Q LANGUAGE - INDEXING

A list is ordered from left to right by the position of its items. The offset of an item from the beginning of the list is called its **index**. Thus, the first item has an index 0, the second item *ifthereisone* has an index 1, etc. A list of count **n** has index domain from **0** to **n–1**.

Index Notation

Given a list **L**, the item at index **i** is accessed by **L[i]**. Retrieving an item by its index is called **item indexing**. For example,

```
q)L:(99;98.7e;`b;`abc;"z")

q)L[0]
99

q)L[1]
98.7e

q)L[4]
"z
```

Indexed Assignment

Items in a list can also be assigned via item indexing. Thus,

```
q)L1:9 8 7

q)L1[2]:66      / Indexed assignment into a simple list
                / enforces strict type matching.

q)L1
9 8 66
```

Lists from Variables

```
q)l1:(9;8;40;200)

q)l2:(1 4 3; `abc`xyz)

q)l:(l1;l2)               / combining the two list l1 and l2

q)l
9 8 40 200
(1 4 3;`abc`xyz)
```

Joining Lists

The most common operation on two lists is to join them together to form a larger list. More precisely, the join operator , appends its right operand to the end of the left operand and returns the result. It accepts an atom in either argument.

```
q)1,2 3 4
1 2 3 4

q)1 2 3, 4.4 5.6     / If the arguments are not of uniform type,
                     / the result is a general list.
1
2
3
4.4
```

```
5.6
```

Nesting

Data complexity is built by using lists as items of lists.

Depth

The number of levels of nesting for a list is called its depth. Atoms have a depth of 0 and simple lists have a depth of 1.

```
q)l1:(9;8;(99;88))

q)count l1
3
```

Here is a list of depth 3 having two items −

```
q)l5
9
(90;180;900 1800 2700 3600)

q)count l5
2

q)count l5[1]
3
```

Indexing at Depth

It is possible to index directly into the items of a nested list.

Repeated Item Indexing

Retrieving an item via a single index always retrieves an uppermost item from a nested list.

```
q)L:(1;(100;200;(300;400;500;600)))

q)L[0]
1

q)L[1]
100
200
300 400 500 600
```

Since the result **L[1]** is itself a list, we can retrieve its elements using a single index.

```
q)L[1][2]
300 400 500 600
```

We can repeat single indexing once more to retrieve an item from the innermost nested list.

```
q)L[1][2][0]
300
```

You can read this as,

> *Get the item at index 1 from L, and from it retrieve the item at index 2, and from it retrieve the item at index 0.*

Notation for Indexing at Depth

There is an alternate notation for repeated indexing into the constituents of a nested list. The last

retrieval can also be written as,

```
q)L[1;2;0]
300
```

Assignment via index also works at depth.

```
q)L[1;2;1]:900

q)L
1
(100;200;300 900 500 600)
```

Elided Indices

Eliding Indices for a General List

```
q)L:((1 2 3; 4 5 6 7); (`a`b`c;`d`e`f`g;`0`1`2);("good";"morning"))

q)L
(1 2 3;4 5 6 7)
(`a`b`c;`d`e`f`g;`0`1`2)
("good";"morning")

q)L[;1;]
4 5 6 7
`d`e`f`g
"morning"

q)L[;;2]
3 6
`c`f`2
"or"
```

Interpret L[;1;] as,

Retrieve all items in the second position of each list at the top level.

Interpret L[;;2] as,

Retrieve the items in the third position for each list at the second level.