **Definition:** abs(x)
- **Description:** Absolute value of x

divide

- **Category:** Arithmetic
- **Scope:** REGULAR
- **Definition:** divide(x, y), x / y
- **Description:** x / y

equal

- **Category:** Logical
- **Scope:** REGULAR
- **Definition:** input1 == input2
- **Description:** Returns true if both inputs are same and returns false otherwise

less

- **Category:** Logical
- **Scope:** REGULAR
- **Definition:** input1 < input2
- **Description:** If input1 < input2 return true, else return false

and

- **Category:** Logical
- **Scope:** REGULAR
- **Definition:** and(input1, input2)
- **Description:** Logical AND operator, returns true if both operands are true and returns false otherwise

or

- **Category:** Logical

- **Scope:** REGULAR
- **Definition:** or(input1, input2)
- **Description:** Logical OR operator returns true if either or both inputs are true and returns false otherwise

not_equal

- **Category:** Logical
- **Scope:** REGULAR
- **Definition:** input1!= input2
- **Description:** Returns true if both inputs are NOT the same and returns false otherwise

not

- **Category:** Logical
- **Scope:** REGULAR
- **Definition:** not(x)
- **Description:** Returns the logical negation of x. If x is true (1), it returns false (0), and if input is false (0), it returns true (1).

greater

- **Category:** Logical
- **Scope:** REGULAR
- **Definition:** input1 > input2
- **Description:** Logic comparison operators to compares two inputs

greater_than_equal

- **Category:** Logical
- **Scope:** REGULAR
- **Definition:** input1 >= input2
- **Description:** Returns true if input1 >= input2, return false otherwise

less_than_equal

- **Category:** Logical
- **Scope:** REGULAR
- **Definition:** input1 <= input2
- **Description:** Returns true if input1 <= input2, return false otherwise

is_nan

- **Category:** Logical
- **Scope:** REGULAR
- **Definition:** is_nan(input)
- **Description:** If (input == NaN) return 1 else return 0

if_else

- **Category:** Logical

- **Scope:** REGULAR
- **Definition:** if_else(input1, input2, input 3)
- **Description:** If input1 is true then return input2 else return input3.
- **Documentation:** https://api.worldquantbrain.com/operators/if_else
- **Documentation Content:**

```
if_else(event_condition, Alpha_expression_1, Alpha_expression_2)


If the event condition provided is true, Alpha_expression_1 will be returned. If the event
condition provided is false, Alpha_expression_2 will be returned.

Example:

We are interested in testing our hypothesis that if the stock price of a company has increased
over the last 2 days, it may decrease in the future. Also, if the number of stocks bought and
sold today is higher than the monthly average, then the reversion effect may be observed more
profoundly.

We will implement this hypothesis by taking positions according to the difference of close
price today and 3 days ago with alpha_2 using the ts_delta operator. When current volume is
higher than average daily volume, we will take a larger position by multiplying by 2 to get
alpha_1.
```

ts_mean

- **Category:** Time Series
- **Scope:** REGULAR
- **Definition:** ts_mean(x, d)
- **Description:** Returns average value of x for the past d days.

ts_sum

- **Category:** Time Series
- **Scope:** REGULAR
- **Definition:** ts_sum(x, d)
- **Description:** Sum values of x for the past d days.

ts_std_dev

- **Category:** Time Series
- **Scope:** REGULAR
- **Definition:** ts_std_dev(x, d)
- **Description:** Returns standard deviation of x for the past d days

ts_scale

- **Category:** Time Series
- **Scope:** REGULAR
- **Definition:** ts_scale(x, d, constant = 0)
- **Description:** Returns (x - ts_min(x, d)) / (ts_max(x, d) - ts_min(x, d)) + constant. This operator is similar to scale down operator but acts in time series space

- **Documentation:** https://api.worldquantbrain.com/operators/ts_scale
- **Documentation Content:**

```
This operator returns (x â€" ts_min(x, d)) / (ts_max(x, d) â€" ts_min(x, d)) + constant
This operator is similar to scale down operator but acts in time series space

Example:
If d = 6 and values for last 6 days are [6,2,8,5,9,4] with first element being todayâ€™s value,
ts_min(x,d) = 2, ts_max(x,d) = 9
ts_scale(x,d,constant = 1) = 1 + (6-2)/(9-2) = 1.57
```

ts_rank

- **Category:** Time Series
- **Scope:** REGULAR
- **Definition:** ts_rank(x, d, constant = 0)
- **Description:** Rank the values of x for each instrument over the past d days, then return the rank of the current value + constant. If not specified, by default, constant = 0.

ts_quantile

- **Category:** Time Series
- **Scope:** REGULAR
- **Definition:** ts_quantile(x,d, driver="gaussian")
- **Description:** It calculates ts_rank and apply to its value an inverse cumulative density function from driver distribution. Possible values of driver (optional) are "gaussian", "uniform", "cauchy" distribution where "gaussian" is the default.

ts_zscore

- **Category:** Time Series
- **Scope:** REGULAR
- **Definition:** ts_zscore(x, d)
- **Description:** Z-score is a numerical measurement that describes a value's relationship to the mean of a group of values. Z-score is measured in terms of standard deviations from the mean: (x - tsmean(x,d)) / tsstddev(x,d). This operator may help reduce outliers and drawdown.

ts_arg_min

- **Category:** Time Series
- **Scope:** REGULAR
- **Definition:** ts_arg_min(x, d)
- **Description:** Returns the relative index of the min value in the time series for the past d days; If the current day has the min value for the past d days, it returns 0; If previous day has the min value for the past d days, it returns 1.
- **Documentation:** https://api.worldquantbrain.com/operators/ts_arg_min
- **Documentation Content:**

```
ts_arg_min(x, d)

It returns the relative index of the min value in the time series for the past d days. If the
current day has the min value for the past d days, it returns 0. If previous day has the min
```

```
value for the past d days, it returns 1.

Example:
If d = 6 and values for past 6 days are [6,2,8,5,9,4] with first element being todayâ€™s value
then min value is 2 and it is present 4 days before today. Hence, ts_arg_min(x, d) = 1
```

kth_element

- **Category:** Time Series
- **Scope:** REGULAR
- **Definition:** kth_element(x, d, k)
- **Description:** Returns K-th value of input by looking through lookback days. This operator can be used to backfill missing data if k=1
- **Documentation:** https://api.worldquantbrain.com/operators/kth_element
- **Documentation Content:**

```
Returns k-th value of input by looking through lookback days while ignoring space separated
scalars in ignore list. This operator is also known as backfill operator as it can be used to
backfill missing data.

ignore parameter is used to provide list of separated scalars to ignore from counting

Example of backfill:
```

ts_corr

- **Category:** Time Series
- **Scope:** REGULAR
- **Definition:** ts_corr(x, y, d)
- **Description:** Returns correlation of x and y for the past d days
- **Documentation:** https://api.worldquantbrain.com/operators/ts_corr
- **Documentation Content:**

```
ts_corr(x, y, d)

Pearson correlation measures the linear relationship between two variables. It's most effective
when the variables are normally distributed and the relationship is linear.


$$Correlation(x,y) = \frac{\sum_{i=t-d+1}^t (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=t-
d+1}^t (x_i - \bar{x})^2 (y_i - \bar{y})^2}}$$

Example:
```

ts_count_nans

- **Category:** Time Series
- **Scope:** REGULAR
- **Definition:** ts_count_nans(x ,d)
- **Description:** Returns the number of NaN values in x for the past d days

ts_covariance

- **Category:** Time Series
- **Scope:** REGULAR
- **Definition:** ts_covariance(y, x, d)
- **Description:** Returns covariance of y and x for the past d days

ts_decay_linear

- **Category:** Time Series
- **Scope:** REGULAR
- **Definition:** ts_decay_linear(x, d, dense = false)
- **Description:** Returns the linear decay on x for the past d days. Dense parameter=false means operator works in sparse mode and we treat NaN as 0. In dense mode we do not.
- **Documentation:** https://api.worldquantbrain.com/operators/ts_decay_linear
- **Documentation Content:**

  ```
  ts_decay_linear(x, d, dense = false)
  ```

  `Returns the linear decay on x for the past d days. Dense parameter=false means operator works in sparse mode and we treat NaN as 0. In dense mode we do not. Data smoothing techniques like linear decay reduce noise in time-series data by applying a decay factor to older observations, which helps to stabilize the dataset.`

  `This operator improves turnover and drawdown.`

  `Example:`

  - `For a stock with the following prices over the last 5 days:`
    - `Day 0: 30 (outlier)`
    - `Day -1: 5`
    - `Day -2: 4`
    - `Day -3: 5`
    - `Day -4: 6`
  - `The calculation would be:`
    - `Numerator = (30â‹…5)+(5â‹…4)+(4â‹…3)+(5â‹…2)+(6â‹…1)=150+20+12+10+6=198`
    - `Denominator=5+4+3+2+1=15`
    - `Weighted Average=198/15=13.2`
  - `The weighted average value of 13.2 is used instead of the outlier value of 20 for assigning weight.`

ts_product

- **Category:** Time Series
- **Scope:** REGULAR
- **Definition:** ts_product(x, d)
- **Description:** Returns product of x for the past d days

ts_regression

- **Category:** Time Series
- **Scope:** REGULAR
- **Definition:** ts_regression(y, x, d, lag = 0, rettype = 0)

- **Description:** Returns various parameters related to regression function
- **Documentation:** https://api.worldquantbrain.com/operators/ts_regression
- **Documentation Content:**

```
ts_regression(y, x, d, lag = 0, rettype = 0)

Given a set of two variablesâ€™ values (X: the independent variable, Y: the dependent variable)
over a course of d days, an approximating linear function can be defined, such that sum of
squared errors on this set assumes minimal value:



Beta and Alpha in second line are OLS Linear Regression coefficients.

ts_regression operator returns various parameters related to said regression. This is governed
by â€œrettypeâ€ keyword argument, which has a default value of 0. Other â€œrettypeâ€ argument
values correspond to:



Here, "di" is current day index, â€œnâ€(may differ from d) is a number of valid (x, y) tuples
used for calculation. All summations are over day index, using only valid tuples.

â€œlagâ€ keyword argument may be optionally specified (default value is zero) to calculate
lagged regression parameters instead:



Example:

  ◦ ▪ ts_regression(est_netprofit, est_netdebt, 252, lag = 0, rettype = 2)
       ▪ Taking the data from the past 252 trading days (1 year), return the Î² coefficient from
         the equation when estimating the est_netprofit using the est_netdebt
```

ts_step

- **Category:** Time Series
- **Scope:** REGULAR
- **Definition:** ts_step(1), step(1)
- **Description:** Returns days' counter

ts_delay

- **Category:** Time Series
- **Scope:** REGULAR
- **Definition:** ts_delay(x, d)
- **Description:** Returns x value d days ago

ts_backfill

- **Category:** Time Series
- **Scope:** REGULAR
- **Definition:** ts_backfill(x,lookback = d, k=1, ignore="NAN")
- **Description:** Backfill is the process of replacing the NAN or 0 values by a meaningful value (i.e., a first non-NaN value)
- **Documentation:** https://api.worldquantbrain.com/operators/ts_backfill
- **Documentation Content:**

```
ts_backfill(x,lookback = d, k=1, ignore="NAN")

The ts_backfill operator replaces NaN values with the last available non-NaN value. If the
input value of the data field x is NaN, the ts_backfill operator will check available input
values of the same data field for the past d number of days, and output the most recent
available non-NaN input value. If the k parameter is set, then the ts_backfill operator will
output the kth most recent available non-NaN input value.

This operator improves weight coverage and may help to reduce drawdown risk.

Example: ts_backfill(x, 252)

  ◦ If the input value for data field x = non-NaN, then output = x
  ◦ If the input value for data field x = NaN, then output = most recent available non-NaN input
    value for x in the past 252 days
```

ts_av_diff

- **Category:** Time Series
- **Scope:** REGULAR
- **Definition:** ts_av_diff(x, d)
- **Description:** Returns x - tsmean(x, d), but deals with NaNs carefully. That is NaNs are ignored during mean computation
- **Documentation:** https://api.worldquantbrain.com/operators/ts_av_diff
- **Documentation Content:**

```
This operator returns x â€“ ts_mean(x, d), but it deals with NaNs carefully

Example:
If d = 6 and values for past 6 days are [6,2,8,5,9,NaN] then ts_mean(x,d) = 6 since NaN are
ignored from mean computation. Hence, ts_av_diff(x,d) = 6-6 = 0
```

hump

- **Category:** Time Series
- **Scope:** REGULAR
- **Definition:** hump(x, hump = 0.01)
- **Description:** Limits amount and magnitude of changes in input (thus reducing turnover)
- **Documentation:** https://api.worldquantbrain.com/operators/hump
- **Documentation Content:**

```
hump(x, hump = 0.01)

This operator limits the frequency and magnitude of changes in the Alpha (thus reducing
turnover). If today's values show only a minor change (not exceeding the Threshold) from
yesterday's value, the output of the hump operator stays the same as yesterday. If the change
is bigger than the limit, the output is yesterday's value plus the limit in the direction of
the change.

This operator may help reduce turnover and drawdown.

Flowchart of the Hump operator:
```

ts_arg_max

- **Category:** Time Series
- **Scope:** REGULAR
- **Definition:** ts_arg_max(x, d)
- **Description:** Returns the relative index of the max value in the time series for the past d days. If the current day has the max value for the past d days, it returns 0. If previous day has the max value for the past d days, it returns 1
- **Documentation:** https://api.worldquantbrain.com/operators/ts_arg_max
- **Documentation Content:**

```
It returns the relative index of the max value in the time series for the past d days. If the
current day has the max value for the past d days, it returns 0. If previous day has the max
value for the past d days, it returns 1.

Example:
If d = 6 and values for past 6 days are [6,2,8,5,9,4] with first element being todayâ€™s value
then max value is 9 and it is present 4 days before today. Hence, ts_arg_max(x, d) = 4
```

last_diff_value

- **Category:** Time Series
- **Scope:** REGULAR
- **Definition:** last_diff_value(x, d)
- **Description:** Returns last x value not equal to current x value from last d days

ts_delta

- **Category:** Time Series
- **Scope:** REGULAR
- **Definition:** ts_delta(x, d)
- **Description:** Returns x - ts_delay(x, d)

days_from_last_change

- **Category:** Time Series
- **Scope:** REGULAR
- **Definition:** days_from_last_change(x)
- **Description:** Amount of days since last change of x

zscore

- **Category:** Cross Sectional
- **Scope:** REGULAR
- **Definition:** zscore(x)
- **Description:** Z-score is a numerical measurement that describes a value's relationship to the mean of a group of values. Z-score is measured in terms of standard deviations from the mean
- **Documentation:** https://api.worldquantbrain.com/operators/zscore
- **Documentation Content:**

```
zscore(x)
```

```
Z-score is a statistical tool that indicates how many standard deviations a data point lies
from the average of a group of values. Essentially, it measures how unusual a data point is in
relation to the mean, making it a handy tool for understanding deviation and comparison.

The formula to calculate a Z-score is:


$$Z\textrm{-}score = \frac{x - mean(x)}{std(x)}$$

Where:

 ∘ x is an individual data point
 ∘ mean(x) is the average of the data set
 ∘ std(x) is the standard deviation of the data set

By this definition, the mean of the Z-scores in a distribution is always 0, and the standard
deviation is always 1.

A Z-score tells you how many standard deviations a particular data point is from the mean. If
the Z-score is positive, the data point is above the mean, and if it's negative, it's below the
mean.

Z-scores may be especially useful for normalizing and comparing different data fields for
different stocks or different data fields. They allow researchers to calculate the probability
of a score occurring within a standard normal distribution and compare two scores that are from
different samples (which may have different means and standard deviations).

This operator may help reduce outliers.
```

normalize

- **Category:** Cross Sectional
- **Scope:** REGULAR
- **Definition:** normalize(x, useStd = false, limit = 0.0)
- **Description:** Calculates the mean value of all valid alpha values for a certain date, then subtracts that mean from each element
- **Documentation:** https://api.worldquantbrain.com/operators/normalize
- **Documentation Content:**

```
normalize(x, useStd = false, limit = 0.0)


This operator calculates the mean value of all valid alpha values for a certain date, then
subtracts that mean from each element. If useStd= true, the operator calculates the standard
deviation of the resulting values and divides each normalized element by it. If limit is not
equal to 0.0, operator puts the limit of the resulting alpha values (between -limit to +
limit).
Example:
If for a certain date, instrument value of certain input x is [3,5,6,2], mean = 4 and standard
deviation = 1.82
normalize(x, useStd = false, limit = 0.0) = [3-4,5-4,6-4,2-4] = [-1,1,2,-2]
normalize(x, useStd = true, limit = 0.0) = [-1/1.82,1/1.82,2/1.82,-2/1.82] = [-0.55,0.55,1.1,-
1.1]
```

quantile

- **Category:** Cross Sectional
- **Scope:** REGULAR

- **Definition:** quantile(x, driver = gaussian, sigma = 1.0)
- **Description:** Rank the raw vector, shift the ranked Alpha vector, apply distribution (gaussian, cauchy, uniform). If driver is uniform, it simply subtract each Alpha value with the mean of all Alpha values in the Alpha vector
- **Documentation:** https://api.worldquantbrain.com/operators/quantile
- **Documentation Content:**

```
quantile(x, driver = gaussian, sigma = 1.0)


Rank the input raw Alpha vector
The ranked Alpha value would be within [0, 1]

  1. Shift the ranked Alpha vector
     For every Alpha value in the ranked Alpha vector, it is shifted as: Alpha_value = 1/N +
     Alpha_value * (1 - 2/N), here assume there are N instruments with value in the Alpha
     vector. The shifted Alpha value would be within [1/N, 1-1/N]
  2. Apply distribution for each Alpha value in the ranked Alpha vector using the specified
     driver. Driver can be one of "gaussian", "uniform", "cauchy".

Note : Sigma only affects the scale of the final value.

This operator may help reduce outliers.

Example:
```

rank

- **Category:** Cross Sectional
- **Scope:** REGULAR
- **Definition:** rank(x, rate=2)
- **Description:** Ranks the input among all the instruments and returns an equally distributed number between 0.0 and 1.0. For precise sort, use the rate as 0
- **Documentation:** https://api.worldquantbrain.com/operators/rank
- **Documentation Content:**

```
rank(x, rate=2):

The Rank operator ranks the value of the input data x for the given stock among all
instruments, and returns float numbers equally distributed between 0.0 and 1.0. When rate is
set to 0, the sorting is done precisely. The default value of rate is 2.

This operator may help reduce outliers and drawdown while improving the Sharpe.

Example:

Rank(close); Rank (close, rate=0) # Sorts precisely

X = (4,3,6,10,2) => Rank(x) = (0.5, 0.25, 0.75, 1, 0)
```

scale

- **Category:** Cross Sectional
- **Scope:** REGULAR
- **Definition:** scale(x, scale=1, longscale=1, shortscale=1)
- **Description:** Scales input to booksize. We can also scale the long positions and short positions to separate scales by mentioning additional parameters to the operator

- **Documentation:** https://api.worldquantbrain.com/operators/scale
- **Documentation Content:**

```
scale (x, scale=1, longscale=1, shortscale=1)

The operator scales the input to the book size. We can optionally tune the book size by
specifying the additional parameter 'scale=booksize_value'. We can also scale the long
positions and short positions to separate scales by specifying additional parameters:
longscale=long_booksize and shortscale=short_booksize. The default value of each leg of the
scale is 0, which means no scaling, unless specified otherwise. Scale the alpha so that the sum
of abs(x) over all instruments equals 1. To scale to a different book size, use Scale(x) *
booksize.

This operator may help reduce outliers.

Please check examples for the application of the same

Examples:

scale(returns, scale=4); scale (returns, scale= 1) + scale (close, scale=20); scale (returns,
longscale=4, shortscale=3)
```

winsorize

- **Category:** Cross Sectional
- **Scope:** REGULAR
- **Definition:** winsorize(x, std=4)
- **Description:** Winsorizes x to make sure that all values in x are between the lower and upper limits, which are specified as multiple of std. Details can be found on wiki

vec_sum

- **Category:** Vector
- **Scope:** REGULAR
- **Definition:** vec_sum(x)
- **Description:** Sum of vector field x

vec_avg

- **Category:** Vector
- **Scope:** REGULAR
- **Definition:** vec_avg(x)
- **Description:** Taking mean of the vector field x

bucket

- **Category:** Transformational
- **Scope:** REGULAR
- **Definition:** bucket(rank(x), range="0, 1, 0.1" or buckets = "2,5,6,7,10")
- **Description:** Convert float values into indexes for user-specified buckets. Bucket is useful for creating group values, which can be passed to GROUP as input
- **Documentation:** https://api.worldquantbrain.com/operators/bucket
- **Documentation Content:**

```
Bucket

Convert float values into indexes for user-specified buckets. Bucket is useful for creating
group values, which can be passed to group operators as input.

If buckets are specified as "num_1, num_2, â€¦, num_N", it is converted into brackets
consisting of [(num_1, num_2, idx_1), (num_2, num_3, idx_2), ..., (num_N-1, num_N, idx_N-1)]

Thus with buckets="2, 5, 6, 7, 10", the vector "-1, 3, 6, 8, 12" becomes "0, 1, 2, 4, 5"

If range if specified as "start, end, step", it is converted into brackets consisting of
[(start, start+step, idx_1), (start+step, start+2*step, idx_2), ..., (start+N*step, end,
idx_N)].

Thus with range="0.1, 1, 0.1", the vector "0.05, 0.5, 0.9" becomes "0, 4, 8"

Note that two hidden buckets corresponding to (-inf, start] and [end, +inf) are added by
default. Use the skipBegin, skipEnd parameters to remove these buckets. Use skipBoth to set
both skipEnd and skipBegin to true.

If you want to assign all NAN values into a separate group of their own, use NANGroup. The
index value will be one after the last bucket

Examples:

my_group = bucket(rank(volume), range="0.1,1,0.1");

group_neutralize(sales/assets, my_group)

my_group = bucket(rank(volume), buckets ="0.2,0.5,0.7", skipBoth=True, NANGroup=True);

group_neutralize(sales/assets, my_group)
```

trade_when

- **Category:** Transformational
- **Scope:** REGULAR
- **Definition:** trade_when(x, y, z)
- **Description:** Used in order to change Alpha values only under a specified condition and to hold Alpha values in other cases. It also allows to close Alpha positions (assign NaN values) under a specified condition
- **Documentation:** https://api.worldquantbrain.com/operators/trade_when
- **Documentation Content:**

```
This operator can be used to change Alpha values only under a specified condition and to retain
Alpha values in other cases. It also allows for closing Alpha positions (assigning NaN values)
under a specified condition.

Trade_When (x=triggerTradeExp, y=AlphaExp, z=triggerExitExp)

If triggerExitExp > 0, Alpha = NaN.

Else if triggerTradeExp > 0, Alpha = AlphaExp;

else, Alpha = previousAlpha

This operator may help reduce correlation and reduce turnover.

Examples:

Trade_When (volume >= ts_sum(volume,5)/5, rank(-returns), -1)

If (volume >= ts_sum(volume,5)/5), Alpha = rank(-returns);

else trade previous Alpha;

exit condition is always false.

Trade_When (volume >= ts_sum(volume,5)/5, rank(-returns), abs(returns) > 0.1)
```

```
If abs(returns) > 0.1, Alpha = nan;

else if volume >= ts_sum(volume,5)/5, Alpha = rank(-returns);

else trade previous Alpha.
```

group_zscore

- **Category:** Group
- **Scope:** REGULAR
- **Definition:** group_zscore(x, group)
- **Description:** Calculates group Z-score - numerical measurement that describes a value's relationship to the mean of a group of values. Z-score is measured in terms of standard deviations from the mean. zscore = (data - mean) / stddev of x for each instrument within its group.

group_backfill

- **Category:** Group
- **Scope:** REGULAR
- **Definition:** group_backfill(x, group, d, std = 4.0)
- **Description:** If a certain value for a certain date and instrument is NaN, from the set of same group instruments, calculate winsorized mean of all non-NaN values over last d days
- **Documentation:** https://api.worldquantbrain.com/operators/group_backfill
- **Documentation Content:**

```
group_backfill(x, group, d, std = 4.0)



If a certain value for a certain date and instrument is NaN, from the set of same group
instruments, calculate winsorized mean of all non-NaN values over last d days. Winsorized mean
means inputs are truncated by std * stddev where stddev is the standard deviation of inputs.

Example:
If d = 4 and there are 3 instruments(i1, i2, i3) in a group whose values for past 4 days are
x[i1] = [4,2,5,5], x[i2] = [7,NaN,2,9], x[i3] = [NaN,-4,2,NaN] where first element is most
recent, then if we want to backfill x, we will only have to backfill x[i3]â€™s first element
because every other instrumentâ€™s first element is non-NaN.

The non-NaN values of other groups are [4,2,5,5,7,2,9,-4,2]. Mean = 3.56, Standard deviation is
3.71 and none of the item is outside the range of 3.56 â€“ 4 * 3.71 and 3.56 + 4 * 3.71. Hence,
we donâ€™t need to clip elements to those limits. Hence, Winsorized mean = backfilled value =
3.56.

For three instruments, group_backfill(x, group, d, std = 4.0) = [4,7,3.56]
```

group_mean

- **Category:** Group
- **Scope:** REGULAR
- **Definition:** group_mean(x, weight, group)
- **Description:** All elements in group equals to the mean

group_rank

- **Category:** Group

- **Scope:** REGULAR
- **Definition:** group_rank(x, group)
- **Description:** Each elements in a group is assigned the corresponding rank in this group
- **Documentation:** https://api.worldquantbrain.com/operators/group_rank
- **Documentation Content:**

```
group_rank(x, group)


Group operators are a type of cross-sectional operator that compares stocks at a finer level,
where the cross-sectional operation is applied within each group, rather than across the entire
market. The group_rank operator allocates the stocks to their specified group, then within each
group, it ranks the stocks based on their input value for data field x and returns an equally
distributed number between 0.0 and 1.0.

This operator may help reduce both outliers and drawdown while reducing correlation.

Example: group_rank(x, subindustry)

 ∘ The stocks are first grouped into their respective subindustry.
 ∘ Within each subindustry, the stocks within that subindustry are ranked based on their input
   value for data field x and assigned an equally distributed number between 0.0 and 1.0.
```

group_neutralize

- **Category:** Group
- **Scope:** REGULAR
- **Definition:** group_neutralize(x, group)
- **Description:** Neutralizes Alpha against groups. These groups can be subindustry, industry, sector, country or a constant
- **Documentation:** https://api.worldquantbrain.com/operators/group_neutralize
- **Documentation Content:**

```
group_neutralize(x, group)


Neutralize alpha against groups. Difference between normalize and group_neutralize is in
normalize, every element is subtracted by mean of all values of all instruments on that day
whereas in group_neutralize, element is subtracted by mean of all values of the group of
instruments that it belongs on that day.

This operator may help reduce correlation, depending on the neutralization used.

Example:
If values of field x on a certain date for 10 instruments is [3,2,6,5,8,9,1,4,8,0] and first 5
instruments belong to one group, last 5 belong to other, then mean of first group =
(3+2+6+5+8)/5 = 4.8 and mean of second group = (9+1+4+8+0)/5 = 4.4. Subtracting means from
instruments of respective groups gives [3-4.8, 2-4.8, 6-4.8, 5-4.8, 8-4.8, 9-4.4, 1-4.4, 4-4.4,
8-4.4, 0-4.4] = [-1.8, -2.8, 1.2, 0.2, 3.2, 4.6, -3.4, -0.4, 3.6, -4.4]
```