Redis TP:

1. BASIC INSTALLATION AND USAGE

A. To install Redis via the official docker image

```
docker run –name my_redis -d redis
```

B. To connect the default client via command line

```
docker run -it –rm –link my-redis:redis redis redis-cli -h redis -p 6379
```

C. If the container already exists:

In Windows command line: docker start container_name

D. To open the default client via Ubuntu command line

In Ubuntu Command line: redis-cli

E. Basic commands on Redis

1. To define a pair (key, value)

➔ set key value

2. To print the value associated with a key

➔ get key (returns the value if exists, otherwise “nil”)

3. To check whether a key exists or not (exists =1, otherwise 0)

    - Exists key

4. To increment/decrement/delete by 1 the value of a key

    - set key value
    - incr key
    - decr key
    - del key

5. To increment/decrement by more x quantity the value of a key

    - incrby key x
    - decrby key x

6. To set and check the expiration time on a key

    - set key value
    - expire key time
    - ttl key (-2 means that the key does not exist anymore, -1 means that the key will never expire – normal creation with "set")
    - set key value ex time (compact way)
    - persist key (to cancel the time to live and making it permanent)

7. To create a list (by adding elements)/print content/len

- rpush

- rpush family “Pablo” (-> family = [“Pablo”])
- rpush family “Mamá” (-> family = [“Pablo”, “Mamá”])
- rpush family “Carlos” “Marta” “Laura” “Esteban” (-> family = [“Pablo”, “Mamá”, “Carlos”,“Marta”,“Laura”, “Esteban”])

- lpush

  - lpush family “Sofi” (-> family = [“Sofi”, “Pablo”, “Mamá”])
  - lpush family “Victor” “Rosa”

  (-> family = [“Victor”, “Rosa”, “Pablo”, “Mamá”, “Carlos”,“Marta”,“Laura”, “Esteban”])

- lrange family start_index end_index

  - lrange family 0 -1 (-> prints the complete list)
  - Lrange family 0 1 (-> prints the first (0) and second (1) element)

- llen list

  - llen family (-> 8)

8. Sets: Adding/removing/printing/testing membership/operations between sets

   - sadd set_name value (adding only one)
   - sadd set_name value1 value2 (you can add many at the same time)
   - sadd set_name value -> returns 0 if the element already belongs to the set and 1 if not

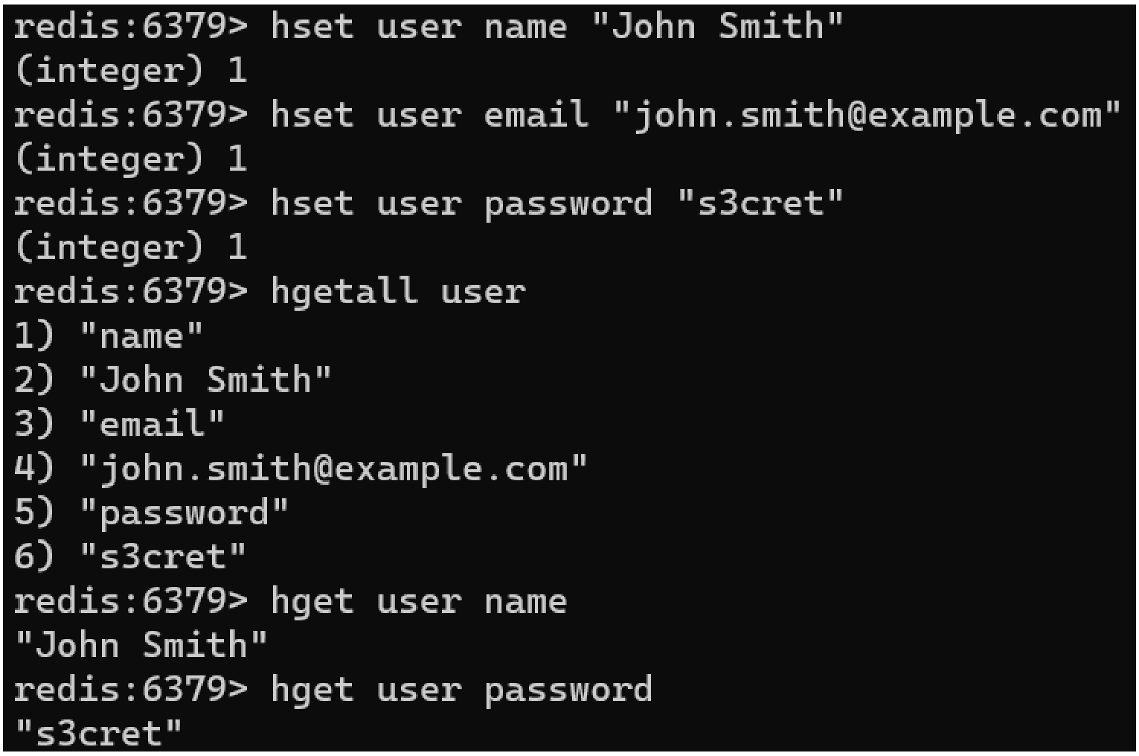
- ➔ srem set_name value (removes from the set set_name the chosen value, returns 1 if the member was and 0 if it wasn't in the set)
- ➔ sismember set_name value (tests if the given value is in the set returning 1 if it is, 0 if not)
- ➔ smembers set_name -> prints the elements from the set
- ➔ sunion set_name1 set_name2 … set_nameN -> prints the elements of all united sets
- ➔ sunionstore new_set set_name1 set_name2 … -> creates a new set called new_set and stores the union of the sets set_name1, set_name2, ….
- ➔ spop set_name number -> removes randomly the number of elements chosen & prints them, number > 0
- ➔ srandmember set_name number -> prints the number of elements chosen without removing them from the set, if number < 0, it returns the same number as in positive but might consider returning repeating numbers

9. Ordered sets

- ➔ zadd ordered_set_name score value -> the score is a number, and the set will be ordered according to it
- ➔ zadd ordered_set_name score1 value1 score2 value2 -> to add multiple members
- ➔ zrange ordered_set_name start_index end_index -> prints content

10. Hashes:

- hset hash_set_name field1 value1 field2 value2 -> to add values with a field associated
- hset hash_set_name field1 value2 -> changes the value for the field (returns 0, if returns 1 means it didn’t exist the field with the value, you just created it)
- hgetall hash_set_name -> prints all content
- hget hash_set_name field -> prints the associated value



```
redis:6379> hset user name "John Smith"
(integer) 1
redis:6379> hset user email "john.smith@example.com"
(integer) 1
redis:6379> hset user password "s3cret"
(integer) 1
redis:6379> hgetall user
1) "name"
2) "John Smith"
3) "email"
4) "john.smith@example.com"
5) "password"
6) "s3cret"
redis:6379> hget user name
"John Smith"
redis:6379> hget user password
"s3cret"
```

➔ hincrby hash_set_name field quantity -> increases the field by the quantity but the field must be an integer otherwise error + if you increase a non-existing field by a certain quantity, it gets created with the quantity

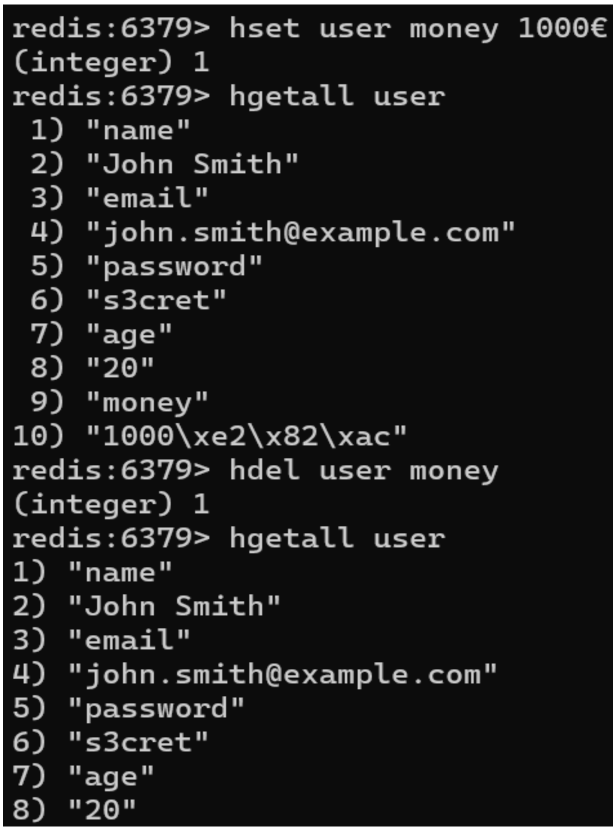
➔ hdel hash_set_name field -> deletes the field + value

F. Multiple Key Operators (Summary)

➔ sunion, sadd, zadd

2. Homework Application: URL Shortener and Decoder

1. We will need to write a Python function to create the URL Shortener and Decoder, so we need to install the library redis-py on Python (pip install redis-py) and then simply import redis to use it.



```
redis:6379> hset user money 1000€
(integer) 1
redis:6379> hgetall user
 1) "name"
 2) "John Smith"
 3) "email"
 4) "john.smith@example.com"
 5) "password"
 6) "s3cret"
 7) "age"
 8) "20"
 9) "money"
10) "1000\xe2\x82\xac"
redis:6379> hdel user money
(integer) 1
redis:6379> hgetall user
1) "name"
2) "John Smith"
3) "email"
4) "john.smith@example.com"
5) "password"
6) "s3cret"
7) "age"
8) "20"
```

2. We need to download WSL (Windows Subsystem for Linux) by running:

- wsl –install

Then, you will need to create a username and password for the Ubuntu account (wrote in Keeper). When finished, you need to update the tool manager of repositories to check for newer versions on libraries by running:

- sudo apt update

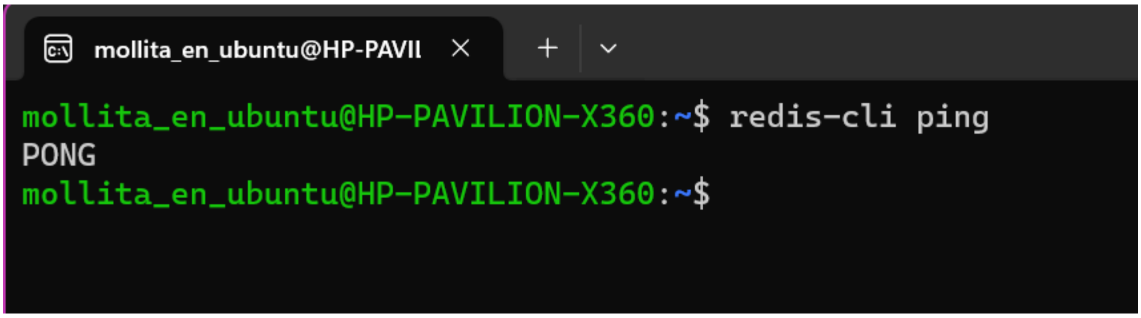
Then install redis-server:

- sudo apt install redis-server

To check that redis-cli which is the client in command line for Redis is installed (it should have been downloaded because it's within redis-server), open the Ubuntu program, and simply run:

- redis-cli ping -> should return PONG

Now that everything works, you can start writing Python files and running them usually on the Integrated development environment (IDE), to do so, you will need to leave opened the Ubuntu program, so the connection can be reached.



```
mollita_en_ubuntu@HP-PAVILION-X360:~$ redis-cli ping
PONG
mollita_en_ubuntu@HP-PAVILION-X360:~$
```