**Реферат**

на тему : «Data Compression and Archivingю Модулі: zlib, gzip, bz2 , lzma, zipfile , tarfile».

Пiдготувала студентка 1 курсу заочного вiддiлення механiко-математичного факультету

Катришева Владамира

ЗМІСТ

[ВВЕДЕННЯ 2](#_Toc88425160)

[1 СТИСНЕННЯ ТА АРХІВУВАННЯ ЧЕРЕЗ МОДУЛІ PYTHON 3](#_Toc88425161)

[1.1 zlib 3](#_Toc88425162)

[1.2 gzip 6](#_Toc88425163)

[1.3 bz2 9](#_Toc88425164)

[1.4 lzma 12](#_Toc88425165)

[1.5 zip– file 16](#_Toc88425166)

[1.6 Tarfile 18](#_Toc88425167)

[УКЛАДЕННЯ 20](#_Toc88425168)

[СПИСОК ВИКОРИСТОВУВАНИХ ДЖЕРЕЛ 21](#_Toc88425169)

# ВВЕДЕННЯ

Одна з переваг Python – це наявність бібліотек, які вирішують мало не всі нагальні проблеми. Так ось для стиснення даних в практично будь– який формат Python– бібліотек величезна безліч. До стандартних відносяться наступні модулі:

1) zlib служить для стиснення даних на основі алгоритму Deflate.

2) gzip використовує вже алгоритм стиснення zlib і служить інтерфейсом, аналогічним утилітам gzip і gunzip.

3) lzma використовує алгоритм LZMA для стиснення, підтримує формати xz і lzma.

4) shutil високорівневий модуль для роботи з архівами різних форматів.

5) bz2 служить для стиснення bzip2, алгоритм стиснення ефективніше вищезгаданого Deflate, але працює повільніше.

6) zip file призначений для роботи з zip– файлами, причому модуль надає можливість створювати, читати, записувати і додавати в архів.

7) tarfile призначений для роботи з tar, підтримує стиснення форматів gzip, bzip2 і lzma.

# 1 СТИСНЕННЯ ТА АРХІВУВАННЯ ЧЕРЕЗ МОДУЛІ PYTHON

# 1.1 zlib

Функції zlib мають безліч опцій і часто повинні використовуватися в певному порядку.

zlib.adler32 (дані[, значення]) обчислює контрольну суму даних Adler– 32. В результаті виходить 32– розрядне ціле число без знака. Якщо значення присутнє, воно використовується в якості початкового значення контрольної суми; в іншому випадку використовується значення за замовчуванням 1. Передача значення дозволяє обчислювати поточну контрольну суму по об'єднанню декількох вхідних даних. Алгоритм не є криптографічно надійним і не повинен використовуватися для аутентифікації або цифрових підписів. Оскільки алгоритм призначений для використання в якості алгоритму контрольної суми, він не підходить для використання в якості загального алгоритму хешування.

zlib.compress (data, /, level=– 1) стискає байти в даних, повертаючи об'єкт bytes, що містить стислі дані. рівень– це ціле число від 0 до 9 або – 1, що керує рівнем стиснення; 1

(Z\_BEST\_SPEED) є найшвидшим і виробляє найменше стиснення, 9

(Z\_BEST\_COMPRESSION) є найповільнішим і виробляє найбільше. 0

(Z\_NO\_COMPRESSION) не є стисненням. Значення за замовчуванням дорівнює – 1

(Z\_DEFAULT\_COMPRESSION). Z\_DEFAULT\_COMPRESSION являє собою компроміс за замовчуванням між швидкістю і стисненням (в даний час еквівалентно рівню 6). Викликає виключення помилки, якщо виникає будь– яка помилка.

zlib.compressobj (level= – 1, method=DEFLATED, wbits=MAX\_WBITS, memLevel=DEF\_MEM\_LEVEL, strategy=Z\_DEFAULT\_STRATEGY[, zdict]) повертає об'єкт стиснення, який використовується для стиснення потоків даних, які не поміщаються в пам'ять відразу.

Рівень– це рівень стиснення– ціле число від 0 до 9 або – 1.

Значення 1 (Z\_BEST\_SPEED) є найшвидшим і виробляє найменше стиснення, в той час як значення 9 (Z\_BEST\_COMPRESSION) є найповільнішим і виробляє найбільше.

0 (Z\_NO\_COMPRESSION) не є стисненням. Значення за замовчуванням дорівнює – 1 (Z\_DEFAULT\_COMPRESSION). Z\_DEFAULT\_COMPRESSION являє собою компроміс за замовчуванням між швидкістю і стисненням (в даний час еквівалентно рівню 6).

Метод– це алгоритм стиснення. В даний час єдине підтримуване значення – ДЕФЛІРОВАННОЕ.

Аргумент wbits визначає розмір буфера історії (або" розмір вікна"), який використовується при стисненні даних, і включає заголовок і трейлер У вихідні дані. Він може приймати кілька діапазонів значень, за замовчуванням до 15 (MAX\_WBITS).

Аргумент memLevel керує обсягом пам'яті, що використовується для стану внутрішнього стиснення. Допустимі значення варіюються від 1 до 9. Більш високі значення споживають більше пам'яті, але працюють швидше і дають менший результат.

Strategy використовується для налаштування алгоритму стиснення. Можливими значеннями є Z\_DEFAULT\_STRATEGY, Z\_FILTERED, Z\_HUFFMAN\_ONLY, Z\_RLE (zlib 1.2.0.1) і Z\_FIXED (zlib 1.2.2.2).

zdict– це зумовлений словник стиснення. Це послідовність байтів (наприклад, об'єкт bytes), що містить підпослідовності, які, як очікується, будуть часто зустрічатися в даних, що підлягають стисненню. Ті підпослідовності, які, як очікується, будуть найбільш поширеними, повинні бути в кінці словника.

zlib.crc32 (data[, value]) обчислює контрольну суму даних CRC (циклічна перевірка надмірності). В результаті виходить 32– розрядне ціле число без знака. Якщо значення присутнє, воно використовується в якості початкового значення контрольної суми; в іншому випадку використовується значення за замовчуванням 0. Передача значення дозволяє обчислювати поточну контрольну суму по об'єднанню декількох вхідних даних. Алгоритм не є криптографічно надійним і не повинен використовуватися для аутентифікації або цифрових підписів. Оскільки алгоритм призначений для використання в якості алгоритму контрольної суми, він не підходить для використання в якості загального алгоритму хешування.

zlib.decompress (data, /, bits=MAX\_WBITS, bufsize=IF\_BUF\_SIZE) розпаковує байти в даних, повертаючи об'єкт bytes, що містить нестиснуті дані. Параметр wbits залежить від формату даних і обговорюється Далі нижче. Якщо заданий bufsize, він використовується в якості початкового розміру вихідного буфера. Викликає виключення помилки, якщо виникає будь– яка помилка.

Створимо масив даних в буфері, а потім стиснемо його і порівняємо розміри до і після стиснення.

>>> import zlib

# створимо масив даних

> > > text = ' Привіт docs– python.ru '

>>> data = []

>>> for \_ in range(10):

data.append(text \* 20)

# перетворення тексту в байти

>>> byte\_data = '\n\n'.join(data).encode('utf– 8')

# стискаємо дані

>>> compress = zlib.compress(byte\_data, level=– 1)

# # довжина нестиснутих даних

>>> len(byte\_data)

# 4418

# # довжина стиснутих даних

>>> >>> les(comperes)

# 85

# знайдемо відсоток стиснення

>>> len(compress) / len(byte\_data)

# 0.01923947487550928

Тепер розпакуємо з буфера стислі дані compress і виведемо кілька символів, щоб переконатися, що дані розпаковані.

# # розпаковуємо стиснення 'compress' з буфера

>>> decompress = zlib.decompress(compress)

# перетворюємо байти в текст

>>> text = decompress.decode('utf– 8')

# виведемо на друк перші 22 Символи

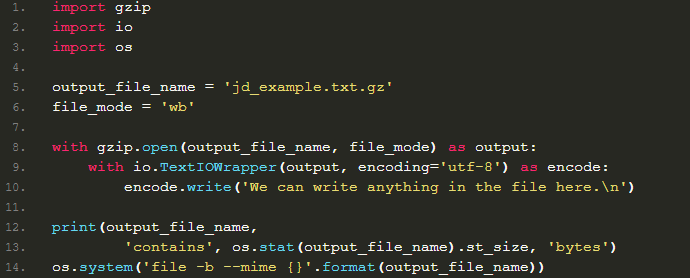
>>> text[0:22]

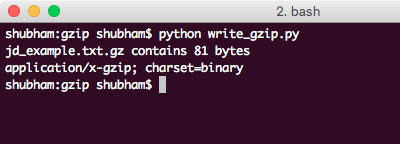
# 'Привіт docs– python.ru Привіт'

# 1.2 gzip

Модуль gzip Python забезпечує дуже простий спосіб стиснення і розпакування файлів і працює аналогічно програмам GNU gzip і gunzip. Цей модуль надає клас Gzip, який містить деякі зручні функції, такі як open (), compress() і deca press(). Перевага класу Gzip полягає в тому, що він читає і записує файли gzip і автоматично стискає і розпаковує їх, так що в програмі вони виглядають так само, як звичайні об'єкти File. Важливо пам'ятати, що інші формати, підтримувані програмами gzip і gunzip, не підтримуються цим модулем.

Почнемо з функції open (), яка створює екземпляр GzipFile і відкриває файл в режимі wb для запису в стислий файл, код даних функцій використовується в малюнку 1.1, а результат виконання код представлений на малюнку 1.2.

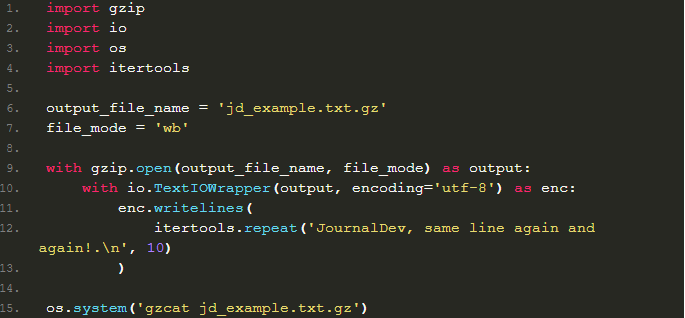


Малюнок 1.1 – Код запису в стислий файл

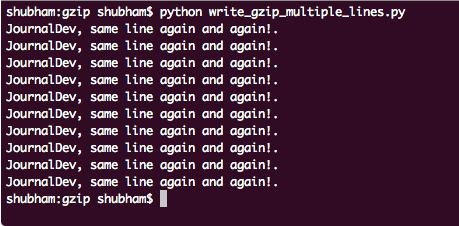
Малюнок 1.2– Результат запису в стислий файл

Щоб записати в стиснутий файл, спочатку потрібно відкрити його в режимі wb і обернути екземпляр gzip File за допомогою TextIOWrapper з модуля io для кодування тексту Unicode в байти, які підходять для стиснення.

Аналогічно таким же чином можна записати кілька рядків за допомогою такого ж сценарію, але з додаванням декількох рядків коду. Код даних функцій використовується в малюнку 1.3, а результат виконання код представлений на малюнку 1.4.

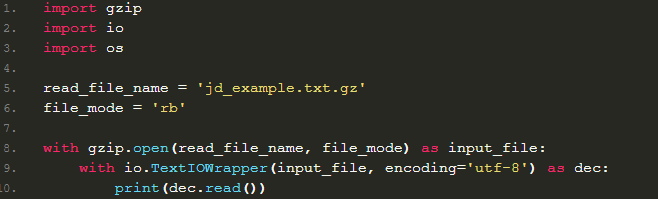


Малюнок 1.3 – Код запису декількох рядків у стислий файл

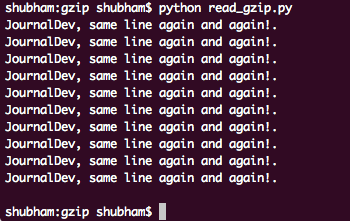


Малюнок 1.4 – Результат запису в стислий файл декількох рядків

Тепер, коли закінчили процес запису файлу, також можемо читати дані зі стисненого файлу. Тепер потрібно використовувати інший файловий режим– rb, режим читання. Код даних функцій використовується в малюнку 1.5, а результат виконання код представлений на малюнку 1.6.



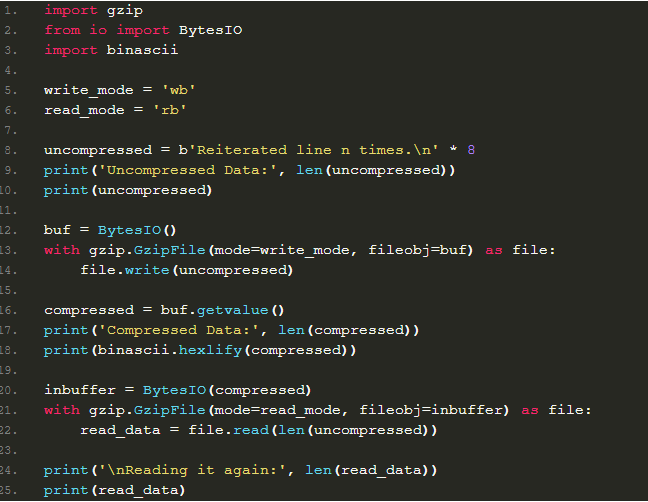
Малюнок 1.5 – Код читання стислих файлів



Малюнок 1.6 – Результат читання стиснутих файлів

Необхідно зверніть увагу, що не зроблено тут нічого особливого з Gzip, крім передачі йому іншого файлового режиму. Процес читання виконується TextIOWrapper, який використовує об'єкт File, як об'єкт, що надається модулем gzip.

Ще одна велика перевага модуля gzip полягає в тому, що його можна використовувати для обгортання інших типів потоків, щоб вони могли також використовувати стиснення. Це надзвичайно корисно, коли необхідно передавати великий обсяг даних через веб– сокети. Код даних функцій використовується в малюнку 1.7, а результат виконання код представлений на малюнку 1.8.



Малюнок 1.7 – Код читання потоків



Малюнок 1.8 – Результат читання потоків

Необхідно звернути увагу, що при записі не потрібно було вказувати параметри довжини. Нам довелося явно передати довжину функції read ().

# 1.3 bz2

Модуль надає вичерпний інтерфейс для стиснення і розпакування даних з використанням алгоритму стиснення bzip2

Модуль bz2 містить:

* функцію open () і клас BZ2File для читання і запису стислих файлів;
* класи BZ2Compressor і BZ2Decompressor для інкрементного стиснення– разжатия;
* функції compress () і decompress () для однокадрового стиснення– разжатия.

Всі класи в цьому модулі можуть бути безпечно доступні з декількох потоків.

bz2.open (filename, mode='r', compresslevel=9, encoding=None, errors=None, newline=None) відкрити bzip2– стиснутий файл в двійковому або текстовому режимі, повернувши файловий об'єкт.

Як і в конструкторі bz2file, аргументом filename може бути фактичне ім'я файлу (str або bytes об'єкт) або існуючий об'єкт файлу для читання або запису.

Аргумент mode може бути будь– яким з 'r', 'rb', 'w', 'wb', 'x', 'xb', 'A ' або' ab ' для двійкового режиму або 'rt', 'wt', 'xt' або 'at' для текстового режиму. Значення за замовчуванням – 'rb'.

Аргумент compresslevel є цілим числом від 1 до 9, як і для конструктора BZ2File.

Для двійкового режиму ця функція еквівалентна конструктору BZ2File: BZ2File (filename, mode, compresslevel=compresslevel). У цьому випадку аргументи encoding, errors та newline не повинні надаватися.

У текстовому режимі створюється bz2file об'єкт, який поміщається в інстанс io.TextIOWrapper з вказаною кодуванням, поведінкою обробки помилок і закінченнями рядків.

class bz2.BZ2File(filename, mode='r', buffering=None, compresslevel=9) відкрити bzip2– стиснутий файл в двійковому режимі.

Якщо filename є str або bytes об'єктом, відкрити іменований файл безпосередньо. В іншому випадку filename повинен бути файловим об'єктом, який буде використовуватися для читання або запису стислих даних.

Аргумент mode може бути ' r ' для читання (за замовчуванням), 'w' для перезапису, 'x' для монопольного створення або 'a' для додавання. Вони можуть бути еквівалентні' rb',' wb',' xb 'і' Ab ' відповідно.

Якщо filename є об'єктом файлу (а не фактичним ім'ям файлу), режим 'w' не усікає файл, а еквівалентний 'a'.

Аргумент buffering ігнорується. Його використання застаріло з Python 3.0.

Якщо mode 'w' або 'a', compresslevel може бути цілим числом між 1 і 9, що визначає ступінь стиснення: 1 забезпечує найменше стиснення, а 9 (За замовчуванням) – найбільше стиснення.

Якщо mode є 'r', вхідний файл може являти собою конкатенацію безлічі стиснутих потоків.

BZ2File надає всі елементи, зазначені io.BufferedIOBase, за винятком detach () і truncate (). Підтримуються ітерація і оператор with.

BZ2File також надає наступний метод: peek ([n]).

Повертає буферизовані дані без просування позиції файлу. Буде повертати принаймні один байт даних (якщо тільки при EOF). Точне число повернутих байтів не вказано.

class bz2.BZ2Compressor(compresslevel=9) створює новий об'єкт компресора. Цей об'єкт може бути використаний для інкрементального стиснення даних. Для однокадрового стиснення використовуйте функцію compress ().

compresslevel, якщо вказано, має бути цілим числом від 1 до 9. Значення за замовчуванням– 9.

compress (data) надасть дані об'єкту компресора. Повертає порцію стиснутих даних, якщо це можливо, або порожній байт рядка в іншому випадку.

Завершивши надання даних компресору, викличте flush () метод для завершення процесу стиснення.

flush () завершує процес стиснення. Повертає стислі дані, що залишилися у внутрішніх буферах.

Об'єкт компресора не може бути використаний після виклику цього методу.

class bz2. BZ2Decompressor створює новий об'єкт декомпресора. Цей об'єкт може бути використаний для інкрементного розпакування даних. Для однокадрового стиснення використовуйте замість цього функцію decompress ().

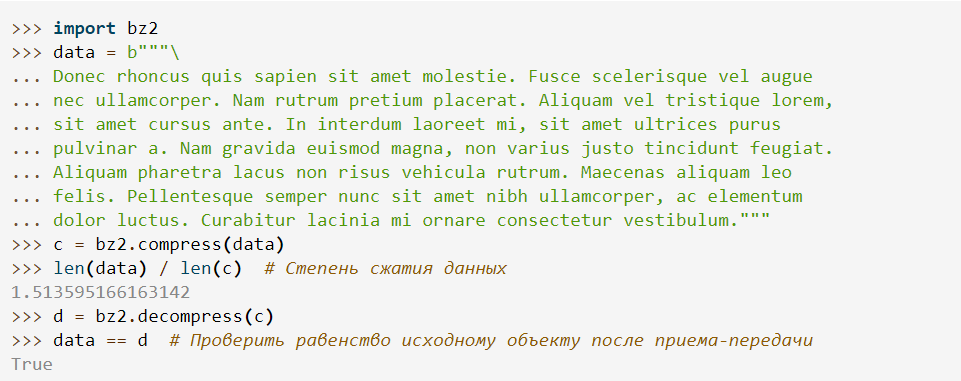
decompress (data, max\_length=– 1) розпакування data (байтоподібного об'єкта), повертаючи нестиснуті дані у вигляді байтів. Деякі з data можуть бути буферизовані всередині системи для використання в наступних викликах на decompress (). Повернуті дані повинні бути об'єднані з виведенням всіх попередніх викликів на decompress ().

Якщо max\_length є невід'ємним, повертає не більше max\_length байт розущільнених даних. Якщо ця межа досягнута і можливий подальший висновок, атрибут needs\_input буде встановлений в False. У цьому випадку наступний виклик decompress () може забезпечити data як b " для отримання більшої кількості вихідних даних.

Якщо всі вхідні дані були розпаковані і повернуті (або через те, що вони були менше max\_length байт, або через негативний max\_length), атрибут needs\_input буде встановлений в True.

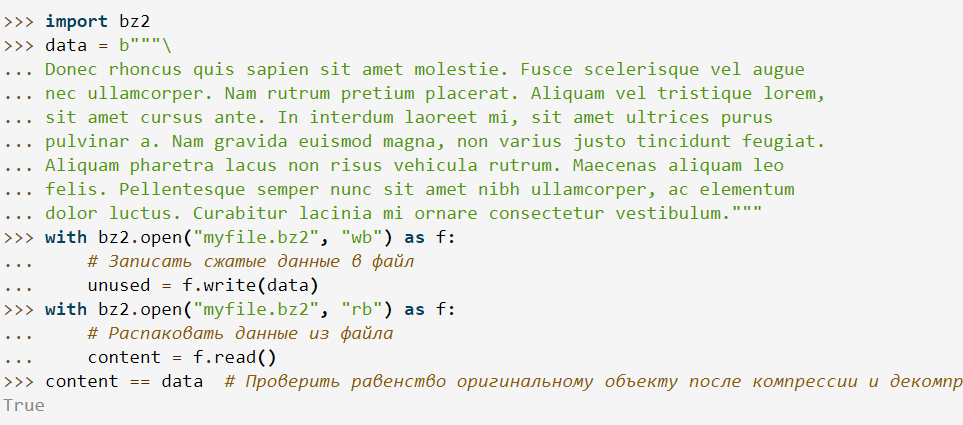
Нижче наведено приклади типового використання модуля bz2.

Використання compress () і decompress () для демонстрації двонаправленого стиснення на малюнку 1.9.



Малюнок 1.9 – Код стиснення

Запис і читання bzip2– стисненого файлу в двійковому режимі представлений на малюнку 1.10.



Малюнок 1.10 – Код читання і запису

# 1.4 lzma

Модуль надає класи і зручні функції для стиснення і розпакування даних з використанням алгоритму стиснення LZMA. Також включений файловий інтерфейс, що підтримує формати .xz і успадкованих файлів .lzma, використовуваний утилітою xz, а також необроблені стислі потоки.

Інтерфейс, наданий цим модулем, дуже схожий на інтерфейс модуля bz2. Однак зверніть увагу, що LZMAFile не є потокобезпечним, на відміну від bz2. BZ2File, тому якщо потрібно використовувати одну lzmafile сутність з декількох потоків, необхідно захистити його блокуванням.

lzma.open (filename, mode= "rb",\*, format=None, check=– 1, preset=None, filters=None, encoding=None, errors=None, newline=None) відкриває файл LZMA– стиснутий в двійковому або текстовому режимі, повертаючи файловий об'єкт.

Аргументом filename може бути або фактичне ім'я файлу (дане як об'єкт str, bytes або шляхоподібний об'єкт), в цьому випадку відкривається іменований файл, або існуючий об'єкт файлу для читання або запису.

Аргумент mode може бути будь– яким з" r"," rb"," w"," wb"," x"," xb"," a "або" ab "для двійкового режиму або" rt"," wt"," xt "або" at " для текстового режиму. Значення за замовчуванням– "rb".

При відкритті файлу для читання аргументи format і filters мають ті ж значення, що і для LZMADecompressor. У цьому випадку аргументи check і preset не повинні бути використовуваний.

При відкритті файлу для запису аргументи format, check, preset і filters мають ті ж значення, що і для LZMACompressor.

Для двійкового режиму ця функція еквівалентна конструктору LZMAFile: LZMAFile (filename, mode,...). У цьому випадку аргументи encoding, errors та newline не повинні бути надані.

Для текстового режиму створюється об'єкт LZMAFile, який поміщається в io.TextIOWrapper сутність із зазначеним кодування, поведінкою обробки помилок і закінченням рядка.

class lzma.LZMAFile (filename=None, mode="r", \*, format=None, check=– 1, preset=None, filters=None) відкриває файл LZMA– стиснутий в двійковому режимі. LZMAFile може обернути вже відкритий файловий об'єкт або впливати безпосередньо на названий файл. Аргумент filename вказує або об'єкт файлу для перенесення, або ім'я файлу для відкриття (як об'єкт str, bytes або шляхоподібний об'єкт). Обгортаючи існуючий об'єкт файлу, загорнутий файл не буде закритий, коли lzmafile буде закритий.

Аргумент mode може бути" r " для читання (за замовчуванням), "w" для перезапису, "x" для монопольного створення або "a" для додавання. Вони можуть бути еквівалентно дані як "rb", "wb", "xb" і "Ab" відповідно.

Якщо filename є файловим об'єктом (а не фактичним ім'ям файлу), режим "w" не усікає файл і замість цього еквівалентний "a".

При відкритті файлу для читання вхідний файл може бути об'єднанням безлічі окремих стиснутих потоків. Вони прозоро декодуються як єдиний логічний потік.

При відкритті файлу для читання аргументи format і filters мають ті ж значення, що і для LZMADecompressor. У цьому випадку аргументи check і preset не повинні бути використовуваний.

При відкритті файлу для запису аргументи format, check, preset і filters мають ті ж значення, що і для LZMACompressor.

LZMAFile підтримує всі елементи, зазначені io.BufferedIOBase, крім detach () і truncate (). Підтримуються ітерація і with інструкцією.

Також передбачений наступний спосіб: peek (size=– 1).

Повертає буферизовані дані без просування позиції файлу. Принаймні один байт даних буде повернутий, якщо не досягнуто значення EOF. Точне число байтів повернений не вказано (аргумент size ігнорується).

class lzma.LZMACompressor (format=FORMAT\_XZ, check=– 1, preset=None, filters=None) создаеь об'єкт компресора, який може бути використовуваний, щоб стиснути дані з прирощенням.

Більш зручний спосіб стиснення одного чанка даних.

Аргумент format вказує, який формат контейнера повинен бути використовуваний.

Аргумент check вказує тип перевірки цілісності для включення в стислі дані. Ця перевірка– використовуваний, розгортаючи, щоб гарантувати, що дані не були зіпсовані.

Якщо зазначена перевірка не підтримана, lzmaerror піднято.

Параметри стиснення можуть бути задані або у вигляді встановленого рівня стиснення (з аргументом preset), або детально у вигляді користувальницької ланцюжка фільтрів (з аргументом filters).

Аргумент preset (якщо він вказаний) повинен бути цілим числом між 0 і 9 (включно), необов'язково або з константою PRESET\_EXTREME. Якщо ні preset, ні filters не задані, за замовчуванням використовується поведінка PRESET\_DEFAULT (встановлений рівень 6). Більш високі налаштування дають менший вихід, але роблять процес стиснення більш повільним.

Аргумент filters (якщо він вказаний) повинен бути специфікатором ланцюжка фільтрів.

compress (data) стискає data (об'єкт bytes), повертаючи об'єкт bytes, що містить стислі дані, принаймні, для частини вхідних даних. Деякі з data можуть бути буферизовані всередині системи для використання в наступних викликах compress () і flush (). Дані повернений повинні конкатеніроваться з вихідними даними всіх попередніх викликів compress ().

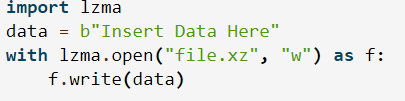
decompress (data, max\_length=– 1) розпаковує data (байтоподібний об'єкт), повертаючи нестиснуті дані в байтах. Деякі з data можуть бути буферизовані всередині системи для використання в наступних викликах decompress (). Дані повернений повинні конкатеніроваться з вихідними даними всіх попередніх викликів decompress ().

Якщо max\_length є невід'ємним, повертає не більше max\_length байт розпакованих даних. Якщо ця межа буде досягнута, і подальший висновок може бути проведена, то needs\_input атрибут буде встановлений в False. У цьому випадку наступний виклик decompress () може забезпечити data як b " для отримання більшої кількості вихідних даних.

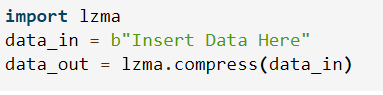
Якщо всі вхідні дані були розгорнуті і повернений (або тому що це було менше, ніж байти max\_length, або тому що max\_length був негативний), needs\_input атрибут буде встановлений в True.

Спроба розпакування даних після досягнення кінця потоку викликає "EOFError". Всі дані, знайдені після закінчення потоку, ігноруються і зберігаються в unused\_data атрибут.

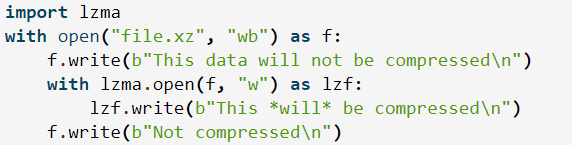
Створення стисненого файлу продемонстровано на малюнку 1.11. Стиснення даних в пам'яті продемонстровано на малюнку 1.12. Запис стислих даних у вже відкритий файл продемонстровано на малюнку 1.13.



Малюнок 1.11 – Код стиснення



Малюнок 1.12 – Код стиснення



Малюнок 1.13 – Код стиснення

# 1.5 zip– file

Модуль в даний час не обробляє багатодискові ZIP– файли. Він може обробляти ZIP– файли з розширеннями ZIP64(тобто ZIP– файли розміром більше 4 ГиБ). Він підтримує розшифровку зашифрованих файлів в ZIP– архівах, але в даний час не може створити зашифрований файл. Розшифровка виконується дуже повільно, оскільки вона реалізована на рідному Python, а не на C.

class zipfile.ZipFile (file, mode='r', compression=ZIP\_STORED, allowZip64=True, compresslevel=None,\*, strict\_timestamps=True) відкриває zip– файл, де file може бути шляхом до файлу (рядок), файлового об'єкта або шляхоподібний об'єкт.

Параметр mode повинен бути' r 'для читання існуючого файлу,' w 'для усічення і запису нового файлу, 'a' для додавання в існуючий файл або' x ' для монопольного створення і запису нового файлу. Якщо mode– це 'x', а file відноситься до існуючого файлу, буде піднято FileExistsError. Якщо mode– 'a', а file відноситься до існуючого файлу ZIP, то до нього додаються додаткові файли. Якщо file не відноситься до файлу ZIP, то до файлу додається новий архів ZIP. Це призначено для додавання Zip– архіву в інший файл (наприклад, python.exe). Якщо mode– 'a', А файл взагалі не існує, він створюється. Якщо mode – 'r' або 'a', файл повинен бути доступний для пошуку.

compression– це метод стиснення ZIP, який використовується при записі архіву, він повинен бути ZIP\_STORED, ZIP\_DEFLATED, ZIP\_BZIP2 або ZIP\_LZMA; нерозпізнані значення викличуть підняття NotImplementedError. Якщо вказано ZIP\_DEFLATED, ZIP\_BZIP2 або ZIP\_LZMA, але відповідний модуль (zlib, bz2 або lzma) недоступний, буде піднято RuntimeError. За замовчуванням– ZIP\_STORED.

Якщо allowZip64– це True (за замовчуванням), zipfile буде створювати ZIP– файли з розширеннями ZIP64, якщо розмір zipfile перевищує 4 ГиБ. Якщо false, zipfile викличе виняток, якщо для файлу ZIP потрібні розширення ZIP64.

Параметр compresslevel управляє рівнем стиснення, використовуваним при записі файлів в архів. При використанні ZIP\_STORED або ZIP\_LZMA він не діє. При використанні ZIP\_DEFLATED приймаються цілі числа від 0 до 9 (додаткову інформацію див.в zlib). При використанні ZIP\_BZIP2 приймаються цілі числа від 1 до 9 (додаткову інформацію див.в bz2).

Аргумент strict\_timestamps, якщо йому задано значення False, дозволяє архівувати файли старше 1 січня 1980 року за рахунок установки мітки часу на 01 січня 1980 року. Аналогічна поведінка відбувається з файлами новіше 2107– 12– 31, тимчасова мітка також встановлена на граничне значення.

Якщо файл створюється в режимі 'w', ' x ' або 'a', а потім closed без додавання файлів в архів, в файл будуть записані відповідні структури ZIP для порожнього архіву.

ZipFile.close() закриває архівний файл. Повинні викликати close () перед виходом з програми, інакше важливі записи не будуть записані.

ZipFile.getinfo (name) повертає Об'єкт ZipInfo з інформацією про елемент архіву name. Виклик getinfo () для імені, яке наразі не міститься в архіві, викличе KeyError.

ZipFile.infolist () повертає список, що містить Об'єкт ZipInfo для кожного елемента архіву. Об'єкти розташовані в тому ж порядку, що і їх записи в реальному ZIP– файлі на диску, якщо існуючий архів був відкритий.

ZipFile.namelist () повертає список елементів архіву за назвою.

ZipFile.open (name, mode='r', pwd=None, \*, force\_zip64=False) доступ до елементу архіву як до двійкового файлового об'єкта. name може бути або ім'ям файлу в архіві, або об'єктом ZipInfo. Якщо параметр mode включений, повинен приймати значення 'r' (за замовчуванням) або 'w'. pwd– це пароль, який використовується для розшифровки зашифрованих файлів ZIP.

ZipFile.extractall (path=None, members=None, pwd=None) розпаковує всі елементи з архіву в поточний робочий каталог. path вказує інший каталог для вилучення. members не є обов'язковим і повинен бути підмножиною списку, що повертається namelist(). PWD– пароль, який використовується для шифрування файлів.

ZipFile.write (filename, arcname=None, compress\_type=None, compresslevel=None) записує файл з ім'ям filename в архів, присвоївши йому ім'я архіву arcname (за замовчуванням це буде те ж саме, що і filename, але без букви диска і з віддаленими роздільниками шляху). Якщо задано, compress\_type перевизначає значення, вказане для параметра compression конструктору нового запису. Точно так само compresslevel перевизначить конструктор, якщо він заданий. Архів повинен бути відкритий в режимі 'w',' x ' або 'a'.

# 1.6 Tarfile

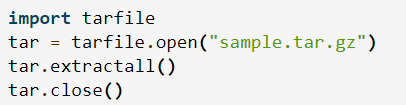
Модуль tarfile дозволяє читати і записувати tar– архіви, в тому числі з використанням стиснення gzip, bz2 і lzma. Використовуйте модуль zipfile для читання або запису .zip файлів або функцій більш високого рівня в shutil.

tarfile.open (name=None, mode='r', fileobj=None, bufsize=10240, \*\*kwargs) повертає Об'єкт TarFile для шляху name. Щоб отримати докладну інформацію про Об'єкти TarFile та дійсні ключові аргументи, перегляньте Об'єкти TarFile. mode повинен бути рядком у формі ' filemode [: compression]', за замовчуванням – 'r'.

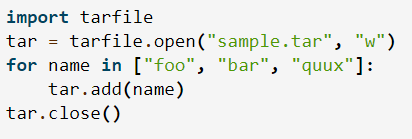
TarFile.extractfile (member) витягує елемент з архіву як файловий об'єкт. member може бути ім'ям файлу або об'єктом TarInfo. Якщо member– звичайний файл або посилання, повертається об'єкт io.BufferedReader. В іншому випадку повертається None.

TarFile.add (name, arcname=None, recursive=True, \*, filter=None) додає в архів файл name. name може бути файлом будь– якого типу (каталог, файл, символічне посилання тощо). Якщо задано, arcname вказує альтернативне ім'я файлу в архіві. За замовчуванням каталоги додаються рекурсивно. Цього можна уникнути, задавши для recursive значення False. Рекурсія додає записи у відсортованому порядку. Якщо задано filter, це повинна бути функція, яка приймає аргумент об'єкта TarInfo і повертає змінений об'єкт TarInfo. Якщо замість цього він поверне None, Об'єкт TarInfo буде виключений з архіву.

Як розпакувати весь tar– архів в поточний робочий каталог показано на малюнку 1.14. Як створити нестиснений tar– архів зі списку імен файлів показано на малюнку 1.15.



Малюнок 1.14 – Розпакування архіву



Малюнок 1.15 – Створення нестисненого архіву

# УКЛАДЕННЯ

Як видно з даного матеріалу, використання модулів стиснення і архівації, дозволяють гнучко використовувати стиснення файлів. Можна архівувати файли в директорії в різні архіви в залежності від їх типу, імені та розміру. Так само можна вирішувати залишати структуру директорії чи ні. Аналогічно при витяганні файлів, можна витягувати їх куди потрібно, на основі власних критеріїв, таких як розмір, і т.д. але при цьому варто пам'ятати, що не кожен модуль стиснення і архівування надає шифрування файлів з чого випливає, що необхідно вибирати модуль стиснення і архівації під необхідні цілі.

# СПИСОК ВИКОРИСТОВУВАНИХ ДЖЕРЕЛ

1 zlib. [Електронний ресурс]. – <https://digitology>. tech/docs/python\_3 /library/zlib.html

1. gzib. [Електронний ресурс]. – <https://digitology>. tech/docs/python\_3 /library/gzib.html
2. bz2. [Електронний ресурс]. – <https://digitology>. tech/docs/python\_3 /library/bz2.html
3. lzma. [Електронний ресурс]. – <https://digitology>. tech/docs/python\_3 /library/lzma.html
4. tarfile. [Електронний ресурс]. – <https://digitology>. tech/docs/python\_3 /library/tarfile.html