# Redis: List & Set Values

**Recap**

In previous article we talked about:

* What is redis?
* Using redis to map String Values.

In this article, we will go through basic commands used to Store List and Set Values.

**Redis: List Values**

Lists in redis are implemented using Linked List. Linked List enabled insertion operations of values in list almost constant. This means, you add first element or thousandth, the time to insert that element will be constant.

It is important to mention that Linked List has two entry/exit points: One is called **Head** (the left side) and one is called **Tail** (the right side). Having said that, let’s begin to learn basic add, read, and delete operations to store list values in Redis using CLI commands:

1. LPUSH – Insert value in List from **Left (Head)**

**Syntax –** LPUSH Key-Name ListElement1 [ListElement2] [ListElement3]

* + If the key-name supplied doesn’t exist then new empty list with that key-name is created. If the key already exists and contains value other than List data type, error is returned.
  + On successful execution of command, an integer is returned that specifies the number of elements inserted.

1. RPUSH – Insert value in List from **Right (Tail)**

**Syntax –** RPUSH Key-Name ListElement1 [ListElement2] [ListElement3]

* + Same as LPUSH command, If the key-name supplied doesn’t exist then new empty list with that key-name is created. If the key already exists and contains value other than List data type, error is returned.
  + It also returns integer on successful execution that specifies number of elements inserted.

1. LRANGE – Prints out elements of List for given input range

**Syntax –** LRANGE Key-Name StartIndex EndIndex

* + If there exists any list elements between range of StartNumber and EndNumber it will return the array of elements, else will return empty array.
  + If considered List to be evaluated from Left to Right, then the index of first element is 0, the second element is 1 and so on.
  + On entering negative indexes in the command, the list will be evaluated from Right to Left. Therefore, the last element will be -1, second last element will be -2 and so on.

1. LPOP – Removes element from Left (Head) and displays that in output

**Syntax** – LPOP key-name

* + If there is nothing to pop or list is empty, it returns nil.

1. RPOP – Removes element from Right (Tail) and displays that in output

**Syntax –** RPOP key-name

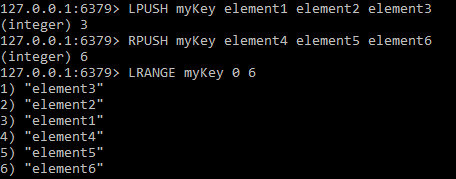
* + If there is nothing to pop or list is empty, it returns nil.

1. LREM – Removes occurrences of elements from the List.

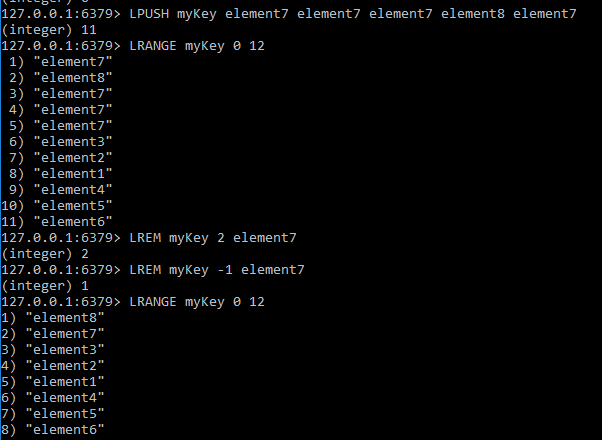
**Syntax –** LREM key-name count value

* + The count parameter in the command determines how many occurrences to remove.
  + If count>0, it will remove the ‘count’ occurrences from Head to Tail of the list
  + If count<0, it will remove the ‘count’ occurrences from Tail to Head of the list
  + If count=0, it will remove all the ‘count’ occurrences of the input value.

These were basic CRUD operations for List values. There are many advanced and useful other list commands such as BLPOP, BRPOP etc but these commands are out of scope for this introductory post. You can check out and play yourself with other list commands from [here](https://redis.io/commands#list)



1Usage of LPUSH, RPUSH and LRANGE



2 LREM command

**Redis: Set Values**

Sets are simply the unordered collection of unique Strings. The primary differences between List and Set is that

* List is an ordered collection while sets are unordered.
* List can contain duplicate elements while sets cannot. The elements in set are unique.

The saved set values are fetched in any random order and can be used in the scenarios where there is no requirement of strict ordering. Following are basic add, read and delete commands for set values in redis.

1. SADD –Add new elements to set

**Syntax –** SADD key-name element1 [element2] [element3]

* + It will add all set elements and return the number of elements added in set if successfully executed.
  + If the element that is added is already in the set, it is ignored and 0 is returned.

1. SMEMBERS – Displays all elements of set

**Syntax –**SMEMBERS key-name

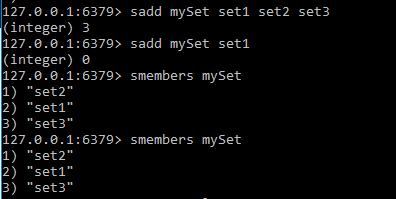
* + Displays the set elements in unordered manner.

1. SPOP – Removes any random number of elements from the set and displays that element

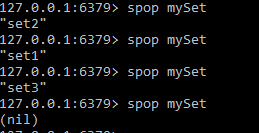
**Syntax –**SPOP myKey [count]

* + Returns nil if set becomes empty

There are many other advanced commands used to perform set operations like union, intersection, difference etc, but these are out of scope of these introductory post. You can check out and play yourself with all set commands [here](https://redis.io/commands#set).



**3** SADD and SMEMBERS commands



4SPOP commands