



ХОЧУ ПОМОЧЬ  
ПРОЕКТУ

Модуль shelve в Python, key/value хранилище

 mango-office.ru



## Виртуальная АТС Расширенная

### 2 000 ₽

[Узнать больше](#)

РЕКЛАМА

Статья / Модуль shelve в Python, key/value хранилище

Ключ для произвольных объектов Python

Модуль shelve — это простое хранилище для произвольных объектов Python, значения которого можно извлекать, что облегчает переход от сценариев на основе [словаря](#) к тем, которые требуют постоянного хранилища shelve используются обычные [строки](#).

Произвольных объектов Python понимается - все, что может обрабатывать [модуль pickle](#). Это включает в себя большинство экземпляров классов, рекурсивных типов данных и объектов, содержащих множество общих подобъектов.

Ограничения:

- Выбор того, какой пакет базы данных будет использоваться, например [dbm.ndbm](#) или [dbm.gnu](#), зависит от того, какой интерфейс доступен в системе. Поэтому небезопасно открывать базу данных напрямую с помощью [dbm](#).
- База данных к сожалению также подвержена ограничениям dbm. Это означает, что сохраненные объекты по средствам [pickle.dumps](#), должны быть довольно маленькими, а в редких случаях совпадения ключей могут привести к тому, что база данных будет отказываться от обновлений.
- Модуль shelve не поддерживает одновременный доступ для чтения/записи к отложенным объектам. Несколько одновременных обращений к чтению [постоянного хранилища shelve](#) безопасны. Но если хранилище открыто для записи, никакая другая программа не должна открывать ее для чтения или записи. Для решения этой проблемы можно использовать блокировку файлов Unix, но она отличается в разных версиях Unix и требует знания используемой реализации базы данных.

Примеры использования:

```
import shelve

# файл может получить суффикс,
# добавленный низкоуровневой библиотекой
db = shelve.open(filename)

# Сохранить данные с ключом 'key', перезапишет
# старые данные, если ключ существует.
db[key] = data

# Получить КОПИЮ данных по ключу, вызывает
# KeyError, если такого ключа нет.
data = db[key]

# Удалить данные, хранящиеся по ключу
# вызывает KeyError, если такого ключа нет.
del db[key]

# True, если ключ существует
flag = key in db

# Список всех существующих ключей - медленно!
klist = list(db.keys())

# Хранилище открыто без аргумента 'writeback=True'
db = shelve.open('example.db')
db['x'] = [0, 1, 2] # Работает как положено, но ...
db['xx'].append(3) # !Это не так!, d['xx'] все еще [0, 1, 2]!
```

```
# Открыв хранилище без аргумента 'writeback=True'
# сделайте следующим образом
temp = db['xx']      # Извлекает копию
temp.append(5)       # Обновляет копию
db['xx'] = temp       # Сохраняет копию в хранилище

# закрывает хранилище
db.close()

# или можно открыть хранилище с 'writeback=True'
with shelve.open(filename, writeback=True) as db:
    # То метод .append будет работать как положено,
    # НО тогда процесс будет занимать больше памяти
    # и замедлять закрытие хранилища [d.close()].
    db['xx'].append(5)
```

Содержание раздела:

- [КРАТКИЙ ОБЗОР МАТЕРИАЛА.](#)
- [Функция open\(\) модуля shelve](#)
- [DbDict: база данных, основанная на dict](#)