به نام خدا

دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده برق و کامپیوتر

آزمایشگاه پایگاه داده

دستور کار شماره 8

نام و نام خانوادگی

معین شیردل 810197535

آبان ماه ۱۴۰۰

آموزش های سایت real-python:

```python
import redis

redis_cli = redis.Redis()
redis_cli.mset(
    {
        "Croatia": "Zagreb",
        "Bahamas": "Nassau",
    }
)
print(redis_cli.get("Bahamas"))
```

```
moein@moein:~/Desktop/DBLab/Lab8$ python3 lab8-redis.py
b'Nassau'
```

با تکه کد بالا، یک دیکشنری درست میکنیم و مقادیری برای کلید های Croatia و Bahamas تعیین می کنیم و مقدار مربوط به Bahamas را دریافت می کنیم.

```python
import redis
import datetime

redis_cli = redis.Redis()
today = datetime.date.today()
stoday = today.isoformat()
visitors = {"dan", "jon", "alex"}
redis_cli.sadd(stoday, *visitors)
print(redis_cli.smembers(stoday))
print(redis_cli.scard(today.isoformat()))
```

```
{b'dan', b'alex', b'jon'}
3
```

به دلیل اینکه تایپ datetime برای ذخیره سازی در ردیس مناسب نیست، با تابع isoformat() آن را به تاریخ استرینگ تبدیل می کنیم و به عنوان مقدار مناسب برای کلید مقادیر داخل ردیس استفاده می کنیم.

```python
import random
from pprint import pprint

redis_cli = redis.Redis(db = 1)

random.seed(444)
hats = {f"hat:{random.getrandbits(32)}": i for i in (
    {
        "color": "black",
        "price": 49.99,
        "style": "fitted",
        "quantity": 1000,
        "npurchased": 0,
    },
    {
        "color": "maroon",
        "price": 59.99,
        "style": "hipster",
        "quantity": 500,
        "npurchased": 0,
    },
    {
        "color": "green",
        "price": 99.99,
        "style": "baseball",
        "quantity": 200,
        "npurchased": 0,
    })
}

with redis_cli.pipeline() as pipe:
    for h_id, hat in hats.items():
        pipe.hmset(h_id, hat)
    print(pipe.execute())


print(redis_cli.bgsave())

pprint(redis_cli.hgetall("hat:56854717"))

print(redis_cli.keys())
```

```
moein@moein: ~/Desktop/DBLab/Lab8
File  Edit  View  Search  Terminal  Help
moein@moein:~/Desktop/DBLab/Lab8$
moein@moein:~/Desktop/DBLab/Lab8$
moein@moein:~/Desktop/DBLab/Lab8$
moein@moein:~/Desktop/DBLab/Lab8$
moein@moein:~/Desktop/DBLab/Lab8$
moein@moein:~/Desktop/DBLab/Lab8$
moein@moein:~/Desktop/DBLab/Lab8$ python3 lab8-redis.py
[True, True, True]
True
{b'color': b'green',
 b'npurchased': b'0',
 b'price': b'99.99',
 b'quantity': b'200',
 b'style': b'baseball'}
moein@moein:~/Desktop/DBLab/Lab8$ python3 lab8-redis.py
[True, True, True]
True
{b'color': b'green',
 b'npurchased': b'0',
 b'price': b'99.99',
 b'quantity': b'200',
 b'style': b'baseball'}
[b'hat:56854717', b'hat:1326692461', b'hat:1236154736']
moein@moein:~/Desktop/DBLab/Lab8$
```

در این قسمت، سه کلاه با مشخصات متفاوت ایجاد شد و به آن ها یک عدد رندوم به عنوان آیدی داده شد. سپس این آبجکت از این کلاه و مشخصاتش روی صفحه چاپ شد. در آخرین خروجی نیز آیدی های رندوم اختصاص داده شده به کلاه ها چاپ شده است.

```
[b'hat:56854717', b'hat:1326692461', b'hat:1236154736']
b'199'
b'1'
moein@moein:~/Desktop/DBLab/Lab8$
```

```python
redis_cli.hincrby("hat:56854717", "quantity", -1)
print(redis_cli.hget("hat:56854717", "quantity"))
redis_cli.hincrby("hat:56854717", "npurchased", 1)
print(redis_cli.hget("hat:56854717", "npurchased"))
```

در این قسمت نیز به کمک تابع hincrby می توان مقادیر عددی ذخیره شده را با مقادیر مختلف جمع زد. حاصل در تصویر مشخص است.

```python
logging.basicConfig()

class OutOfStockError(Exception):
    """Raised when PyHats.com is all out of today's hottest hat"""

def buyitem(redis_cli: redis.Redis, itemid: int) -> None:
    with redis_cli.pipeline() as pipe:
        error_count = 0
        while True:
            try:
                # Get available inventory, watching for changes
                # related to this itemid before the transaction
                pipe.watch(itemid)
                nleft: bytes = redis_cli.hget(itemid, "quantity")
                if nleft > b"0":
                    pipe.multi()
                    pipe.hincrby(itemid, "quantity", -1)
                    pipe.hincrby(itemid, "npurchased", 1)
                    pipe.execute()
                    break
                else:
                    # Stop watching the itemid and raise to break out
                    pipe.unwatch()
                    raise OutOfStockError(
                        f"Sorry, {itemid} is out of stock!"
                    )
            except redis.WatchError:
                # Log total num. of errors by this user to buy this item,
                # then try the same process again of WATCH/HGET/MULTI/EXEC
                error_count += 1
                logging.warning(
                    "WatchError #%d: %s; retrying",
                    error_count, itemid
                )
    return None
```

```
moein@moein:~/Desktop/DBLab/Lab8$ python3 lab8-redis.py
[b'193', b'7']
buyitem(redis_cli, "hat:56854717")
buyitem(redis_cli, "hat:56854717")
buyitem(redis_cli, "hat:56854717")
print(redis_cli.hmget("hat:56854717", "quantity", "npurchased"))
```

در کد بالا، یک تابع برای ثبت خرید یک کالا داریم که فرآیندهای لازم در هنگام خرید کالا را بررسی می کند. مثلا اگر موجودی یک کالا صفر باشد، اجازه خرید آن را نمی دهد. به عبارتی هرگاه موجودی منفی شد، عملیات را برمیگرداند و undo می کند و یک exception می دهد. حاصل سه بار خرید این کالا را در تصویر دوم می توان مشاهده کرد. البته به علت اجرای متوالی این کد پیش از گرفتن عکس، نتیجه در تصویر کمی متفاوت از انتظار است.

```
moein@moein:~/Desktop/DBLab/Lab8$ python3 lab8-redis.py
[b'0', b'200']
```

با دیدن مقدار ۰ به عنوان موجودی کلاه مورد نظر به کمک انجام ۲۰۰ خرید متوالی، حال می توان با یک خرید کلاه دیگر exception اتمام موجودی را دید و تغییر نکردن مقادیر تعداد خرید و موجودی را مشاهده کرد.

```
moein@moein:~/Desktop/DBLab/Lab8$ python3 lab8-redis.py
Traceback (most recent call last):
  File "lab8-redis.py", line 49, in <module>
    buyitem(redis_cli, "hat:56854717")
  File "lab8-redis.py", line 34, in buyitem
    f"Sorry, {itemid} is out of stock!"
__main__.OutOfStockError: Sorry, hat:56854717 is out of stock!
```

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

```
# setex: "SET" with expiration
redis_cli.setex(
    "runner",
    timedelta(minutes=1),
    value="now you see me, now you don't"
)
```

به کمک کد بالا یک رکورد با کلید runner به دیتابیس اضافه می کنیم که بعد از یک دقیقه پاک خواهد شد و ttl یک دقیقه ای دارد. در تصویر پایین هم مقدار ttl یا زمان باقیمانده تا پاک شدن کلید runner مشخص است که حدود ۵۴ ثانیه است.

```
54
53861
moein@moein:~/Desktop/DBLab/Lab8$
```

```
print(redis_cli.ttl("runner"))
print(redis_cli.pttl("runner"))
```

```
redis_cli = redis.Redis(db=5)

redis_cli.lpush("ips", "51.218.112.236")
redis_cli.lpush("ips", "90.213.45.98")
redis_cli.lpush("ips", "115.215.230.176")
redis_cli.lpush("ips", "51.218.112.236")
```

به کمک تکه کد بالا تعدادی ip را به لیستی با نام ips اضافه کرده ایم. با تابع lpush(). خروجی آن نیز طول لیست پس از هر دفعه اجرای تابع است.

```python
# 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)
```

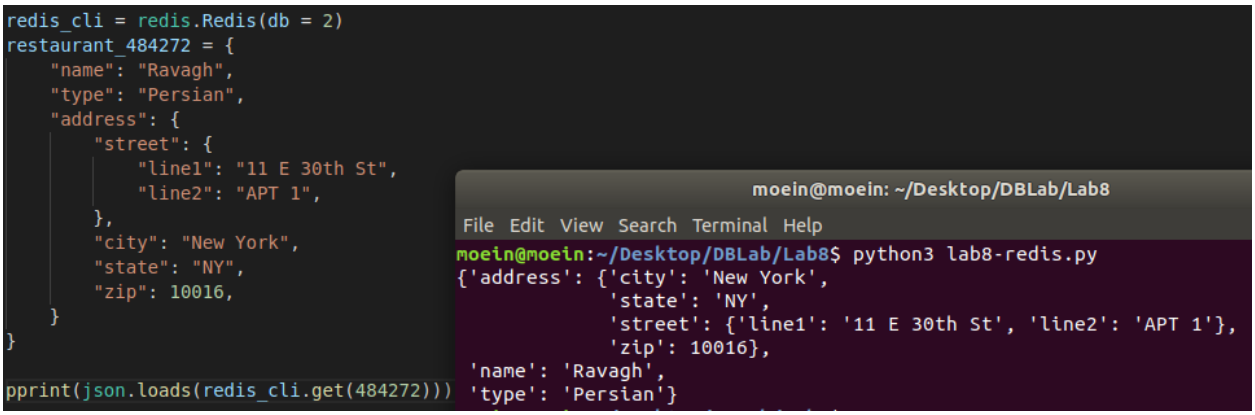
```python
import redis

redis_cli = redis.Redis(db=5)

for _ in range(20):
    redis_cli.lpush("ips", "104.174.118.18")
```

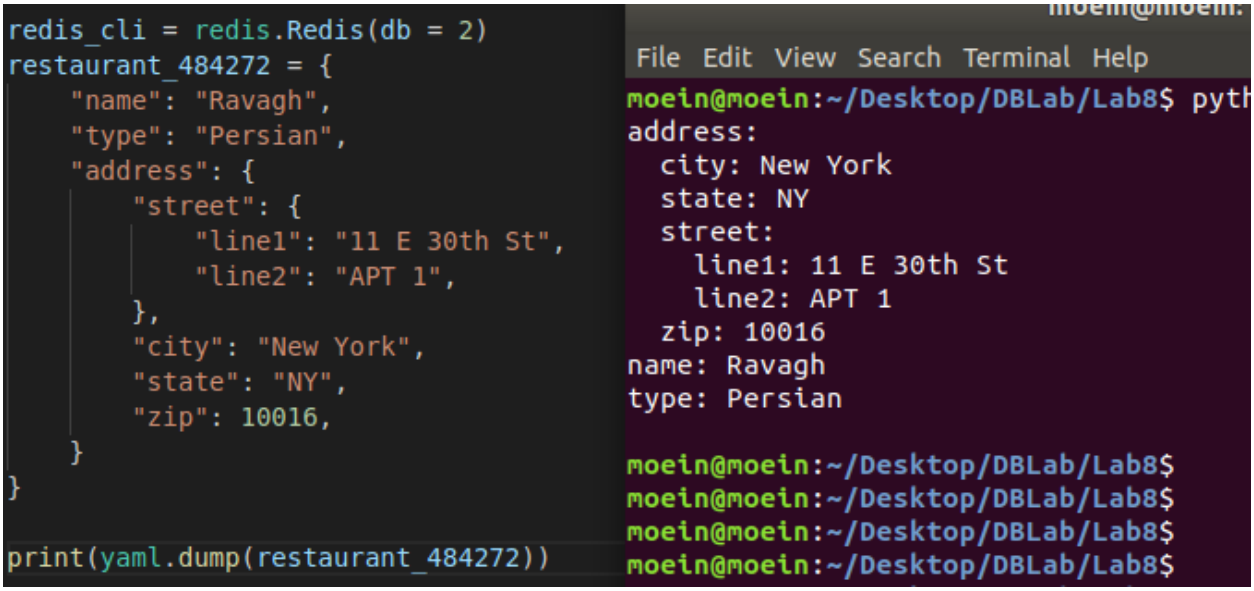
به کمک تکه کد سمت چپ، یک ربات تهیه شد که به هر ip به اندازه ۱۵ بار بازدید از سایت را می دهد. یک بازدید از سایت، به هر بار insert شدن در لیست ips گفته می شود. در کد سمت راست نیز، ۲۰ بار بازدید انجام می شود و خروجی پایین دریافت می شود. به ازای ۱۵ دفعه اول ربات ip را می بیند و پس از آن این اجازه را نمی دهد.

```
2021-12-29 19:05:41.129659:  saw 104.174.118.18
2021-12-29 19:05:41.129885:  saw 104.174.118.18
2021-12-29 19:05:41.130037:  saw 104.174.118.18
2021-12-29 19:05:41.130207:  saw 104.174.118.18
2021-12-29 19:05:41.130437:  saw 104.174.118.18
2021-12-29 19:05:41.130667:  saw 104.174.118.18
2021-12-29 19:05:41.130844:  saw 104.174.118.18
2021-12-29 19:05:41.131043:  saw 104.174.118.18
2021-12-29 19:05:41.131247:  saw 104.174.118.18
2021-12-29 19:05:41.131456:  saw 104.174.118.18
2021-12-29 19:05:41.131606:  saw 104.174.118.18
2021-12-29 19:05:41.131754:  saw 104.174.118.18
2021-12-29 19:05:41.131903:  saw 104.174.118.18
2021-12-29 19:05:41.132053:  saw 104.174.118.18
Hat bot detected!:  104.174.118.18
Hat bot detected!:  104.174.118.18
Hat bot detected!:  104.174.118.18
Hat bot detected!:  104.174.118.18
Hat bot detected!:  104.174.118.18
Hat bot detected!:  104.174.118.18
```



در این قسمت، برای ذخیره سازی یک دیکشنری از نوع json به کمک تابع json.loads() توانستیم آن را ذخیره کنیم.

روش پایین یز یک روش دیگر برای serialization است. هر دو روش گفته شده تا الان، آبجکت json را در یک رشته serialize می کنند.



```python
redis_cli = redis.Redis(db = 1)


def setflat_skeys(
    r: redis.Redis,
    obj: dict,
    prefix: str,
    delim: str = ":",
    *,
    _autopfix=""
) -> None:
    """Flatten `obj` and set resulting field-value pairs into `r`.

    Calls `.set()` to write to Redis instance inplace and returns None.

    `prefix` is an optional str that prefixes all keys.
    `delim` is the delimiter that separates the joined, flattened keys.
    `_autopfix` is used in recursive calls to created de-nested keys.

    The deepest-nested keys must be str, bytes, float, or int.
    Otherwise a TypeError is raised.
    """
    allowed_vtypes = (str, bytes, float, int)
    for key, value in obj.items():
        key = _autopfix + key
        if isinstance(value, allowed_vtypes):
            redis_cli.set(f"{prefix}{delim}{key}", value)
        elif isinstance(value, MutableMapping):
            setflat_skeys(
                redis_cli, value, prefix, delim, _autopfix=f"{key}{delim}"
            )
        else:
            raise TypeError(f"Unsupported value type: {type(value)}")
```

در اینجا یک تابع به نام setflat_keys تعریف شده است که آبجک جیسونی ما را flatten می کند و وارد دیتابیس ردیسی ما می کند. این عمل را روی restaurant_484272 انجام می دهیم و برای دیدن حاصل از تکه کد زیر استفاده می کنیم که فیلدها را یکی یکی چاپ می کند. به عنوان آخرین خروجی نیز، مقدار فیلد line1 از خیابان موجود در آدرس چاپ شده است و خروجی مشخص است.

```python
restaurant_484272 = {
    "name": "Ravagh",
    "type": "Persian",
    "address": {
        "street": {
            "line1": "11 E 30th St",
            "line2": "APT 1",
        },
        "city": "New York",
        "state": "NY",
        "zip": 10016,
    }
}

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

for key in sorted(redis_cli.keys("484272*")):
    print(f"{repr(key):35}{repr(redis_cli.get(key)):15}")


print("\n\n\n")
print(redis_cli.get("484272:address:street:line1"))
```

```
moein@moein:~/Desktop/DBLab/Lab8$ python3 lab8-redis.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'
```

- به علت نیاز به نوشتن گزارش در google docs، امکان رعایت دقیق فرمت گزارش ها را نداشتم. از این بابت عذرخواهی میکنم.