# 以下是key类型数据操作

## set [key] [value]

存入一个键值对（当redis重启后依然可以从redis中取到）

如果Set同样的key则value覆盖



## set [key] [value]

通过key获取值，获取不到则为nil





## dbsize

获取当前数据库中的数据数量



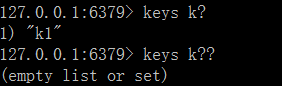
## select [number]

选择数据库。Redis自带16个数据库，编号从0开始。



## keys […]

查询键，？表示占位符占一位，若括号中填\*则表示查询当前库中的所有key。



## flushdb

删除当前数据库中的所有数据



## flushall

删除所有数据库中的所有数据（不推荐使用很可能会误删除数据）



## exists [key]

查询数据库中是否含存在此键的数据

1表示有

0表示没有



## move [key] [number]

将key为k3的数据转移到3号数据库



## ttl [key]

查询此键的数据过期状态

-1表示永久有效

-2表示已过期（数据库中不再含有此键的数据）

正数表示还是多少时间过期



## type [key]

查询此键数据的数据类型



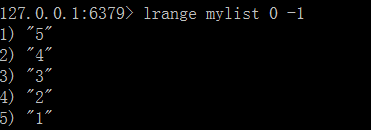


## lpush [listname] [values]

创建list类型的数据



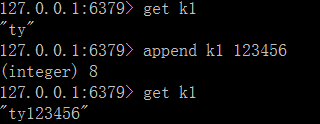
遍历list类型的数据



# 以下是string类型数据操作

## 合并字符串append [key] [string]

将键为key的字符串加上后面的string



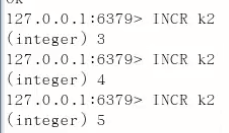
## 获得字符串长度strlen [key]



## 数据加一（仅限number类型的数据）

**Incr [key]**

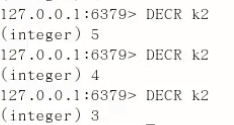
**通常用在统计点击量，快速实现数据的增长。**



## 数据减一（仅限number类型的数据）

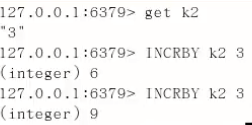
**decr [key]**

**通常用在统计点击量，快速实现数据的减少。**



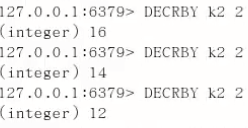
## 数据加number（仅限number类型的数据）

**Incrby [key] number**



## 数据减number（仅限number类型的数据）

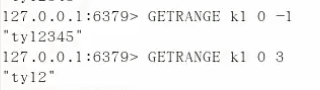
**decrby [key] number**



## 截取字符串

**Getrange [key] [start] [end]**

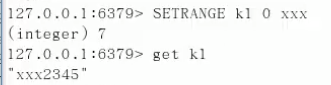
Start为开始角标，end为结束角标



## 替换字符串

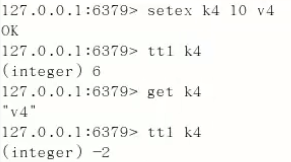
**Setrange [key] [start] [string]**

Start为开始角标，string为所要替换的字符串



## 设置字符串过期时间

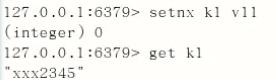
**Setex [key] [time] [value]**



## 不覆盖设值

**Setnx [key] [value]**

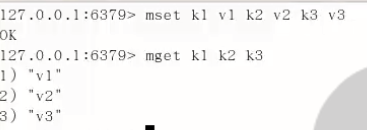
若数据库中已存在此键数据则不覆盖



## 批量获取和赋值

**mset [keys…] [values…]**

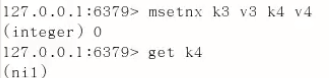
**mget [keys…] [values…]**



## 批量赋值不覆盖

**msetnx [keys…] [values…]**

只要有一个key已存在，就都不赋值



# 以下是List类型数据操作

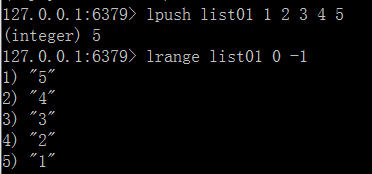
## 向list的左边插入数据

插入

**Lpush [listkey] [values…]**

从list的左边开始读取

**Lrange [listkey] [start] [end]**



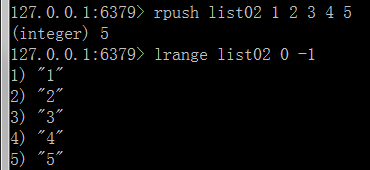
## 向list的右边插入数据

插入

**rpush [listkey] [values…]**

从list的左边开始读取（都是从左边读的，没有从右边读的）

**Lrange [listkey] [start] [end]**

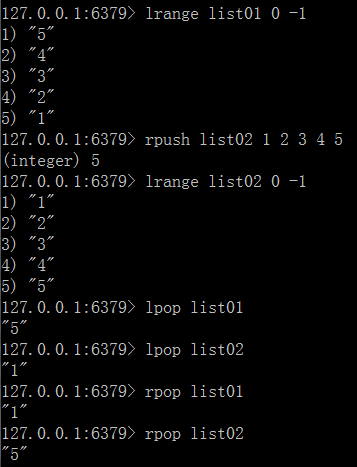


## List出栈操作

**Lpop [listkey] 从顶部出栈**

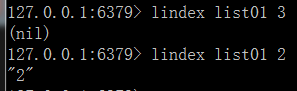
**Rpop [listkey] 从底部出栈**

如果不清楚方向，可以先将list数据读出来，应用lrange



## 根据list角标获取值

**Lindex [listkey] [角标]**



## 获取list的长度

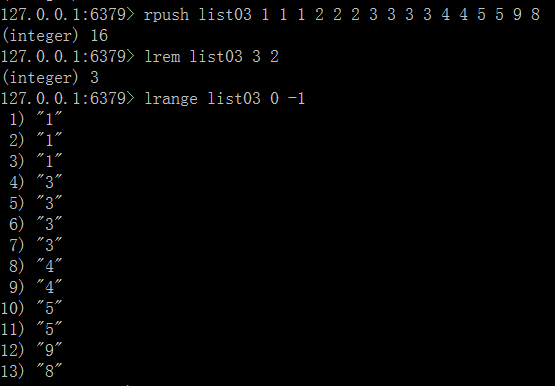
**Llen [listkey]**



## 删除n个值为x的元素

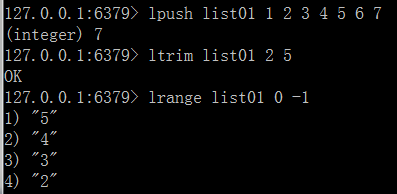
**Lrem [listkey] [n] [x]**

下图将list03中的3个2都去除了



## 截断list从start到end的数据再赋给list

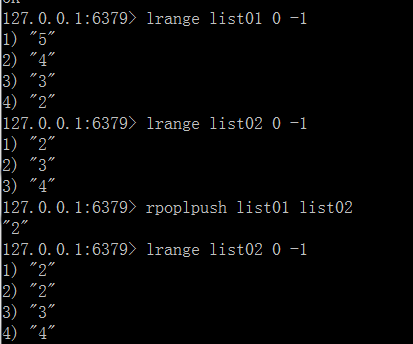
**Ltrim [listkey] [start] [end]**



## 从list01出栈再压栈到list02

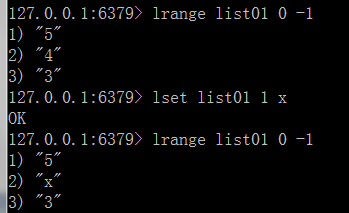
**Rpoplpush [listkey01] [listkey02]**

此操作值转移一个数据，返回信息为操作的数据。



## 根据角标替换对应数据

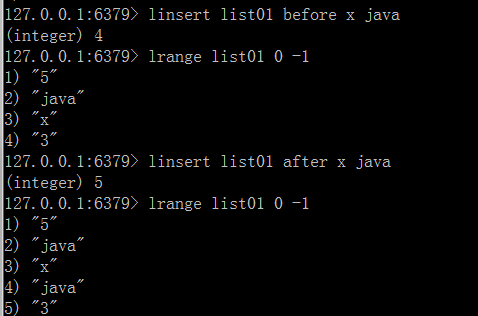
**Lset [listkey] [index] [value]**



## 在值为x的数据之前或之后插入数据

**Linsert [listkey] before [valuex] [value] 之前插入**

**Linsert [listkey] after [valuex] [value] 之后插入**



# 以下是set类型数据操作

## 向set添加数据

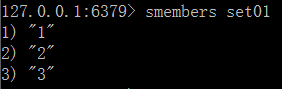
**Sadd [setkey] [values…]**

**重复值不重复插入**



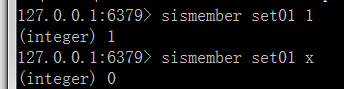
## 获取set的所有数据

**Smembers [setkey]**



## 判断set中是否含有此值数据

**Sismember [setkey] [value]**



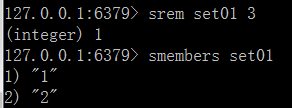
## 获取set里的元素个数

**Scard [setkey]**



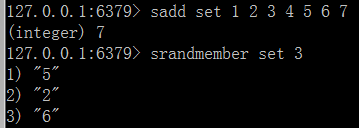
## 根据值删除元素

**Srem [setkey] [values] 可以同时删除多个**



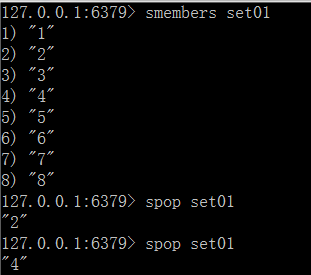
## 从set中随机抽取几个数

**Srandmember [setkey] [count]**



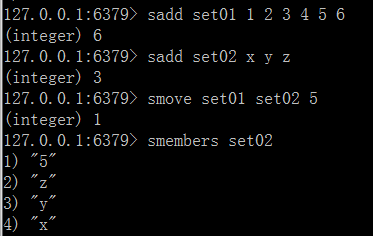
## 随机出栈

**Spop [setkey]**



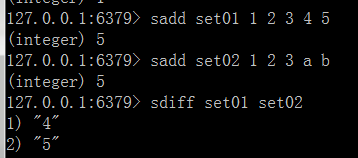
## 将set01的值移动到set02

**Smove [setkey01] [setkey02] [value]**



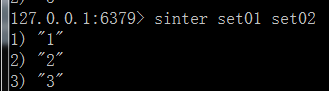
## 将set01与set02对比，取出set01中特有的值

**Sdiff [setkey01] [setkey02]**



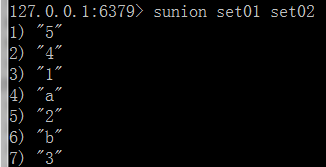
## 取出set01和set02的交集

**Sinter [setkey01] [setkey02]**



## 获得set01和set02的并集

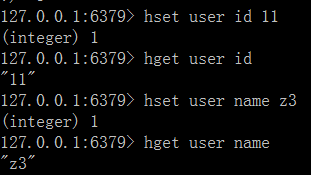
**Sunion [setkey01] [setkey02]**



# 以下是hash类型数据操作

## 向hashset添加数据

**Hset [对象key] [属性key] [属性值]**

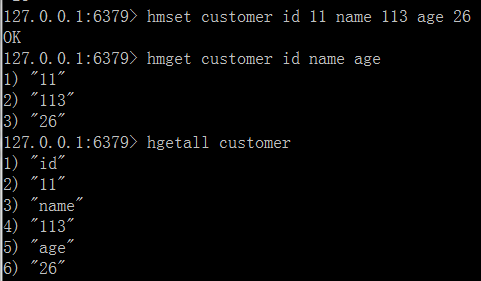


## 向hashset批量插入数据

**Hmset [对象key] [属性key] [属性值] …**

**Hmget [对象key] [属性key] … (通过key批量获取值)**

**Hgetall [对象key] (获取对象所有值和键)**



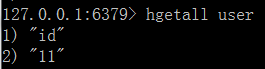
## 根据属性key删除属性

**Hdel [对象key] [属性key]**



## 获取对象中属性的个数

**Hgetall [对象key]**



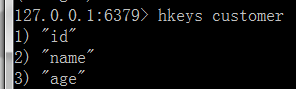
## 判断对象中是否含有该属性

**Hexists [对象key] [属性key]**

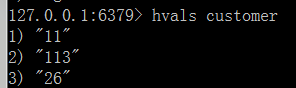


## 获得对象的所有key

**Hkeys [对象key]**

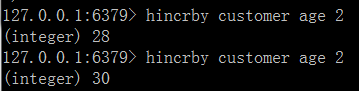


## 获得对象的所有value

**Hvals [对象key]**

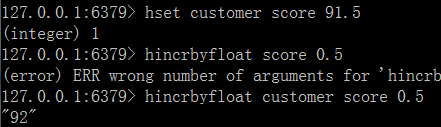
## 对对象的属性加值

**Hincrby [对象key] [对象属性key] [count]**

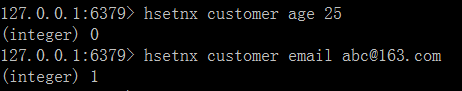


## 对对象的属性加小数

**Hincrbyfloat [对象key] [属性key] [count小数]**



## 向对象添加属性若已存在则不添加



# 以下是zset类型数据操作

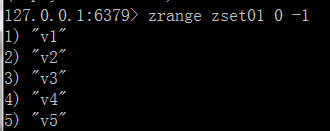
## 在set添加数据

**Zadd [zsetkey] [score] [value] …**



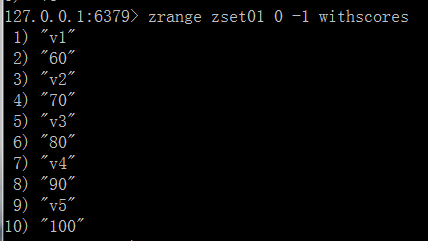
## 获取zset的所有value

**Zrange [zsetkey] [start] [end]**



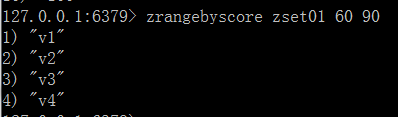
## 获取zset的所有value和score

**Zrange [zsetkey] [start] [end] withscores**



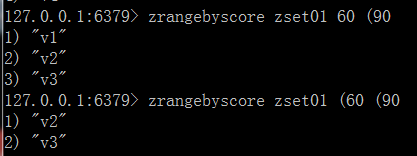
## 获取score范围内的value

**Zrangebyscore [zsetkey] min max**



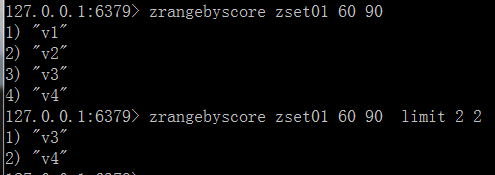
## 获取score范围内的value(不包含)

**Zrangebyscore [zsetkey] (min (max (加一个”(”就是不包含)**



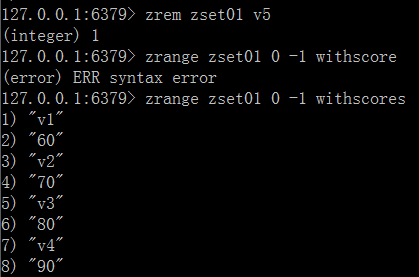
## 在有限的结果集中再截取

**Zrangebyscore [zsetkey] min max limit indexstart indexend**



## 通过value删除

**Zrem [zsetkey] value**



## 获得zset的元素个数（score和value为一个）

**Zcard [zsetkey]**



## 获得score范围内元素个数

**Zcount [zsetkey] min max**



## 通过value获得角标

**Zrank [zsetkey] [value]**



## 通过value获得score

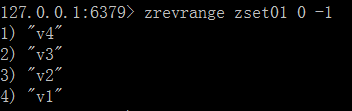
**Zscore [zsetkey] [value]**



## 逆序并通过value获得下标

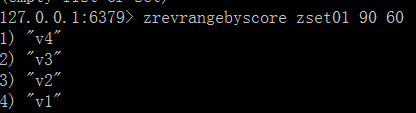
**Zrevrank [zsetkey] [value]**





## 通过score反转zset

**Zrerangebyscore [zetkey] [endscore] [startscore]**



# 为redis设置密码

平常最好别设，设置之后redis做任何操作都需要输入密码。

