Quickstart

```python
# ---------- QUICKSTART ----------
    # redis leverages on in-memory data storage (computer's RAM) that is not persistent across machine system reboots by design, but redis data can be made persistent using the following
        # rdb snapshots
        # append only files
    # redis is mainly used for caching, session management, leaderboards, messaging, rate limiting, counting, and dealing with both sorted and unsorted sets
    # redis can be interacted with similar commands in the CLI and using a GUI client called RedisInsight
    # however, redis is most often used via import as a library in Python

import redis

# ----- GENERAL -----
    # redis.Redis(host='<hostname>', port=<portNumber>) => connects to the specified redis server
    # .set(<valueName>, <value>) => assigns a value to the specified value name in memory
    # .get(<valueName>) => retrieves a value from memory based on the specified value name
    # .decode() => used to decode the bytes to a readable string
    # .hset(<hashName>, mapping=<hashValue>) => creates a new hash under the specified hash name, where the value is the hash
    # .hget(<hashName>, <key>) => retrieves a value from the specified hash using the given key
    # .rpsuh(<listName>, <comma-delimited value(s) to be appended>) => appends any number of comma-delimited values to the specified list
    # .lindex(<index>, <listName>) => retrives a value from the specified list based on the given index
    # .sadd(<setName>, *<setValue>) => adds/assigns elements to the specified set name
    # * => aestricks operator unpacks the List into multiple individual arguments as required by the .sadd() method
    # .sismember(<setName>, <queriedMember>) => returns a boolean value based on whether the queried element is a member of the specified set
    # .zadd(<orderedSetName>, <orderedSetValue>) => adds/assigns elements to the specified ordered set name
    # .zrangebyscore(<orderedSetName>, min=<minimumValue>, max=<maximumValue>) => retrieves elements within the specified score ranges from the specified ordered set

r = redis.Redis(host='localhost', port=6379)

r.set('name', 'Alice')
name = r.get('name')
print(name.decode()) # redis Strings are stored as sequences of bytes and have to be decoded

user_data = {'name': 'Bob', 'age': 30, 'city': 'New York'}
r.hset('user:1', mapping=user_data)
age = r.hget('user:1', 'age')
print(age.decode())

r.rpush('shopping_list', 'apples', 'bread', 'milk')
first_item = r.lindex(0, 'shopping_list')
print(first_item.decode())

fruits = {'apple', 'banana', 'orange'}
r.sadd('fruits', *fruits)
is_member = r.sismember('fruits', 'mango')
print(is_member)

leaderboard = {('Alice', 100), ('Bob', 80), ('Charlie', 90)}
r.zadd('scores', leaderboard)
top_3 = r.zrangebyscore('scores', min=85, max=100)
print(top_3)
```