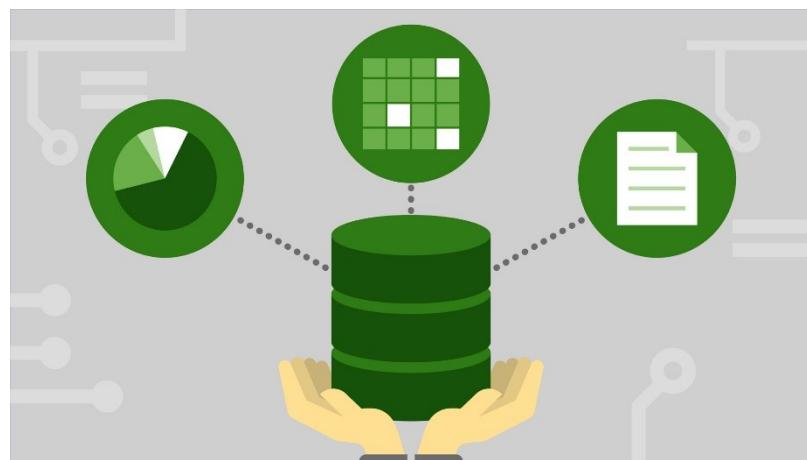


به نام خدا



دانشگاه تهران
پردیس دانشکده‌های فنی
دانشکده برق و کامپیوتر



آزمایشگاه پایگاه داده

ستور کار شماره ۸

آتیه آرمین

۸۱۰۱۹۷۶۴۸

مهرماه ۱۴۰۰

گزارش دستورکار انجام شده

* در این آزمایش در هر عکس هم تکه کد و هم خروجی آن نشان داده شده است.

- 1 - تکه کد زیر در واقع معادل MSET و GET در Redis است که این اجازه را میدهد که به ازای چند key یک value میتوان get معادل آن key را خواند.

```
1 import redis
2 r = redis.Redis()
3 r.mset({"Croatia": "Zagreb", "Bahamas": "Nassau"})
4 print(r.get("Bahamas"))
```

- 2 - برای استفاده از مثلاً تایپ date باید آن را به string تبدیل کنیم. در تکه کد زیر تعدادی visitor داریم که میخواهیم آن ها را به عنوان value تاریخ امروز اضافه کنیم. برای اینکار باید تاریخ امروز را ابتدا به str تبدیل کنیم سپس با استفاده از sadd دستور را پیاده‌سازی میکنیم.

```
import datetime
today = datetime.date.today()
visitors = {"dan", "jon", "alex"}
stoday = today.isoformat() # Python 3.7+, or use str(today)
print('today date:', stoday)

r.sadd(stoday, *visitors) # sadd: set-add

print(r.smembers(stoday))

print(r.scard(today.isoformat()))
```

- 3 - در این تکه کد ابتدا یک دیکشنری از ۳ کلاه که شماره آنها به صورت رندوم انتخاب می‌شود درست میکنیم. هر کلاه اطلاعاتی مانند قیمت، رنگ و استایل در خود دارد. حال با دستور hmset این اطلاعات را به ازای هر کلاه در Redis نگه داری میکنیم.

```
Users > atieharmin > Desktop > untitled.py > ...
1 import redis
2 import random
3
4 random.seed(444)
5 hats = {f"hat:{random.getrandbits(32)}": i for i in (
6     {
7         "color": "black",
8         "price": 49.99,
9         "style": "fitted",
10        "quantity": 1000,
11        "npurchased": 0,
12    },
13    {
14        "color": "maroon",
15        "price": 59.99,
16        "style": "hipster",
17        "quantity": 500,
18        "npurchased": 0,
19    },
20    {
21        "color": "green",
22        "price": 99.99,
23        "style": "baseball",
24        "quantity": 200,
25        "npurchased": 0,
26    })
27 }
28
29 r = redis.Redis(db=1)
30 with r.pipeline() as pipe:
31     for h_id, hat in hats.items():
32         print(pipe.hmset(h_id, hat))
33     pipe.execute()
34
35 print(r.bgsave())
```

۴- برای آنکه اطمینان حاصل کنیم که این اطلاعات در دیتابیس ذخیره شده است با دستور `hgetall` تمام اطلاعات کلاه با شماره خاص را نشان می‌دهیم. همینطور با دستور `keys()` شماره ۳ کلاه ذخیره شده را میتوانیم مشاهده کنیم.

```

import redis
r = redis.Redis(db=1)
print(r.hgetall("hat:56854717"))
print(r.keys())

```

```

Desktop — zsh — 80x24
(base) atieharmin@Atiehs-MacBook-Pro Desktop % python untitled.py
[{"b'color': b'green', b'price': b'99.99', b'style': b'baseball', b'quantity': b'2
00', b'nurchased': b'0'}]
[b'hat:1236154736', b'hat:56854717', b'hat:1326692461']
(base) atieharmin@Atiehs-MacBook-Pro Desktop %

```

۵- در تکه کد زیر میخواهیم اگر کاربری روی `purchase` کلیک کرد، اگر آن کلاه موجود بود `npurchases` آن را یکی زیاد کرده و `quantity` را یکی کم کنیم. این تغییر ها را با دستور `hincrby` میتوان انجام داد.

```

Users > atieharmin > Desktop > ✎ untitled.py > ...
1 import redis
2
3 r = redis.Redis(db=1)
4 print(r.hincrby("hat:56854717", "quantity", -1))
5
6 print(r.hget("hat:56854717", "quantity"))
7
8 print(r.hincrby("hat:56854717", "nurchased", 1))

```

```

Desktop — zsh — 80x24
(base) atieharmin@Atiehs-MacBook-Pro Desktop % python untitled.py
199
b'199'
1
(base) atieharmin@Atiehs-MacBook-Pro Desktop %

```

۶- در تکه کد زیر دستور بالا به صورت یک تابع پیاده‌سازی شده است. نکته مهم این که با تابع `watch` ما این آیتم را برای هرگونه تغییری در مقدار آن نگاه می‌کنیم. اگر در این مدت Redis یک اروری بدهد، یک `watch error` میدهد که آن را `catch` میکنیم. در این تکه کد ۳ بار یک کلاه را با تابع مذکور خریداری کرده و سپس اطلاعات آن را چاپ کرده‌ایم تا کارایی این تابع را بسنجیم. تعداد موجود این کلاه ۱۹۹ بود زیرا در دستور قبلی یکی از آن کم کردیم. همچنان مقدار `nurchased` هم ۱ بود.

```

Users > atieharmin > Desktop > ✎ untitled.py > ...
1 import logging
2 import redis
3
4 r = redis.Redis(db=1)
5 logging.basicConfig()
6
7 class OutOfStockError(Exception):
8     """Raised when PyHats.com is all out of today's hottest hat"""
9
10 def buyitem(r: redis.Redis, itemid: int) -> None:
11     with r.pipeline() as pipe:
12         error_count = 0
13         while True:
14             try:
15                 # Get available inventory, watching for changes
16                 # related to this itemid before the transaction
17                 pipe.watch(itemid)
18                 nleft: bytes = r.hget(itemid, "quantity")
19                 if nleft > b"0":
20                     pipe.multi()
21                     pipe.hincrby(itemid, "quantity", -1)
22                     pipe.hincrby(itemid, "nurchased", 1)
23                     pipe.execute()
24                     break
25                 else:
26                     # Stop watching the itemid and raise to break out
27                     pipe.unwatch()
28                     raise OutOfStockError(
29                         f"Sorry, {itemid} is out of stock!"
30                     )
31             except redis.WatchError:
32                 # Log total num. of errors by this user to buy this item,
33                 # then try the same process again of WATCH/HGET/MULTI/EXEC
34                 error_count += 1
35                 logging.warning(
36                     "WatchError #id: %s; retrying",
37                     error_count, itemid
38                 )
39     return None
40

```

```
buyitem(r, "hat:56854717")
buyitem(r, "hat:56854717")
buyitem(r, "hat:56854717")
print(r.hmget("hat:56854717", "quantity", "npurchased"))
```

```
(base) atieharmin@Atiehs-MacBook-Pro Desktop % python untitled.py
[b'196', b'4']
(base) atieharmin@Atiehs-MacBook-Pro Desktop %
```

7 - حال با یک حلقه تمام کلاه های موجود از این یک کلاه را میخربم.

```
for _ in range(196):
    buyitem(r, "hat:56854717")
print(r.hmget("hat:56854717", "quantity", "npurchased"))
```

```
(base) atieharmin@Atiehs-MacBook-Pro Desktop % python untitled.py
[b'0', b'200']
(base) atieharmin@Atiehs-MacBook-Pro Desktop %
```

8 - حال دوباره برای خرید آن اقدام میکنیم. که چون موجودی تمام شده است، با خطای `out of stock` روبه رو میشویم.

```
buyitem(r, "hat:56854717")
```

```
(base) atieharmin@Atiehs-MacBook-Pro Desktop % python untitled.py
Traceback (most recent call last):
  File "untitled.py", line 41, in <module>
    buyitem(r, "hat:56854717")
  File "untitled.py", line 28, in buyitem
    raise OutOfStockError(
__main__.OutOfStockError: Sorry, hat:56854717 is out of stock!
(base) atieharmin@Atiehs-MacBook-Pro Desktop %
```

9 - حال با تابع `timedelta` میخواهیم زمانی را برای `expire` شدن یک `key` مشخص کنیم. سپس با توابع `ttl` و `pttl` میتوان مقدار زمانی که از `expire` شدن آن مانده است را مشاهده کرد. همچنین با تابع `get` و `key` اگر این شده باشد، مقدار `none` و اگر نشده باشد مقدار دیگری نشان داده میشود.

```
sers > atieharmin > Desktop > untitled.py > ...
1 import logging
2 import redis
3 from datetime import timedelta
4
5 r = redis.Redis(db=1)
6
7 # setex: "SET" with expiration
8 r.setex(
9     "runner",
10    timedelta(minutes=1),
11    value="now you see me, now you don't"
12 )
```

```
6
7 # setex: "SET" with expiration
8 print(r.ttl("runner")) # "Time To Live", in seconds
9
10 print(r.pttl("runner")) # Like ttl, but milliseconds
11
12
```

```
(base) atieharmin@Atiehs-MacBook-Pro Desktop % python untitled.py
32
31799
(base) atieharmin@Atiehs-MacBook-Pro Desktop %
```

```
sers > atieharmin > Desktop > untitled.py > ...
1 import logging
2 import redis
3 from datetime import timedelta
4
5 r = redis.Redis(db=1)
6 print(r.get("runner"))
7
8 ##logging.basicConfig()
```

```
(base) atieharmin@Atiehs-MacBook-Pro Desktop % python untitled.py
b'now you see me, now you don't'
(base) atieharmin@Atiehs-MacBook-Pro Desktop %
```

```
5 r = redis.Redis(db=1)
6 print(r.get("runner"))
7 print(r.exists("runner"))
8
```

```
(base) atieharmin@Atiehs-MacBook-Pro Desktop % python untitled.py
None
0
(base) atieharmin@Atiehs-MacBook-Pro Desktop %
```

حال با تکه کد زیر میخواهیم ip هایی را به صورت یک لیست نگه داریم.

- 10

```
import redis

r = redis.Redis(db=5)

print(r.lpush("ips", "51.218.112.236"))
print(r.lpush("ips", "90.213.45.98"))
print(r.lpush("ips", "115.215.230.176"))
print(r.lpush("ips", "51.218.112.236"))
```

```
(base) atieharmin@Atiehs-MacBook-Pro Desktop % python untitled.py
1
2
3
4
(base) atieharmin@Atiehs-MacBook-Pro Desktop %
```

- ۱ ۱ - حال در یک shell دیگر یک ipwatcher خواهیم داشت که بر روی لیست ip ها حلقه میزند و سپس آن را به یک proper address تبدیل میکند. سپس زمانی که ipwatcher این ip را دیده است را به صورت یک key string نگه داری میکند. اگر این آدرس بیش از یک مقدار مشخص دیده شود مشخص میشود که این ip متعلق به یک bot است. خروجی این تابع همان ابتدا در عکس زیر آمده است.

```
# New shell window or tab

import datetime
import ipaddress

import redis

# Where we put all the bad egg IP addresses
blacklist = set()
MAXVISITS = 15

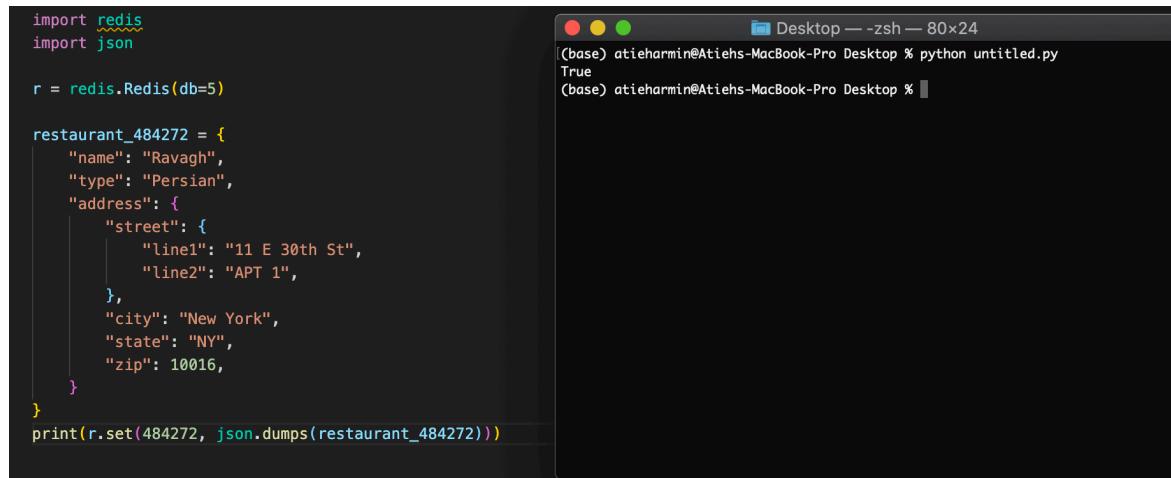
ipwatcher = redis.Redis(db=5)

while True:
    _, addr = ipwatcher.blpop("ips")
    addr = ipaddress.ip_address(addr.decode("utf-8"))
    now = datetime.datetime.utcnow()
    addrts = f"{addr}:{now.minute}"
    n = ipwatcher.incrby(addrts, 1)
    if n >= MAXVISITS:
        print(f"!Hat bot detected!: {addr}")
        blacklist.add(addr)
    else:
        print(f"{now}: saw {addr}")
    _ = ipwatcher.expire(addrts, 60)
```

- ۱ ۲ - حال ۲۰ به این لیست در shell قبلی اضافه میکنیم و نتیجه‌ی آن را در shell دوم به سرعت می‌بینیم.

```
Users > atieharmin > Desktop > untitled.py > ...
1 import redis
2
3 r = redis.Redis(db=5)
4
5 for _ in range(20):
6     r.lpush("ips", "104.174.118.18")
```

- ۱ ۳ - حال میخواهیم عمل serialize مقدار را به string انجام دهیم. حال با دستور json.dumps میتوان string را به json ساخته شده را به string تبدیل کرد. راه حل دیگر برای serialization استفاده از yaml است (عکس ۳).



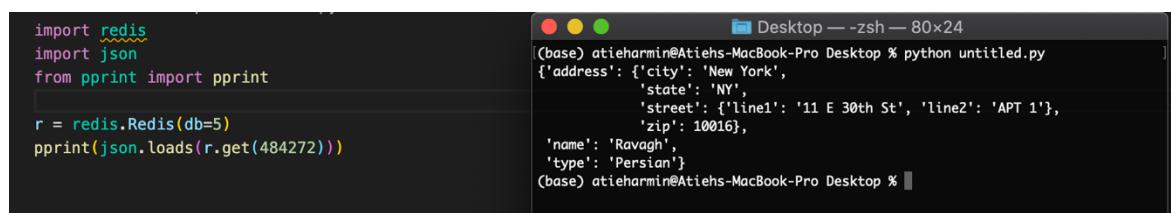
```

import redis
import json

r = redis.Redis(db=5)

restaurant_484272 = {
    "name": "Ravagh",
    "type": "Persian",
    "address": {
        "street": {
            "line1": "11 E 30th St",
            "line2": "APT 1",
        },
        "city": "New York",
        "state": "NY",
        "zip": 10016,
    }
}
print(r.set(484272, json.dumps(restaurant_484272)))

```

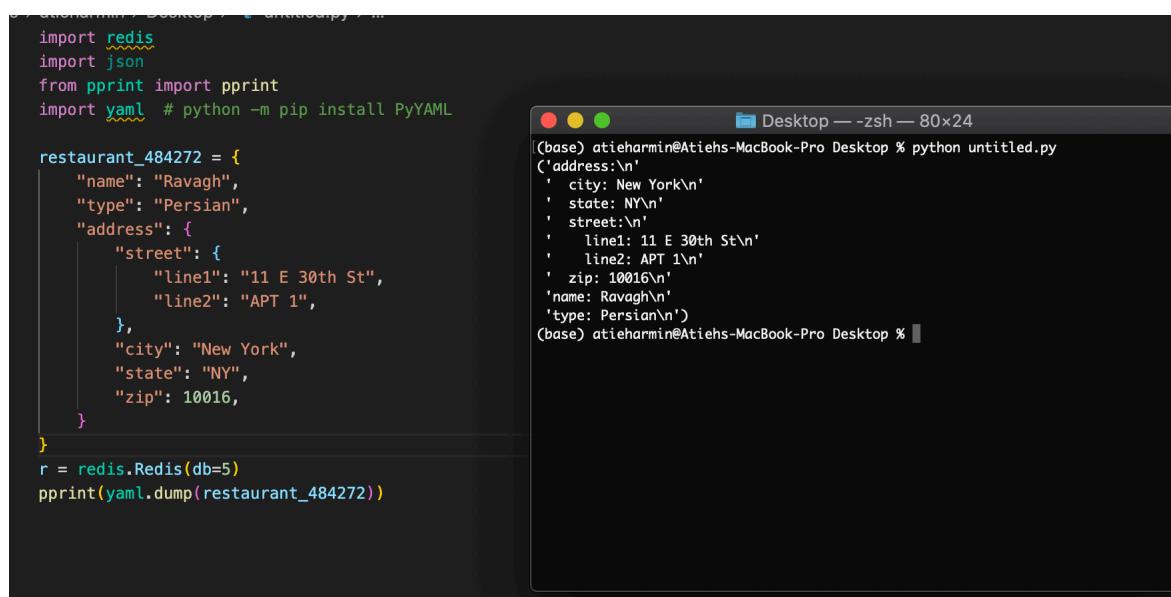


```

import redis
import json
from pprint import pprint

r = redis.Redis(db=5)
pprint(json.loads(r.get(484272)))

```



```

import redis
import json
from pprint import pprint
import yaml # python -m pip install PyYAML

restaurant_484272 = {
    "name": "Ravagh",
    "type": "Persian",
    "address": {
        "street": {
            "line1": "11 E 30th St",
            "line2": "APT 1",
        },
        "city": "New York",
        "state": "NY",
        "zip": 10016,
    }
}
r = redis.Redis(db=5)
pprint(yaml.dump(restaurant_484272))

```

روش دیگر استفاده از delimiter string key value در delimiter است. در تکه کد زیر، ابتدا روی تمام key value ها حلقه زده و نوع آن را چک میکنیم. اگر نوع آن به صورت دیکشنری بود، دوباره وار آن شده و تمام value هایش را چک کرده و به همین صورت ادامه میدهیم. اگر نه باتابع set میتوانیم آن را flat کنیم.

```

4 import yaml # python -m pip install PyYAML
5
6 restaurant_484272 = {
7     "name": "Ravagh",
8     "type": "Persian",
9     "address": {
10         "street": {
11             "line1": "11 E 30th St",
12             "line2": "APT 1",
13         },
14         "city": "New York",
15         "state": "NY",
16         "zip": 10016,
17     }
18 }
19 r = redis.Redis(db=5)
20
21 from collections.abc import MutableMapping
22
23 def setflat_skeys(
24     r: redis.Redis,
25     obj: dict,
26     prefix: str,
27     delim: str = ";",
28     *,
29     _autoprefix=""
30 ) -> None:
31     allowed_vtypes = (str, bytes, float, int)
32     for key, value in obj.items():
33         key = _autoprefix + key
34         if isinstance(value, allowed_vtypes):
35             r.set(f"{prefix}{delim}{key}", value)
36         elif isinstance(value, MutableMapping):
37             setflat_skeys(
38                 r, value, prefix, delim, _autoprefix=f"{key}{delim}"
39             )
40         else:
41             raise TypeError(f"Unsupported value type: {type(value)}")
42

```

```

r.flushdb() # Flush database: clear old entries
setflat_skeys(r, restaurant_484272, 484272)

for key in sorted(r.keys("484272*")): # Filter to this pattern
    print(f"{repr(key):35}{repr(r.get(key)):15}")

print(r.get("484272:address:street:line1"))

```

```

Desktop — zsh — 80x24
(base) atieharmin@Atiehs-MacBook-Pro Desktop % python untitled.py
b'484272:address:city'      b'New York'
b'484272:address:state'     b'NY'
b'484272:address:street:line1'  b'11 E 30th St'
b'484272:address:street:line2'  b'APT 1'
b'484272:address:zip'        b'10016'
b'484272:name'              b'Ravagh'
b'484272:type'              b'Persian'
b'11 E 30th St'

(base) atieharmin@Atiehs-MacBook-Pro Desktop %

```