1. Redis-py

- Official Python client for Redis, maintained by Redis
- GitHub: redis/redis-py
- Install using:
  ```
  pip install redis
  ```

2. Connecting to Redis

Use this template to connect to your Redis server:

```
import redis

redis_client = redis.Redis(
    host='localhost',      # or '127.0.0.1'
    port=6379,             # default port
    db=2,                  # Redis DB (0-15)
    decode_responses=True  # decode bytes to strings
)
```

- `decode_responses=True` ensures output is in string format instead of bytes.

3. Redis Commands Documentation

- Full command list: Redis Commands
- Redis-py docs: Redis-py Docs
- Use filters to explore commands by data type (e.g., list, hash, set).

4. String Commands

```
# Set and Get
r.set('clickCount:/abc', 0)
val = r.get('clickCount:/abc')

# Increment
r.incr('clickCount:/abc')
print(f'click count = {r.get("clickCount:/abc")}')

# Multiple values
redis_client.mset({'key1': 'val1', 'key2': 'val2', 'key3': 'val3'})
print(redis_client.mget('key1', 'key2', 'key3'))
# Output: ['val1', 'val2', 'val3']
```

Other Common String Commands:

- `set()`, `mset()`, `setex()`, `setnx()`
- `get()`, `mget()`, `getdel()`, `getex()`
- `incr()`, `decr()`, `incrby()`, `decrby()`
- `strlen()`, `append()`

5. List Commands

```
# Right push values into a list
redis_client.rpush('names', 'mark', 'sam', 'nick')
print(redis_client.lrange('names', 0, -1))  # ['mark', 'sam', 'nick']
```

Other Common List Commands:

- `lpush()`, `lpop()`, `rpush()`, `rpop()`
- `lrange()`, `llen()`, `lset()`, `lrem()`, `lpos()`
- Advanced: move between lists, pop from multiple lists

6. Hash Commands

```
redis_client.hset('user-session:123', mapping={
    'first': 'Sam',
    'last': 'Uelle',
    'company': 'Redis',
    'age': 30
})

print(redis_client.hgetall('user-session:123'))
# {'first': 'Sam', 'last': 'Uelle', 'company': 'Redis', 'age': '30'}
```

Other Common Hash Commands:

- `hget()`, `hgetall()`, `hkeys()`
- `hdel()`, `hexists()`, `hlen()`, `hstrlen()`

7. Redis Pipelines

- Use pipelines to **bundle multiple Redis calls** together, reducing network overhead:

```
r = redis.Redis(decode_responses=True)
pipe = r.pipeline()

for i in range(5):
    pipe.set(f"seat:{i}", f"#{i}")

pipe.execute()  # [True, True, True, True, True]

# Chain commands
pipe = r.pipeline()
result = pipe.get("seat:0").get("seat:3").get("seat:4").execute()
print(result)  # ['#0', '#3', '#4']
```

8. Redis in ML/DS Context

- Redis can be used for:
    - **Feature stores** for fast ML model access
    - **Real-time data pipelines**
    - **Low-latency serving of features and session data**
- Helpful in MLOps setups for caching, monitoring, and model interaction