Introducción a Redis

Al abrir el sitio web oficial de Redis, la primera frase indica que Redis es un almacén de estructuras de datos en memoria de código abierto, comúnmente utilizado como base de datos y caché. `Redis` es muy utilizado.

Instalación de Redis

Puedes instalar `Redis` desde su sitio web oficial. Al igual que con `SQLite`, una vez que hayas completado la instalación, ¿cómo puedes usar `Redis` en `Python`?

```
pip install redis
```

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

La documentación de Python proporciona algunos ejemplos. Aquí aparece algo como `pip`. `pip` es una herramienta de gestión de paquetes. Puedes consultar qué es una herramienta de gestión de paquetes en el capítulo "Familiarizarse con el entorno de programación". `pip` es para `python` lo que `Homebrew` es para el sistema `macOS`.

`pip` generalmente viene incluido cuando instalas `Python`. Si tienes varias versiones de `Python` y `Pip` en tu computadora, puedes agregar las siguientes dos líneas en tu archivo `~/.bash_profile`:

```
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
```

Se refiere a especificar una versión de `python` y `pip`. Una forma de hacerlo es utilizando `Homebrew` para la instalación. También es posible construir e instalar desde el código fuente.

```
make
make test
make install
```

```
$ redis-server
87684:C 10 Mar 2021 14:46:06.056 # oOOOoOOOoOOOo Redis está iniciando oOOOoOOOoOOOo
87684:C 10 Mar 2021 14:46:06.056 # Versión de Redis=6.2.1, bits=64, commit=00000000, modificado=0, pid=8
87684:C 10 Mar 2021 14:46:06.056 # Advertencia: no se especificó un archivo de configuración, se está u
87684:M 10 Mar 2021 14:46:06.057 * Número máximo de archivos abiertos incrementado a 10032 (originalmen
87684:M 10 Mar 2021 14:46:06.057 * Reloj monótono: POSIX clock_gettime
...
Redis 6.2.1 (00000000/0) 64 bit
...
87684:M 10 Mar 2021 14:46:06.058 # Servidor inicializado
87684:M 10 Mar 2021 14:46:06.058 * Listo para aceptar conexiones
```

Aquí se muestra un extracto del contenido. Como se puede ver, ya lo hemos instalado. La versión es `6.2.1`, la más reciente en el sitio web oficial. Abre otra ventana de terminal. Puedes intentar jugar un poco con él:

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

Ejecuta el siguiente código.

```
import redis

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

Salida:

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

Ejemplo de caché con Redis

Para implementar la secuencia de Fibonacci en una versión de `Redis`, puedes seguir los siguientes pasos. La idea es utilizar `Redis` para almacenar y recuperar los valores de la secuencia

de Fibonacci de manera eficiente.

Paso 1: Configurar Redis

Primero, asegúrate de tener `Redis` instalado y en funcionamiento. Puedes instalar `Redis` siguiendo las instrucciones en la documentación oficial de Redis.

Paso 2: Crear un script para calcular la secuencia de Fibonacci

Vamos a crear un script en Python que interactúe con `Redis` para calcular y almacenar los valores de la secuencia de Fibonacci.

```
import redis

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

def fibonacci(n):
    # Verificar si el valor ya está en Redis
    if r.exists(n):
        return int(r.get(n))

    # Caso base
    if n <= 1:
        r.set(n, n)
        return n

    # Calcular Fibonacci recursivamente y almacenar en Redis
    fib_value = fibonacci(n-1) + fibonacci(n-2)
    r.set(n, fib_value)
    return fib_value

# Ejemplo de uso
n = 10
print(f"Fibonacci({n}) = {fibonacci(n)}")
```

Paso 3: Ejecutar el script

Guarda el script en un archivo, por ejemplo `fibonacci_redis.py`, y ejecútalo:

```
python fibonacci_redis.py
```

Explicación del código

1. **Conexión a Redis**: El script se conecta a una instancia local de `Redis` en el puerto predeterminado `6379`.
2. **Función** `fibonacci(n)`: Esta función calcula el valor de Fibonacci para un número `n`. Primero verifica si el valor ya está almacenado en `Redis`. Si es así, lo recupera y lo devuelve. Si no, calcula el valor recursivamente y lo almacena en `Redis` para futuras consultas.
3. **Almacenamiento en Redis**: Los valores de Fibonacci se almacenan en `Redis` con la clave correspondiente al índice `n` y el valor de Fibonacci calculado.

Paso 4: Verificar los valores en Redis

Puedes verificar los valores almacenados en `Redis` utilizando el cliente de línea de comandos de `Redis`:

```
redis-cli
```

Luego, puedes obtener los valores almacenados utilizando el comando `GET`:

```
GET 10
```

Esto debería devolver el valor de Fibonacci para `n = 10`.

Conclusión

Este enfoque utiliza `Redis` como una caché para almacenar y recuperar los valores de la secuencia de Fibonacci, lo que puede ser útil para optimizar el cálculo de valores grandes en la secuencia. Además, `Redis` es una herramienta poderosa para manejar datos en memoria, lo que hace que este método sea eficiente para aplicaciones que requieren cálculos repetitivos.

```
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))
```

¡Y listo! Así de fácil.