String Template Class & String Formatting using %
List Methods Set 1, Set 2, Set 3
Tuples & Sets
Dictionary Methods Set 1, Set 2
ChainMap
Explore More...
Control Flow
Loops and Control Statements
Counters & Accessing Counters
Iterators & Iterator Functions Set 1, Set 2
Generators
Explore More...
Functions
Function Decorators
Returning Multiple Values
Yield instead of Return
Python Closures & Coroutine
Explore More...
Modules
Introduction
Numeric Functions & Logarithmic and Power functions
Calender Functions Set 1, Set 2
Complex Numbers Introduction & Important functions
Explore More...
Object Oriented Concepts
Class, Object and Members
Data Hiding and Object Printing
Inheritance, Subclass and super
Class method vs static method & Class or Static Variables

Explore More...
Exception Handling
Exception Handling
User-Defined Exceptions
Built-in Exceptions
Libraries and Functions
Timeit
Numpy Set 1, Set 2
Get and Post
import module & reload module
Collection Modules Deque, Namedtuple & Heap
Explore More...
Machine Learning with Python
Classifying data using Support Vector Machines(SVMs) in Python
K means Clustering
How to get synonyms/antonyms from NLTK WordNet in Python?
Explore More...
Misc
Sql using Python & MongoDB and Python
Json formatting & Python Virtual environment
Metaprogramming with Metaclasses in Python
Python Input Methods for Competitive Programming
Explore More...
Applications and Projects
Creating a proxy webserver Set 1, Set 2
Send Messsage to FB friend
Twitter Sentiment Analysis & Whatsapp using Python

Desktop Notifier & Junk File Organizer

Explore More...

Working with zip files in Python

This article explains how one can perform various operations on a zip file using a simple python program.

What is a zip file?

ZIP is an archive file format that supports lossless data compression. By lossless compression, we mean that the compression algorithm allows the original data to be perfectly reconstructed from the compressed data. So, a ZIP file is a single file containing one or more compressed files, offering an ideal way to make large files smaller and keep related files together.

Why do we need zip files?

- To reduce storage requirements.
- To improve transfer speed over standard connections.

To work on zip files using python, we will use an inbuilt python module called zipfile.

1. Extracting a zip file

```
# importing required modules
from zipfile import ZipFile

# specifying the zip file name
file_name = "my_python_files.zip"

# opening the zip file in READ mode
with ZipFile(file_name, 'r') as zip:
    # printing all the contents of the zip file
    zip.printdir()

    # extracting all the files
    print('Extracting all the files now...')
    zip.extractall()
    print('Done!')
```

Run on IDE

The above program extracts a zip file named “my_python_files.zip” in the same directory as of this python script.
The output of above program may look like this:

```
File Name                                             Modified             Size
python_files/python_basic.pdf                 2016-12-08 23:05:34       210098
python_files/python_wiki.txt                  2016-12-08 23:02:40          990
python_files/logos/python_logo.jpg            2016-12-08 23:00:42        46623

Extracting all the files now...
Done!
```

Let us try to understand the above code in pieces:

- `from zipfile import ZipFile`

  ZipFile is a class of zipfile module for reading and writing zip files. Here we import only class ZipFile from zipfile module.

- `with ZipFile(file_name, 'r') as zip:`

  Here, a ZipFile object is made by calling ZipFile constructor which accepts zip file name and mode parameters. We create a ZipFile object in **READ** mode and name it as **zip**.

- `zip.printdir()`

  printdir() method prints a table of contents for the archive.

- `zip.extractall()`

  extractall() method will extract all the contents of the zip file to the current working directory. You can also call **extract()** method to extract any file by specifying its path in the zip file.
  For example:

  ```
  zip.extract('python_files/python_wiki.txt')
  ```

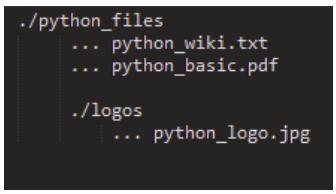
  This will extract only the specified file.

  If you want to read some specific file, you can go like this:

  ```
  data = zip.read(name_of_file_to_read)
  ```

2. Writing to a zip file

Consider a directory (folder) with such a format:



Here, we will need to crawl whole directory and its sub-directories in order to get a list of all file-paths before writing them to a zip file.
The following program does this by crawling the directory to be zipped:

```
# importing required modules
from zipfile import ZipFile
import os

def get_all_file_paths(directory):

    # initializing empty file paths list
    file_paths = []

    # crawling through directory and subdirectories
    for root, directories, files in os.walk(directory):
        for filename in files:
            # join the two strings in order to form the full filepath.
            filepath = os.path.join(root, filename)
            file_paths.append(filepath)

    # returning all file paths
    return file_paths

def main():
    # path to folder which needs to be zipped
    directory = './python_files'

    # calling function to get all file paths in the directory
    file_paths = get_all_file_paths(directory)

    # printing the list of all files to be zipped
    print('Following files will be zipped:')
    for file_name in file_paths:
        print(file_name)

    # writing files to a zipfile
    with ZipFile('my_python_files.zip','w') as zip:
        # writing each file one by one
        for file in file_paths:
            zip.write(file)

    print('All files zipped successfully!')
```

```
if __name__ == "__main__":
    main()
```

Run on IDE

The output of above program looks like this:

```
Following files will be zipped:
./python_files\python_basic.pdf
./python_files\python_wiki.txt
./python_files\logos\python_logo.jpg
All files zipped successfully!
```

Let us try to understand above code by dividing into fragments:

- ```
  def get_all_file_paths(directory):
      file_paths = []

      for root, directories, files in os.walk(directory):
          for filename in files:
              filepath = os.path.join(root, filename)
              file_paths.append(filepath)

      return file_paths
  ```

  First of all, to get all file paths in our directory, we have created this function which uses the **os.walk()** method. In each iteration, all files present in that directory are appended to a list called **file_paths**.
  In the end, we return all the file paths.

- ```
  file_paths = get_all_file_paths(directory)
  ```

  Here we pass the directory to be zipped to the **get_all_file_paths()** function and obtain a list containing all file paths.

- ```
  with ZipFile('my_python_files.zip','w') as zip:
  ```

  Here, we create a ZipFile object in WRITE mode this time.

- ```
  for file in file_paths:
              zip.write(file)
  ```

  Here, we write all the files to the zip file one by one using **write** method.

3. Getting all information about a zip file

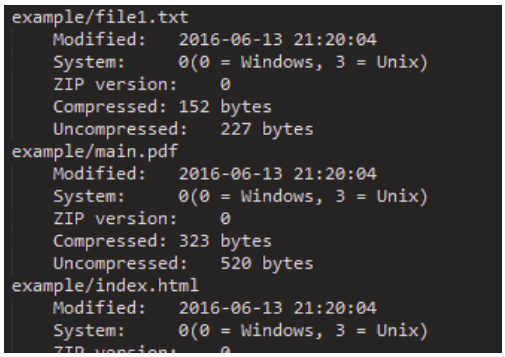
```
# importing required modules
from zipfile import ZipFile
import datetime

# specifying the zip file name
file_name = "example.zip"

# opening the zip file in READ mode
with ZipFile(file_name, 'r') as zip:
    for info in zip.infolist():
            print(info.filename)
            print('\tModified:\t' + str(datetime.datetime(*info.date_time)))
            print('\tSystem:\t\t' + str(info.create_system) + '(0 = Windows, 3 = Unix)')
            print('\tZIP version:\t' + str(info.create_version))
            print('\tCompressed:\t' + str(info.compress_size) + ' bytes')
            print('\tUncompressed:\t' + str(info.file_size) + ' bytes')
```

Run on I

The output of above program may look like this:




```
for info in zip.infolist():
```

Here, **infolist()** method creates an instance of **ZipInfo** class which contains all the information about the zip file.
We can access all information like last modification date of files, file names, system on which files were created, Zip version, size of files in compressed and uncompressed form, etc.

This article is contributed by **Nikhil Kumar**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GATE CS Corner Company Wise Coding Practice

GBlog Project Python

Login to Improve this Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Recommended Posts:

Downloading files from web using Python
Working with csv files in Python
Using Iterations in Python Effectively
Jar files in Java
Student Data Management in C++
Fetch top 10 starred repositories of user on GitHub | Python
Natural Language Programming — Teaching Kids
Activation functions in Neural Networks
Natural Language Programming
Cloud Based Services

(Login to Rate)

0 Average Difficulty : **0/5.0**
No votes yet.

Basic Easy Medium Hard Expert

Add to TODO List
Mark as DONE

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

Share this post!

 Contact Us! About Us! Careers! Privacy Policy