Запись и чтение архивных zip-файлов

Последнее обновление: 24.11.2022

Zip представляет наиболее популярный формат архивации и сжатия файлов. И язык Python имеет встроенный модуль для работы с ними - **zipfile**. С помощью этого модуля можно создавать, считывать, записывать zip-файлы, получать их содержимое и добавлять в них файлы. Также поддерживается шифрование, но не поддерживается дешифрование.

Для представления zip-файла в этом модуле определен класс **ZipFile**. Он имеет следующий конструктор:

```
ZipFile(file, mode='r', compression=ZIP_STORED, allowZip64=True, compresslevel=None,
```

Параметры:

- `file`: путь к zip-файлу
- `mode`: режим открытия файла. Может принимать следующие значения:
  - `r`: применяется для чтения существующего файла
  - `w`: применяется для записи нового файла
  - `a`: применяется для добавления в файл
- `compression`: тип сжатия файла при записи. Может принимать значения:
  - `ZIP_STORED`: архивация без сжатия (значение по умолчанию)
  - `ZIP_DEFLATED`: стандартный тип сжатия при архивации в zip
  - `ZIP_BZIP2`: сжатие с помощью способа BZIP2
  - `ZIP_LZMA`: сжатие с помощью способа LZMA
- `allowZip64`: если равно True, то zip-файл может быть больше 4 Гб

- `compresslevel`: уровень сжатия при записи файла. Для типов сжатия `ZIP_STORED` и `ZIP_LZMA` не применяется. Для типа `ZIP_DEFLATED` допустимые значения от 0 до 9, а для типа `ZIP_BZIP2` допустимые значения от 1 до 9.
- `strict_timestamps`: при значении False позволяет работать с zip-файлами, созданными ранее 01.01.1980 и позже 31.12.2107
- `metadata_encoding`: применяется для декодирования метаданных zip-файла (например, коментариев)

Для работы с файлами этот класс предоставляет ряд методов:

- **close()**: закрывает zip-файл
- **getinfo()**: возвращает информацию об одном файле из архива в виде объекта ZipInfo
- **namelist()**: возвращает список файлов архива
- **infolist()**: возвращает информацию обо всех файлах из архива в виде списока объектов ZipInfo
- **open()**: предоставляет доступ к одному из файлов в архиве
- **read()**: считывает файл из архива в набор байтов
- **extract()**: извлекает из архива один файл
- **extractall()**: извлекает все элементы из архива
- **setpassword()**: устанавливает пароль для zip-файла
- **printdir()**: выводит на консоль содержимое архива

Создание и закрытие файла

Для создания архивного файла в конструктор ZipFile передается режим "w" или "a":

```
1 from zipfile import ZipFile
2
3 myzip = ZipFile("metanit.zip", "w")
```

После выполнения кода в текущей папке будет создаваться пустой архивный файл "metanit.zip".

После окончания работы с архивом для его закрытия применяется метод `close()`:

```
1 from zipfile import ZipFile
2
```

```
myzip = ZipFile("metanit.zip", "w")
myzip.close()
```

Но так как ZipFile также представляет менеджер контекста, то он поддерживает выражение **with**, которое определяет контекст и автоматически закрывает файл по завершению контекста:

```
from zipfile import ZipFile

with ZipFile("metanit.zip", "w") as myzip:
    pass
```

Запись файлов в архив

Для записи файлов в архив применяется файл **write()**:

```
write(filename, arcname=None, compress_type=None, compresslevel=None)
```

Первый параметр представляет файл, который записиывается в архив. Второй параметр - `arcname` устанавливает произвольное имя для файла внутри архива (по умолчанию это само имя файла). Третий параметр - `compress_type` представляет тип сжатия, а параметр `compresslevel` - уровень сжатия.

Например, запишем в архив "metanit.zip" файл "hello.txt" (который, как предполагается, находится в той же папке, где и текущий скрипт python):

```
from zipfile import ZipFile

with ZipFile("metanit.zip", "w") as myzip:
    myzip.write("hello.txt")
```

Стоит учитывать, что при открытии файла в режиме "w" при всех последующих записях текущее содержимое будет затираться, то есть фактически архивный файл будет создаваться заново. Если нам необходимо добавить, то необходимо определять zip-файл в режиме "a":

```
from zipfile import ZipFile

with ZipFile("metanit.zip", "a") as myzip:
    myzip.write("hello2.txt")
    myzip.write("forest.jpg")
```

Стоит отметить, что по умолчанию сжатие не применяется. Но при необходимости можно применить какой-нибудь способ сжатия и уровень сжатия"

```
from zipfile import ZipFile, ZIP_DEFLATED

with ZipFile("metanit.zip", "w", compression=ZIP_DEFLATED, compresslevel=3) as myzip
    myzip.write("hello.txt")
```

Необходимо учитывать, что если мы попробуем добавить в архив файлы с уже имеющимися именами, то консоль выведет предупреждение. Чтобы избежать наличия файлов с дублирующимися именами можно через второй папаметр метода write явным образом определить для них уникальное имя внутри архива:

```
from zipfile import ZipFile

with ZipFile("metanit.zip", "a") as myzip:
    myzip.write("hello.txt", "hello1.txt")
    myzip.write("hello.txt", "hello2.txt")
    myzip.write("hello.txt", "hello3.txt")
```

Получение информации о файлах в архиве

Метод **infolist()** возвращает информацию о файлах в архиве с виде списка, где каждый отдельный файл представлен объектом ZipInfo:

```
from zipfile import ZipFile

with ZipFile("metanit.zip", "a") as myzip:
    print(myzip.infolist())
```

Класс **ZipInfo** предоставляет ряд атрибутов для хранения информации о файле. Основные из них:

- `filename`: название файла
- `date_time`: дата и время последнего изменения файла в виде кортежа в формате `(год, месяц, день, час, минута, секунда)`
- `compress_type`: тип сжатия
- `compress_size`: размер после сжатия
- `file_size`: оригинальный размер файла до сжатия

Получим эти данные по каждому отдельному файлу в архиве:

```
from zipfile import ZipFile

with ZipFile("metanit.zip", "r") as myzip:
    for item in myzip.infolist():
        print(f"File Name: {item.filename} Date: {item.date_time} Size: {item.file_s
```

Примерный консольный вывод:

```
File Name: hello.txt  Date: (2022, 11, 23, 20, 21, 34)  Size: 18
File Name: forest.jpg  Date: (2022, 11, 19, 20, 46, 52)  Size: 103956
File Name: hello1.txt  Date: (2022, 11, 23, 20, 21, 34)  Size: 18
File Name: hello2.txt  Date: (2022, 11, 23, 20, 21, 34)  Size: 18
File Name: hello3.txt  Date: (2022, 11, 23, 20, 21, 34)  Size: 18
```

С помощью метода `is_dir()` можно проверить, является ли элемент в архиве папкой:

```
from zipfile import ZipFile

with ZipFile("metanit.zip", "r") as myzip:
    for item in myzip.infolist():
        if(item.is_dir()):
            print(f"Папка: {item.filename}")
        else:
            print(f"Файл: {item.filename}")
```

Если надо получить только список имен входящих в архив файлов, то применяется метод **namelist()**:

```
from zipfile import ZipFile

with ZipFile("metanit.zip", "r") as myzip:
    for item in myzip.namelist():
        print(item)
```

Консольный вывод в моем случае:

```
hello.txt
forest.jpg
hello1.txt
hello2.txt
hello3.txt
```

С помощью метода **getinfo()** можно получить данные по одному из архивированных файлов, передав в метод его имя в архиве. Результат метода - объект ZipInfo:

```
from zipfile import ZipFile

with ZipFile("metanit.zip", "r") as myzip:
    try:
        hello_file = myzip.getinfo("hello.txt")
        print(hello_file.file_size)
    except KeyError:
        print("Указанный файл отсутствует")
```

Если в архиве не окажется элемента с указанным именем, то метод сгенерирует ошибку KeyError.

Извлечение файлов из архива

Для извлечения всех файлов из архива применяется метод **extractall()**:

```
extractall(path=None, members=None, pwd=None)
```

Первый параметр метода устанавливает каталог для извлечения архива (по умолчанию извлечение идет в текущий каталог). Параметр `members` представляет список строк - список названий файлов, которые надо извлечт из архива. И третий параметр - `pwd` представляет пароль, в случае если архив закрыт паролем.

Например, извлечем все файлы из архива:

```
from zipfile import ZipFile

with ZipFile("metanit.zip", "r") as myzip:
    myzip.extractall()
```

Извлечение в определенную папку:

```
myzip.extractall(path="metanit")
```

Извлечение части файлов:

```
# извлекаем файлы  "hello.txt", "forest.jpg" в папку "metanit2"
myzip.extractall(path="metanit2", members=["hello.txt", "forest.jpg"])
```

Для извлечения одного файла применяется метод **extract()**, в который в качестве обязательного параметра передается имя извлекаемого файла:

```
myzip.extract("hello.txt")
```

Считывание файла

Метод **read()** позволяет считать содержимое файла из архива в набор байтов:

```
from zipfile import ZipFile

with ZipFile("metanit.zip", "r") as myzip:
    content = myzip.read("hello5.txt")
    print(content)
```

Открытие файла

Метод **open()** позволяет открывать отдельные файлы из архива без непосредственного их извлечения:

```
open(name, mode='r', pwd=None, *, force_zip64=False)
```

В качестве первого обязательного параметра передается имя файла внутри архива. Второй параметр - `mode` устанавливает режим открытия. Параметр `pwd` задает пароль, если файл защищен паролем. И параметр `force_zip64` при значении True позволяет открывать файлы больше 4 Гб.

Этот файл может быть полезен для манипулирования файлом, например, для считывания его содержимого или, наоборот, для записи в него. Например, откроем файл и считаем его содержимое:

```
from zipfile import ZipFile

with ZipFile("metanit.zip", "a") as myzip:
    # записываем в архив новый файл "hello5.txt"
    with myzip.open("hello5.txt", "w") as hello_file:
        encoded_str = bytes("Python...", "UTF-8")
        hello_file.write(encoded_str)
```

Назад **Содержание** **Вперед**