## 一 redis的简单介绍

### 1.1 redis启动

cmd中运行redis-cli（前提要把redis写入环境变量）

### 1.2 redis的数据库db

redis中有16个db数据库（db(0)~db(15)）,开启redis后默认使用索引为0的db。

|  |
| --- |
| 切换db |
| select index  例如：切换到db1 🡪 select 1 |

如何查看当前使用的是哪个数据库：没有这个命令！！！！

### 1.3 redis中类型

|  |  |
| --- | --- |
| 类型 | 表示 |
| 字符串（数字属于特殊的字符串） | string |
| 哈希（字典） | hash |
| 列表（有序，相当于python中的list） | list |
| 无序集合 | set |
| 有序集合 | zset |

## 二 键操作

### 2.1 普通操作：

|  |  |
| --- | --- |
| 操作 | 命令 |
| 查找 | keys pattern |
| 删除 | del [key] |
| 检查是否存在 | exists [key] |
| 查看键的类型 | type key |

### 2.2 过期时间操作

|  |  |
| --- | --- |
| 操作 | 命令 |
| 设置过期时间（单位秒） | expire key second |
| 设置过期时间（单位毫秒） | pexpire key milliseconds |
| 设置过期时间（时间戳） | expireat key timestamp |
| 移除过期时间 | persist key |
| 返回过期时间（秒） | ttl key |
| 返回过期时间（毫秒） | pttl key |

### 2.3 更改keyname

|  |  |
| --- | --- |
| 命令 | 说明 |
| rename key key\_new | 修改key名字（如果key\_new已经存在,  相当于删除了原来的key\_new键） |
| renamenx key key\_new | 修改key名字（如果key\_new已经存在，  不能完成这次操作） |

### 2.4 键的移动

|  |
| --- |
| move key db  例如：  move key1 1 🡪把key1移动到db1中 |

### 2.5 补充命令

|  |  |
| --- | --- |
| 操作 | 命令 |
| 随机获取一个key | randomkey |

## 三 string

### 3.1 字符串与数字

如果一个字符串对象都是数字，没有其他英文字母，例如”1”，则认为这个对象是一个数字，可以做加减法

### 3.2设置值

|  |  |
| --- | --- |
| 操作 | 命令 |
| 设置值 | set key value |
| 设置多个值 | mset [key value] |
| 设置值和过期时间（单位秒） | setex key second value |
| 设置值和过期时间（单位毫秒） | psetex key milliseconds value |
| 只有key不存在时，才被赋值（单个） | setnx key value |
| 只有所有key都不存在时，才被赋值（多个） | msetnx [key value] |
| 拼接value，如果key不存在，就创建 | append key value |

### 3.3获取值

|  |  |
| --- | --- |
| 操作 | 命令 |
| 获取一个值 | get key |
| 获取多个值 | mget [key] |
| 获取一个值得切片 | getrange key start stop |
| 取值后再赋值 | getset key value |
| 获取value的长度 | strlen key |

### 3.4运算

|  |  |
| --- | --- |
| 操作 | 命令 |
| value += 1（value必须是整数） | incr key |
| value += 一个整数（value必须是整数） | incrby key increment |
| value += 小数 | incrbyfloat key increment |
| value -= 1 （value必须是整数） | decr key |
| value -= 一个整数（value必须是整数） | decr key increment |

## 四 hash

### 4.1 hash的特点

类似python中的字典

key相当于字典对象

field相当于键

value为值

例如：key = { field1 : value1 , field2 : value2 }

### 4.2设置值

|  |  |
| --- | --- |
| 操作 | 命令 |
| 设置一个键值对（如果存在，则替换） | hset key field value |
| 设置多个键值对（如果存在，则替换） | hmset key [field value] |
| 设置一个键值对（仅当不存在，才能成功） | hsetnx key field value |
| 删除field | hdel key field |

### 4.3获取值

|  |  |
| --- | --- |
| 操作 | 命令 |
| 获取一个value | hget key field |
| 获取多个value | hmget key [field] |
| 获取key中所有的field和value | hgetall key |
| 获取key中所有的field | hkeys key |
| 获取key中所有的value | hvals key |
| 判断key中是否存在否个field | hexists key field |
| 获取key中field的个数 | hlen key |
| 获取某个value的长度 | hstrlen key field |

### 4.4操作

|  |  |
| --- | --- |
| 操作 | 命令 |
| value += 一个整数 | hincrby key increment |
| value += 小数 | hincrbyfloat key increment |

## 五 list

### 5.1 list的特点

redis总的list和python中的list几乎一致

### 5.2 设置值

|  |  |
| --- | --- |
| 操作 | 命令 |
| 最左边插入 | lpush key [value] |
| 最右边插入 | rpush key [value] |
| 最左边插入（key存在才能操作成功） | lpushx key value |
| 最右边插入（key存在才能操作成功） | rpushx key value |
| 根据索引改变 | lset key index value |

注：

1.

lpush和rpush：插入前如果没有这个key，则会创建一个key

lpushx和rpushx：只能对已存在的key进行插入

2.

lpush key value1 value2 value3

* lpush key value1 ，lpush key value2 ， lpush key value3
* 最左面是：value3 value2 value1 。。。

### 5.3 获取值

|  |  |
| --- | --- |
| 操作 | 命令 |
| 最左边出栈 | lpop key |
| 最右边出栈 | rpop key |
| 最左边出栈，最大等待时间timeout | blpop key timeout |
| 最右边出栈，最大等待时间timeout | brpop key timeout |
| 根据index获取value（不改变栈） | lindex key index |
| 得到key的长度 | llen key |
| 显示范围的value | lrange key start stop |

### 5.4 扩展命令

|  |  |
| --- | --- |
| 操作 | 命令 |
| 从key1队列中rpop然后lpush进key2 | rpoplpush key1 key2 |
| 在value1前或后插入value2 | linsert key before|after value1 value2 |

## 六 set

### 6.1 set的特点

无序，不能有重复的member

### 6.2 设置值

|  |  |
| --- | --- |
| 操作 | 命令 |
| 向key中添加一个或多个member（如果key不存在则创建） | sadd key [member] |
| 删除指定的members | srem key [member] |

### 6.3 获取

|  |  |
| --- | --- |
| 操作 | 命令 |
| 查看key中所有的member | smembers key |
| 随机出栈count个member | spop key count |
| 随机查看count个member | srandmember key count |
| 查看key中member的个数 | scard key |
| 判断member是否在key中 | sismember key member |

### 6.4 操作

|  |  |
| --- | --- |
| 操作 | 命令 |
| 交集 | sinter [key] |
| 并集 | sunion [key] |
| 差集 | sdiff [key] |

## 七、zset

### 7.1说明

1.有序集合

2.一个key中不会存在两个相同的member

3.每个member对应一个唯一的score权重

4.相同权重的member，按先插入在左侧，后插入在右侧排列

### 7.2 设置值

#### 7.2.1 命令表

|  |  |
| --- | --- |
| 操作 | 命令 |
| 添加member和score | zadd key [score member] |
| 删除member | zrem key [member] |
| 根据排序删除 | zremrangebyrank key start stop |
| 根据score范围删除 | zremrangebyscore key min max |

注：

start和stop是整个key从左到右排列member对应的索引

min和max是score的范围

#### 7.2.2 排序问题

zadd k1 1 a 1 b 3 e 4 f

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| member | a | b | e | f |
| index | 0 | 1 | 2 | 3 |
| score | 1 | 1 | 3 | 4 |

zadd k1 1 c 2 d

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| member | a | b | c | d | e | f |
| index | 0 | 1 | 2 | 3 | 4 | 5 |
| score | 1 | 1 | 1 | 2 | 3 | 4 |

zadd k1 2 a

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| member | b | c | d | a | e | f |
| index | 0 | 1 | 2 | 3 | 4 | 5 |
| score | 1 | 1 | 2 | 2 | 3 | 4 |

### 7.3 获取

|  |  |
| --- | --- |
| 操作 | 命令 |
| 根据index范围返回member（正序） | zrange key start stop |
| 根据index范围返回member（反序） | zrevrange key start stop |
| 根据score范围返回member（正序） | zrangebyscore key min max |
| 根据score范围返回member（反序） | zrevrangebyscore key min max |
| 根据score范围返回member数目 | zcount key min max |
| 返回member对应的score | zscore key member |
| 返回member的index | zrank key member |
| 返回member反方向的index | zrevrank key member |

### 7.4 操作

|  |  |
| --- | --- |
| 操作 | 命令 |
| member += increment（整数小数都可以） | zincrby key increment member |

## 八、python操作redis

### 8.1 安装redis

pip install redis

### 8.2 建立连接的方式

#### 8.2.1 StrictRedis

|  |
| --- |
| import redis  conn = redis.StrictRedis(  host = ‘localhost’,  port = 6379,  password = ‘’  )  ret = conn.execute\_command(‘redis语句’)  print(ret) |

#### 8.2.2 Redis

|  |
| --- |
| import redis  conn = redis.Redis(  host = ‘localhost’,  port = 3306,  password = ‘’,  )  ret = conn.execute\_command(‘redis语句’)  print(ret) |

#### 8.2.3 ConnectionPool

|  |
| --- |
| import redis  # 建立连接池  redis\_datasource = redis.ConnectionPool(  host = ‘localhost’,  port = 3306,  password = ‘’,  )  # 二选一  conn = redis.Redis(connection\_pool = redis\_datasource)  # 二选一  conn = redis.StrictRedis(connection\_pool = redis\_datasource)  ret = conn.execute\_command(‘redis语句’)  print(ret) |

### 8.3 管道pipeline

避免数据库语句零散执行，一起发送大量的redis语句

|  |
| --- |
| import redis  # 生成管道  pipe = conn.pipeline()  # 管道中的命令不会立刻发送到redis中 pipe.execute\_command(‘…….’)  pipe.execute\_command(‘…….’)  pipe.execute\_command(‘…….’)  … …  # 提交管道命令  pipe.execute() |

注意：ret = pipe.execute\_command(‘get key’) 🡪 返回实一个地址，而不是值