

# How to create a zip archive of a directory?

Asked 13 years, 9 months ago Modified 1 month ago Viewed 731k times



How can I create a zip archive of a directory structure in Python?

851

[python](#) [zip](#) [archive](#) [python-zipfile](#)



Share Follow



edited Dec 26, 2021 at 22:09



[Tomerikoo](#)

Tome 18.4k ● 16 ● 47 ● 61

asked Dec 6, 2009 at 11:12



[Martha Yi](#)

8,653 ● 3 ● 18 ● 8

76 Don't use the solution suggested in the accepted answer but the one further down using `make_archive` from `shutil` (if you want to zip a single directory recursively). – [malana](#) Oct 8, 2018 at 21:39

yes, agree with @malana - Martha Yi seems to be unregistered - so is there now way to change the accepted answer through a community process? – [Romeo Kienzler](#) Sep 20, 2021 at 14:22

one caveat with `shutil.make_archive` - it doesn't seem to follow symlinks – [LRE](#) Dec 2, 2021 at 16:21

2 The accepted answer is the only one that is actually thread safe in regards to read/write access while creating the zipfile from a directory since each file is opened individually, locking read access to it until the file is closed. – [Beefcake](#) Jun 16, 2022 at 17:03

Sorted by:

Highest score (default)

29 Answers



The easiest way is to use `shutil.make_archive`. It supports both zip and tar formats.

2091

```
import shutil
shutil.make_archive(output_filename, 'zip', dir_name)
```



If you need to do something more complicated than zipping the whole directory (such as skipping certain files), then you'll need to dig into the `zipfile` module as others have suggested.

Share Follow

edited Mar 11, 2019 at 15:01



[phoenix](#)

8,088 ● 6 ● 39 ● 45

answered Sep 3, 2014 at 17:26



[crdavis](#)

21k ● 3 ● 14 ● 9

280 `shutil` is part of the standard python library. This should be the top answer – [Alex](#) Apr 28, 2017 at 19:19

12 This is the most concise answer here and also has the advantage of adding all subdirectories and files to the archive directly, rather than having everything included in a top-level folder (which results in a

redundant level in the folder structure when unzipping). – [aitch-hat](#) Jun 22, 2017 at 8:05

- 3 @cmcginty could you please be a bit more specific as to what aspect of it is not thread-safe? Will running multiple threads while one calls this cause the interpreter to crash? – [std"OrgnlDave](#) Nov 4, 2017 at 22:49
- 25 Be warned that prior to Python 3.4, `shutil.make_archive` does not support ZIP64 and will fail on creating ZIP files larger than 2GB. – [dvs](#) Jan 18, 2018 at 23:42
- 9 @Teekin No. If you look at the bugs report ([bugs.python.org/issue30511](https://bugs.python.org/issue30511)), you'll see that `shutil.make_archive` uses `os.chdir()`. From what I'm reading about `os.chdir()`, it operates globally. – [Sam Malayek](#) Jul 25, 2018 at 0:36



706



As others have pointed out, you should use [zipfile](#). The documentation tells you what functions are available, but doesn't really explain how you can use them to zip an entire directory. I think it's easiest to explain with some example code:

```
import os
import zipfile

def zipdir(path, ziph):
    # ziph is zipfile handle
    for root, dirs, files in os.walk(path):
        for file in files:
            ziph.write(os.path.join(root, file),
                      os.path.relpath(os.path.join(root, file),
                                      os.path.join(path, '..')))

with zipfile.ZipFile('Python.zip', 'w', zipfile.ZIP_DEFLATED) as zipf:
    zipdir('tmp/', zipf)
```

Share Follow

edited Mar 18, 2022 at 11:15



[Nico Schlömer](#)

54k ● 27 ● 201 ● 250

answered Dec 6, 2009 at 11:23



[Mark Byers](#)

813k ● 193 ● 1583 ● 1452

- 150 I would add a second argument to the write call, passing `os.path.relpath(os.path.join(root, file), os.path.join(path, '..'))`. That would let you zip a directory from any working directory, without getting the full absolute paths in the archive. – [Reimund](#) Jun 29, 2013 at 14:35
- 13 There's a funny recursion going on when I try to zip a folder and output the resultant zip to the same folder. :-) – [Sibbs Gambling](#) Mar 23, 2017 at 4:22
- 38 `shutil` makes it really easy in just a single line. Please check the answer below.. – [droidlabour](#) May 30, 2017 at 18:24
- 7 you may be more interested by doing `ziph.write(os.path.join(path, file), arcname=file)` so that the filenames inside the archive are not relative to the hard drive – [Christophe Blin](#) Jun 14, 2017 at 7:42
- 2 The next answer does not handle anything related to compression, it just stores the results in a ZIP archive. If you are looking for actual compression, the correct answer is this one and not the next with `shutil`. – [José L. Patiño](#) Jul 11, 2021 at 16:13

To add the contents of `mydirectory` to a new zip file, including all files and subdirectories:



87



```
import os
import zipfile

zf = zipfile.ZipFile("myzipfile.zip", "w")
for dirname, subdirs, files in os.walk("mydirectory"):
    zf.write(dirname)
    for filename in files:
        zf.write(os.path.join(dirname, filename))
zf.close()
```

Share Follow

edited Dec 6, 2009 at 11:36

answered Dec 6, 2009 at 11:28



Ben James

121k ● 26 ● 193 ● 155

For me this code throwing below error `TypeError: invalid file: <zipfile.ZipFile [closed]>` – [Nishad Up](#)  
Aug 23, 2017 at 12:41

22 Can't you use a `with` instead of having to call `close()` yourself at the end? – [ArtOfWarfare](#) Jan 12, 2018 at 16:37

example: ``with zipfile.ZipFile("myzipfile.zip", "w") as zf: pass`` – [Subramanya Rao](#) Feb 2, 2021 at 11:30

2 This rebuilds the full path to "mydirectory" within the resulting zipfile. i.e. only works as desired if "mydirectory" is in root of your filesystem – [Maile Cupo](#) Nov 3, 2021 at 4:24

Using the `arcname` parameter of the write function will solve the issue where the entire directory branch is getting zipped rather than just the contents. – [Prince](#) Apr 21, 2022 at 15:22



63



## How can I create a zip archive of a directory structure in Python?

### In a Python script

In Python 2.7+, `shutil` has a `make_archive` function.

```
from shutil import make_archive
make_archive(
    'zipfile_name',
    'zip',          # the archive format - or tar, bztar, gztar
    root_dir=None,  # root for archive - current working dir if None
    base_dir=None)  # start archiving from here - cwd if None too
```

Here the zipped archive will be named `zipfile_name.zip`. If `base_dir` is farther down from `root_dir` it will exclude files not in the `base_dir`, but still archive the files in the parent dirs up to the `root_dir`.

I did have an issue testing this on Cygwin with 2.7 - it wants a `root_dir` argument, for cwd:

```
make_archive('zipfile_name', 'zip', root_dir='.')
```

## Using Python from the shell

You can do this with Python from the shell also using the `zipfile` module:

```
$ python -m zipfile -c zipname sourcedir
```

Where `zipname` is the name of the destination file you want (add `.zip` if you want it, it won't do it automatically) and `sourcedir` is the path to the directory.

## Zippping up Python (or just don't want parent dir):

If you're trying to zip up a python package with a `__init__.py` and `__main__.py`, and you don't want the parent dir, it's

```
$ python -m zipfile -c zipname sourcedir/*
```

And

```
$ python zipname
```

would run the package. (Note that you can't run subpackages as the entry point from a zipped archive.)

## Zippping a Python app:

If you have python3.5+, and specifically want to zip up a Python package, use [zipapp](#):

```
$ python -m zipapp myapp
$ python myapp.pyz
```

Share Follow

edited Jan 6, 2017 at 17:29

answered Mar 31, 2016 at 18:57



[Russia Must Remove Putin](#) ♦

375k ● 89 ● 403 ● 331

This is the best and simplest solution. – [Cary](#) Apr 13 at 10:05



44

This function will recursively zip up a directory tree, *compressing* the files, and recording the correct relative filenames in the archive. The archive entries are the same as those generated by `zip -r output.zip source_dir`.

```
import os
import zipfile
def make_zipfile(output_filename, source_dir):
    relroot = os.path.abspath(os.path.join(source_dir, os.pardir))
    with zipfile.ZipFile(output_filename, "w", zipfile.ZIP_DEFLATED) as zip:
        for root, dirs, files in os.walk(source_dir):
            # add directory (needed for empty dirs)
            zip.write(root, os.path.relpath(root, relroot))
            for file in files:
                filename = os.path.join(root, file)
                if os.path.isfile(filename): # regular files only
                    arcname = os.path.join(os.path.relpath(root, relroot), file)
                    zip.write(filename, arcname)
```

Share Follow

edited Apr 18, 2018 at 15:38

answered Jun 13, 2013 at 6:57

Guillaume Jacquenot  
11.3k ● 6 ● 43 ● 49George V. Reilly  
15.9k ● 7 ● 44 ● 38

Sweet, I was wondering if `zipfile` could be used in a `with` statement. Thanks for pointing out it can. – Mitms Nov 16, 2021 at 8:02

With python 3.9, `pathlib` & `zipfile` module you can create a zip files from anywhere in the system.

29

```
def zip_dir(dir: Union[Path, str], filename: Union[Path, str]):
    """Zip the provided directory without navigating to that directory using `pathlib`
    module"""

    # Convert to Path object
    dir = Path(dir)

    with zipfile.ZipFile(filename, "w", zipfile.ZIP_DEFLATED) as zip_file:
        for entry in dir.rglob("*"):
            zip_file.write(entry, entry.relative_to(dir))
```

It is neat, typed, and has less code.

Share Follow

edited Aug 17, 2021 at 12:11

answered Aug 17, 2021 at 12:05

JD Solanki  
898 ● 8 ● 18

Modern Python (3.6+) using the `pathlib` module for concise OOP-like handling of paths, and `pathlib.Path.rglob()` for recursive globbing. As far as I can tell, this is equivalent to George V. Reilly's answer: zips with compression, the topmost element is a directory, keeps empty dirs, uses relative paths.

```
from pathlib import Path
from zipfile import ZIP_DEFLATED, ZipFile

from os import PathLike
from typing import Union
```

```
def zip_dir(zip_name: str, source_dir: Union[str, PathLike]):
    src_path = Path(source_dir).expanduser().resolve(strict=True)
    with ZipFile(zip_name, 'w', ZIP_DEFLATED) as zf:
        for file in src_path.rglob('*'):
            zf.write(file, file.relative_to(src_path.parent))
```

Note: as optional type hints indicate, `zip_name` can't be a Path object ([would be fixed in 3.6.2+](#)).

Share Follow

answered Mar 31, 2017 at 13:01



monk-time

2,123 ● 1 ● 14 ● 17

6 Fantastic! Concise! Modern! – [ingyhere](#) Apr 27, 2020 at 6:46 ✎



Use shutil, which is part of python standard library set. Using shutil is so simple(see code below):

27



- 1st arg: Filename of resultant zip/tar file,
- 2nd arg: zip/tar,
- 3rd arg: dir\_name



Code:

```
import shutil
shutil.make_archive('/home/user/Desktop/Filename', 'zip', '/home/username/Desktop/Directory')
```

Share Follow

edited Oct 12, 2018 at 10:27

answered Oct 12, 2018 at 9:46



Qback

4,310 ● 3 ● 25 ● 38



Vadiraj S J

639 ● 10 ● 17

1 With all the `shutil.make_archive` examples here, they all create empty root folders leading up to the folder that you actually want to archive. I do not want my archive file unarchive with `"/home/user/Desktop"` so everyone can see where folder of interest was originally. How do i just zip `"/Directory"` and leave out all traces of parent folders? – [kravb](#) Feb 12, 2021 at 23:14

1 This has been said 3 times already. And it's definitely not the best answer. – [José L. Patiño](#) Jul 11, 2021 at 16:17



For adding compression to the resulting zip file, check out [this link](#).

13

You need to change:



```
zip = zipfile.ZipFile('Python.zip', 'w')
```



to



```
zip = zipfile.ZipFile('Python.zip', 'w', zipfile.ZIP_DEFLATED)
```

Share Follow

edited Oct 20, 2014 at 8:57

answered Apr 27, 2013 at 17:14



philshem

24.8k ● 8 ● 61 ● 127



E Smith

137 ● 1 ● 3



7



I've made some changes to [code given by Mark Byers](#). Below function will also adds empty directories if you have them. Examples should make it more clear what is the path added to the zip.

```
#!/usr/bin/env python
import os
import zipfile

def addDirToZip(zipHandle, path, basePath=""):
    """
    Adding directory given by \a path to opened zip file \a zipHandle

    @param basePath path that will be removed from \a path when adding to archive

    Examples:
        # add whole "dir" to "test.zip" (when you open "test.zip" you will see only
        "dir")
        zipHandle = zipfile.ZipFile('test.zip', 'w')
        addDirToZip(zipHandle, 'dir')
        zipHandle.close()

        # add contents of "dir" to "test.zip" (when you open "test.zip" you will see
        only it's contents)
        zipHandle = zipfile.ZipFile('test.zip', 'w')
        addDirToZip(zipHandle, 'dir', 'dir')
        zipHandle.close()

        # add contents of "dir/subdir" to "test.zip" (when you open "test.zip" you will
        see only contents of "subdir")
        zipHandle = zipfile.ZipFile('test.zip', 'w')
        addDirToZip(zipHandle, 'dir/subdir', 'dir/subdir')
        zipHandle.close()

        # add whole "dir/subdir" to "test.zip" (when you open "test.zip" you will see
        only "subdir")
        zipHandle = zipfile.ZipFile('test.zip', 'w')
        addDirToZip(zipHandle, 'dir/subdir', 'dir')
        zipHandle.close()

        # add whole "dir/subdir" with full path to "test.zip" (when you open "test.zip"
        you will see only "dir" and inside it only "subdir")
        zipHandle = zipfile.ZipFile('test.zip', 'w')
        addDirToZip(zipHandle, 'dir/subdir')
        zipHandle.close()

        # add whole "dir" and "otherDir" (with full path) to "test.zip" (when you open
        "test.zip" you will see only "dir" and "otherDir")
        zipHandle = zipfile.ZipFile('test.zip', 'w')
        addDirToZip(zipHandle, 'dir')
        addDirToZip(zipHandle, 'otherDir')
        zipHandle.close()
```

```

"""
basePath = basePath.rstrip("\\\\") + ""
basePath = basePath.rstrip("\\\\")
for root, dirs, files in os.walk(path):
    # add dir itself (needed for empty dirs
    zipHandle.write(os.path.join(root, "."))
    # add files
    for file in files:
        filePath = os.path.join(root, file)
        inZipPath = filePath.replace(basePath, "", 1).lstrip("\\\\")
        #print filePath + " , " + inZipPath
        zipHandle.write(filePath, inZipPath)

```

Above is a simple function that should work for simple cases. You can find more elegant class in my Gist: <https://gist.github.com/Eccenux/17526123107ca0ac28e6>

Share Follow

edited May 23, 2017 at 12:10

answered Jun 10, 2013 at 9:28

Community Bot  
1 ● 1Nux  
9,286 ● 5 ● 59 ● 72

1 The path handling could be greatly simplified by using [os.path](#). See my answer. – George V. Reilly Jun 13, 2013 at 18:09

Bug: zipHandle.write(os.path.join(root, ".")) does not take basePath into consideration. – Petter Aug 30, 2014 at 9:16

Yes, you're probably right. I've later enhanced this a bit ;-)  
[gist.github.com/Eccenux/17526123107ca0ac28e6](https://gist.github.com/Eccenux/17526123107ca0ac28e6) – Nux Aug 30, 2014 at 18:00



6



To retain the folder hierarchy under the parent directory to be archived:

```

from pathlib import Path
import zipfile

fp_zip = Path("output.zip")
path_to_archive = Path("./path-to-archive")

with zipfile.ZipFile(fp_zip, "w", zipfile.ZIP_DEFLATED) as zipf:
    for fp in path_to_archive.glob("**/*"):
        zipf.write(fp, arcname=fp.relative_to(path_to_archive))

```

Share Follow

edited May 12 at 13:01

answered Jul 9, 2019 at 8:48

ryanjdillon  
17.7k ● 9 ● 86 ● 110

4



I have another code example that may help, using python3, pathlib and zipfile. It should work in any OS.

```

from pathlib import Path
import zipfile
from datetime import datetime

DATE_FORMAT = '%y%m%d'

```





```

def date_str():
    """returns the today string year, month, day"""
    return '{}'.format(datetime.now().strftime(DATE_FORMAT))

def zip_name(path):
    """returns the zip filename as string"""
    cur_dir = Path(path).resolve()
    parent_dir = cur_dir.parents[0]
    zip_filename = '{}/{}_{}.zip'.format(parent_dir, cur_dir.name, date_str())
    p_zip = Path(zip_filename)
    n = 1
    while p_zip.exists():
        zip_filename = ('{}_{}_{}.zip'.format(parent_dir, cur_dir.name,
                                              date_str(), n))
        p_zip = Path(zip_filename)
        n += 1
    return zip_filename

def all_files(path):
    """iterator returns all files and folders from path as absolute path string"""
    for child in Path(path).iterdir():
        yield str(child)
        if child.is_dir():
            for grand_child in all_files(str(child)):
                yield str(Path(grand_child))

def zip_dir(path):
    """generate a zip"""
    zip_filename = zip_name(path)
    zip_file = zipfile.ZipFile(zip_filename, 'w')
    print('create:', zip_filename)
    for file in all_files(path):
        print('adding... ', file)
        zip_file.write(file)
    zip_file.close()

if __name__ == '__main__':
    zip_dir('.')
    print('end!')

```

Share Follow

answered Oct 2, 2015 at 10:03



duncanmonty

41 ● 1 ● 3



3

If you want a functionality like the compress folder of any common graphical file manager you can use the following code, it uses the [zipfile](#) module. Using this code you will have the zip file with the path as its root folder.



```

import os
import zipfile

def zipdir(path, ziph):
    # Iterate all the directories and files

```



```

for root, dirs, files in os.walk(path):
    # Create a prefix variable with the folder structure inside the path folder.
    # So if a file is at the path directory will be at the root directory of the
zip file
    # so the prefix will be empty. If the file belongs to a containing folder of
path folder
    # then the prefix will be that folder.
    if root.replace(path, '') == '':
        prefix = ''
    else:
        # Keep the folder structure after the path folder, append a '/' at the
end
        # and remove the first character, if it is a '/' in order to have a
path like
        # folder1/folder2/file.txt
        prefix = root.replace(path, '') + '/'
        if (prefix[0] == '/'):
            prefix = prefix[1:]
    for filename in files:
        actual_file_path = root + '/' + filename
        zipped_file_path = prefix + filename
        zipf.write( actual_file_path, zipped_file_path)

zipf = zipfile.ZipFile('Python.zip', 'w', zipfile.ZIP_DEFLATED)
zipdir('/tmp/justtest/', zipf)
zipf.close()

```

Share Follow

edited Mar 22, 2016 at 11:15

answered Mar 21, 2016 at 13:40



VGe0rge

1,030 ● 3 ● 15 ● 18



3

So many answers here, and I hope I might contribute with my own version, which is based on the original answer (by the way), but with a more graphical perspective, also using context for each `zipfile` setup and sorting `os.walk()`, in order to have a ordered output.



Having these folders and them files (among other folders), I wanted to create a `.zip` for each `cap_` folder:



```

$ tree -d
.
├── cap_01
│   ├── 0101000001.json
│   ├── 0101000002.json
│   └── 0101000003.json
├── cap_02
│   ├── 0201000001.json
│   ├── 0201000002.json
│   └── 0201001003.json
├── cap_03
│   ├── 0301000001.json
│   ├── 0301000002.json
│   └── 0301000003.json
└── docs
    ├── map.txt
    └── main_data.xml

```

```

|
├─ core_files
│   ├── core_master
│   └── core_slave

```

Here's what I applied, with comments for better understanding of the process.

```

$ cat zip_cap_dirs.py
""" Zip 'cap_*' directories. """
import os
import zipfile as zf

for root, dirs, files in sorted(os.walk('.')):
    if 'cap_' in root:
        print(f"Compressing: {root}")
        # Defining .zip name, according to Capítulo.
        cap_dir_zip = '{}.zip'.format(root)
        # Opening zipfile context for current root dir.
        with zf.ZipFile(cap_dir_zip, 'w', zf.ZIP_DEFLATED) as new_zip:
            # Iterating over os.walk list of files for the current root dir.
            for f in files:
                # Defining relative path to files from current root dir.
                f_path = os.path.join(root, f)
                # Writing the file on the .zip file of the context
                new_zip.write(f_path)

```

Basically, for each iteration over `os.walk(path)`, I'm opening a context for `zipfile` setup and afterwards, iterating over `files`, which is a `list` of files from `root` directory, forming the relative path for each file based on the current `root` directory, appending to the `zipfile` context which is running.

And the output is presented like this:

```

$ python3 zip_cap_dirs.py
Compressing: ./cap_01
Compressing: ./cap_02
Compressing: ./cap_03

```

To see the contents of each `.zip` directory, you can use `less` command:

```

$ less cap_01.zip

```

Length	Method	Size	Cmpr	Date	Time	CRC-32	Name
22017	Defl:N	2471	89%	2019-09-05	08:05	7a3b5ec6	cap_01/0101000001.json
21998	Defl:N	2471	89%	2019-09-05	08:05	155bec7	cap_01/0101000002.json
23236	Defl:N	2573	89%	2019-09-05	08:05	55fcd20	cap_01/0101000003.json
67251		7515	89%				3 files

Share Follow

answered Sep 5, 2019 at 13:10



ivanleonz

9,150 ● 7 ● 57 ● 49



A solution using `pathlib.Path`, which is independent of the OS used:

3



```
import zipfile
from pathlib import Path

def zip_dir(path: Path, zip_file_path: Path):
    """Zip all contents of path to zip_file"""
    files_to_zip = [
        file for file in path.glob('*') if file.is_file()]
    with zipfile.ZipFile(
        zip_file_path, 'w', zipfile.ZIP_DEFLATED) as zip_f:
        for file in files_to_zip:
            print(file.name)
            zip_f.write(file, file.name)

current_dir = Path.cwd()
tozip_dir = current_dir / "test"
zip_dir(
    tozip_dir, current_dir / 'dir.zip')
```

Share Follow

edited Mar 9 at 16:10

answered Jan 5, 2021 at 9:09



Alex

2,793 ● 2 ● 32 ● 46



You probably want to look at the `zipfile` module; there's documentation at <http://docs.python.org/library/zipfile.html>.

2



You may also want `os.walk()` to index the directory structure.



Share Follow

edited Apr 7, 2010 at 16:32

answered Dec 6, 2009 at 11:17



me\_and

15.2k ● 8 ● 60 ● 96



To give more flexibility, e.g. select directory/file by name use:

2



```
import os
import zipfile

def zipall(ob, path, rel=""):
    basename = os.path.basename(path)
    if os.path.isdir(path):
        if rel == "":
            rel = basename
        ob.write(path, os.path.join(rel))
        for root, dirs, files in os.walk(path):
            for d in dirs:
                zipall(ob, os.path.join(root, d), os.path.join(rel, d))
            for f in files:
                ob.write(os.path.join(root, f), os.path.join(rel, f))
            break
    elif os.path.isfile(path):
        ob.write(path, os.path.join(rel, basename))
```

```
else:
    pass
```

For a file tree:

```

.
├── dir
│   ├── dir2
│   │   └── file2.txt
│   ├── dir3
│   │   └── file3.txt
│   └── file.txt
├── dir4
│   ├── dir5
│   └── file4.txt
├── listdir.zip
├── main.py
├── root.txt
└── selective.zip
```

You can e.g. select only `dir4` and `root.txt`:

```

cwd = os.getcwd()
files = [os.path.join(cwd, f) for f in ['dir4', 'root.txt']]

with zipfile.ZipFile("selective.zip", "w" ) as myzip:
    for f in files:
        zipall(myzip, f)
```

Or just `listdir` in script invocation directory and add everything from there:

```

with zipfile.ZipFile("listdir.zip", "w" ) as myzip:
    for f in os.listdir():
        if f == "listdir.zip":
            # Creating a listdir.zip in the same directory
            # will include listdir.zip inside itself, beware of this
            continue
        zipall(myzip, f)
```

Share Follow

answered Sep 25, 2018 at 13:32



pbn

2,416 ● 2 ● 26 ● 39

This zips, but doesn't compress. – Alex Dec 12, 2018 at 16:50



1



Here is a variation on the answer given by Nux that works for me:

```

def WriteDirectoryToZipFile( zipHandle, srcPath, zipLocalPath = "", zipOperation =
zipfile.ZIP_DEFLATED ):
    basePath = os.path.split( srcPath )[ 0 ]
    for root, dirs, files in os.walk( srcPath ):
        p = os.path.join( zipLocalPath, root [ ( len( basePath ) + 1 ) : ] )
        # add dir
```



```
zipHandle.write( root, p, zipOperation )
# add files
for f in files:
    filePath = os.path.join( root, f )
    fileInZipPath = os.path.join( p, f )
    zipHandle.write( filePath, fileInZipPath, zipOperation )
```

Share Follow

answered Jul 30, 2014 at 23:13



M Katz

5,098 ● 3 ● 44 ● 66

**Try the below one .it worked for me.**

1



```
import zipfile, os
zipf = "compress.zip"
def main():
    directory = r"Filepath"
    toZip(directory)
def toZip(directory):
    zippedHelp = zipfile.ZipFile(zipf, "w", compression=zipfile.ZIP_DEFLATED )

    list = os.listdir(directory)
    for file_list in list:
        file_name = os.path.join(directory,file_list)

        if os.path.isfile(file_name):
            print file_name
            zippedHelp.write(file_name)
        else:
            addFolderToZip(zippedHelp,file_list,directory)
            print "-----Directory Found-----"
    zippedHelp.close()

def addFolderToZip(zippedHelp,folder,directory):
    path=os.path.join(directory,folder)
    print path
    file_list=os.listdir(path)
    for file_name in file_list:
        file_path=os.path.join(path,file_name)
        if os.path.isfile(file_path):
            zippedHelp.write(file_path)
        elif os.path.isdir(file_name):
            print "-----sub directory found-----"
            addFolderToZip(zippedHelp,file_name,path)

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

Share Follow

answered Apr 23, 2015 at 11:18



Chandra

111 ● 1 ● 3 ● 12



Say you want to Zip all the folders(sub directories) in the current directory.

1



```
for root, dirs, files in os.walk("."):
    for sub_dir in dirs:
        zip_you_want = sub_dir+".zip"
        zip_process = zipfile.ZipFile(zip_you_want, "w", zipfile.ZIP_DEFLATED)
        zip_process.write(file_you_want_to_include)
        zip_process.close()

    print("Successfully zipped directory: {sub_dir}".format(sub_dir=sub_dir))
```

Share Follow

edited Jun 17, 2019 at 18:29



Uli Köhler

13k ● 16 ● 70 ● 120

answered Apr 9, 2019 at 8:25



snowpeak

797 ● 9 ● 25



1



Zip a file or a tree (a directory and its sub-directories).

```
from pathlib import Path
from zipfile import ZipFile, ZIP_DEFLATED

def make_zip(tree_path, zip_path, mode='w', skip_empty_dir=False):
    with ZipFile(zip_path, mode=mode, compression=ZIP_DEFLATED) as zf:
        paths = [Path(tree_path)]
        while paths:
            p = paths.pop()
            if p.is_dir():
                paths.extend(p.iterdir())
                if skip_empty_dir:
                    continue
            zf.write(p)
```

To append to an existing archive, pass `mode='a'`, to create a fresh archive `mode='w'` (the default in the above). So let's say you want to bundle 3 different directory trees under the same archive.

```
make_zip(path_to_tree1, path_to_arch, mode='w')
make_zip(path_to_tree2, path_to_arch, mode='a')
make_zip(path_to_file3, path_to_arch, mode='a')
```

Share Follow

edited Sep 20, 2020 at 22:59

answered Sep 17, 2020 at 5:50



Michael Ekoka

19.1k ● 12 ● 78 ● 79



1



The obvious way to go would be to go with `shutil`, Like the second top answer says so, But if you still wish to go with `ZipFile` for some reason, And if you are getting some trouble doing that (Like ERR 13 in Windows etc), You can use this fix:

```
import os
import zipfile

def retrieve_file_paths(dirName):
    file_paths = []
    for root, directories, files in os.walk(dirName):
        for filename in files:
```

```

    filePath = os.path.join(root, filename)
    filePaths.append(filePath)
    return filePaths

def main(dir_name, output_filename):
    filePaths = retrieve_file_paths(dir_name)

    zip_file = zipfile.ZipFile(output_filename+'.zip', 'w')
    with zip_file:
        for file in filePaths:
            zip_file.write(file)

main("my_dir", "my_dir_archived")

```

This one recursively iterates through every sub-folder/file in your given folder, And writes them to a zip file instead of attempting to directly zip a folder.

Share Follow

answered Mar 10, 2021 at 5:09



GuruPrasaathM  
23 ● 5



0



Here's a modern approach, using pathlib, and a context manager. Puts the files directly in the zip, rather than in a subfolder.

```

def zip_dir(filename: str, dir_to_zip: pathlib.Path):
    with zipfile.ZipFile(filename, 'w', zipfile.ZIP_DEFLATED) as zipf:
        # Use glob instead of iterdir(), to cover all subdirectories.
        for directory in dir_to_zip.glob('*'):
            for file in directory.iterdir():
                if not file.is_file():
                    continue
                # Strip the first component, so we don't create an unneeded subdirectory
                # containing everything.
                zip_path = pathlib.Path(*file.parts[1:])
                # Use a string, since zipfile doesn't support pathlib directly.
                zipf.write(str(file), str(zip_path))

```

Share Follow

answered Dec 14, 2016 at 21:50



Turtles Are Cute  
3,208 ● 6 ● 30 ● 38



0



I prepared a function by consolidating Mark Byers' solution with Reimund and Morten Zilmer's comments (relative path and including empty directories). As a best practice, `with` is used in ZipFile's file construction.

The function also prepares a default zip file name with the zipped directory name and '.zip' extension. Therefore, it works with only one argument: the source directory to be zipped.

```

import os
import zipfile

def zip_dir(path_dir, path_file_zip=''):
    if not path_file_zip:

```



```

path_file_zip = os.path.join(
    os.path.dirname(path_dir), os.path.basename(path_dir)+'.zip')
with zipfile.ZipFile(path_file_zip, 'wb', zipfile.ZIP_DEFLATED) as zip_file:
    for root, dirs, files in os.walk(path_dir):
        for file_or_dir in files + dirs:
            zip_file.write(
                os.path.join(root, file_or_dir),
                os.path.relpath(os.path.join(root, file_or_dir),
                                os.path.join(path_dir, os.path.pardir)))

```

Share Follow

edited Dec 26, 2016 at 15:15

answered Dec 24, 2016 at 20:32



Gürol Canbek

1,106 ● 1 ● 14 ● 20



0



```

# import required python modules
# You have to install zipfile package using pip install

import os,zipfile

# Change the directory where you want your new zip file to be

os.chdir('Type your destination')

# Create a new zipfile ( I called it myfile )

zf = zipfile.ZipFile('myfile.zip','w')

# os.walk gives a directory tree. Access the files using a for loop

for dirnames,folders,files in os.walk('Type your directory'):
    zf.write('Type your Directory')
    for file in files:
        zf.write(os.path.join('Type your directory',file))

```

Share Follow

answered Jun 15, 2017 at 6:24



Praveen

1 ● 1



0



Well, after reading the suggestions I came up with a very similar way that works with 2.7.x without creating "funny" directory names (absolute-like names), and will only create the specified folder inside the zip.

Or just in case you needed your zip to contain a folder inside with the contents of the selected directory.

```

def zipDir( path, ziph ) :
    """
    Inserts directory (path) into zipfile instance (ziph)
    """
    for root, dirs, files in os.walk( path ) :
        for file in files :
            ziph.write( os.path.join( root, file ) , os.path.basename( os.path.normpath( path )
            ) + "\\\" + file )

def makeZip( pathToFolder ) :

```

```

"""
Creates a zip file with the specified folder
"""

zipf = zipfile.ZipFile( pathToFolder + 'file.zip', 'w', zipfile.ZIP_DEFLATED )
zipDir( pathToFolder, zipf )
zipf.close()
print( "Zip file saved to: " + pathToFolder)

makeZip( "c:\\path\\to\\folder\\to\\insert\\into\\zipfile" )

```

Share Follow

answered Aug 15, 2018 at 21:27



Xedret

1,823 ● 18 ● 25



Function to create zip file.

0



```

def CREATEZIPFILE(zipname, path):
    #function to create a zip file
    #Parameters: zipname - name of the zip file; path - name of folder/file to be put
    in zip file

    zipf = zipfile.ZipFile(zipname, 'w', zipfile.ZIP_DEFLATED)
    zipf.setpassword(b"password") #if you want to set password to zipfile

    #checks if the path is file or directory
    if os.path.isdir(path):
        for files in os.listdir(path):
            zipf.write(os.path.join(path, files), files)

    elif os.path.isfile(path):
        zipf.write(os.path.join(path), path)
    zipf.close()

```

Share Follow

answered Nov 19, 2018 at 3:55



sushh

115 ● 1 ● 10

please explain with an example so that i can correct my answer – [sushh](#) Jan 29, 2019 at 8:45

However, zipfile "currently cannot create an encrypted file" (from [docs.python.org/3.9/library/zipfile.html](https://docs.python.org/3.9/library/zipfile.html)) – [Georg](#) May 14, 2020 at 14:05



0



One thing completely missed by previous answers is the fact that using `os.path.join()` can easily return POSIX-incompatible paths when you run the code on Windows. The resulting archive will contain files with literal backslashes in their names when processing it with any common archive software on Linux, which is not what you want. Use `path.as_posix()` for the `arcname` parameter instead!

```

import zipfile
from pathlib import Path
with zipfile.ZipFile("archive.zip", "w", zipfile.ZIP_DEFLATED) as zf:

```

```
for path in Path("include_all_of_this_folder").rglob("*"):
    zf.write(path, path.as_posix())
```

Share Follow

answered Jan 7 at 14:54



Zyl

2,720 ● 2 ● 22 ● 27



0



While using `shutil` note that you should include output directory path in `base_name` arg:

```
import shutil

shutil.make_archive(
    base_name=output_dir_path + output_filename_without_extension,
    format="zip",
    root_dir=input_root_dir)
```

Share Follow

answered Aug 7 at 12:29



AmiNadimi

5,149 ● 3 ● 39 ● 55



**Highly active question.** Earn 10 reputation (not counting the [association bonus](#)) in order to answer this question. The reputation requirement helps protect this question from spam and non-answer activity.