Functions Explained

Functions are expressions or terms that can be used as building blocks to combine Price Sources and Value Sources to answer questions or create decision points. They cannot be used in isolation, and don't inherently provide any numerical quantities without being applied to a Price Source, Value Source or a combination thereof.

Arithmetic Operators

You can use basic arithmetic such as '+', '-', '*', and '/' in your custom prices. In addition, you can use any number of parenthesis. There is also an advanced operator for the exponent '^'

Examples

```
dbmarket + 5g
(dbmarket / 2) + crafting
((dbmarket + 10g) + (dbhistorical + 10g)) * 2
(vendorbuy + 100g) / 2
crafting / 0.95
```

> **Tip**
>
> Dividing by `/ 0.95` will account for the 5% Auction House Sale Cut

Percent of Price Source

You can use '%' as a shorthand for '/100 *' (which is the definition of percentage)

Examples

```
50% dbmarket
110% crafting
vendorbuy
```

> **Tip**
>
> 100% is implied if no % is used. In the example, `vendorbuy` is the same as `100% vendorbuy`

min() and max()

The min() and max() functions will evaluate to the lowest or highest of the valid parameters respectively. If any of the parameters are invalid, they will be silently ignored (unless they all are).

Examples

```
min(50% dbmarket, 100g)
min(dbmarket, dbregionmarketavg)
max(vendorsell, dbminbuyout, avgsell, 20g)
```

first()

Returns the first parameter which is a valid price. This can be useful to specify a fallback price in case something is invalid (i.e. if you're using market value and the item has not been posted to your auction house in 14 days).

Examples

```
first(dbmarket, dbregionmarketavg)
first(avgsell, crafting, 100g)
```

avg()

Returns the average, or mean, of the valid parameters between the parentheses, i.e the sum divided by the count.

Examples

```
avg(dbmarket, dbregionmarketavg)
avg(crafting, avgbuy)
```

check()

This is a simple logic check in the format **`check(a, b, c)`**.
If the first parameter **a** is greater than 0, the second parameter **b** is returned. Otherwise, if the first parameter **a** is equal to or less than 0, the third parameter **c** is returned.

Example

```
check(dbmarket - 1000g, 95% dbmarket, 50% dbmarket)

check(50g-100g, 1g, 2g)
```

convert()

This is a special function for dealing with item conversions. The convert() function has one required parameter which is the price source to be used in calculating the value of the items which convert to the target item. This parameter cannot be a custom price string and must be simply a price source. Conversions include milling, prospecting, transforming (e.g. greater essences to lesser essences and vice versa) and also vendor trades for inks. The function will return the lowest cost of all possible conversions of the item.

Examples

```
convert(dbmarket)
```

Item Parameters

Any price source can be evaluated as a function where you pass an item link or item string in order to evaluate the price source for that specific item. By default, the price source will be evaluated for the item which the entire custom price is being evaluated for. This also works for convert() as the second parameter.

Examples

```
dbminbuyout([Ghost Iron Ore])
matprice([Ink of Dreams])
dbmarket + convert(dbminbuyout, item:79251)
```

Logic Functions

Advanced Logic Functions allow you to input multiple Price or Value Sources and evaluate a single output through various logic gates and operations. The **`IF`** functions will take 3 or 4 parameters where the 4th parameter is optional.

ifgt() and ifgte()

"If Greater Than" in the format **`ifgt(a, b, c, d)`**
"If Greater Than or Equal To" in the format **`ifgte(a, b, c, d)`**
If parameter **a** is greater than (or equal to) to parameter **b**, then parameter **c** is returned. Otherwise, if parameter **a** is not greater than (or equal to) parameter **b**, then parameter **d** is returned. Parameter **d** can be omitted, and would result in the operation being "invalid" if parameter **d** would have otherwise been returned.

Examples

```
ifgt(dbmarket, 100g, 50% dbmarket, 20g)
ifgt(dbregionsalerate, 0.3, 10% crafting, 50% crafting)
ifgte(numinventory, 2000, 50% avgsell)
```

iflt() and iflte()

"If Less Than" in the format **`iflt(a, b, c, d)`**
"If Less Than or Equal To" in the format **`iflte(a, b, c, d)`**
If parameter **a** is less than (or equal to) to parameter **b**, then parameter **c** is returned. Otherwise, if parameter **a** is not less than (or equal to) parameter **b**, then parameter **d** is returned. Parameter **d** can be omitted, and would result in the operation being "invalid" if parameter **d** would have otherwise been returned.

Examples

```
iflt(crafting, 100g, 50)
iflt(itemlevel, 300, 50% dbmarket, ifgt(itemlevel, 350, 90% dbregionmarketavg)
iflte(salerate, 0.5, 500, 5000)
```

ifeq()

"If Equal To" in the format **`ifeq(a, b, c, d)`**
If parameter **a** is equal to parameter **b**, then parameter **c** is returned. Otherwise, if parameter **a** is not equal to parameter **b**, then parameter **d** is returned. Parameter **d** can be omitted, and would result in the operation being "invalid" if parameter **d** would have otherwise been returned.

Examples

```
ifeq(numinventory, 10, 50g)
ifeq(dbregionsalerate, 0.1, 50% dbmarket)
```

Rounding

Rounding functions will take 1 or 2 parameters, where the 1st parameter is the numerical quantity in gold you want to round, and the 2nd parameter is the factor you want to round to. If the 2nd parameter is not defined, rounding is done to the nearest copper, as appropriate.

"Standard Round" in the format **`round(a, b)`**
"Round Up" or Ceiling, in the format **`roundup(a, b)`**
"Round Down" or Floor, in the format **`rounddown(a, b)`**

Examples

```
round(crafting, 10s)
roundup(avg(dbmarket, dbregionmarketavg), 10g))
rounddown(avgbuy)
```