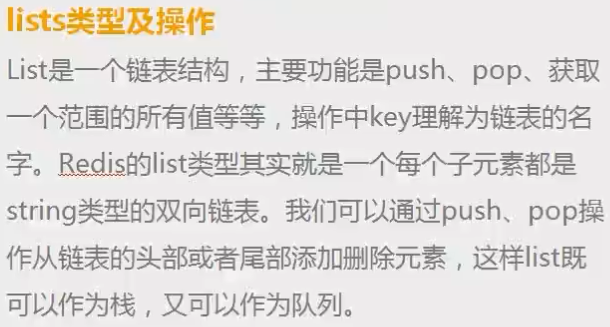
List类型

1. **List类型：**

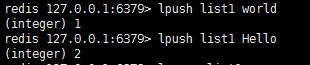


1. **push、pop方法。**
2. 栈： 先入后出FILO。 lpush与lrange就构成了栈。

队列：先入先出FIFO。rpush与lrange就构成了对列。

1. **lpush方法：在key对应list的头部添加字符串元素**。返回压住的value的序数。（第几个）

**lpush listname value #压栈**



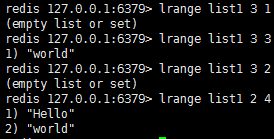
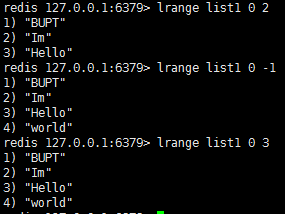


1. lrange 方法：取出一定范围数据

lrange listname fromindex endindex 取出数据：

注意： 如果lrange listname 0 -1 #表示的是取出全部数据。

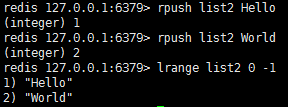
索引值0是刚刚进栈的数据，就是往下压一样，进去一个，原来数据的索引值就加一。



**如果endfrom超出了list的最大索引值，不报错，输出到索引最大值即可。**

1. **rpush ： 与lpush相反，从list的尾部压入数据**。

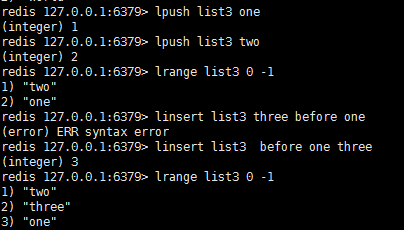
**rpush listname value**



1. **linsert方法：在某个value值前插入一个value。**

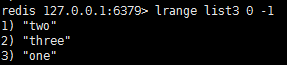
**linsert listname before value1 newvalue**

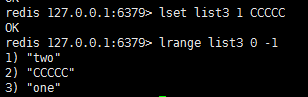
**注意：list表中前总是从尾部到顶部为前方向，及索引值减小的方向为前。**



1. **lset方法**：设置list中指定下标的元素值。

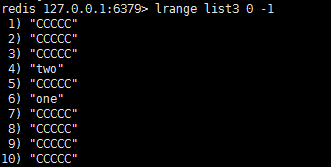
**lset listname index value #更改特定索引对应的value值**



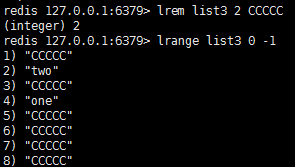


1. lrem 方法：删除list中和value值相同的元素，可以指定删除个数。

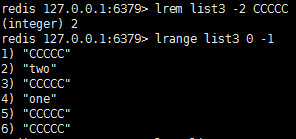
**lrem listname n value # 如果n<0，则从尾部删除|n|个value，如果n=0，则删除所有的value元素。返回值，就是成功删除的个数，如果没有删除，则返回0.**



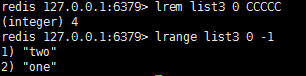
（1）从头部开始删除



（2）从尾部开始删除



（3）删除所有的



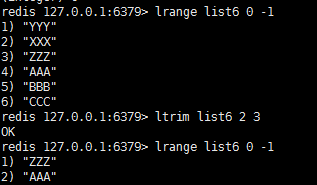
1. **ltrim ：保留指定索引范围的元素，并删除两端的元素。**

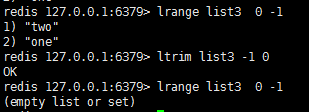
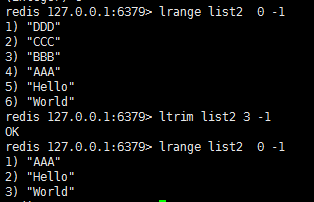
**ltrim listname fromindex endindex # 删除两端的元素**

**ltrim listname fromindex -1 # 删除后端的元素**

**ltrim listname -1 endindex #删除所有的元素，不是删除前面的元素。**

**注意 范围是保留的范围。**



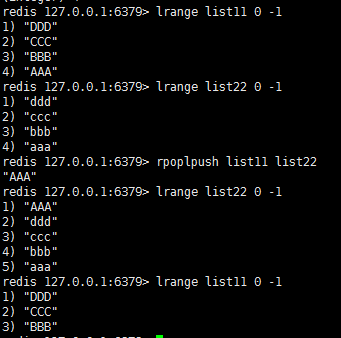


1. **lpop 方法：从list的头部删除元素，并返回删除的元素。**

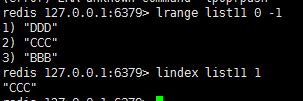
rpop 方法：从list的尾部删除元素，并返回删除的元素。

1. **rpoplpush 方法：从一个list的尾部删除一个元素，并将该元素添加到第2个元素的头部。返回的就是操作的元素。**

**注意没有lpoplpush、rpoprpush、lpoprpush等。**



1. **lindex方法：返回list中index位置对应的元素值。返回元素值。**



1. **llen 方法： 返回list中元素的个数，即list的长度。返回长度。**

