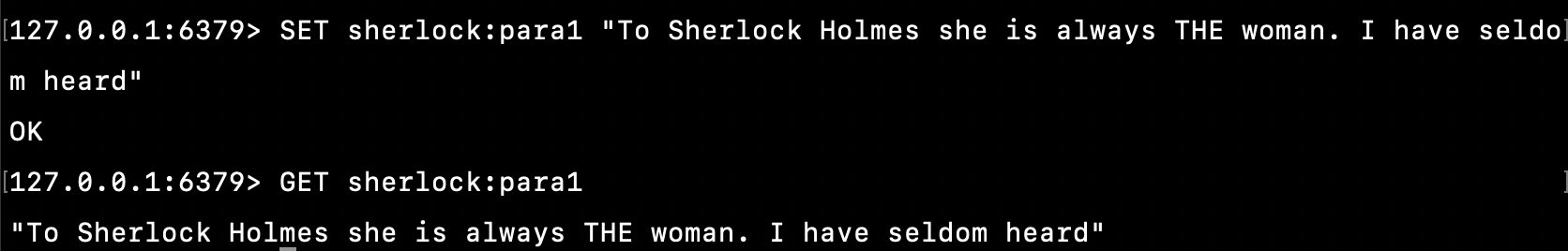
Tutorial 3 Exercise – Redis 1

Excercises

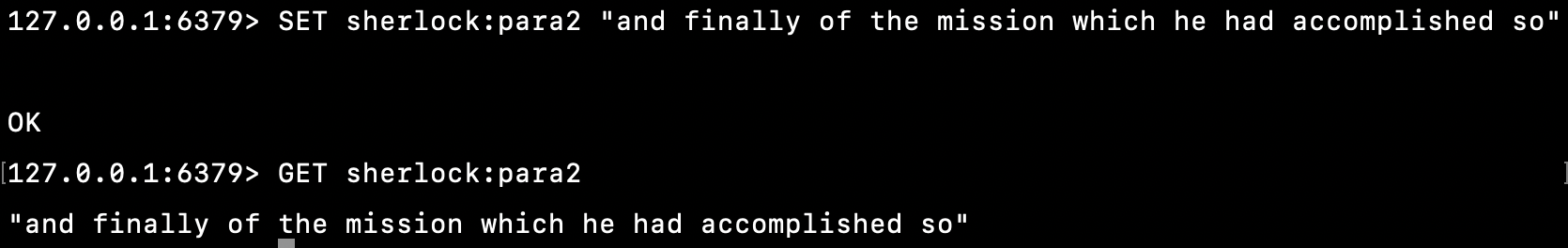
# STRING

1. Set the key "sherlock:para1" to the first sentence of the "Adventures of Sherlock Holmes", the paragraph that begins 'To Sherlock Holmes she is always THE woman.' (The text is available from <http://www.gutenberg.org/cache/epub/1661/pg1661.txt>)

Confirm this, by GETting the paragraph back out again



1. Continue setting keys of the form "sherlock:paraXXXX", where XXXX is the number of the paragraph, based on this text. Make the assumption that a blank line indicates the beginning of a new paragraph. Retain the code you use here, as you will be able to adapt it for later exercises.



## Useful commands: GETSET

## Exercise

1. Play around with this function to get used to how it functions. What does it respond if the key does not exist?

GETSET <http://redis.io/commands/getset>

* Create a new key with value specified if key is not found.

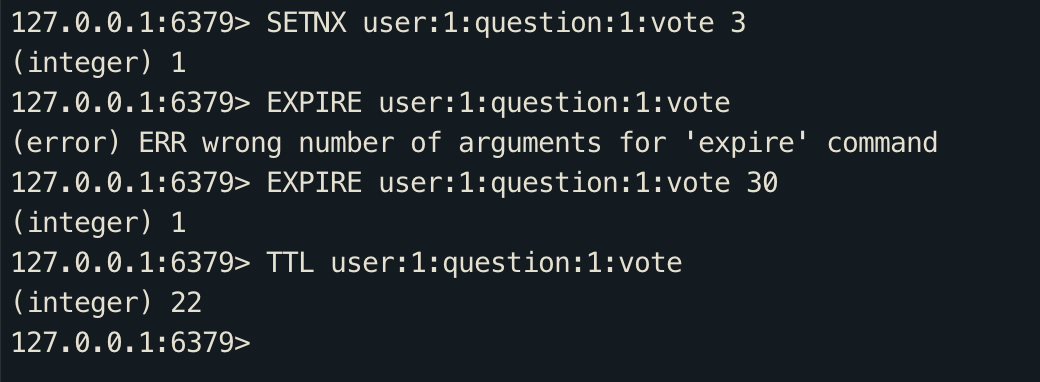
Handling expire key

Excercises

1. What happens if you set a timeout for a non-existing key? Or attempt to persist one?

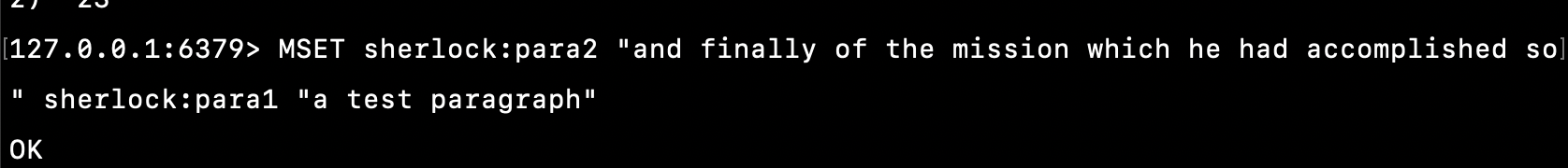
Cannot set, redis return 0

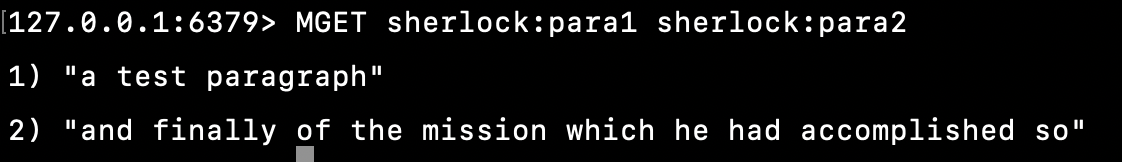
1. Consider (and if desired, build) a system that makes sure that a user of a web app can only vote on a poll once every 30 seconds. During 30 seconds, user cannot vote the same question again. If 30 seconds passed, user can re-vote.



Batch set and get

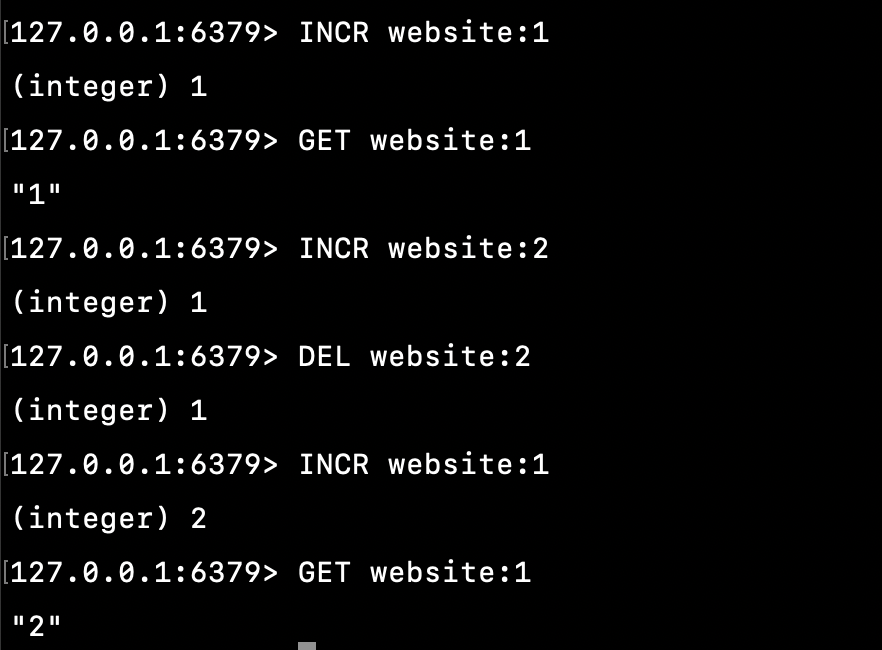
1. Adapt the Sherlock paragraph setting code you wrote previously (in section 1 of this) to set the paragraphs in configurable batches, using MSET.





Commands acting on number

1. Imagine it's 1993 and a client wants to have a click counter on their website, with a counter per page. How would you build it using Redis?



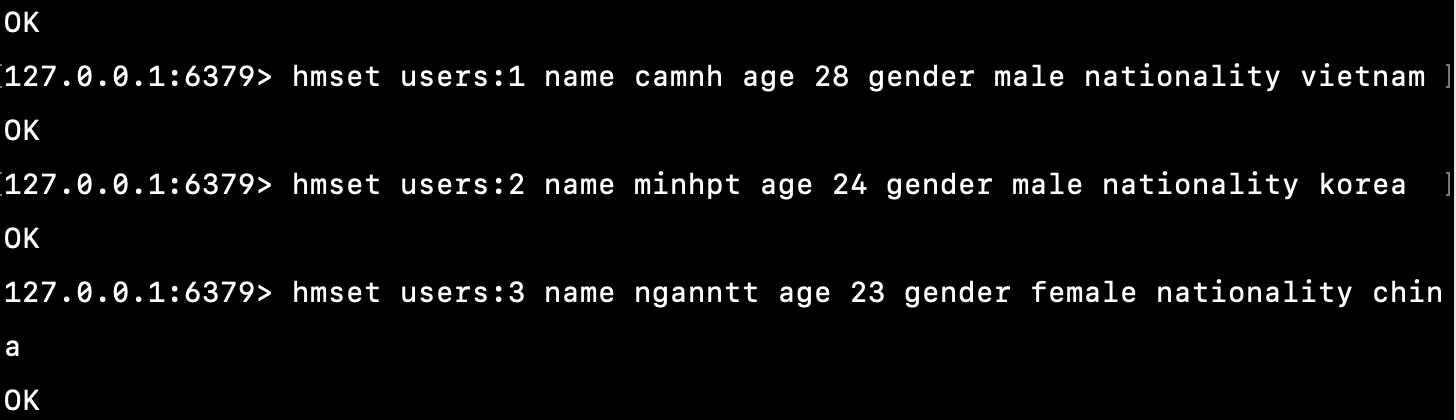
# Hash

1. Store sample records in users in each hash like this below

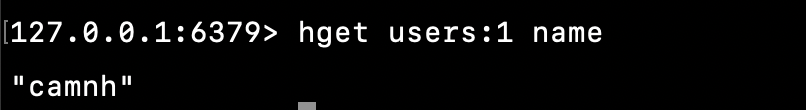
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| id | name | Age | Gender | Nationality |
| 1 | camnh | 28 | Male | vietnam |
| 2 | Minhpt | 24 | Male | korea |
| 3 | nganntt | 23 | Female | china |

Based on table above, enter commands performs listed requirements

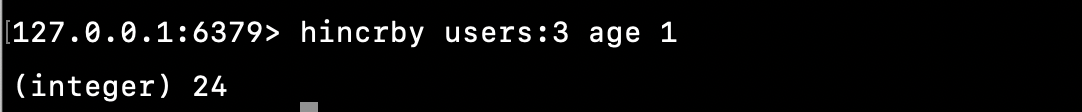
* Print all data in hash (no need to specify keys to show)



* Print name of hash key of record id = 1



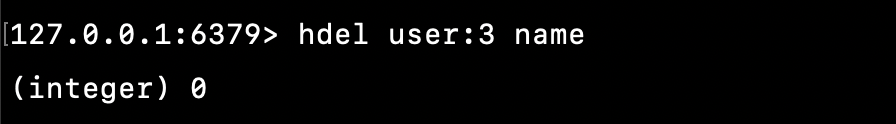
* Increase age of record Id = 3 from 23 to 24.



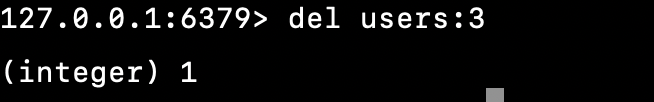
* Get all keys in users table



* Unset person name in hash where id = 3



* Delete person where id = 3

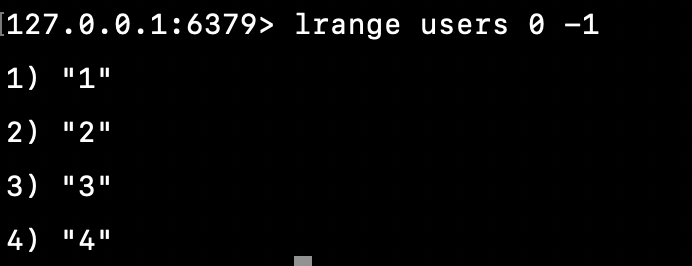


# List

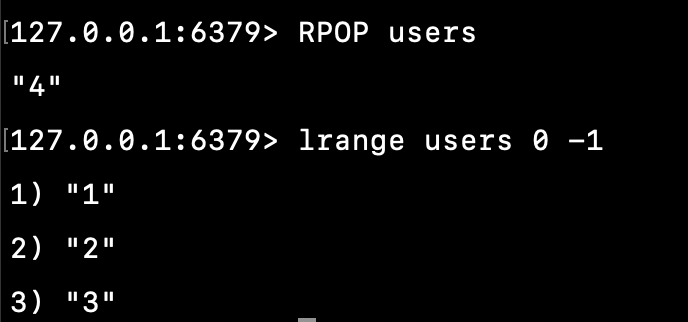
* 1. Create a list contain 3 numbers 1,2,3



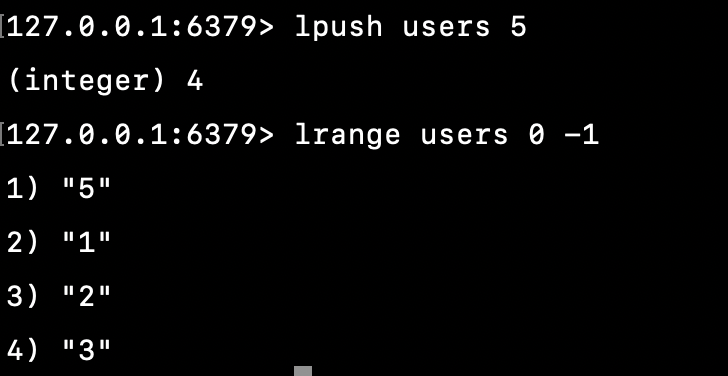
* Show number of ids stored in list (expected: 1 2 3 4)



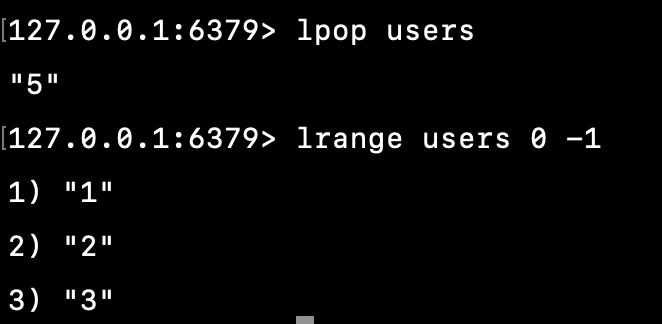
* Remove 4 from list



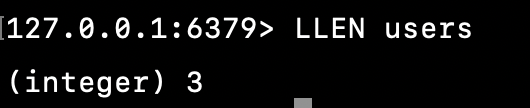
* Add id = 5 into head of the list (expected: 5 1 2 3)



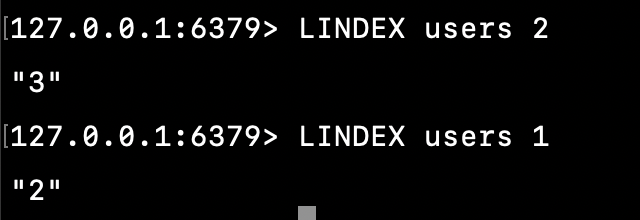
* Pop out id = 5 out of list



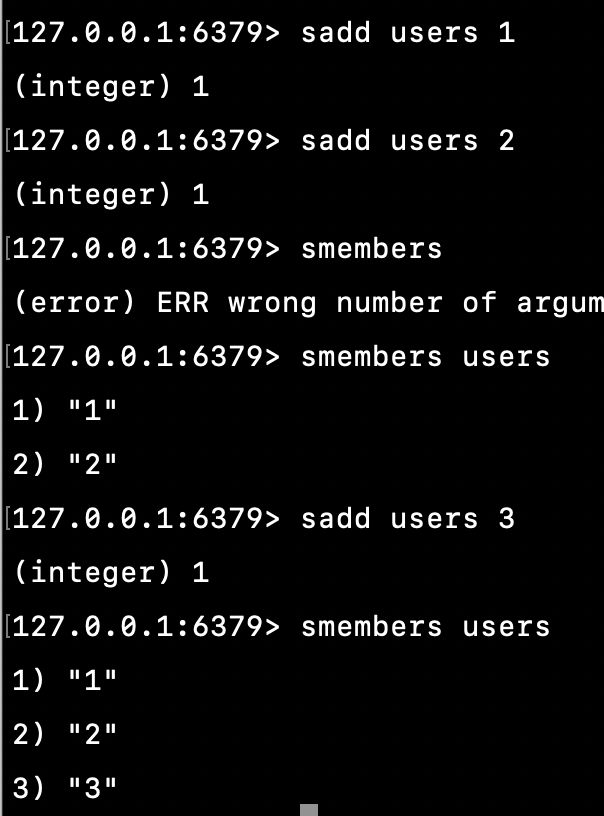
* Get number of items in list



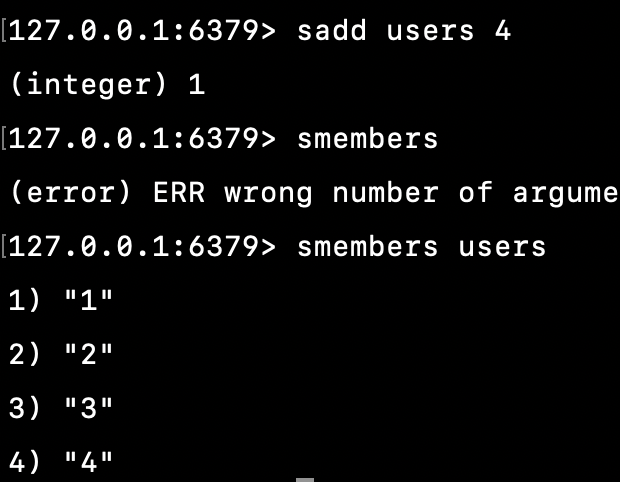
* Get element name with index = 2 (expected 1)



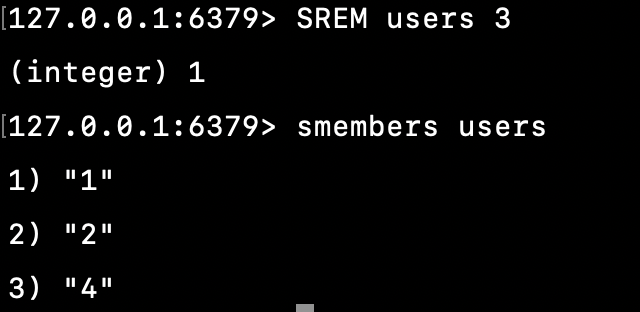
# Set



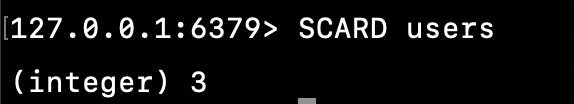
* Add new id = 4 to set created above + Show number of ids stored in list (expected: 1 2 3 4)



* Remove set item 3 from list created above (expected: 1 2 4)



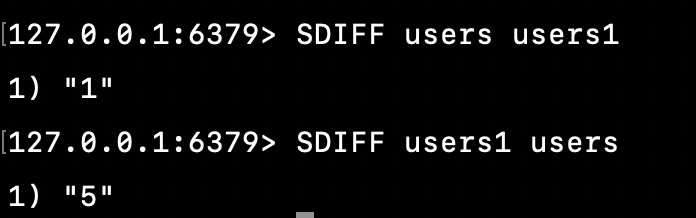
* Get number of items in set



* Now create a new set called “users1” which include 3 elements 2, 4, 5. Write Redis query to get common values between “users” and “users1” (expected 2,4)



* Use set above, show the difference between 2 sets **users** and **users1** respectively
* Use set above, show the difference between 2 sets **users1** and **users** respectively

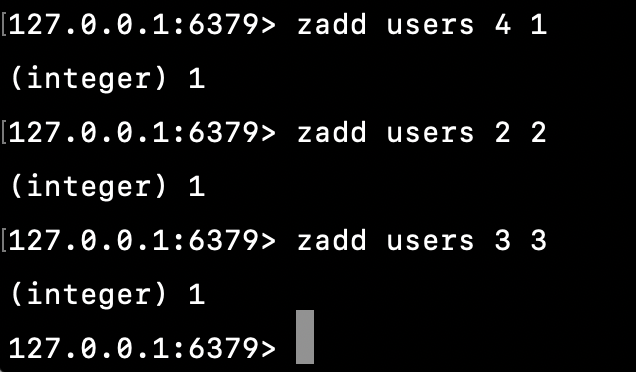


# Zset

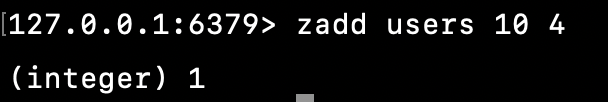
Store sample records in users in each set like this below

|  |  |
| --- | --- |
| id | score |
| 1 | 4 |
| 2 | 2 |
| 3 | 3 |

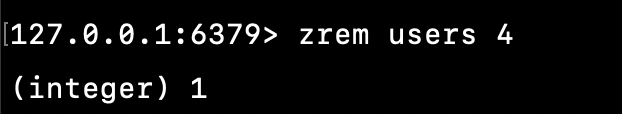
* Add data to zset



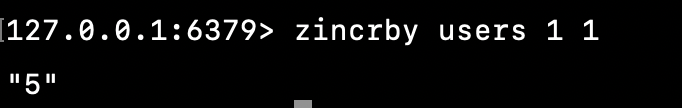
* Add new item id = 4 and score 10



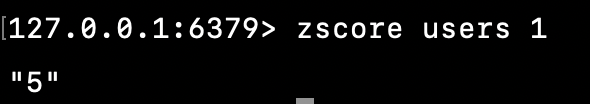
* Remove that item with id = 4



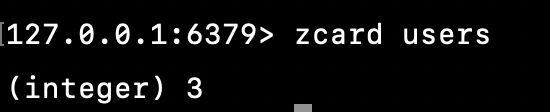
* Increase score of item with id = 1 by 1



* Get score of item with id = 1

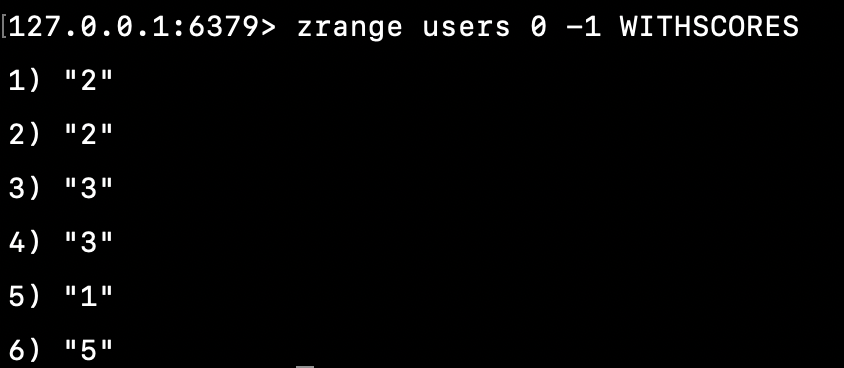


* Show numbers of item in zset (expected : 3)



* Print the list of items in normal and reverse order

A->Z order



Z->A order

