Q LANGUAGE - BUILT-IN FUNCTIONS

The **q** programming language has a set of rich and powerful built-in functions. A built-in function can be of the following types −

- **String function** − Takes a string as input and returns a string.
- **Aggregate function** − Takes a list as input and returns an atom.
- **Uniform function** − Takes a list and returns a list of the same count.
- **Mathematical function** − Takes numeric argument and returns a numeric argument.
- **Miscellaneous function** − All functions other than above mentioned.

String Functions

Like − pattern matching

```
q)/like is a dyadic, performs pattern matching, return 1b on success else 0b

q)"John" like "J??n"
1b

q)"John My Name" like "J*"
1b
```

ltrim − removes leading blanks

```
q)/ ltrim - monadic ltrim takes string argument, removes leading blanks

q)ltrim " Rick "
"Rick "
```

rtrim − removes trailing blanks

```
q)/rtrim - takes string argument, returns the result of removing trailing blanks

q)rtrim " Rick "
" Rick"
```

ss − string search

```
q)/ss - string search, perform pattern matching, same as "like" but return the indices of
the matches of the pattern in source.

q)"Life is beautiful" ss "i"
1 5 13
```

trim − removes leading and trailing blanks

```
q)/trim - takes string argument, returns the result of removing leading & trailing blanks

q)trim " John "
"John"
```

Mathematical Functions

acos − inverse of cos

```
q)/acos - inverse of cos, for input between -1 and 1, return float between 0 and pi

q)acos 1
0f

q)acos -1
3.141593

q)acos 0
1.570796
```

cor − gives correlation

```
q)/cor - the dyadic takes two numeric lists of same count, returns a correlation between
the items of the two arguments

q)27 18 18 9 0 cor 27 36 45 54 63
-0.9707253
```

cross − Cartesian product

```
q)/cross - takes atoms or lists as arguments and returns their Cartesian product

q)9 18 cross `x`y`z

9 `x
9 `y
9 `z

18 `x
18 `y
18 `z
```

var − variance

```
q)/var - monadic, takes a scaler or numeric list and returns a float equal to the
mathematical variance of the items

q)var 45
0f

q)var 9 18 27 36
101.25
```

wavg

```
q)/wavg - dyadic, takes two numeric lists of the same count and returns the average of
the second argument weighted by the first argument.

q)1 2 3 4 wavg 200 300 400 500
400f
```

Aggregate Functions

all − & operation

```
q)/all - monadic, takes a scaler or list of numeric type and returns the result of &
applied across the items.

q)all 0b
```

```
0b

q)all 9 18 27 36
1b

q)all 10 20 30
1b
```

Any − | operation

```
q)/any - monadic, takes scaler or list of numeric type and the return the result of |
applied across the items

q)any 20 30 40 50
1b

q)any 20012.02.12 2013.03.11
'20012.02.12
```

prd − arithmetic product

```
q)/prd - monadic, takes scaler, list, dictionary or table of numeric type and returns
the arithmetic product.

q)prd `x`y`z! 10 20 30
6000

q)prd ((1 2; 3 4);(10 20; 30 40))

10 40
90 160
```

Sum − arithmetic sum

```
q)/sum - monadic, takes a scaler, list,dictionary or table of numeric type and returns
the arithmetic sum.

q)sum 2 3 4 5 6
20

q)sum (1 2; 4 5)
5 7
```

Uniform Functions

Deltas − difference from its previous item.

```
q)/deltas -takes a scalar, list, dictionary or table and returns the difference of each
item from its predecessor.

q)deltas 2 3 5 7 9
2 1 2 2 2

q)deltas `x`y`z!9 18 27

x | 9
y | 9
z | 9
```

fills − fills nulls value

```
q)/fills - takes scalar, list, dictionary or table of numeric type and returns a c copy
of the source in which non-null items are propagated forward to fill nulls
```

```
q)fills 1 0N 2 0N 4
1 1 2 2 4

q)fills `a`b`c`d! 10 0N 30 0N

a | 10
b | 10
c | 30
d | 30
```

maxs − cumulative maximum

```
q)/maxs - takes scalar, list, dictionary or table and returns the cumulative maximum of
the source items.

q)maxs 1 2 4 3 9 13 2
1 2 4 4 9 13 13

q)maxs `a`b`c`d!9 18 0 36

a | 9
b | 18
c | 18
d | 36
```

Miscellaneous Functions

Count − return number of element

```
q)/count - returns the number of entities in its argument.

q)count 10 30 30
3

q)count (til 9)
9

q)count ([]a:9 18 27;b:1.1 2.2 3.3)
3
```

Distinct − return distinct entities

```
q)/distinct - monadic, returns the distinct entities in its argument

q)distinct 1 2 3 4 2 3 4 5 6 9
1 2 3 4 5 6 9
```

Except − element not present in second arg.

```
q)/except - takes a simple list (target) as its first argument and returns a list
containing the items of target that are not in its second argument

q)1 2 3 4 3 1 except 1
2 3 4 3
```

fill − fill null with first argument

```
q)/fill (^) - takes an atom as its first argument and a list(target) as its second
argument and return a list obtained by substituting the first argument for every
occurrence of null in target

q)42^ 9 18 0N 27 0N 36
```

```
9 18 42 27 42 36

q)";"^"Life is Beautiful"
"Life;is;Beautiful"
```