


ХОЧУ ПОМОЧЬ  
ПРОЕКТУ

Модуль zlib в Python, библиотека ОС zlib

mango-office.ru

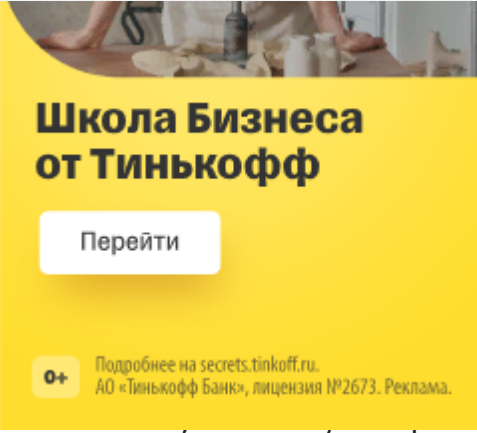
РЕКЛАМА

Виртуальная АТС Базовая

До 50 функций. С многоканальным номером и функциями обработки обращений вы не...

Узнать больше

Ссылка



Ссылка

Модуль zlib в Python, библиотека ОС zlib

Ссылка

данных с использованием библиотека ОС zlib

Для

ия данных, функции [модуля zlib](#) позволяют выполнять сжатие и распаковку с

ис

б. У библиотеки zlib есть собственная домашняя страница по адресу <http://www.zlib.net>.

Из

у модулем Python и версиями библиотеки zlib ранее 1.1.3, т. к. она имеет уязвимость

бе

мендуем использовать версии 1.1.4 или новее.

Функции модуля zlib имеют много опций и часто должны использоваться в определенном порядке. Обратитесь к руководству zlib по адресу <http://www.zlib.net/manual.html> для получения достоверной информации.

Для чтения и записи файлов .gz смотрите [модуль gzip](#).

Примечание: В некоторых случаях, модуль zlib может отсутствовать в стандартной библиотеке из за того, что при сборке Python из исходных кодов (компилировании) не был найден системный пакет zlib1g-dev по причине его отсутствия в системе Unix. В этом случае просто установите его командой `sudo apt install -y zlib1g-dev` (для Debian подобных ОС) и повторите [сборку Python из исходных кодов](#).

### Примеры использования модуля zlib:

Создадим массив данных в буфере, а затем сожжем его и сравним размеры до и после сжатия.

```
>>> import zlib
# создадим массив данных
>>> text = 'Привет docs-python.ru '
>>> data = []
>>> for _ in range(10):
...     data.append(text * 20)
...
# преобразование текста в байты
>>> byte_data = '\n\n'.join(data).encode('utf-8')
# сжимаем данные
>>> compress = zlib.compress(byte_data, level=-1)
# длина не сжатых данных
>>> len(byte_data)
# 4418

# длина сжатых данных
>>> len(compress)
# 85

# найдем процент сжатия
>>> len(compress) / len(byte_data)
# 0.01923947487550928
```

Теперь распакуем из буфера сжатые данные compress и выведем несколько символов, что бы убедиться, что данные распакованы.

```
# распаковываем сжатые 'compress' из буфера
>>> decompress = zlib.decompress(compress)
# Вверх берем байты в текст
>>> text = decompress.decode('utf-8')
```

```
# выведем на печать первые 22 символа
>>> text[0:22]
# 'Привет docs-python.ru Привет'
```

В этом примере создадим массив текстовых данных, запишем их в файл, а потом будем читать файл частями, изображая поток данных. Этот поток данных будем сжимать, выводить на печать и параллельно записывать в файл.

```
import zlib, pprint, binascii, os

# подготовим данные
text = 'Привет docs-python.ru '
data = []
for _ in range(50):
    data.append(text * 25)

# запишем данные в файл
with open('sample.txt', 'w') as fp:
    fp.write('\n\n'.join(data))

# создадим объекта сжатия `Compress`
compressor = zlib.compressobj(1)

# открываем созданный файл в двоичном
# режиме, читаем блоками и сжимаем
with open('sample.txt', 'rb') as fp, \
    open('sample.zl', 'wb') as fz:
    while True:
        block = fp.read(4096)
        if block:
            # сжимаем
            compressed = compressor.compress(block)
            if compressed:
                print(f'Compressed: {binascii.hexlify(compressed)}')
                # пишем
                fz.write(compressed)
            else:
                print('buffering...')
        else:
            break

    # данные кончились, сбросим буфер
    # и запишем остатки
    remaining = compressor.flush()
    fz.write(remaining)
    print('Flushed:')
    pprint.pprint(binascii.hexlify(remaining), width=60)

# Compressed: b'7801'
# buffering...
# buffering...
# buffering...
# buffering...
# buffering...
# buffering...
# Flushed:
# (b'edd8b10dc2501403c03e536402e648c90c8c8094820eb1083b202131'
#  b'c6cf4669e9e2c24a752de5d7d9cf61bcb7e7f88dcff86eaff9fa58d6'
#  b'dbe5becec3afdee1cfc334112119c70dc189064d2e09279c70e2a6b6'
#  b'6eaa3ed127fa449fe89324059c70c2c999ff7bd927f64992384e38e1'
#  b'c43eb14f921470c20927766c2b05faa4f59276ac1d9b3433279c70e2'
#  b'eeb83b490a38e18413df3bad14e893d64bdab1766cd2cc9c70c289bb'
#  b'e3ee2429e084134e7cefb452a04f5a2f69c7dab1493373c20927ee8e'
#  b'bb93a480134e38f1bdd34a813e69bda41d6bc726cdcc09279cb83bee'
#  b'4e92024e38e1c4f74e2b05faa4f59276ac1d9b3433279c70e2eeb83b'
#  b'490a38e18413df3bad14e893d64bdab1766cd2cc9c70c289bbe3ee24'
#  b'29e084134ecfcded901ec6bfe0a')

>>> import os
>>> os.remove('sample.zl')
>>> os.system(f'file sample.zl')
# sample.zl: zlib compressed data
```

Вверх

```
>>> os.path.getsize('sample.txt')
# 27598
>>> os.path.getsize('sample.zl')
# 294
```

Теперь будем распаковывать поток данных, поступающий из файла sample.zl, который был создан на предыдущем этапе.

```
import zlib, os

# создадим декомпрессор
unpack = zlib.decompressobj()

# открываем файлы на чтение 'sample.zl' и запись
# 'unpack-sample.txt' - оба в бинарных режимах.
with open('sample.zl', 'rb') as fpr, \
     open('unpack-sample.txt', 'wb') as fpw:
    while True:
        # читаем частями по 32 байта
        block = fpr.read(32)
        if block:
            # распаковываем
            data = unpack.decompress(block)
            # пишем данные в текстовый файл
            fpw.write(data)
        else:
            break

# смотрим что получилось
>>> os.system('file 'unpack-sample.txt')
# unpack-sample.txt: UTF-8 Unicode text, with very long lines
```



Содержание раздела:

- [КРАТКИЙ ОБЗОР МАТЕРИАЛА.](#)
- [Функция adler32\(\) модуля zlib](#)
- [Функция crc32\(\) модуля zlib](#)
- [Функция compress\(\) модуля zlib](#)
- [Функция decompress\(\) модуля zlib](#)
- [Функция compressobj\(\) модуля zlib](#)
- [Функция decompressobj\(\) модуля zlib](#)