Types of Functions

Functions can be classified in a number of ways. Here we have classified them based on the number and type of argument they take and the result type. Functions can be,

- **Atomic** – Where the arguments are atomic and produce atomic results
- **Aggregate** – atom from list
- **Uniform listfromlist** – Extended the concept of atom as they apply to lists. The count of the argument list equals the count of the result list.
- **Other** – if the function is not from the above category.

Binary operations in mathematics are called **dyadic functions** in q; for example, “+”. Similarly unary operations are called **monadic functions**; for example, “abs” or “floor”.

Frequently Used Functions

There are quite a few functions used frequently in q programming. Here, in this section, we will see the usage of some popular functions –

**abs**

```
q) abs -9.9 / Absolute value, Negates -ve number & leaves non -ve number
   9.9
```

**all**

```
q) all 4 5 0 -4 / Logical AND (numeric min), returns the minimum value
   0b
```

**Max [&, Min |], and Not !**

```
q) /And, Or, and Logical Negation
q) 1b & 1b        / And (Max)
   1b
q) 1b|0b              / Or (Min)
   1b
q) not 1b             /Logical Negate (Not)
   0b
```

**asc**

```
q)asc 1 3 5 7 -2 0 4    / Order list ascending, sorted list
   / in ascending order
s returned
   `s#-2 0 1 3 4 5 7
q)/attr - gives the attributes of data, which describe how it's sorted.
   `s denotes fully sorted, `u denotes unique and `p and `g are used to
   refer to lists with repetition, with `p standing for parted and `g for grouped
```

**avg**
q)avg 3 4 5 6 7               / Return average of a list of numeric values
5f

q)/Create on trade table

q)trade:([]time:3?(.Z-200);sym:3?(`ibm`msft`apple);price:3?99.0;size:3?100)

by

q)/ by - Groups rows in a table at given sym
q)select sum price by sym from trade    / find total price for each sym

<table>
<thead>
<tr>
<th>sym</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>apple</td>
<td>140.2165</td>
</tr>
<tr>
<td>ibm</td>
<td>16.11385</td>
</tr>
</tbody>
</table>

cols

q)cols trade / Lists columns of a table
`time`sym`price`size

count

q)count (til 9) / Count list, count the elements in a list and
/ return a single int value 9

port

q)\p 9999 / assign port number

q)/csv - This command allows queries in a browser to be exported to
   excel by prefixing the query, such as http://localhost:9999/.csv?select from trade
   where sym =`ibm

cut

q)/ cut - Allows a table or list to be cut at a certain point
q)(1 3 5) cut "abcdefghijkl"
   / the argument is split at 1st, 3rd and 5th letter.
   "bc"
   "de"
   "fghijkl"
q)5 cut "abcdefghijkl"  / cut the right arg. Into 5 letters part
   / until its end.
   "abcde"
   "fghij"
   "kl"

Delete

q)/delete - Delete rows/columns from a table
q)delete price from trade

<table>
<thead>
<tr>
<th>time</th>
<th>sym</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td>2009.06.18T06:04:42.919</td>
<td>apple</td>
<td>36</td>
</tr>
</tbody>
</table>
Distinct

q)/distinct - Returns the distinct element of a list

q)distinct 1 2 3 2 3 4 5 2 1 3       / generate unique set of number
1 2 3 4 5

enlist

q)/enlist - Creates one-item list.

q)enlist 37
,37

q)type 37           / -ve type value
-7h

q)type enlist 37    / +ve type value
7h

Fill

q)/fill - used with nulls. There are three functions for processing null values.

The dyadic function named fill replaces null values in the right argument with the atomic left argument.

q)100 ^ 3 4 0N 0N -5
3 4 100 100 -5

q)`Hello^`jack`herry`john`
`jack`herry`Hello`john`Hello

Fills

q)/fills - fills in nulls with the previous not null value.

q)fills 1 0N 2 0N 0N 2 3 0N -5 0N
1 1 2 2 2 2 3 3 -5 -5

First

q)/first - returns the first atom of a list

q)first 1 3 34 5 3
1

Flip

q)/flip - Monadic primitive that applies to lists and associations. It interchange the top two levels of its argument.

q)trade

<table>
<thead>
<tr>
<th>time</th>
<th>sym</th>
<th>price</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td>2009.11.14T12:42:34.653</td>
<td>ibm</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>2009.12.27T17:02:11.518</td>
<td>apple</td>
<td>97</td>
<td></td>
</tr>
<tr>
<td>2009.11.14T12:42:34.653</td>
<td>apple</td>
<td>97</td>
<td></td>
</tr>
<tr>
<td>2009.12.27T17:02:11.518</td>
<td>apple</td>
<td>97</td>
<td></td>
</tr>
</tbody>
</table>
q) flip trade

<table>
<thead>
<tr>
<th>time</th>
<th>2009.06.18T06:04:42.919</th>
<th>2009.11.14T12:42:34.653</th>
<th>2009.12.27T17:02:11.518</th>
</tr>
</thead>
<tbody>
<tr>
<td>sym</td>
<td>apple</td>
<td>ibm</td>
<td>apple</td>
</tr>
<tr>
<td>price</td>
<td>72.05742</td>
<td>16.11385</td>
<td>68.15909</td>
</tr>
<tr>
<td>size</td>
<td>36</td>
<td>12</td>
<td>97</td>
</tr>
</tbody>
</table>

iasc

q) iasc - Index ascending, return the indices of the ascended sorted list relative to the input list.

q) iasc 5 4 0 3 4 9

2 3 1 4 0 5

Idesc

q) idesc - Index desceding, return the descended sorted list relative to the input list.

q) idesc 0 1 3 4

3 2 1 0

in

q) in - In a list, dyadic function used to query list (on the right-handside) about their contents.

q) (2 4) in 1 2 3

1 0 b

insert

q) insert - Insert statement, upload new data into a table.

q) insert[`trade;((.z.Z);`samsung;48.35;99)],3

```
q) trade

<table>
<thead>
<tr>
<th>time</th>
<th>sym</th>
<th>price</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td>2009.06.18T06:04:42.919</td>
<td>apple</td>
<td>72.05742</td>
<td>36</td>
</tr>
<tr>
<td>2009.11.14T12:42:34.653</td>
<td>ibm</td>
<td>16.11385</td>
<td>12</td>
</tr>
<tr>
<td>2009.12.27T17:02:11.518</td>
<td>apple</td>
<td>68.15909</td>
<td>97</td>
</tr>
<tr>
<td>2015.04.06T10:03:36.738</td>
<td>samsung</td>
<td>48.35</td>
<td>99</td>
</tr>
</tbody>
</table>
```

key

q) key - three different functions i.e. generate +ve integer number, gives content of a directory or key of a table/dictionary.

q) key 9

0 1 2 3 4 5 6 7 8

q) key `:c:

`$RECYCLE.BIN`Config.Msi`Documents and Settings`Drivers`Geojit`hiberfil.sys`I..
lower

q)/lower - Convert to lower case and floor

q)lower ("JoHn";`HERRY`SYM)
  "john"
  `herry`sym

Max and Min

q)/Max and Min / a|b and a&b

q)9|7
  9
q)9&5
  5

null

q)/null - return 1b if the atom is a null else 0b from the argument list

q)null 1 3 3 0N
  0001b

Peach

q)/peach - Parallel each, allows process across slaves

q)foo peach list1       / function foo applied across the slaves named in list1
  'list1
q)foo:{x+27}
q)list1:(0 1 2 3 4)
q)foo peach list1       / function foo applied across the slaves named in list1
  27 28 29 30 31

Prev

q)/prev - returns the previous element i.e. pushes list forwards

q)prev 0 1 3 4 5 7
  0N 0 1 3 4 5

Random?

q)/random - syntax - n?list, gives random sequences of ints and floats

q)9?5
  0 0 4 0 3 2 2 0 1
q)3?9.9
  0.2426823 1.674133 3.901671

Raze

q)/raze - Flattn a list of lists, removes a layer of indexing from a list of lists. for
instance:

```
q)raze (( 12 3 4; 30 0);("hello";7 8); 1 3 4)

12 3 4
30 0
"hello"
7 8
1
3
4
```

read0

```
q)/read0 - Read in a text file

q)read0 `:c:/q/README.txt    / gives the contents of *.txt file
```

read1

```
q)/read1 - Read in a q data file

q)read1 `:c:/q/t1

0xff016200630b00050000073796d00746966d65007072696369616e73206b657973206f66207965617231
```

reverse

```
q)/reverse - Reverse a list

q)reverse 2 30 29 1 3 4

4 3 1 29 30 2

q)reverse "HelloWorld"

"dlroWolleH"
```

set

```
q)/set - set value of a variable

q)`x set 9
`
x

q)x

9

q)`:c:/q/test12 set trade

`:c:/q/test12

q)get `:c:/q/test12

<table>
<thead>
<tr>
<th>time</th>
<th>sym</th>
<th>price</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td>2009.06.18T06:04:42.919</td>
<td>apple</td>
<td>72.05742</td>
<td>36</td>
</tr>
<tr>
<td>2009.11.14T12:42:34.653</td>
<td>ibm</td>
<td>16.11385</td>
<td>12</td>
</tr>
<tr>
<td>2009.12.27T17:02:11.518</td>
<td>apple</td>
<td>68.15909</td>
<td>97</td>
</tr>
<tr>
<td>2015.04.06T10:03:36.738</td>
<td>samsung</td>
<td>48.35</td>
<td>99</td>
</tr>
<tr>
<td>2015.04.06T10:03:47.540</td>
<td>samsung</td>
<td>48.35</td>
<td>99</td>
</tr>
<tr>
<td>2015.04.06T10:04:44.844</td>
<td>samsung</td>
<td>48.35</td>
<td>99</td>
</tr>
</tbody>
</table>
**ssr**

q)/ssr - String search and replace, syntax - ssr["string";searchstring;replaced-with]

q)ssr["HelloWorld";"o";"0"]

"Hell0WOrld"

**string**

q)/string - converts to string, converts all types to a string format.

q)string (1 2 3; `abc"XYZ";0b)

(,"1";,"2";,"3")

"abc"

(,"X";,"Y";,"Z")

,"θ"

**SV**

q)/sv - Scalar from vector, performs different tasks dependent on its arguments.

It evaluates the base representation of numbers, which allows us to calculate the number of seconds in a month or convert a length from feet and inches to centimeters.

q)24 60 60 sv 11 30 49

41449 / number of seconds elapsed in a day at 11:30:49

**system**

q)/system - allows a system command to be sent,

q)system "dir *.py"

" Volume in drive C is New Volume"

" Volume Serial Number is 8CD2-05B2"

""

" Directory of C:\\Users\\myaccount-raj"

""

"09/14/2014  06:32 PM   22 hello1.py"

"  1 File(s)   22 bytes"

**tables**

q)/tables - list all tables

q)tables `

`s#`tab1`tab2`trade

**Til**

q)/til - Enumerate

q)til 5

0 1 2 3 4
trim

q)/trim - Eliminate string spaces

q)trim " John "
"John"

vs

q)/vs - Vector from scaler , produces a vector quantity from a scaler quantity

q)
"\"" vs "20150204\msft\20.45"
"20150204"
"msft"
"20.45"

xasc

q)/xasc - Order table ascending, allows a table (right-hand argument) to be sorted such that (left-hand argument) is in ascending order

q)`price xasc trade

<table>
<thead>
<tr>
<th>time</th>
<th>sym</th>
<th>price</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td>2009.11.14T12:42:34.653</td>
<td>ibm</td>
<td>16.11385</td>
<td>12</td>
</tr>
<tr>
<td>2015.04.06T10:03:36.738</td>
<td>samsung</td>
<td>48.35</td>
<td>99</td>
</tr>
<tr>
<td>2015.04.06T10:03:47.540</td>
<td>samsung</td>
<td>48.35</td>
<td>99</td>
</tr>
<tr>
<td>2015.04.06T10:04:44.844</td>
<td>samsung</td>
<td>48.35</td>
<td>99</td>
</tr>
<tr>
<td>2009.12.27T17:02:11.518</td>
<td>apple</td>
<td>68.15909</td>
<td>97</td>
</tr>
<tr>
<td>2009.06.18T06:04:42.919</td>
<td>apple</td>
<td>72.05742</td>
<td>36</td>
</tr>
</tbody>
</table>

xcol

q)/xcol - Renames columns of a table

q)`timeNew`symNew xcol trade

<table>
<thead>
<tr>
<th>timeNew</th>
<th>symNew</th>
<th>price</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td>2009.06.18T06:04:42.919</td>
<td>apple</td>
<td>72.05742</td>
<td>36</td>
</tr>
<tr>
<td>2009.11.14T12:42:34.653</td>
<td>ibm</td>
<td>16.11385</td>
<td>12</td>
</tr>
<tr>
<td>2009.12.27T17:02:11.518</td>
<td>apple</td>
<td>68.15909</td>
<td>97</td>
</tr>
<tr>
<td>2015.04.06T10:03:36.738</td>
<td>samsung</td>
<td>48.35</td>
<td>99</td>
</tr>
<tr>
<td>2015.04.06T10:03:47.540</td>
<td>samsung</td>
<td>48.35</td>
<td>99</td>
</tr>
<tr>
<td>2015.04.06T10:04:44.844</td>
<td>samsung</td>
<td>48.35</td>
<td>99</td>
</tr>
</tbody>
</table>

xcols

q)/xcols - Reorders the columns of a table,

q)`size`price xcols trade

<table>
<thead>
<tr>
<th>size</th>
<th>price</th>
<th>time</th>
<th>sym</th>
</tr>
</thead>
<tbody>
<tr>
<td>36</td>
<td>72.05742</td>
<td>2009.06.18T06:04:42.919</td>
<td>apple</td>
</tr>
<tr>
<td>12</td>
<td>16.11385</td>
<td>2009.11.14T12:42:34.653</td>
<td>ibm</td>
</tr>
<tr>
<td>97</td>
<td>68.15909</td>
<td>2009.12.27T17:02:11.518</td>
<td>apple</td>
</tr>
<tr>
<td>99</td>
<td>48.35</td>
<td>2015.04.06T10:03:36.738</td>
<td>samsung</td>
</tr>
<tr>
<td>99</td>
<td>48.35</td>
<td>2015.04.06T10:03:47.540</td>
<td>samsung</td>
</tr>
<tr>
<td>99</td>
<td>48.35</td>
<td>2015.04.06T10:04:44.844</td>
<td>samsung</td>
</tr>
</tbody>
</table>
**xdesc**

q)/xdesc - Order table descending, allows tables to be sorted such that the left-hand argument is in descending order.

```
q)`price xdesc trade

<table>
<thead>
<tr>
<th>time</th>
<th>sym</th>
<th>price</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td>2009.06.18T06:04:42.919</td>
<td>apple</td>
<td>72.05742</td>
<td>36</td>
</tr>
<tr>
<td>2009.12.27T17:02:11.518</td>
<td>apple</td>
<td>68.15909</td>
<td>97</td>
</tr>
<tr>
<td>2015.04.06T10:03:36.738</td>
<td>samsung</td>
<td>48.35</td>
<td>99</td>
</tr>
<tr>
<td>2015.04.06T10:03:47.540</td>
<td>samsung</td>
<td>48.35</td>
<td>99</td>
</tr>
<tr>
<td>2015.04.06T10:04:44.844</td>
<td>samsung</td>
<td>48.35</td>
<td>99</td>
</tr>
<tr>
<td>2009.11.14T12:42:34.653</td>
<td>ibm</td>
<td>16.11385</td>
<td>12</td>
</tr>
</tbody>
</table>
```

**xgroup**

q)/xgroup - Creates nested table

```
q)`x xgroup ([].x:9 18 9 18 27 9 9;y:10 20 10 20 30 40)
'

```
q)`x xgroup ([].x:9 18 9 18 27 9 9;y:10 20 10 20 30 40 10)

```

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>10 10 40 10</td>
</tr>
<tr>
<td>18</td>
<td>20 20</td>
</tr>
<tr>
<td>27</td>
<td>,30</td>
</tr>
</tbody>
</table>
```

**xkey**

q)/xkey - Set key on table

```
q)`sym xkey trade

<table>
<thead>
<tr>
<th>sym</th>
<th>time</th>
<th>price</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td>apple</td>
<td>2009.06.18T06:04:42.919</td>
<td>72.05742</td>
<td>36</td>
</tr>
<tr>
<td>ibm</td>
<td>2009.11.14T12:42:34.653</td>
<td>16.11385</td>
<td>12</td>
</tr>
<tr>
<td>apple</td>
<td>2009.12.27T17:02:11.518</td>
<td>68.15909</td>
<td>97</td>
</tr>
<tr>
<td>samsung</td>
<td>2015.04.06T10:03:36.738</td>
<td>48.35</td>
<td>99</td>
</tr>
<tr>
<td>samsung</td>
<td>2015.04.06T10:03:47.540</td>
<td>48.35</td>
<td>99</td>
</tr>
<tr>
<td>samsung</td>
<td>2015.04.06T10:04:44.844</td>
<td>48.35</td>
<td>99</td>
</tr>
</tbody>
</table>
```

**System Commands**

System commands control the q environment. They are of the following form –

```
\cmd \[p\]  where p may be optional
```

Some of the popular system commands have been discussed below –

**\a [ namespace] - List tables in the given namespace**

```
q)/Tables in default namespace
q)`a
`\,`trade
q)`a .o  / table in .o namespace.
`\,`TI
\b - View dependencies

q)/ views/dependencies
q)a:: x+y / global assignment
q)b:: x+1
q)\b
  `s#`a`b

\B - Pending views / dependencies

q)/ Pending views/dependencies
q)a::x+1 / a depends on x
q)\B / the dependency is pending
  / the dependency is pending
q)\B
  `s#`a`b
q)b
  `s#`a`b
q)b
29
q)a
29
q)\B
  `symbol$()

\cd - Change directory

q)/ change directory, \cd [name]
q)\cd "C:\\Users\\myaccount-raj"
q)\cd ../new-account
q)\cd "C:\\Users\\new-account"

\d - sets current namespace

q)/ sets current namespace \d [namespace]
q)\d /default namespace
q)\d .o /change to .o
q.o)\d`
o
q.o)\d . / return to default
q)key `o`/ lists namespaces other than .z
  `Q`h`j`o
q)\d .john /change to non-existent namespace

q.john\d .
`.john
q.john)\d .
q)\d .

\l - load file or directory from db

q)\ Load file or directory, \l

q)\l test2.q / loading test2.q which is stored in current path.

ric |     date     ex      openP    closeP    MCap
----------- | -------------------------------------------------
JPMORGAN | 2008.05.23  SENSEX  18.30185  17.16319  17876
HSBC     | 2002.05.21  NIFTY   2.696749  16.58846  26559
JPMORGAN | 2006.09.07  NIFTY   14.15219  20.05624  14557
HSBC     | 2010.10.11  SENSEX  7.394497  25.45859  29366
JPMORGAN | 2007.10.02  SENSEX  1.558085  25.61478  20390

ric |     date     ex      openP    closeP    MCap
---------- | ------------------------------------------------
INFOSYS  | 2003.10.30  DOW    21.2342   7.565652   2375
RELIANCE | 2004.08.12  DOW    12.34132  17.68381   4201
SBIN    | 2008.02.14  DOW    1.830857  9.006485   1546
INFOSYS  | 2009.06.11  HENSENG 19.47664  12.05208   11143
SBIN    | 2010.07.05  DOW    18.55637  10.54082   15873

\p - port number

q)/ assign port number, \p

q)\p
5001i
q)\p 8888
q)\p 8888i

\\ - Exit from q console

\\ - exit
Exit form q.