一、KEY

```
127.0.0.1:6379> keys *   #查看所有的key
(empty array)
127.0.0.1:6379> set k1 v1 #设置键保存字符串值。
OK
127.0.0.1:6379> exists k1 #判断某个key是否存在
(integer) 1
127.0.0.1:6379> del k1 #删除key
(integer) 1
127.0.0.1:6379> keys *
(empty array)
127.0.0.1:6379> set k1 v1 EX 10 #设置过期时间
OK
127.0.0.1:6379> ttl k1 #查看剩余时间
(integer) 5
127.0.0.1:6379> ttl k1
(integer) 3
127.0.0.1:6379> ttl k1
(integer) 2
127.0.0.1:6379> set k2 v2
OK
127.0.0.1:6379> get k2 #获取key对应的value
"v2"
127.0.0.1:6379>
```

二、String

```
127.0.0.1:6379> keys *  #查看所有的key
1) "k2"
2) "num"
127.0.0.1:6379> get k2
"v2"
127.0.0.1:6379> get num
"2"
127.0.0.1:6379> incr num #自增 +1
(integer) 3
127.0.0.1:6379> incr num
```

```
(integer) 4
127.0.0.1:6379> decr num #减 -1 incr、decr 一定要是数字才能进行加减，+1 和 -1。
(integer) 3
127.0.0.1:6379> incrby num 10 #incrby、decrby 命令将 key 中储存的数字加上指定的增量值。
(integer) 13
127.0.0.1:6379> decrby num 5
(integer) 8
127.0.0.1:6379> set k3 asdfghj
OK
127.0.0.1:6379> getrange k3 0 -1 #getrange 获取指定区间范围内的值
"asdfghj"
127.0.0.1:6379> getrange k3 0 3
"asdf"
127.0.0.1:6379> get k3
"asdfghj"
127.0.0.1:6379> setrange k3 1 xx #setrange 设置指定区间范围内的值，格式是setrange key 值 具体值
(integer) 7
127.0.0.1:6379> get k3
"axxfghj"
127.0.0.1:6379> setex k3 10 exprire #设置过期时间
OK
127.0.0.1:6379> ttl k3 #查看剩余的时间
(integer) 6
127.0.0.1:6379> setnx k4 "redis" #如果不存在就设置，成功返回1
(integer) 1
127.0.0.1:6379> get k4
"redis"
127.0.0.1:6379> setnx k4 "redis"
(integer) 0
127.0.0.1:6379> mset k10 v10 k11 v11 k12 v12 #同时设置一个或多个 key-value 对。
OK
127.0.0.1:6379> keys *
1) "k2"
2) "k12"
3) "k4"
4) "k10"
5) "k11"
6) "num"
```

```
127.0.0.1:6379> mget k10 k11 k12 k13 # 命令返回所有(一个或多个)给定 key 的值。如果给定的 key 里面，有某个 key 不存在，那么这个 key 返回特殊值 nil
1) "v10"
2) "v11"
3) "v12"
4) (nil)
127.0.0.1:6379> msetnx k10 v10 k15 v15 #当所有 key 都成功设置，返回 1。如果所有给定 key 都设置失败(至少有一个 key 已经存在)，那么返回 0 。原子操作。
(integer) 0
127.0.0.1:6379> get key15
(nil)
127.0.0.1:6379> set user:1 value(json数据) #传统对象缓存，可以用来缓存对象
OK
127.0.0.1:6379> keys *
1) "k2"
2) "user:1"
3) "k12"
4) "k4"
5) "k10"
6) "k11"
7) "num"
127.0.0.1:6379> get user:1
"value(json\xe6\x95\xb0\xe6\x8d\xae)"
127.0.0.1:6379> mset user:1:name zhangsan user:1:age 16 #批量设置键和值
OK
127.0.0.1:6379> mget user:1:name user:1:age #批量获取值
1) "zhangsan"
2) "16"
127.0.0.1:6379> getset db mongodb #没有旧值，返回 nil
(nil)
127.0.0.1:6379> get db
"mongodb"
127.0.0.1:6379> getset db redis
"mongodb"
127.0.0.1:6379> get db #返回旧值 mongodb
"redis"
127.0.0.1:6379> strlen db #获取字符串的长度 (integer)
(integer) 5

```

三、List

```
1
2  # ====================================================
3  # Lpush:将一个或多个值插入到列表头部。(左)
4  # rpush:将一个或多个值插入到列表尾部。(右)
5  # lrange:返回列表中指定区间内的元素，区间以偏移量 START 和 END 指定。 # 其中 0 表示列表的第一个元素， 1 表示列表的第二个元素，以此类推。
6
7  # 你也可以使用负数下标，以 -1 表示列表的最后一个元素， -2 表示列表的倒数第二个元素，以此 类推。
8  # ====================================================
9   127.0.0.1:6379> LPUSH list "one"
10 (integer) 1
11 127.0.0.1:6379> LPUSH list "two"
12 (integer) 2
13 127.0.0.1:6379> RPUSH list "right"
14 (integer) 3
15 127.0.0.1:6379> Lrange list 0 -1
16 1) "two"
17 2) "one"
18 3) "right"
19 127.0.0.1:6379> Lrange list 0 1
20 1) "two"
21 2) "one"
22 # ====================================================
23 # lpop 命令用于移除并返回列表的第一个元素。当列表 key 不存在时，返回 nil 。
24 # rpop 移除列表的最后一个元素，返回值为移除的元素。
25 # ====================================================
26 127.0.0.1:6379> Lpop list
27 "two"
28 127.0.0.1:6379> Rpop list
29 "right"
30 127.0.0.1:6379> Lrange list 0 -1
31 1) "one"
32 # ====================================================
33 # Lindex，按照索引下标获得元素(-1代表最后一个，0代表是第一个)
34 # ====================================================
35 127.0.0.1:6379> Lindex list 1
36 (nil)
```

```
127.0.0.1:6379> Lindex list 0
"one"
127.0.0.1:6379> Lindex list -1
"one"
# ====================================================
# llen 用于返回列表的长度。
# ====================================================
127.0.0.1:6379> flushdb
OK
127.0.0.1:6379> Lpush list "one" (integer) 1
127.0.0.1:6379> Lpush list "two" (integer) 2
127.0.0.1:6379> Lpush list "three" (integer) 3
127.0.0.1:6379> Llen list # 返回列表的长度 (integer) 3
# ====================================================
# lrem key 根据参数 COUNT 的值，移除列表中与参数 VALUE 相等的元素。 #
====================================================
127.0.0.1:6379> lrem list 1 "two"
(integer) 1
127.0.0.1:6379> Lrange list 0 -1
1) "three"
2) "one"
# ====================================================
# Ltrim key 对一个列表进行修剪(trim)，就是说，让列表只保留指定区间内的元素，不在指定区 间之内
的元素都将被删除。
# ====================================================
127.0.0.1:6379> RPUSH mylist "hello"
(integer) 1
127.0.0.1:6379> RPUSH mylist "hello"
(integer) 2
127.0.0.1:6379> RPUSH mylist "hello2"
(integer) 3
127.0.0.1:6379> RPUSH mylist "hello3"
(integer) 4
127.0.0.1:6379> ltrim mylist 1 2
OK
127.0.0.1:6379> lrange mylist 0 -1
1) "hello"
2) "hello2"
# ====================================================
```

```
# rpoplpush 移除列表的最后一个元素，并将该元素添加到另一个列表并返回。 # ==================================================
127.0.0.1:6379> rpush mylist "hello"
(integer) 1
127.0.0.1:6379> rpush mylist "foo"
(integer) 2
127.0.0.1:6379> rpush mylist "bar"
(integer) 3
127.0.0.1:6379> rpoplpush mylist myotherlist
"bar"
127.0.0.1:6379> lrange mylist 0 -1
1) "hello"
2) "foo"
127.0.0.1:6379> lrange myotherlist 0 -1
1) "bar"
# ==================================================
# lset key index value 将列表 key 下标为 index 的元素的值设置为 value 。
# ==================================================
127.0.0.1:6379> exists list # 对空列表(key 不存在)进行 LSET (integer) 0
127.0.0.1:6379> lset list 0 item # 报错
(error) ERR no such key
127.0.0.1:6379> lpush list "value1" # 对非空列表进行 LSET (integer) 1
127.0.0.1:6379> lrange list 0 0
1) "value1"
127.0.0.1:6379> lset list 0 "new" # 更新值 OK
127.0.0.1:6379> lrange list 0 0
1) "new"
127.0.0.1:6379> lset list 1 "new" # index 超出范围报错 (error) ERR index out of range
# ==================================================
# linsert key before/after pivot value 用于在列表的元素前或者后插入元素。
# 将值 value 插入到列表 key 当中，位于值 pivot 之前或之后。
# ==================================================
redis> RPUSH mylist "Hello"
(integer) 1
redis> RPUSH mylist "World"
(integer) 2
redis> LINSERT mylist BEFORE "World" "There"
(integer) 3
redis> LRANGE mylist 0 -1
```

```
1) "Hello"
2) "There"
3) "World"

```

四、Set

```
# ====================================================
# sadd 将一个或多个成员元素加入到集合中，不能重复
# smembers 返回集合中的所有的成员。
# sismember 命令判断成员元素是否是集合的成员。
# ====================================================
127.0.0.1:6379> sadd myset "hello"
(integer) 1
127.0.0.1:6379> sadd myset "daming"
(integer) 1
127.0.0.1:6379> sadd myset "daming"
(integer) 0
127.0.0.1:6379> SMEMBERS myset
1) "daming"
2) "hello"
127.0.0.1:6379> SISMEMBER myset "hello"
(integer) 1
127.0.0.1:6379> SISMEMBER myset "world"
(integer) 0
# ====================================================
# scard，获取集合里面的元素个数
# ====================================================
127.0.0.1:6379> scard myset
(integer) 2
# ====================================================
# srem key value 用于移除集合中的一个或多个成员元素
# ====================================================
127.0.0.1:6379> srem myset "daming"
(integer) 1
127.0.0.1:6379> SMEMBERS myset
1) "hello"
# ====================================================
# srandmember key 命令用于返回集合中的一个随机元素。
```

```
# ====================================================
127.0.0.1:6379> SMEMBERS myset
1) "daming"
2) "world"
3) "hello"
127.0.0.1:6379> SRANDMEMBER myset
"hello"
127.0.0.1:6379> SRANDMEMBER myset 2
1) "world"
2) "daming"
127.0.0.1:6379> SRANDMEMBER myset 2
1) "daming"
2) "hello"
# ====================================================
# spop key 用于移除集合中的指定 key 的一个或多个随机元素
# ====================================================
127.0.0.1:6379> SMEMBERS myset
1) "daming"
2) "world"
3) "hello"
127.0.0.1:6379> spop myset
"world"
127.0.0.1:6379> spop myset
"daming"
127.0.0.1:6379> spop myset
"hello"
# ====================================================
# smove SOURCE DESTINATION MEMBER
# 将指定成员 member 元素从 source 集合移动到 destination 集合。
# ====================================================
127.0.0.1:6379> sadd myset "hello"
(integer) 1
127.0.0.1:6379> sadd myset "world"
(integer) 1
127.0.0.1:6379> sadd myset "daming"
(integer) 1
127.0.0.1:6379> sadd myset2 "set2"
(integer) 1
127.0.0.1:6379> smove myset myset2 "daming"
(integer) 1
```

```
127.0.0.1:6379> SMEMBERS myset
1) "world"
2) "hello"
127.0.0.1:6379> SMEMBERS myset2
1) "daming"
2) "set2"
# ================================================
- 数字集合类
- 差集: sdiff - 交集: sinter
- 并集: sunion
# ================================================
127.0.0.1:6379> sadd key1 "a"
(integer) 1
127.0.0.1:6379> sadd key1 "b"
(integer) 1
127.0.0.1:6379> sadd key1 "c"
(integer) 1
127.0.0.1:6379> sadd key2 "c"
(integer) 1
127.0.0.1:6379> sadd key2 "d"
(integer) 1
127.0.0.1:6379> sadd key2 "e"
(integer) 1
127.0.0.1:6379> SDIFF key1 key2 # 差集
1) "a"
2) "b"
127.0.0.1:6379> SINTER key1 key2 # 交集
1) "c"
127.0.0.1:6379> SUNION key1 key2 # 并集
1) "a"
2) "b"
3) "c"
4) "e"
5) "d"

```

五、Hash

```
# ================================================
```

```
# hset、hget 命令用于为哈希表中的字段赋值 。
# hmset、hmget 同时将多个field-value对设置到哈希表中。会覆盖哈希表中已存在的字段。
# hgetall 用于返回哈希表中，所有的字段和值。
# hdel 用于删除哈希表 key 中的一个或多个指定字段
# ====================================================
127.0.0.1:6379> hset myhash field1 "daming"
(integer) 1
127.0.0.1:6379> hget myhash field1
"daming"
127.0.0.1:6379> HMSET myhash field1 "Hello" field2 "World"
OK
127.0.0.1:6379> HGET myhash field1
"Hello"
127.0.0.1:6379> HGET myhash field2
"World"
127.0.0.1:6379> hgetall myhash
1) "field1"
2) "Hello"
3) "field2"
4) "World"
127.0.0.1:6379> HDEL myhash field1
(integer) 1
127.0.0.1:6379> hgetall myhash
1) "field2"
2) "World"
# ====================================================
# hlen 获取哈希表中字段的数量。
# ====================================================
127.0.0.1:6379> hlen myhash
(integer) 1
127.0.0.1:6379> HMSET myhash field1 "Hello" field2 "World"
OK
127.0.0.1:6379> hlen myhash
(integer) 2
# ====================================================
# hexists 查看哈希表的指定字段是否存在。
# ====================================================
127.0.0.1:6379> hexists myhash field1
(integer) 1
127.0.0.1:6379> hexists myhash field3
```

```
(integer) 0
# =====================================================
# hkeys 获取哈希表中的所有域(field)。
# hvals 返回哈希表所有域(field)的值。
# =====================================================
127.0.0.1:6379> HKEYS myhash
1) "field2"
2) "field1"
127.0.0.1:6379> HVALS myhash
1) "World"
2) "Hello"
# =====================================================
# hincrby 为哈希表中的字段值加上指定增量值。
# =====================================================
127.0.0.1:6379> hset myhash field 5
(integer) 1
127.0.0.1:6379> HINCRBY myhash field 1
(integer) 6
127.0.0.1:6379> HINCRBY myhash field -1
(integer) 5
127.0.0.1:6379> HINCRBY myhash field -10
(integer) -5
# =====================================================
# hsetnx 为哈希表中不存在的的字段赋值 。
# =====================================================
127.0.0.1:6379> HSETNX myhash field1 "hello" (integer) 1 # 设置成功，返回 1 。
127.0.0.1:6379> HSETNX myhash field1 "world" (integer) 0 # 如果给定字段已经存在，返回 0 。
127.0.0.1:6379> HGET myhash field1
"hello"

```

六、Sorted set

```
# =====================================================
# zadd 将一个或多个成员元素及其分数值加入到有序集当中。
# zrange 返回有序集中，指定区间内的成员
# =====================================================
127.0.0.1:6379> zadd myset 1 "one"
```

```
(integer) 1
127.0.0.1:6379> zadd myset 2 "two" 3 "three"
(integer) 2
127.0.0.1:6379> ZRANGE myset 0 -1
1) "one"
2) "two"
3) "three"
# ================================================
# zrangebyscore 返回有序集合中指定分数区间的成员列表。有序集成员按分数值递增(从小到大) 次序排列。
# ================================================
127.0.0.1:6379> zadd salary 2500 xiaoming
(integer) 1
127.0.0.1:6379> zadd salary 5000 xiaohong
(integer) 1
127.0.0.1:6379> zadd salary 500 daming
(integer) 1
# Inf无穷大量+∞,同样地,-∞可以表示为-Inf。
127.0.0.1:6379> ZRANGEBYSCORE salary -inf
1) "daming"
2) "xiaoming"
3) "xiaohong"
127.0.0.1:6379> ZRANGEBYSCORE salary -inf
1) "daming"
2) "500"
3) "xiaoming"
4) "2500"
5) "xiaohong"
6) "5000"
127.0.0.1:6379> ZREVRANGE salary 0 -1 WITHSCORES # 递减排列
1) "xiaohong"
2) "5000"
3) "xiaoming"
4) "2500"
5) "daming"
6) "500"
127.0.0.1:6379> ZRANGEBYSCORE salary -inf 2500 WITHSCORES # 显示工资 <=2500 的所有成员
1) "daming"
2) "500"
```

```
3) "xiaoming"
4) "2500"
# ====================================================
# zrem 移除有序集中的一个或多个成员
# ====================================================
127.0.0.1:6379> ZRANGE salary 0 -1
1) "daming"
2) "xiaoming"
3) "xiaohong"
127.0.0.1:6379> zrem salary daming
(integer) 1
127.0.0.1:6379> ZRANGE salary 0 -1
1) "xiaoming"
2) "xiaohong"
# ====================================================
# zcard 命令用于计算集合中元素的数量。
# ====================================================
127.0.0.1:6379> zcard salary
(integer) 2
OK
# ====================================================
# zcount 计算有序集合中指定分数区间的成员数量。
# ====================================================
127.0.0.1:6379> zadd myset 1 "hello"
(integer) 1
127.0.0.1:6379> zadd myset 2 "world" 3 "daming"
(integer) 2
127.0.0.1:6379> ZCOUNT myset 1 3
(integer) 3
127.0.0.1:6379> ZCOUNT myset 1 2
(integer) 2
# ====================================================
# zrank 返回有序集中指定成员的排名。其中有序集成员按分数值递增(从小到大)顺序排列。 #
==================================================
127.0.0.1:6379> zadd salary 2500 xiaoming
(integer) 1
127.0.0.1:6379> zadd salary 5000 xiaohong
(integer) 1
127.0.0.1:6379> zadd salary 500 daming
(integer) 1
```

```
127.0.0.1:6379> ZRANGE salary 0 -1 WITHSCORES # 显示所有成员及其 score 值
1) "daming"
2) "500"
3) "xiaoming"
4) "2500"
5) "xiaohong"
6) "5000"
127.0.0.1:6379> zrank salary daming # 显示 daming 的薪水排名，最少 (integer) 0
127.0.0.1:6379> zrank salary xiaohong # 显示 xiaohong 的薪水排名，第三 (integer) 2
# ================================================
# zrevrank 返回有序集中成员的排名。其中有序集成员按分数值递减(从大到小)排序。
# ================================================
127.0.0.1:6379> ZREVRANK salary daming # 大明第三
(integer) 2
127.0.0.1:6379> ZREVRANK salary xiaohong # 小红第一 (integer) 0
```