□□□□□ का परिचय

□□□□□ की आधिकारिक वेबसाइट खोलें, तो पहला वाक्य यह कहता है कि □□□□□ एक ओपन-सोर्स, इन-मेमोरी डेटा स्ट्रक्चर स्टोर है, जिसे अक्सर डेटाबेस और कैशिंग के लिए उपयोग किया जाता है। Redis बहुत ही आम तौर पर उपयोग किया जाता है।

□□□□□ स्थापित करना

आप Redis को उसके आधिकारिक वेबसाइट से इंस्टॉल कर सकते हैं। यह SQLite की तरह ही है। इंस्टॉल हो जाने के बाद, Python में Redis का उपयोग कैसे करें?

```
pip install redis
```

```
>>> import redis
>>> r = redis.Redis(host='localhost', port=6379, db=0)
>>> r.set('foo', 'bar')
True
>>> r.get('foo')
b'bar'
```

(नोट: कोड ब्लॉक को अनुवादित नहीं किया गया है क्योंकि यह प्रोग्रामिंग सिंटैक्स है और इसे अपरिवर्तित रखना आवश्यक है।)

□□□□□□ डॉक्यूमेंटेशन में कुछ उदाहरण दिए गए हैं। यहां pip जैसी चीज़ें दिखाई देती हैं। pip एक पैकेज मैनेजमेंट टूल है। पैकेज मैनेजमेंट टूल क्या है, यह "प्रोग्रामिंग वातावरण से परिचित होना" अध्याय में देखा जा सकता है। pip का python से वही संबंध है, जो Homebrew का macOS सिस्टम से है।

pip आमतौर पर Python इंस्टॉल करते समय पहले से ही शामिल होता है। यदि आपके कंप्यूटर में कई संस्करणों के Python और Pip हैं, तो आप ~/.bash_profile में निम्नलिखित दो पंक्तियाँ जोड़ सकते हैं:

```
alias python=/usr/local/Cellar/python@3.9/3.9.1_6/bin/python3
alias pip=/usr/local/Cellar/python@3.9/3.9.1_6/bin/pip3
```

(नोट: कोड ब्लॉक को अनुवादित नहीं किया गया है क्योंकि यह एक टर्मिनल कमांड है और इसे हिंदी में अनुवादित करने की आवश्यकता नहीं है।)

इसका मतलब है कि python और pip का एक विशिष्ट संस्करण निर्दिष्ट करना। एक तरीका यह है कि Homebrew का उपयोग करके इसे इंस्टॉल किया जाए। इसके अलावा, इसे सोर्स कोड से भी बनाया और इंस्टॉल किया जा सकता है।

```
make
make test
make install
```

```
$ redis-server
87684:C 10 Mar 2021 14:46:06.056 # oOOOoOOOoOOOo Redis          oOOOoOOOoOOOo
87684:C 10 Mar 2021 14:46:06.056 # Redis version=6.2.1, bits=64, commit=00000000, modified=0, pid=87684
87684:C 10 Mar 2021 14:46:06.056 #    :                        ,
87684:M 10 Mar 2021 14:46:06.057 *                10032          (        4864      )
87684:M 10 Mar 2021 14:46:06.057 *        : POSIX clock_gettime
...
Redis 6.2.1 (00000000/0) 64 bit
...
87684:M 10 Mar 2021 14:46:06.058 #
87684:M 10 Mar 2021 14:46:06.058 *
```

यहां कुछ सामग्री का एक अंश दिया गया है। जैसा कि आप देख सकते हैं, हमने इसे सफलतापूर्वक इंस्टॉल कर लिया है। संस्करण संख्या `6.2.1` है, जो कि आधिकारिक वेबसाइट पर उपलब्ध नवीनतम संस्करण है। एक और टर्मिनल विंडो खोलें। आप इसे आज़मा सकते हैं और इसके साथ खेल सकते हैं:

```
$ redis-cli
127.0.0.1:6379> set a 2
OK
127.0.0.1:6379> get a
"2"
```

कोड को चलाएं।

```
import redis

r = redis.Redis(host='localhost', port=6379, db=0)
r.set('foo', 'bar')
print(r.get('foo'))
```

यह कोड □□□□□ डेटाबेस से जुड़ता है और एक कुंजी-मान जोड़ी को सेट करता है। `r.set('foo', 'bar')` कमांड '□□□' कुंजी के साथ '□□□' मान को सेट करता है। `r.get('foo')` कमांड '□□□' कुंजी से जुड़े मान को प्राप्त करता है और उसे प्रिंट करता है।

आउटपुट:

```
$ python fib_redis.py
b'bar'
```

□□□□□ कैश उदाहरण

फिबोनाची अनुक्रम का Redis संस्करण लागू करने के लिए, हम □□□□□ को एक कैशिंग लेयर के रूप में उपयोग कर सकते हैं। यह हमें पहले से गणना किए गए फिबोनाची मानों को स्टोर करने और उन्हें बाद में पुनः प्राप्त करने की अनुमति देता है, जिससे प्रदर्शन में सुधार होता है।

नीचे एक उदाहरण दिया गया है कि कैसे हम □□□□□ का उपयोग करके फिबोनाची अनुक्रम को लागू कर सकते हैं:

```
import redis
import time

# Redis
r = redis.Redis(host='localhost', port=6379, db=0)

def fibonacci(n):
    #      Redis          ,
    if r.exists(n):
        return int(r.get(n))

    #
    if n <= 1:
        return n

    #
    result = fibonacci(n-1) + fibonacci(n-2)

    #       Redis
    r.set(n, result)

    return result

#        , 10
start_time = time.time()
print(fibonacci(10))
end_time = time.time()
print(f"Time taken: {end_time - start_time} seconds")
```

व्याख्या:

1. □□□□□ **कनेक्शन**: हम `redis.Redis` का उपयोग करके □□□□□ सर्वर से कनेक्ट होते हैं।
2. **फिबोनाची फ़ंक्शन**: `fibonacci` फ़ंक्शन पुनरावर्ती रूप से फिबोनाची अनुक्रम की गणना करता है। यदि मान पहले से ही □□□□□ में स्टोर है, तो उसे सीधे वापस लौटा दिया जाता है।
3. **कैशिंग**: गणना किए गए मानों को □□□□□ में स्टोर किया जाता है ताकि भविष्य में उन्हें दोबारा गणना करने की आवश्यकता न पड़े।
4. **प्रदर्शन**: □□□□□ का उपयोग करने से पुनरावर्ती गणना का समय काफी कम हो जाता है, खासकर बड़े `n` के लिए।

उदाहरण आउटपुट:

```
55
Time taken: 0.0005 seconds
```

यह कोड फिबोनाची अनुक्रम की गणना करता है और □□□□□ का उपयोग करके प्रदर्शन को अनुकूलित करता है।

```
import redis

r = redis.Redis(host='localhost', port=6379, db=0)

def f(n):
    nr = r.get(n)
    if nr is not None:
        return int(nr)
    res_n = 0
    if n < 2:
        res_n = n
    else:
        res_n = f(n-1) + f(n-2)

    r.set(n, res_n)
    return res_n

print(f(10))
```

इस तरह से यह काम हो गया।