Redis Cheat Sheet

Objects in Redis

- must have a key
- keys are colon separated
- value can be hash, set, list, sortedset, string
- can have an expiry

Hashes:

- Use to store "objects" from object oriented programming

```
"blogs:234": {
        "id": 234,
        "title": "Introduction to Redis",
        "views": 1024
    }
```

- hmset blogs:234 id 234 title "Introduction to Redis" views 1024
- hgetall blogs:234
- hincrby blogs:234 views 1

Sets:

- Only unique items. No duplicates
- Elements are not ordered
- Can check if element exists very quickly
- Use case: Count unique items
- Use case: Check if item exists
- Use case: Perform logical UNION / INTERSECTION operations
- Two ways to represent same information, choose based on use case set of integers consumes less memory, is very efficient

```
"blogs:234:tags" ==> set("aws", "redis", "cache")
    v/s
"tags:aws:blogs" ==> set(234, 532, 332)
```

Lists:

- Duplicates allowed.
- Very well defined order of elements
- Expensive to check if element already exists in list

- Operations near head / tail very fast.
- Operations near centre elements slower
- Can use as a queue or a stack
- Operations can block if no elements are available, useful
- Easy to restrict length using ltrim command
- Use case: Recent items, recent logs
- Use case: Queue to maintain tasks

Sorted Sets

- Like a set, only allows unique items
- Every element can have a numeric score
- Score can represent anything - sales, count of orders, points in a game etc.
- Elements are always maintained in order of score
- If score of two elements are same, elements are sorted in lexicographical order - the way you see in oxford's dictionary
- Use case: Leader boards - Top 10 players, Products with least sales,
- Use case: Autocomplete API

Strings:

- Strings are binary, and can store integers, floats, text
- You can treat the string as a very large bitmap
- You can split the string into smaller fields, and then set/get each field using the bitfield command